HITACHI
Inspire the Next

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

4010-1J-500-50

前書き

■ 対象製品

RT-1V63-5021 Hitachi Microservices Platform - Paxos Commit Transaction Orchestrator 01-06 (適用 OS：Red Hat Universal Base Image 8(64-bit x86_64)、Red Hat Universal Base Image 8 Minimal(64-bit x86_64)、Red Hat Universal Base Image 9(64-bit x86_64)、Red Hat Universal Base Image 9 Minimal(64-bit x86_64)、Almalinux 9 Minimal 9.5(64-bit x86_64))

RT-1V63-8021 Hitachi Microservices Platform - Bridge Option for OpenTP1 01-06 (適用 OS：Red Hat Universal Base Image 8(64-bit x86_64)、Red Hat Universal Base Image 8 Minimal(64-bit x86_64)、Red Hat Universal Base Image 9(64-bit x86_64)、Red Hat Universal Base Image 9 Minimal(64-bit x86_64))

■ 輸出時の注意

本製品を輸出される場合には、外国為替および外国貿易法の規制なら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

2026 年 4 月 4010-1J-500-50

■ 著作権

All Rights Reserved. Copyright (C) 2024, 2026, Hitachi, Ltd.

変更内容

変更内容（4010-1J-500-50） Hitachi Microservices Platform - Paxos Commit Transaction Orchestrator 01-06

追加・変更内容	変更個所
Jaeger v2 に対応しました。	1.1、1.2.5、2.2.17(1)(a)、2.2.17(2)、2.6.2、2.6.3、2.6.4、2.6.5、2.6.6、2.6.7、2.6.8、2.7.2(7)、2.7.2(8)、3.7.1、3.7.6、3.7.13(2)、3.8.3、3.12.1(1)(c)、3.12.5(2)、4.1、7.3、7.5.1、11.3.2、11.3.5、11.4.1(1)、11.4.2(1)、11.5.2(2)(a)、11.5.3(1)、11.7.1(2)(b)、付録 D
Amazon Web Services 環境でトラブルシュート情報を収集できるようにしました。	1.2.5、1.2.6、1.4.1、2.1.4、2.4、2.5.1、2.5.2、3.1.4、3.5、3.6.1、3.6.2、3.6.2(4)、3.7.1、3.7.2、3.7.5、3.7.10、3.7.16、3.8.2、3.9.1、3.9.3(20)、3.11.1(1)、3.11.1(9)、4.2.1、4.2.2、4.3.1、4.3.2、4.3.6(2)、4.5.1、4.5.2、11.2.2、11.3.2、11.4.2、11.5.2(1)(a)、11.5.3(2)(e)、11.6.1、11.6.2(4)、11.7.1(1)(e)、11.7.1(2)(e)、11.7.1(3)(e)、付録 D
システム運用に必要となる機能の構成要素を変更しました。 • Microsoft Azure 環境での分散トレース収集および閲覧は Jaeger、永続化は Elasticsearch を使用するようにしました。これに伴い、トラブルシュート機能（Microsoft Azure）用ライブラリの提供を廃止しました。 • Red Hat OpenShift Container Platform 環境で分散トレースを収集する場合は、OpenTelemetry Collector の利用を必須としました。	1.2.5、2.2.17(2)、2.3.3(1)、2.7.2(7)、2.7.2(8)、3.1.2、3.4.3(1)、3.4.4(1)、3.7.1、3.7.12、3.7.20、3.9.3(1)、3.9.3(15)、3.9.3(27)、3.12.6(2)、9.2.1(3)、9.3.8(1)、11.3.2、11.3.5、11.5.2、11.6.1、11.7.1(2)(c)、11.7.1(2)(g)、付録 A.1(1)、付録 C.2
Red Hat OpenShift Container Platform 環境でメッセージログおよび分散トレース情報を収集できるようにしました。	1.2.5、3.9.3(22)、3.11.1(1)、3.11.1(9)、7.6.11、7.6.12、7.6.13、7.6.14、7.6.15、7.6.16、11.2.2、11.3.2、11.3.5、11.5.2(1)(d)、11.5.2(2)(d)、11.6.1、11.6.2(6)、11.7.1(1)(g)、11.7.1(2)(g)
Mediator が稼働した日が稼働日ファイルとして保存されるようになりました。	1.5、3.7.2、3.8.2、3.9.3(15)、11.8
トランザクションモデルとして、呼び出し元がトランザクションブランチからの受信処理を反応的（リアクティブ）に実行する「リアクティブ分散トランザクション」をサポートしました。 これに伴い、Spring Framework の WebClient を使用できるようにしました。	2.2.8(2)、2.2.8(3)(b)、2.2.8(4)、2.2.10(3)、7.3、8.2.1、8.2.7、付録 A.1(1)

追加・変更内容	変更個所
また、HMP-PCTO のトランザクション制御で使用する API に TxPropagationWebClientFactory クラスを追加しました。	
スケーリング対応機能を使用する場合も、gRPC 通信を使用できるようにしました。 これに伴い、HMP-PCTO のトランザクション制御で使用する API に次のクラスを追加しました。 • TxPropagationGrpcNameResolverProvider クラス • TxPropagationGrpcLoadBalancerProvider クラス	2.2.10、2.2.10(1)、2.2.10(2)(b)、2.2.11、2.2.11(1)、2.2.11(2)(a)、2.2.11(3)、7.5.3、8.2.1、8.2.11、8.2.12、8.2.19、8.2.20、9.2.3(1)、付録 C.1、付録 C.2(3)(a)
uCosminexus Application Runtime for Spring Boot 01-21 をサポートしました。これに伴い、次の説明を追加しました。 • コンテナイメージのロケール設定を UTF-8 に設定するための説明 • java コマンドの詳細説明	2.2.19、2.3.3(1)、2.3.4(1)、2.3.5(1)、2.3.6(1)、2.3.7(1)、2.3.8(1)、2.3.9(1)、2.3.10(1)、2.3.11(1)、2.6.2、2.6.3、2.6.4、2.6.5、2.6.6、2.6.7、2.6.8、3.4.3(1)、3.4.4(1)、6.3.1、6.3.2、6.4.1、6.4.2、6.4.3、6.4.4、6.4.5、6.4.6、6.4.7、6.4.8、6.4.9
Kubernetes の Service の Service 名と Namespace 名の長さの合計の説明を修正しました。	2.6、2.6.4、2.6.5、2.6.6、2.6.8、3.8、3.8.3
Kubernetes マニフェストの記載例の次のパラメタの指定形式を変更しました。 • server.tomcat のパラメタ accesslog.locale	2.6.2、2.6.3、2.6.4、2.6.5、2.6.6、2.6.7、2.6.8
HMP-PCTO が Kubernetes と連携して Pod 間の通信を制御するための、「アプリケーショングループ定義」の説明を追加しました。 これに伴い、次のパラメタを追加しました。 Mediator および TP1-Bridge のパラメタ • labels.hmppctoAppGroup	2.6.2、2.6.3、2.6.4、2.6.5、2.6.6、2.6.7、2.6.8、3.8.2、3.8.3、7.2、7.3、7.6.1、7.6.2、7.6.3、7.6.4、7.6.5、7.6.6、7.6.7、7.6.8、7.6.9、9.3.1(1)、9.3.2(1)、付録 B.1
RAS の Kubernetes アプリケーションのデプロイ方法を、Helm によるデプロイからユーザ任意の方法によるデプロイに変更しました。 これに伴い、RAS の Kubernetes アプリケーションの Kubernetes マニフェストの説明を追加しました。	3.7.1、3.7.6、3.7.11、3.7.12、3.7.19、3.7.20、3.7.21、3.8、3.8.1、3.8.2、3.8.5、3.9.3(1)、3.9.3(15)、3.9.3(27)、3.12.5(1)(b)、3.12.5(3)(c)、3.12.5(4)(b)、3.12.5(5)(c)、7.1、7.6.8、7.6.10、7.6.11、7.6.12、7.6.13、7.6.14、7.6.15、7.6.16、9.1、9.3.1(1)、9.3.4(1)、9.3.5(1)、9.3.6、9.3.7(1)、9.3.8(1)、9.3.9(1)、付録 A.1(1)
スレッドプールに関するパラメタの説明を変更しました。	3.12.1(3)、3.12.1(3)(a)、3.12.1(3)(b)、3.12.1(4)、3.12.1(4)(a)、3.12.1(4)(b)、3.12.1(6)(b)、9.2.5(1)
Mediator および TP1-Bridge の場合のコンフィグ設定の記載を修正しました。	7.6.8、7.6.9
tracing.grpc.debug パラメタの説明を修正しました。	9.2.1(3)

追加・変更内容	変更個所
次のメッセージを追加しました。 KFSG63112-E (Y)M、KFSG63113-E (Y)M、KFSG63114-E (Y)M、KFSG64603-E (Y)M、KFSG64604-E (Y)M	10.2.1
次のメッセージを修正しました。 KFSG60001-E (Y)M、KFSG60006-E (Y)M、KFSG60008-E (Y)M、KFSG60013-W (Y)M+J、KFSG60018-W (Y)M+J、KFSG60023-E (Y)M、KFSG60025-W (Y)M、KFSG60028-W (Y)M、KFSG60033-W (Y)M、KFSG60036-E (Y)M、KFSG61001-E (Y)M、KFSG61002-E (Y)M、KFSG61004-W (Y)M、KFSG61005-W (Y)M、KFSG61007-W (Y)M、KFSG61008-E (Y)M、KFSG61101-E (Y)M、KFSG61102-W (Y)M、KFSG61104-E (Y)M、KFSG61106-E (Y)M、KFSG61110-E (Y)M、KFSG61111-W (Y)M、KFSG61115-W (Y)M、KFSG62015-W (Y)M、KFSG62201-E (Y)J+M、KFSG62300-E (Y)J+M～KFSG62302-E(Y)J+M、KFSG63003-E (Y)J+M、KFSG63100-E (Y)M～KFSG63105-W (Y)J+M、KFSG63107-W (Y)M～KFSG63111-E (Y)J+M、KFSG63200-E (Y)M、KFSG63207-E (Y)M、KFSG63209-E (Y)M、KFSG63300-E (Y)C～KFSG63307-E (Y)C、KFSG63309-E (Y)C、KFSG63311-W (Y)C、KFSG63312-E (Y)C、KFSG64001-E (Y)M、KFSG64002-E (Y)M、KFSG64602-E (Y)M、KFSG74001-E (Y)M、KFSG74002-E (Y)M、KFSG74100-I (Y)M、KFSG74101-I (Y)M、KFSG74105-I (Y)M、KFSG74200-I (Y)M、KFSG74201-I (Y)M、KFSG74203-I (Y)M、KFSG74204-I (Y)M、KFSG74206-I (Y)M、KFSG74401-I (Y)M～KFSG74406-I (Y)M、KFSG74408-W (Y)M、KFSG80000-I (Y)M～KFSG80008-I (Y)M、KFSG81001-I (Y)M、KFSG81002-I (Y)M、KFSG81101-I (Y)M、KFSG81102-I (Y)M、KFSG81105-I (Y)M、KFSG81107-I (Y)M～KFSG81109-I (Y)M、KFSG82006-I (Y)M、KFSG82200-I (Y)M	10.2.1、10.2.2、10.2.3
提供ファイル一覧に技術補助資材配置ディレクトリを追加しました。	付録 A.1(1)
HMP-PCTO に関連する OSS および他社製品のパラメタ一覧の説明を、HMP-PCTO 固有の指定が必要なパラメタと、不要なパラメタに分けて記載しました。	付録 B.1、付録 B.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
- HMP-PCTO のオプション機能の設定および運用方法
- 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。 (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　20
1.1　HMP-PCTO の Kubernetes アプリケーションの一覧と分類　21
1.2　構成要素　23
1.2.1　HMP-PCTO の Kubernetes クラスタの構成要素　23
1.2.2　SQL-Participant を使用する場合の構成要素（SQL-Participant 限定）　23
1.2.3　TCC-Participant を使用する場合の構成要素（TCC-Participant 限定）　24
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）　25
1.2.5　システム運用に必要となる機能の構成要素　25
1.2.6　スケーリング対応機能の構成要素　28
1.3　最小機器構成　29
1.3.1　SQL-Participant を使用する場合の最小機器構成（SQL-Participant 限定）　29
1.3.2　TCC-Participant を使用する場合の最小機器構成（TCC-Participant 限定）　33
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）　37
1.4　前提条件　40
1.4.1　前提とする Kubernetes クラスタ構成　40
1.4.2　前提とする共有ファイルシステム　45
1.4.3　前提とする Red Hat OpenShift Container Platform の設定　45
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　47

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　49
2.1　ユーザ責務の Kubernetes アプリケーションの開発の流れ　50
2.1.1　オンプレミス環境の場合　50
2.1.2　Microsoft Azure 環境の場合　51
2.1.3　Google Cloud Platform 環境の場合　51
2.1.4　Amazon Web Services 環境の場合　52
2.1.5　Red Hat OpenShift Container Platform 環境の場合　53
2.2　Java アプリケーションの開発　55

2.2.1 作業項目の一覧 55
2.2.2 事前準備 58
2.2.3 ライブラリのセットアップ（通常版限定） 58
2.2.4 コンポーネントスキャンの追加 58
2.2.5 DI に使用するアノテーションの注意事項 58
2.2.6 アノテーションによるトランザクション境界の設定 59
2.2.7 アノテーションによるトランザクションブランチの作成（SQL-Participant 限定） 61
2.2.8 通信によるトランザクションブランチの作成（REST 通信-クライアント） 62
2.2.9 通信によるトランザクションブランチの作成（REST 通信-サーバ） 69
2.2.10 通信によるトランザクションブランチの作成（gRPC 通信-クライアント） 72
2.2.11 通信によるトランザクションブランチの作成（gRPC 通信-サーバ） 77
2.2.12 JDBC Proxy Driver の使用（SQL-Participant 限定） 80
2.2.13 TccService インタフェースの実装クラスの作成（TCC-Participant 限定） 81
2.2.14 業務ロジックからの executeTry メソッドの呼び出し（TCC-Participant 限定） 88
2.2.15 TP1-Bridge の使用（通常版かつ TP1-Bridge 限定） 88
2.2.16 ロギングライブラリの設定ファイルのカスタマイズ 90
2.2.17 分散トレース機能の設定 97
2.2.18 Quartz スケジューラ機能の利用 98
2.2.19 アプリケーションが出力するメッセージの文字コード 98
2.2.20 Spring Boot の組み込み HTTP サーバの設定 98
2.3 コンテナイメージ作成 101
2.3.1 作業項目の一覧 101
2.3.2 事前準備 102
2.3.3 uCosminexus Application Runtime for Spring Boot コンテナベースイメージの作成（通常版限定） 102
2.3.4 JDK コンテナベースイメージの作成 107
2.3.5 Orchestrator-Service コンテナイメージの作成 110
2.3.6 Relay-Service コンテナイメージの作成 114
2.3.7 Entity-Service(SQL)コンテナイメージの作成（SQL-Participant 限定） 117
2.3.8 Entity-Service(TCC)コンテナイメージの作成（TCC-Participant 限定） 124
2.3.9 Orchestrator-Service(SQL)コンテナイメージの作成（SQL-Participant 限定） 131
2.3.10 Alternate-Service コンテナイメージの作成 138
2.3.11 Alternate-Service(SQL)コンテナイメージの作成（SQL-Participant 限定） 142
2.4 永続ボリュームとプロビジョニング方法の選定（通常版限定） 150
2.5 共有ファイルシステムでの作業（通常版限定） 152
2.5.1 作業項目の一覧（通常版限定） 152
2.5.2 マネージドファイルサービスでのプロビジョニング（通常版限定） 152
2.5.3 uCosminexus Application Runtime 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） 153
2.6 Kubernetes マニフェスト作成 154

2.6.1 作業項目の一覧 154
2.6.2 Orchestrator-Service の Kubernetes マニフェストの作成 155
2.6.3 Relay-Service の Kubernetes マニフェストの作成 164
2.6.4 Entity-Service(SQL)の Kubernetes マニフェストの作成（SQL-Participant 限定） 172
2.6.5 Entity-Service(TCC)の Kubernetes マニフェストの作成（TCC-Participant 限定） 183
2.6.6 Orchestrator-Service(SQL)の Kubernetes マニフェストの作成（SQL-Participant 限定） 192
2.6.7 Alternate-Service の Kubernetes マニフェストの作成 203
2.6.8 Alternate-Service(SQL)の Kubernetes マニフェストの作成（SQL-Participant 限定） 212
2.6.9 Kubernetes リソース取得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 222
2.6.10 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 Kubernetes マニフェストの作成 225
2.6.11 uCosminexus Application Runtime 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） 225
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 226
2.7.1 作業項目の一覧 226
2.7.2 Kubernetes コントロールプレーンでの作業 227

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

3 HMP-PCTO のコントロールプレーンの Kubernetes アプリケーションおよび RAS のデプロイ 238
3.1 HMP-PCTO のコントロールプレーンの Kubernetes アプリケーションおよび RAS のデプロイの流れ 239
3.1.1 オンプレミス環境の場合 239
3.1.2 Microsoft Azure 環境の場合 240
3.1.3 Google Cloud Platform 環境の場合 241
3.1.4 Amazon Web Services 環境の場合 243
3.1.5 Red Hat OpenShift Container Platform 環境の場合 244
3.2 Mediator のビルド（通常版限定） 245
3.2.1 作業項目の一覧（通常版限定） 245
3.2.2 事前準備（通常版限定） 245
3.2.3 ライブラリのセットアップ（通常版限定） 246
3.2.4 JAR ファイルのビルド（通常版限定） 246
3.3 TP1-Bridge のビルド（通常版かつ TP1-Bridge 限定） 248
3.3.1 作業項目の一覧（通常版かつ TP1-Bridge 限定） 248
3.3.2 事前準備（通常版かつ TP1-Bridge 限定） 248
3.3.3 ライブラリのセットアップ（通常版かつ TP1-Bridge 限定） 249
3.3.4 JAR ファイルのビルド（通常版かつ TP1-Bridge 限定） 249
3.4 コンテナイメージ作成（通常版限定） 250

3.4.1 作業項目の一覧（通常版限定） 250
3.4.2 事前準備（通常版限定） 250
3.4.3 Mediator コンテナイメージの作成（通常版限定） 251
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） 255
3.4.5 Dependency-Checker コンテナイメージの作成（通常版限定） 261
3.4.6 EADS サーバ用コンテナイメージの作成（通常版限定） 262
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） 262
3.5 永続ボリュームとプロビジョニング方法の選定 263
3.6 共有ファイルシステムでの作業 265
3.6.1 作業項目の一覧 265
3.6.2 マネージドファイルサービスでのプロビジョニング 265
3.6.3 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） 268
3.6.4 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 268
3.6.5 uCosminexus Application Runtime 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） 269
3.6.6 EADS 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） 269
3.7 Kubernetes マニフェスト作成 270
3.7.1 作業項目の一覧 270
3.7.2 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 272
3.7.3 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 279
3.7.4 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 284
3.7.5 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） 287
3.7.6 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 298
3.7.7 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 304
3.7.8 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 305
3.7.9 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 307
3.7.10 uCosminexus Application Runtime 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） 309
3.7.11 Filebeat の Kubernetes マニフェスト作成 312
3.7.12 Jaeger の Kubernetes マニフェスト作成 315
3.7.13 Prometheus の Kubernetes マニフェスト作成 318
3.7.14 Microsoft Azure のマネージド Prometheus によるメトリクス収集の Kubernetes マニフェストの作成 322
3.7.15 Google Cloud Platform のマネージド Prometheus によるメトリクス収集の Kubernetes マニフェスト作成 324

3.7.16 Amazon Web Services のマネージド Prometheus によるメトリクス収集の Kubernetes マニフェスト作成 326
3.7.17 New Relic によるメトリクス収集の Kubernetes マニフェスト作成 327
3.7.18 Red Hat OpenShift Container Platform モニタリングの Prometheus によるメトリクス収集の Kubernetes マニフェスト作成 328
3.7.19 Metricbeat の Kubernetes マニフェスト作成 331
3.7.20 Elasticsearch の Kubernetes マニフェスト作成 332
3.7.21 Logstash の Kubernetes マニフェスト作成 338
3.8 Helm チャートのカスタマイズ 340
3.8.1 作業項目の一覧 340
3.8.2 Mediator の Helm チャートのカスタマイズ 340
3.8.3 TP1-Bridge の Helm チャートのカスタマイズ（通常版かつ TP1-Bridge 限定） 344
3.8.4 EADS の Helm チャートのカスタマイズ（通常版限定） 348
3.8.5 Ext-Cons の Helm チャートのカスタマイズ（トライアル版限定） 348
3.9 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 350
3.9.1 作業項目の一覧 350
3.9.2 Kubernetes 各ワーカーノードでの作業 356
3.9.3 Kubernetes コントロールプレーンでの作業 359
3.10 RAS のマネージドサービスの運用 398
3.10.1 New Relic を使用する場合の作業 398
3.11 ログの取得および監視の設定 399
3.11.1 ログおよびトラブルシュート情報の取得の設定 399
3.11.2 監視の設定 403
3.12 パラメタの設計とチューニング 407
3.12.1 スレッドプール 407
3.12.2 タイマ 443
3.12.3 G1GC 450
3.12.4 ディスク 452
3.12.5 システム運用に必要となる機能 452
3.12.6 分散トレース 460

第 4 編 HMP-PCTO のオプション機能の設定および運用

4 スケーリング対応機能の運用 466
4.1 スケーリング対応機能の概要 467
4.2 Metrics Server でメトリクスを監視する場合のオートスケールのセットアップ 470
4.2.1 Metrics Server でメトリクスを監視する場合のオートスケールのセットアップの流れ 470
4.2.2 Metrics Server のインストール（オンプレミス環境または Amazon Web Services 環境限定） 471
4.2.3 Metrics Server のデプロイ（オンプレミス環境限定） 471
4.2.4 HPA の Kubernetes マニフェストの作成 471

4.2.5 HPA の設定のカスタマイズ 473
4.2.6 HPA のデプロイ 474
4.2.7 Kubernetes マニフェストの作成および Helm チャートのカスタマイズ 475
4.2.8 Kubernetes アプリケーションのデプロイ 476
4.3 Prometheus でメトリクスを監視する場合のオートスケールのセットアップ 477
4.3.1 Prometheus でメトリクスを監視する場合のオートスケールのセットアップの流れ 477
4.3.2 KEDA のインストールとデプロイ（オンプレミス環境または Amazon Web Services 環境限定） 479
4.3.3 KEDA アドオンのインストール（Microsoft Azure 環境限定） 479
4.3.4 カスタム指標 Stackdriver アダプタのインストール（Google Cloud Platform 環境限定） 479
4.3.5 Custom Metrics Autoscaler のインストール（Red Hat OpenShift Container Platform 環境限定） 480
4.3.6 HPA の Kubernetes マニフェストの作成 480
4.3.7 HPA の設定のカスタマイズ 483
4.3.8 HPA のデプロイ 483
4.3.9 Kubernetes マニフェストの作成および Helm チャートのカスタマイズ 484
4.3.10 Kubernetes アプリケーションのデプロイ 485
4.4 オートスケールのパラメタの設計とチューニング 486
4.4.1 設計指針 486
4.4.2 評価対象メトリクスの選定 487
4.4.3 チューニング方法 489
4.4.4 オートスケールによる Pod 数の増減を確認する方法 491
4.4.5 チューニング例 492
4.5 オートスケールの障害対策 494
4.5.1 Metrics Server でメトリクスを監視する場合の障害対策 494
4.5.2 Prometheus でメトリクスを監視する場合の障害対策 497

第 5 編 Kubernetes クラスタ外の HMP-PCTO の運用

5 TP1-Converter の運用（通常版かつ TP1-Bridge 限定） 502
5.1 TP1-Converter の運用の流れ（通常版かつ TP1-Bridge 限定） 503
5.2 データ変換定義ファイルの作成（通常版かつ TP1-Bridge 限定） 504
5.2.1 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） 504
5.2.2 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） 505
5.2.3 XML スキーマ（通常版かつ TP1-Bridge 限定） 506
5.2.4 マッピング定義ファイル（通常版かつ TP1-Bridge 限定） 506
5.2.5 データ変換定義ファイル（通常版かつ TP1-Bridge 限定） 506
5.3 J2EE サーバのセットアップ（通常版かつ TP1-Bridge 限定） 507
5.4 TP1-Converter の資材の配置（通常版かつ TP1-Bridge 限定） 508
5.5 J2EE サーバの開始（通常版かつ TP1-Bridge 限定） 509

5.6 TP1-Converter のインポート（通常版かつ TP1-Bridge 限定） 510
5.7 TP1-Converter の開始（通常版かつ TP1-Bridge 限定） 511
5.8 データ変換定義の再作成（通常版かつ TP1-Bridge 限定） 512
5.9 キャッシュのリフレッシュ（通常版かつ TP1-Bridge 限定） 513
5.10 TP1-Converter の終了（通常版かつ TP1-Bridge 限定） 515
5.11 J2EE サーバの終了（通常版かつ TP1-Bridge 限定） 516
5.12 キャッシュの参照（通常版かつ TP1-Bridge 限定） 517
5.13 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） 519
5.13.1 TP1-Converter の削除（通常版かつ TP1-Bridge 限定） 519
5.13.2 J2EE サーバのアンセットアップ（通常版かつ TP1-Bridge 限定） 519

第 6 編 リファレンス

6 コンテナイメージの作成要件 520
6.1 コンテナイメージの一覧 521
6.2 コンテナイメージの提供方式 524
6.3 各コンテナベースイメージの作成要件 525
6.3.1 uCosminexus Application Runtime for Spring Boot コンテナベースイメージ（通常版限定） 525
6.3.2 JDK コンテナベースイメージ 526
6.4 各コンテナイメージの作成要件 528
6.4.1 Orchestrator-Service コンテナイメージ 528
6.4.2 Relay-Service コンテナイメージ 529
6.4.3 Entity-Service(SQL)コンテナイメージ（SQL-Participant 限定） 531
6.4.4 Entity-Service(TCC)コンテナイメージ（TCC-Participant 限定） 532
6.4.5 Orchestrator-Service(SQL)コンテナイメージ（SQL-Participant 限定） 534
6.4.6 Alternate-Service コンテナイメージ 536
6.4.7 Alternate-Service(SQL)コンテナイメージ（SQL-Participant 限定） 538
6.4.8 Mediator コンテナイメージ 540
6.4.9 TP1-Bridge コンテナイメージ（通常版かつ TP1-Bridge 限定） 542
6.4.10 Dependency-Checker コンテナイメージ 543
6.4.11 Ext-Cons コンテナイメージ（トライアル版限定） 544

7 Kubernetes クラスタの作成要件 546
7.1 Kubernetes アプリケーションのデプロイ方法 547
7.2 Kubernetes アプリケーションのグループ定義 548
7.3 Kubernetes クラスタ内外の通信暗号化 552
7.4 Kubernetes アプリケーションのタイムゾーン設定方針 555
7.5 Kubernetes アプリケーションのグレースフルシャットダウン 556
7.5.1 HMP-PCTO の Kubernetes アプリケーションのグレースフルシャットダウンのサポート有無 556
7.5.2 HMP-PCTO の Kubernetes アプリケーションのグレースフルシャットダウンが開始できる条件 557

7.5.3 グレースフルシャットダウンの実現方法 559
7.5.4 HMP-PCTO の Kubernetes アプリケーションが強制シャットダウンしたときの挙動 561
7.6 各 Kubernetes アプリケーションの作成要件 562
7.6.1 Orchestrator-Service の要件 562
7.6.2 Relay-Service の要件 569
7.6.3 Entity-Service(SQL)の要件（SQL-Participant 限定） 577
7.6.4 Entity-Service(TCC)の要件（TCC-Participant 限定） 584
7.6.5 Orchestrator-Service(SQL)の要件（SQL-Participant 限定） 592
7.6.6 Alternate-Service の要件 599
7.6.7 Alternate-Service(SQL)の要件（SQL-Participant 限定） 607
7.6.8 Mediator の要件 615
7.6.9 TP1-Bridge の要件（通常版かつ TP1-Bridge 限定） 623
7.6.10 Ext-Cons の要件（トライアル版限定） 631
7.6.11 Filebeat の要件 635
7.6.12 Jaeger の要件 636
7.6.13 Prometheus の要件 638
7.6.14 Metricbeat の要件 640
7.6.15 Elasticsearch の要件 642
7.6.16 Logstash の要件 643
7.6.17 Metrics Server の要件 645
7.7 Kubernetes アプリケーションで使用するスクリプト 646
7.7.1 デプロイ依存関係チェックスクリプト（dependency_check.sh） 646
7.7.2 トランザクションヘルスチェックスクリプト（health_check.sh） 649
7.7.3 未決着トランザクション待機スクリプト（wait_transactions.sh） 652

8 API 656
8.1 API の記述形式 657
8.2 トランザクション制御の API 仕様 659
8.2.1 クラス・インタフェースの一覧 659
8.2.2 アノテーションの一覧 660
8.2.3 例外の一覧 660
8.2.4 XidHeaders インタフェース（SQL-Participant 限定） 661
8.2.5 TxPropagationRestTemplateInterceptor クラス 662
8.2.6 TxPropagationGrpcClientInterceptor クラス 664
8.2.7 TxPropagationWebClientFactory クラス 665
8.2.8 TxPropagationOptionalSettings クラス 666
8.2.9 TxPropagationGrpcServerInterceptor クラス 667
8.2.10 GrpcTraceInterceptorFactory クラス 668
8.2.11 TxPropagationGrpcNameResolverProvider クラス 669

8.2.12 TxPropagationGrpcLoadBalancerProvider クラス 670
8.2.13 @PctoBranchRollbackFor アノテーション 670
8.2.14 @PctoGrpcBranchRollbackFor アノテーション 671
8.2.15 @PctoTransactional アノテーション 672
8.2.16 @PctoTransactionBranch アノテーション 673
8.2.17 RollbackTerminationException 例外 674
8.2.18 TerminationFailureException 例外 674
8.2.19 PodNotFoundException 例外 675
8.2.20 ServiceNotFoundException 例外 675
8.2.21 TransactionBranchException 例外 676
8.2.22 TransactionSystemException 例外 676
8.3 JDBC Proxy Driver の API 仕様（SQL-Participant 限定） 678
8.3.1 クラス・インタフェースの一覧（SQL-Participant 限定） 678
8.3.2 例外の一覧（SQL-Participant 限定） 679
8.3.3 PctoDataSource クラス（SQL-Participant 限定） 680
8.3.4 PctoConnection クラス（SQL-Participant 限定） 682
8.3.5 PctoStatement クラス（SQL-Participant 限定） 697
8.3.6 PctoPreparedStatement クラス（SQL-Participant 限定） 713
8.3.7 PctoCallableStatement クラス（SQL-Participant 限定） 738
8.3.8 PctoResultSet クラス（SQL-Participant 限定） 760
8.3.9 PctoDatabaseMetaData クラス（SQL-Participant 限定） 800
8.3.10 PctoResultSetMetaData クラス（SQL-Participant 限定） 802
8.3.11 PctoSQLXML クラス（SQL-Participant 限定） 806
8.3.12 PctoBlob クラス（SQL-Participant 限定） 809
8.3.13 PctoClob クラス（SQL-Participant 限定） 812
8.3.14 PctoArray クラス（SQL-Participant 限定） 814
8.3.15 PctoSqlFeatureNotSupportedException 例外（SQL-Participant 限定） 816
8.3.16 PctoSQLClientInfoException 例外（SQL-Participant 限定） 817
8.3.17 PctoInternalContradictionException 例外（SQL-Participant 限定） 817
8.3.18 PctoTransactionStartException 例外（SQL-Participant 限定） 818
8.3.19 PctoPrePreparedException 例外（SQL-Participant 限定） 818
8.3.20 PctoConnectionLackedException 例外（SQL-Participant 限定） 819
8.3.21 SQLState・ErrorCode 一覧（SQL-Participant 限定） 819
8.4 TCC-Participant の API 仕様（TCC-Participant 限定） 821
8.4.1 クラス・インタフェースの一覧（TCC-Participant 限定） 821
8.4.2 アノテーションの一覧（TCC-Participant 限定） 821
8.4.3 例外の一覧（TCC-Participant 限定） 821
8.4.4 TccResponse<T extends java.io.Serializable>クラス（TCC-Participant 限定） 822
8.4.5 TccService<I, O extends java.io.Serializable>インタフェース（TCC-Participant 限定） 824

8.4.6 @Tcc アノテーション（TCC-Participant 限定） 827

9 パラメタ 829
9.1 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 830
9.2 ユーザのコンポーネントに設定するパラメタ 832
9.2.1 共通 832
9.2.2 JDBC Proxy Driver 839
9.2.3 Orchestrator 841
9.2.4 Entity-Module 846
9.2.5 SQL-Participant 847
9.2.6 TCC-Participant 856
9.2.7 Spring Boot の組み込み HTTP サーバ 861
9.3 HMP-PCTO と RAS のコンポーネントに設定するパラメタ 866
9.3.1 Mediator 866
9.3.2 TP1-Bridge（通常版かつ TP1-Bridge 限定） 878
9.3.3 EADS（通常版限定） 893
9.3.4 Ext-Cons（トライアル版限定） 901
9.3.5 Filebeat 903
9.3.6 Jaeger 903
9.3.7 Prometheus 904
9.3.8 Metricbeat 906
9.3.9 Logstash 907
9.3.10 Metrics Server 907
9.4 各 RDBMS の使用に必要なパラメタおよび設定（SQL-Participant 限定） 909
9.4.1 PostgreSQL（SQL-Participant 限定） 909
9.4.2 SQL Server（SQL-Participant 限定） 910
9.4.3 HiRDB（SQL-Participant 限定） 910
9.4.4 Oracle Database（SQL-Participant 限定） 912
9.5 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） 913
9.5.1 rap リスナーサービス定義（通常版かつ TP1-Bridge 限定） 913
9.5.2 システム共通定義（通常版かつ TP1-Bridge 限定） 913
9.5.3 トランザクションサービス定義（通常版かつ TP1-Bridge 限定） 914
9.5.4 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） 914
9.6 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） 915
9.6.1 usrconf.properties（通常版かつ TP1-Bridge 限定） 915
9.6.2 usrconf.cfg（通常版かつ TP1-Bridge 限定） 917

10 メッセージ 918
10.1 メッセージの概要 919
10.1.1 メッセージの記述形式 919
10.1.2 メッセージ系統 920
10.1.3 メッセージの説明 921
10.2 メッセージ一覧 924
10.2.1 KFSG60000～KFSG69999 924
10.2.2 KFSG70000～KFSG79999 991
10.2.3 KFSG80000～KFSG89999 1000

第７編 システム運用に必要となる機能

11 システム運用に必要となる機能 1013
11.1 概要 1014
11.1.1 取得する情報および各情報の紐づけ方法 1014
11.2 メッセージログ 1027
11.2.1 概要 1027
11.2.2 サポートするアプリケーションおよびサービス 1028
11.2.3 取得方針 1029
11.2.4 パラメタ 1029
11.3 分散トレース 1030
11.3.1 概要 1030
11.3.2 サポートするアプリケーションおよびサービス 1030
11.3.3 分散トレース機能の使用 1032
11.3.4 トレースの単位 1033
11.3.5 Span へ付加する情報 1034
11.3.6 パラメタ 1040
11.4 メトリクス 1041
11.4.1 概要 1041
11.4.2 サポートするアプリケーションおよびサービス 1043
11.4.3 パラメタ 1047
11.5 データおよびファイル 1048
11.5.1 メッセージログ 1048
11.5.2 分散トレース 1053
11.5.3 メトリクス 1057
11.5.4 定義するデータ形式 1062
11.6 HMP-PCTO の障害調査 1064
11.6.1 取得する情報 1064
11.6.2 情報の収集方法 1068

11.6.3 障害原因の調査方法 1072
11.7 システム運用に必要となる機能の障害対策 1080
11.7.1 障害の原因および対策 1080
11.8 稼働日ファイルに関する注意事項 1089

付録 1090
付録 A 提供ファイル 1091
付録 A.1 提供ファイル一覧 1091
付録 B HMP-PCTO に関連する OSS および他社製品のパラメタ一覧 1099
付録 B.1 HMP-PCTO 固有の指定が必要なパラメタ 1099
付録 B.2 HMP-PCTO 固有の指定が不要なパラメタ 1100
付録 B.3 Azure Kubernetes Service（AKS）関連パラメタ 1102
付録 B.4 Helm 関連パラメタ 1103
付録 B.5 KEDA 関連パラメタ 1103
付録 B.6 Prometheus 関連パラメタ 1103
付録 B.7 Spring Boot 関連パラメタ 1104
付録 C HMP-PCTO の分散トレース取得項目 1105
付録 C.1 Span の一覧 1105
付録 C.2 各スパンの詳細 1194
付録 D HMP-PCTO のメトリクス取得項目 1223
付録 E システム運用時の注意事項 1227
付録 E.1 システム時刻の補正時の注意事項 1227
付録 F このマニュアルの参考情報 1228
付録 F.1 関連ドキュメント 1228
付録 F.2 マニュアルでの表記 1230
付録 F.3 英略語 1231
付録 F.4 KB（キロバイト）などの単位表記について 1232
付録 G HMP-PCTO で使用する用語 1233

索引 1237

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
ユーザのアプリケーションと HMP-PCTO のライブラリが混在するアプリケーション※1	○	−	−
Mediator	−	○	−
TP1-Bridge（TP1 連携機能を使用する場合）	−	○	−
EADS※2（通常版の場合）	−	○	−
Ext-Cons（トライアル版の場合）	−	○	−
Filebeat	−	−	○
Jaeger	−	−	○
Prometheus	−	−	○
Metricbeat	−	−	○
Elasticsearch	−	−	○
Logstash	−	−	○
Metrics Server	○	−	−

（凡例）

○：該当します。

−：該当しません。

注※1

該当するアプリケーションを下記に示します。

- Orchestrator-Service
- Relay-Service
- Entity-Service(SQL)
- Entity-Service(TCC)
- Orchestrator-Service(SQL)
- Alternate-Service
- Alternate-Service(SQL)

注※2

EADS の詳細については、取扱説明書「EADS コンテナオーケストレーションツール対応機能」を参照してください。

1.2　構成要素

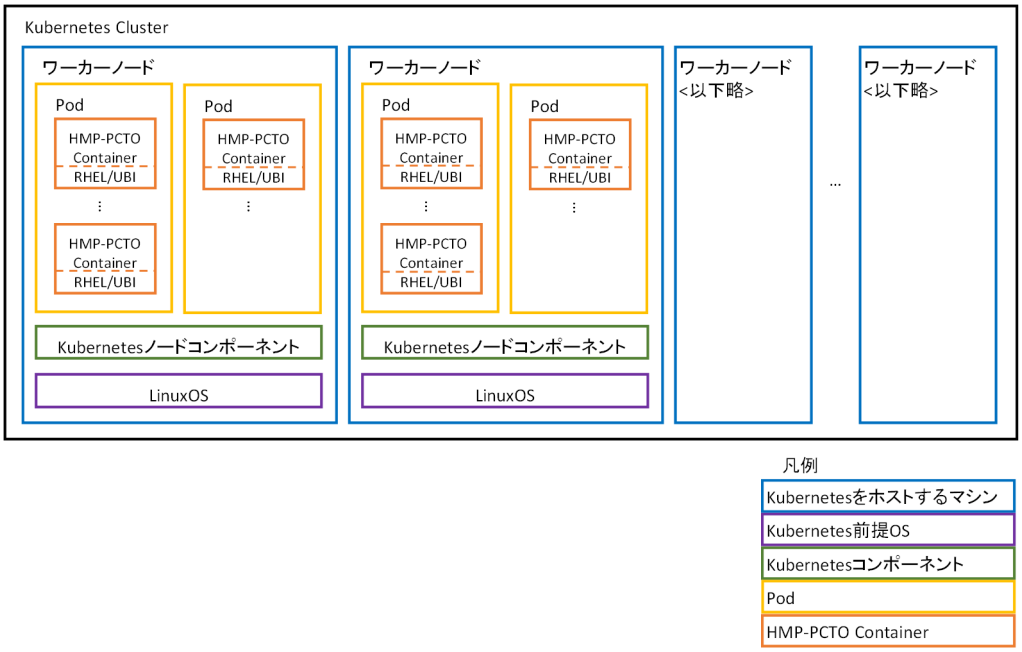
HMP-PCTO のクラスタの構成要素、および使用する Participant や機能ごとの構成要素（Pod（Kubernetes アプリケーション）、Container、サービスなど）について説明します。

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（Kubernetes アプリケーション）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（Kubernetes アプリケーション）および Container	persistent volume	備考
HMP-PCTO	Orchestrator-Service	−	なし
	Entity-Service(SQL)	−	なし
	Mediator	○	なし
	通常版の場合 　EADS トライアル版の場合 　Mock サーバ	○	なし
	EADS 生存監視	−	CronJob で定期実行するため pod は常設しない

（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（Kubernetes アプリケーション）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（Kubernetes アプリケーション）および Container	persistent volume	備考
HMP-PCTO	Orchestrator-Service	−	なし
	Entity-Service(TCC)	−	なし
	Mediator	○	なし
	通常版の場合 　EADS トライアル版の場合 　Mock サーバ	○	なし
	EADS 生存監視	−	CronJob で定期実行するため pod は常設しない

（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（Kubernetes アプリケーション）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（Kubernetes アプリケーション）および Container	persistent volume	備考
HMP-PCTO	Orchestrator-Service	−	なし
	TP1-Bridge	−	なし
	Mediator	○	なし
	EADS	○	なし
	EADS 生存監視	−	CronJob で定期実行するため pod は常設しない

（凡例）○：必要です。　−：規定しません。

1.2.5　システム運用に必要となる機能の構成要素

システム運用に必要となる機能の構成要素（Pod（Kubernetes アプリケーション）、および Container ならびにサービス）を次の表に示します。

表 1-5　システム運用に必要となる機能の構成要素

役割	Pod（Kubernetes アプリケーション）、および Container ならびにサービス					
	Elastic Stack、Jaeger および Prometheus 使用時	Microsoft Azure 環境	Google Cloud Platform 環境	Amazon Web Services 環境	New Relic 使用時	Red Hat OpenShift Container Platform 環境
ログ収集（コンテナ標準出力）	Filebeat	Azure Monitor ※1	Cloud Logging※2	Fluent Bit	New Relic※3	Red Hat OpenShift Logging Operator※4
ログ収集（Pod 内ファイル）	Filebeat	−	−	−	−	−
ログ永続化	• Logstash • Elasticsearch	Azure Monitor Log Analytics※1	Cloud Logging※2	Amazon CloudWatch ※5	New Relic※3	Loki Operator ※4

役割	Pod（Kubernetes アプリケーション）、および Container ならびにサービス					
	Elastic Stack、Jaeger および Prometheus 使用時	Microsoft Azure 環境	Google Cloud Platform 環境	Amazon Web Services 環境	New Relic 使用時	Red Hat OpenShift Container Platform 環境
ログ閲覧	• Elasticsearch • Kibana	Azure Monitor※1	Cloud Logging※2	Amazon CloudWatch※5	New Relic UI（Web UI）※6	Red Hat OpenShift Container Platform（Web UI）※4
分散トレース収集	• Jaeger • OpenTelemetry Collector※7	• Jaeger • OpenTelemetry Collector※7	• Cloud Trace※2 • OpenTelemetry Collector※7	• Jaeger • OpenTelemetry Collector※7	• OpenTelemetry Collector※7 • New Relic※3	• Tempo Operator • OpenTelemetry Collector※7、※8
分散トレース永続化	Elasticsearch	Elasticsearch	Cloud Trace※2	Amazon OpenSearch Service※5	New Relic※3	TempoStack
分散トレース閲覧	Jaeger	Jaeger	Cloud Trace※2	Jaeger	New Relic UI（Web UI）※6	Red Hat OpenShift Container Platform（Web UI）※4
メトリクス収集	• Prometheus • Metricbeat	Prometheus 用のマネージドサービス※1	Google Cloud Managed Service for Prometheus※2	• AWS Distro for OpenTelemetry Collector • Amazon Managed Service for Prometheus※5	New Relic※3	Red Hat OpenShift Container Platform モニタリングの Prometheus※4
メトリクス永続化	Elasticsearch	Azure Monitor※1	Cloud Monitoring※2	Amazon CloudWatch※5	New Relic※3	Red Hat OpenShift Container Platform モニタリングの Prometheus※4

役割	Pod（Kubernetes アプリケーション）、および Container ならびにサービス					
	Elastic Stack、Jaeger および Prometheus 使用時	Microsoft Azure 環境	Google Cloud Platform 環境	Amazon Web Services 環境	New Relic 使用時	Red Hat OpenShift Container Platform 環境
メトリクス閲覧	Prometheus	Grafana 用のマネージドサービス※1	Cloud Monitoring※2	Amazon CloudWatch※5	New Relic UI（Web UI）※6	Red Hat OpenShift Container Platform（Web UI）※4

（凡例）－：規定しません。

注※1

Microsoft Azure のマネージドサービスとして提供されるため、Pod および Container としての配置は不要です。

注※2

Google Cloud Platform のマネージドサービスとして提供されるため、Pod および Container としての配置は不要です。

注※3

New Relic が提供しているコンポーネントの Pod および Container を配置します。

注※4

Red Hat OpenShift Container Platform モニタリングが提供しているコンポーネントの Pod および Container としての配置は不要です。

注※5

Amazon Web Services のマネージドサービスとして提供されるため、Pod およびコンテナとしての配置は不要です。

注※6

New Relic のマネージドサービスとして提供されるため、Pod および Container としての配置は不要です。

注※7

確率的サンプリングおよびテイルベースサンプリングなどを実施する場合は、分散トレース収集アプリケーションやサービスにデータを送信するコンポーネントで実施する必要があります。そのため、分散トレース収集アプリケーションやサービスにデータを送信するコンポーネントとして、OpenTelemetry Collector を経由して分散トレースを収集するように配置してください。

注※8

Red Hat OpenShift Container Platform 環境で分散トレースを収集するときは、必ず OpenTelemetry Collector を使用してください。

1.2.6　スケーリング対応機能の構成要素

収集したメトリクスを使用してオートスケールをする場合の構成要素（Pod（Kubernetes アプリケーション）、および Container ならびにサービス）を次の表に示します。

表 1-6　スケーリング対応機能の構成要素

役割	Pod（Kubernetes アプリケーション）、および Container ならびにサービス					
	Prometheus 使用時	Microsoft Azure 環境	Google Cloud Platform 環境	Amazon Web Services 環境	New Relic 使用時	Red Hat OpenShift Container Platform 環境
Kubernetes のメトリクス収集	Metrics Server	Metrics Server	カスタム指標 Stackdriver アダプタ（custom-metrics-stackdriver-adapter）	Metrics Server	－※	Metrics Server
HMP-PCTO のメトリクス収集	Prometheus	Prometheus 用のマネージドサービス	Google Cloud Managed Service for Prometheus	Amazon Managed Service for Prometheus	－※	Red Hat OpenShift Container Platform モニタリングの Prometheus

（凡例）－：規定しません。

注※

該当するメトリクスを使用して、HMP-PCTO のスケーリング対応機能は使用できません。

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。

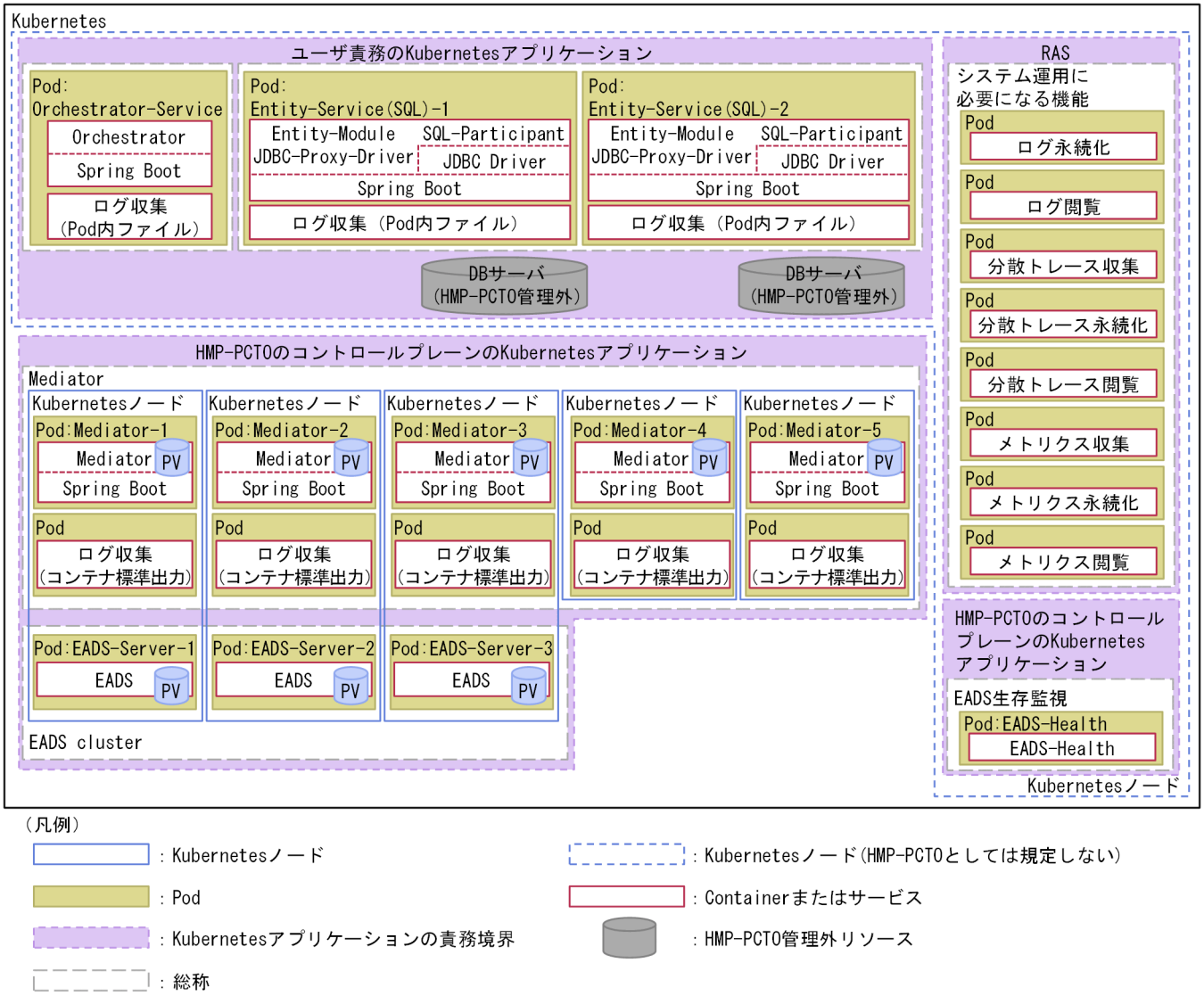
補助合意用の EADS クラスタ内の EADS サーバの Pod は可用性を確保するため、別々の Kubernetes ワーカーノードに配置し、物理的にも別のマシンとする必要があります。

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。

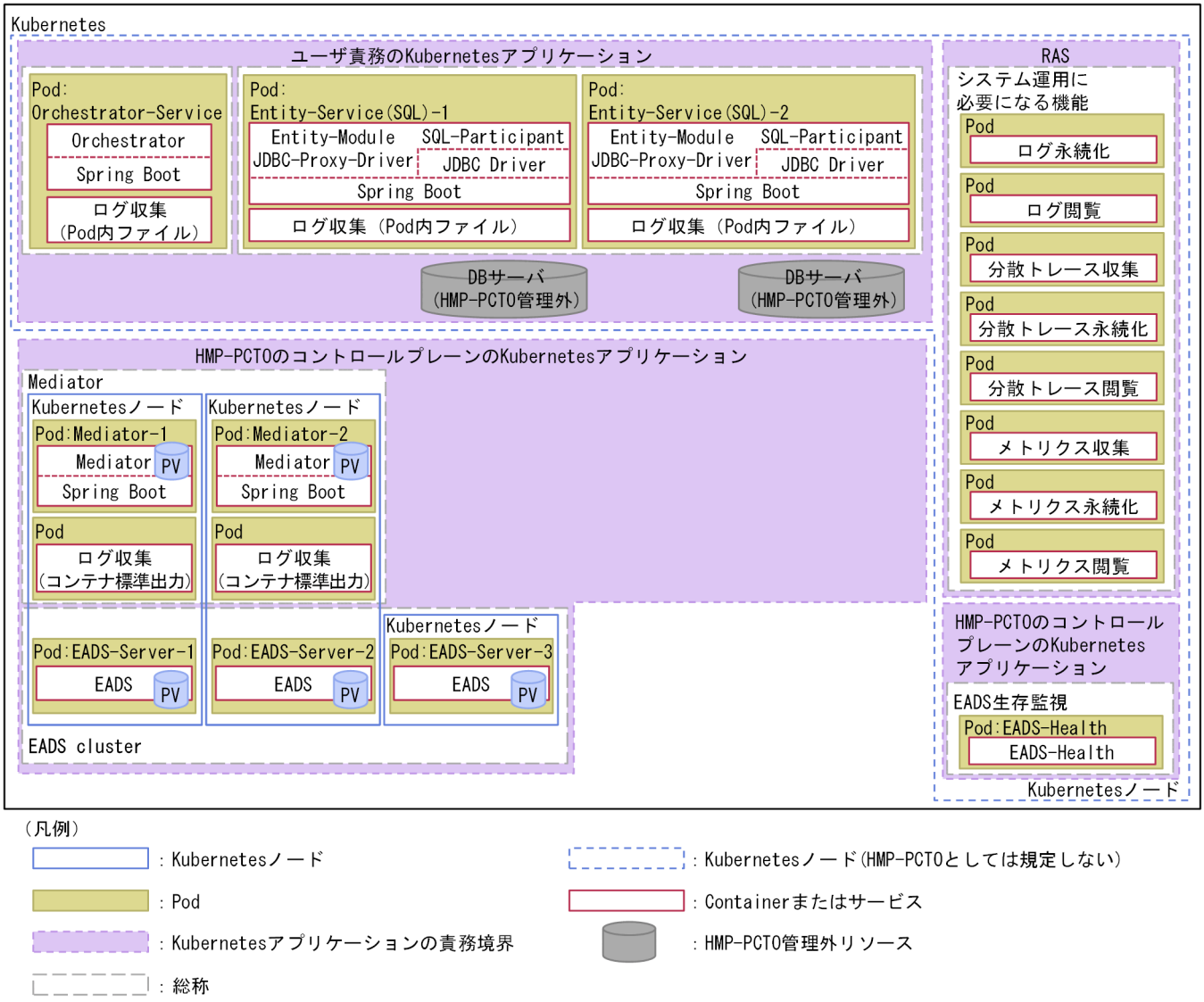
図 1-2　SQL-Participant を使用する場合の最小機器構成（ThreeQuarters 合意方式（通常版））



- 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））



(1)　トライアル版の場合

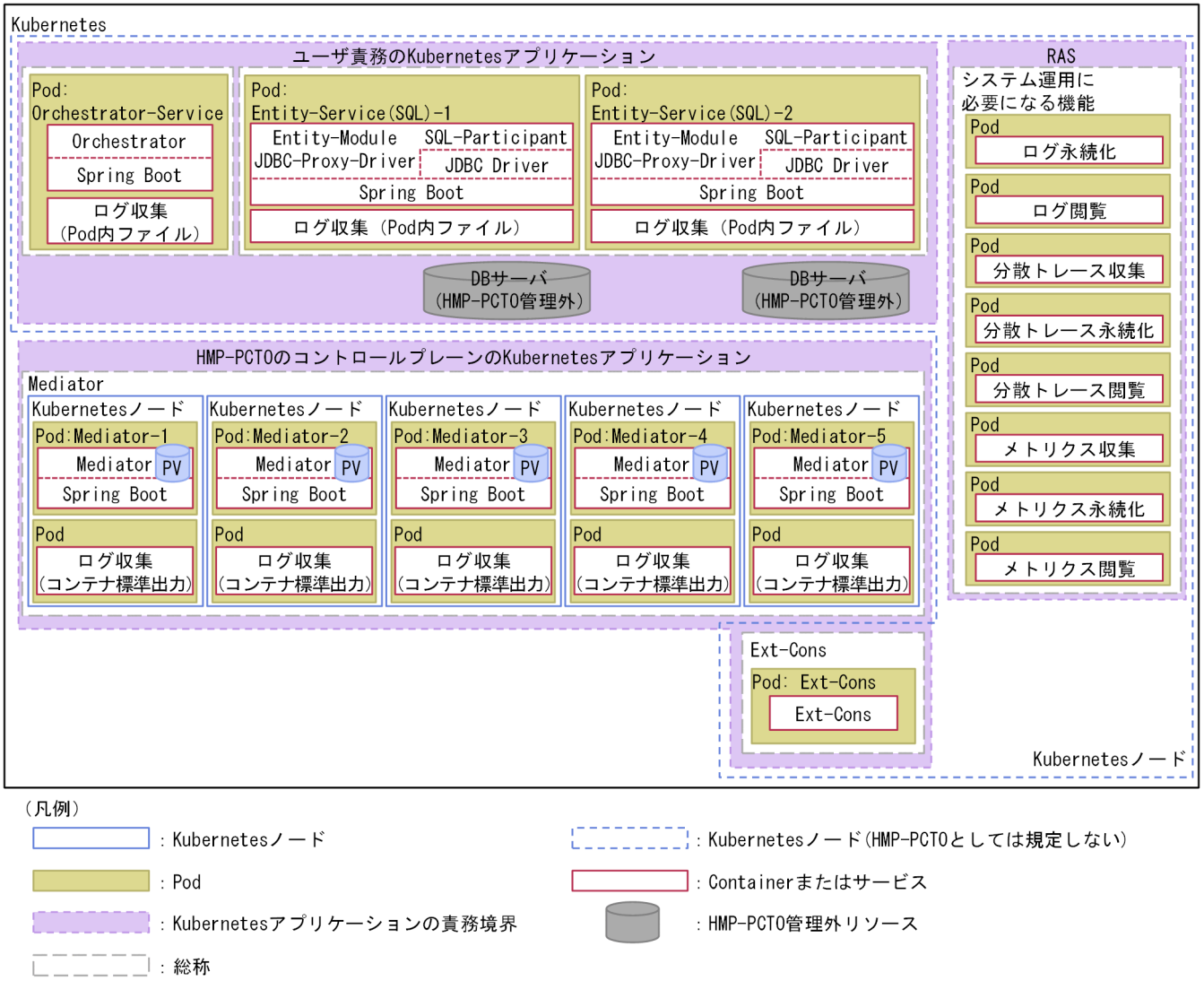
トライアル版では、EADS の代わりに Mock サーバを使用します。Mock サーバの場合、コンテナは 1 つで可用性を考慮しないため、別ノードとする必要はありません。

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。

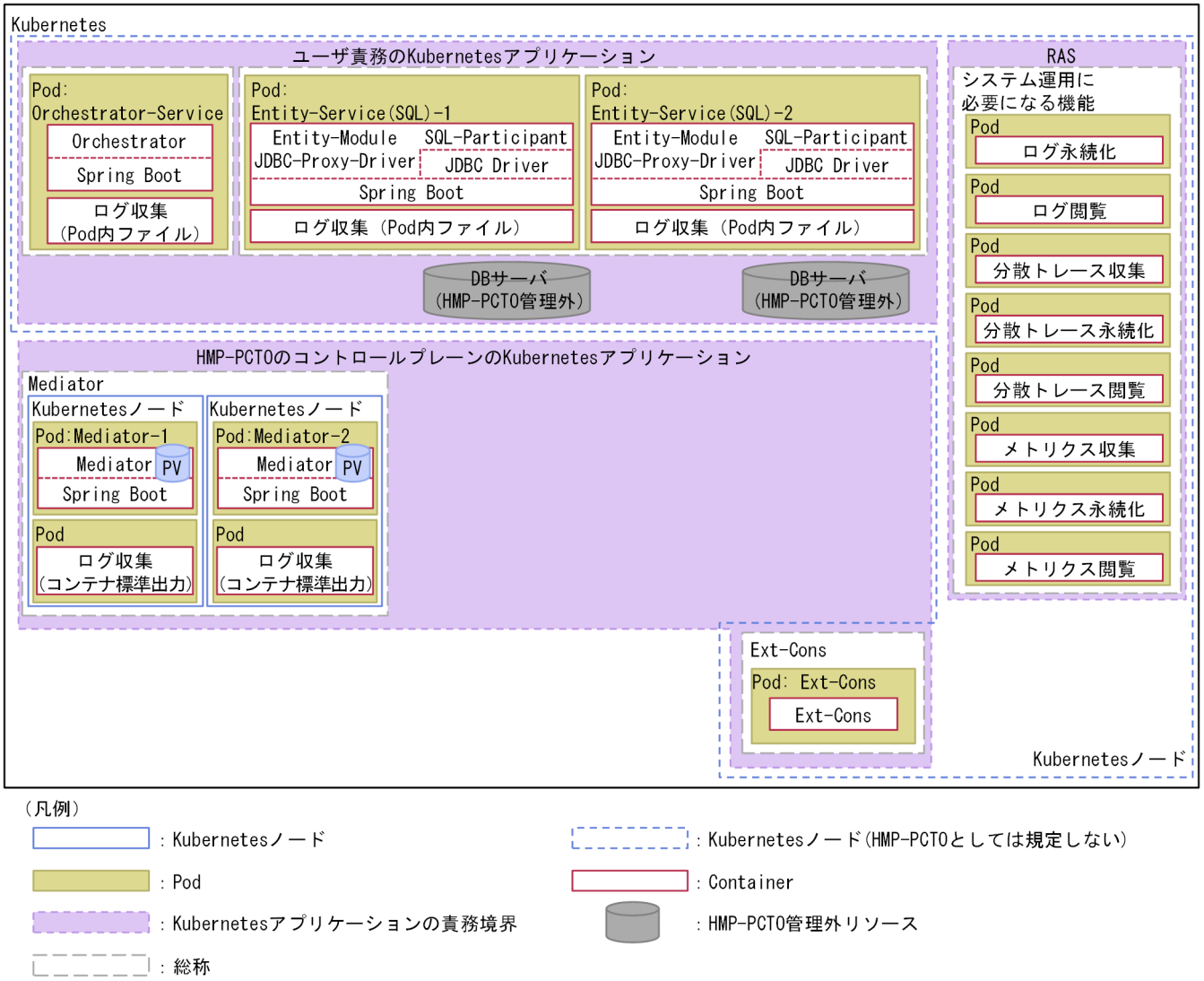
図 1-4　SQL-Participant を使用する場合の最小機器構成（ThreeQuarters 合意方式（トライアル版））



- 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。

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。

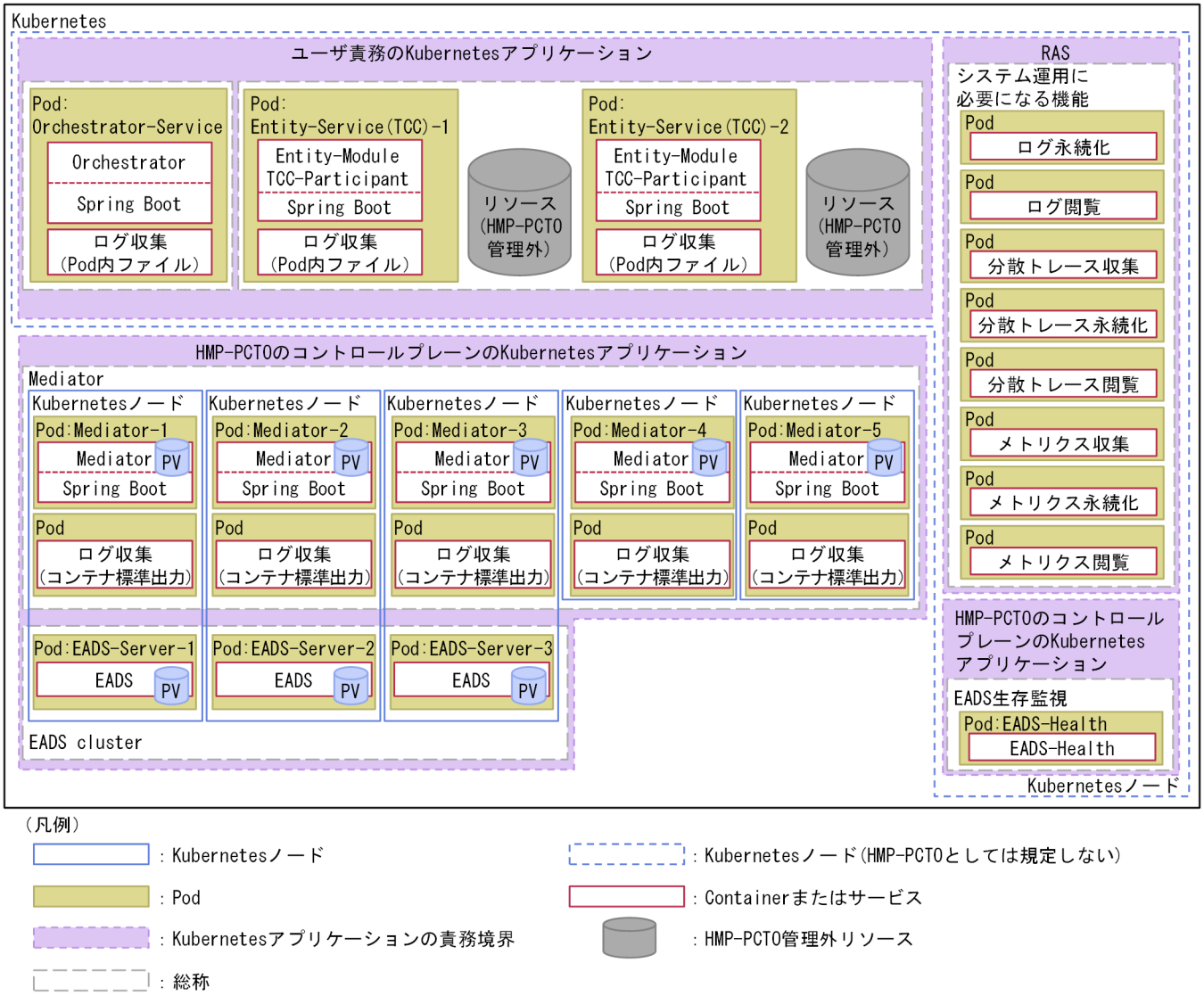
Mediator の Pod と EADS サーバの Pod は、同一のワーカーノードにも配置できます。

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。

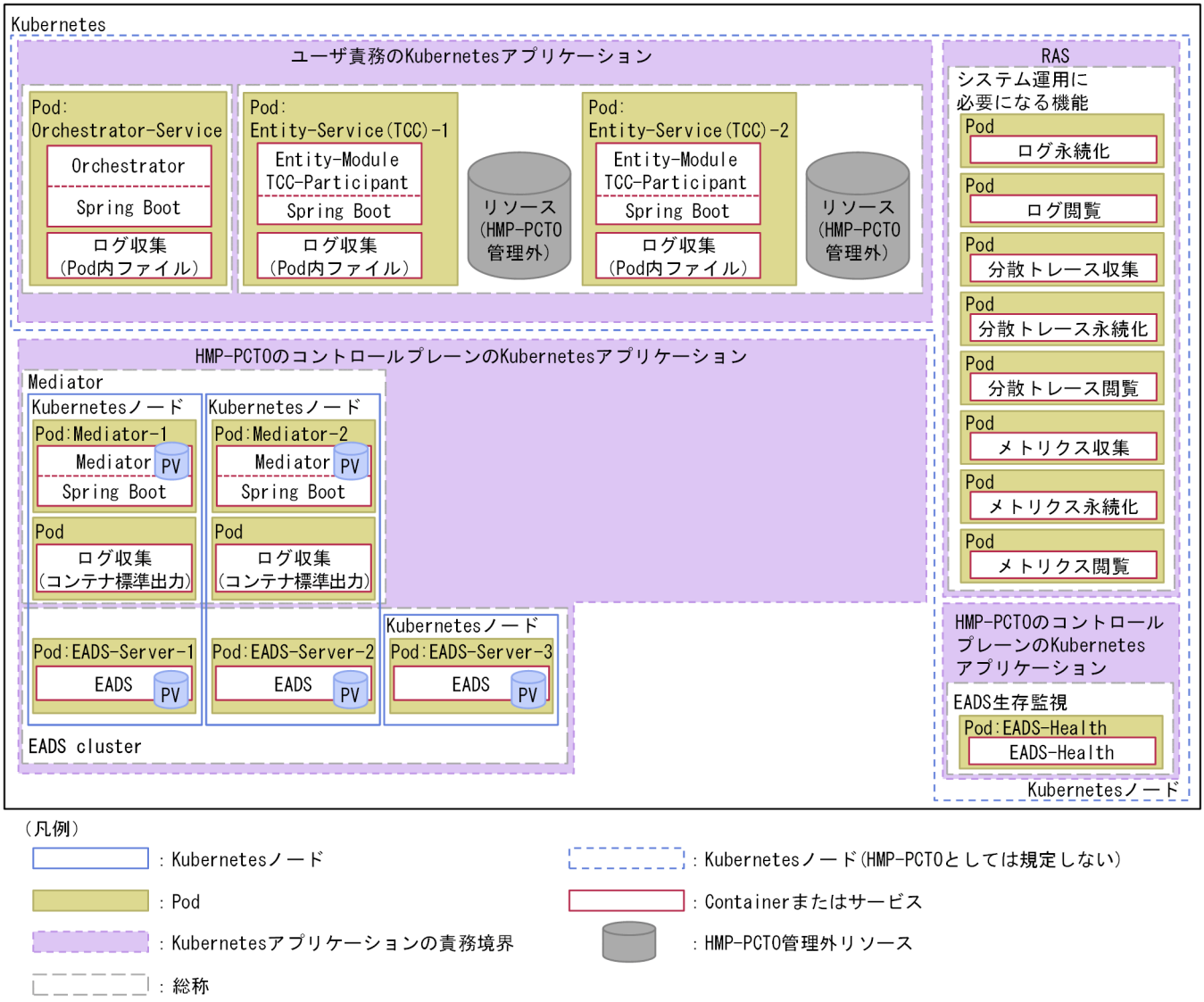
  図 1-6　TCC-Participant を使用する場合の最小機器構成（ThreeQuarters 合意方式（通常版））



- 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））



(1)　トライアル版の場合

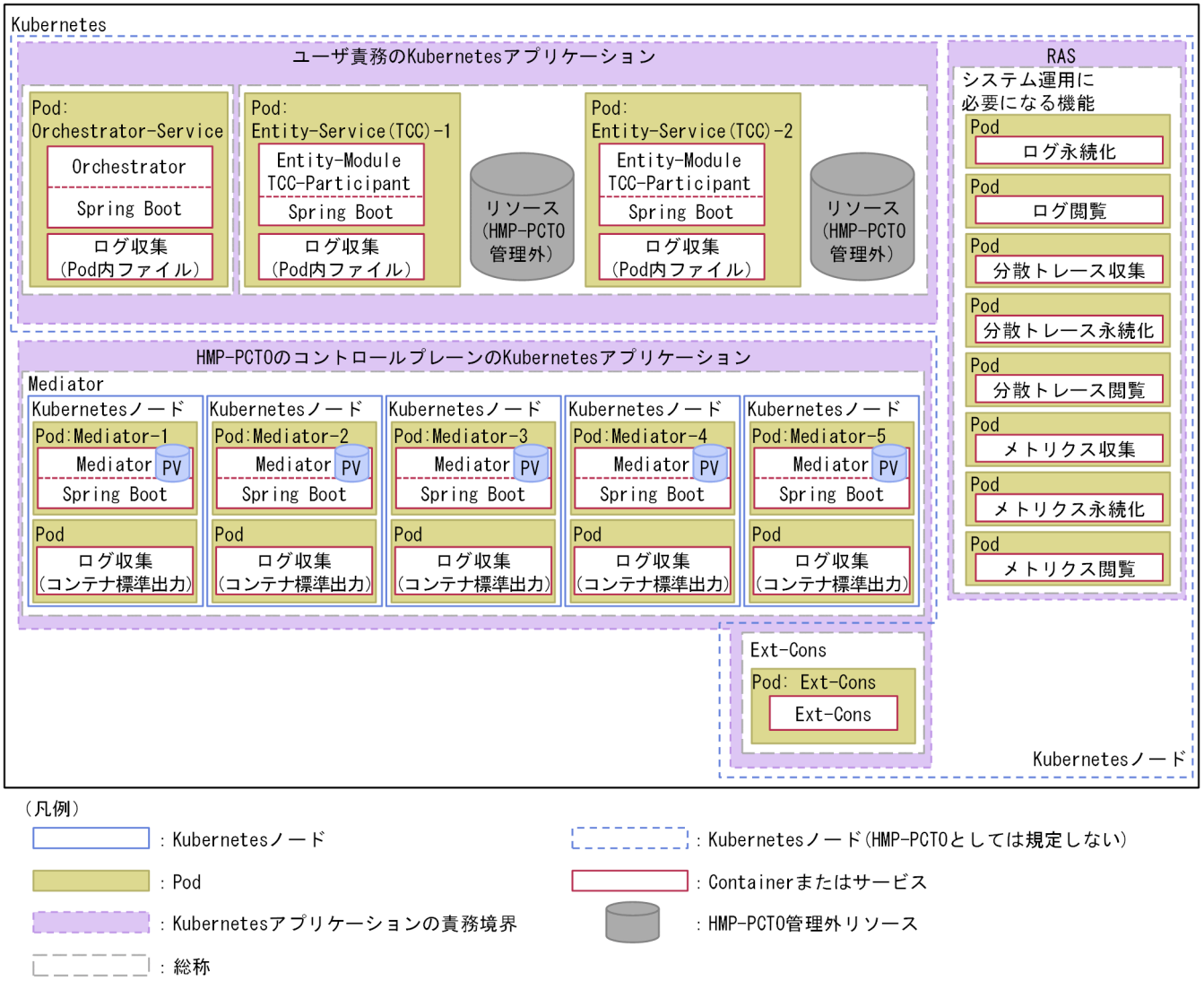
トライアル版では、EADS の代わりに Mock サーバを使用します。Mock サーバの場合、コンテナは 1 つで可用性を考慮しないため、別ノードとする必要はありません。

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。

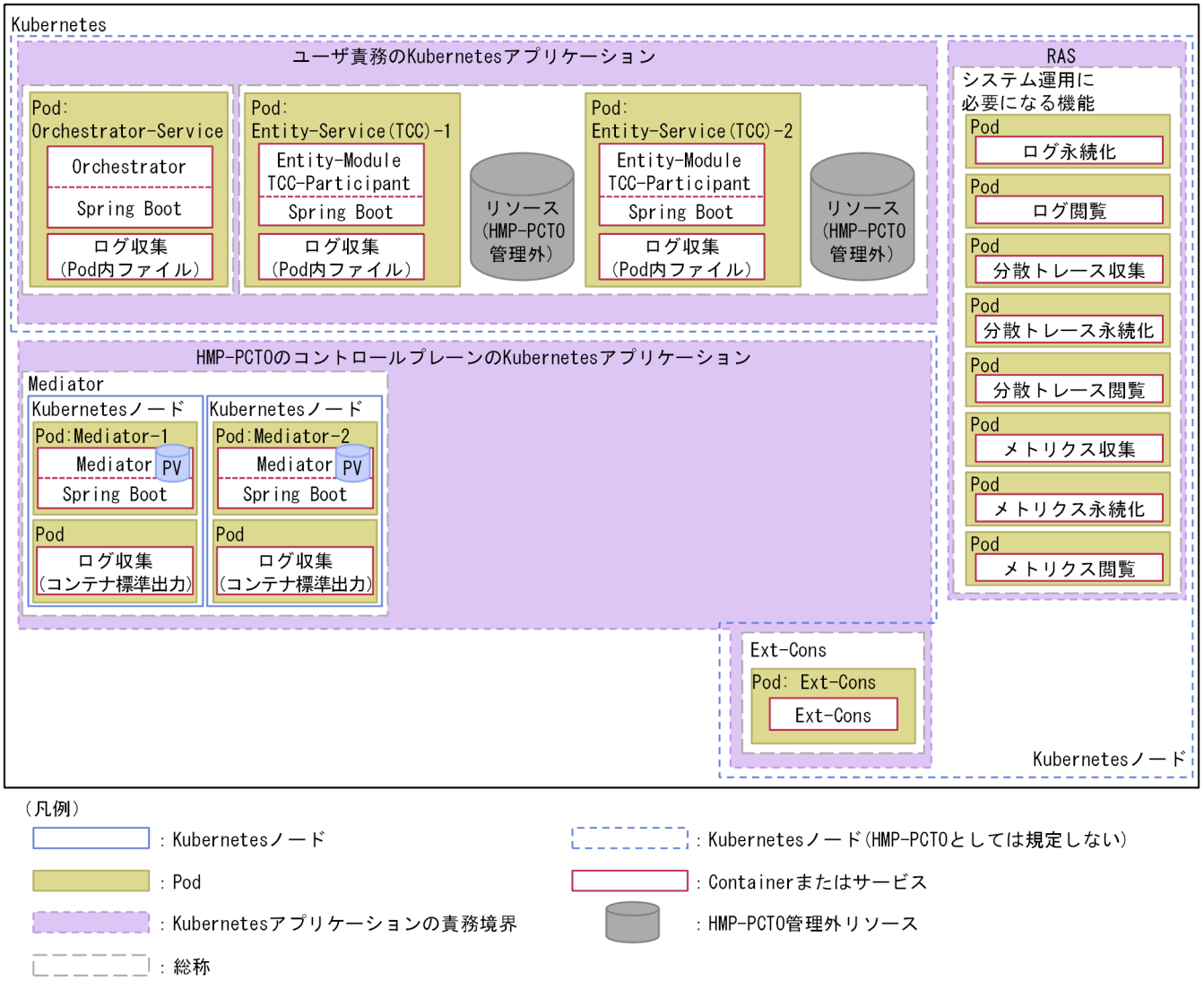
図 1-8　TCC-Participant を使用する場合の最小機器構成（ThreeQuarters 合意方式（トライアル版））



- 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。

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。

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。

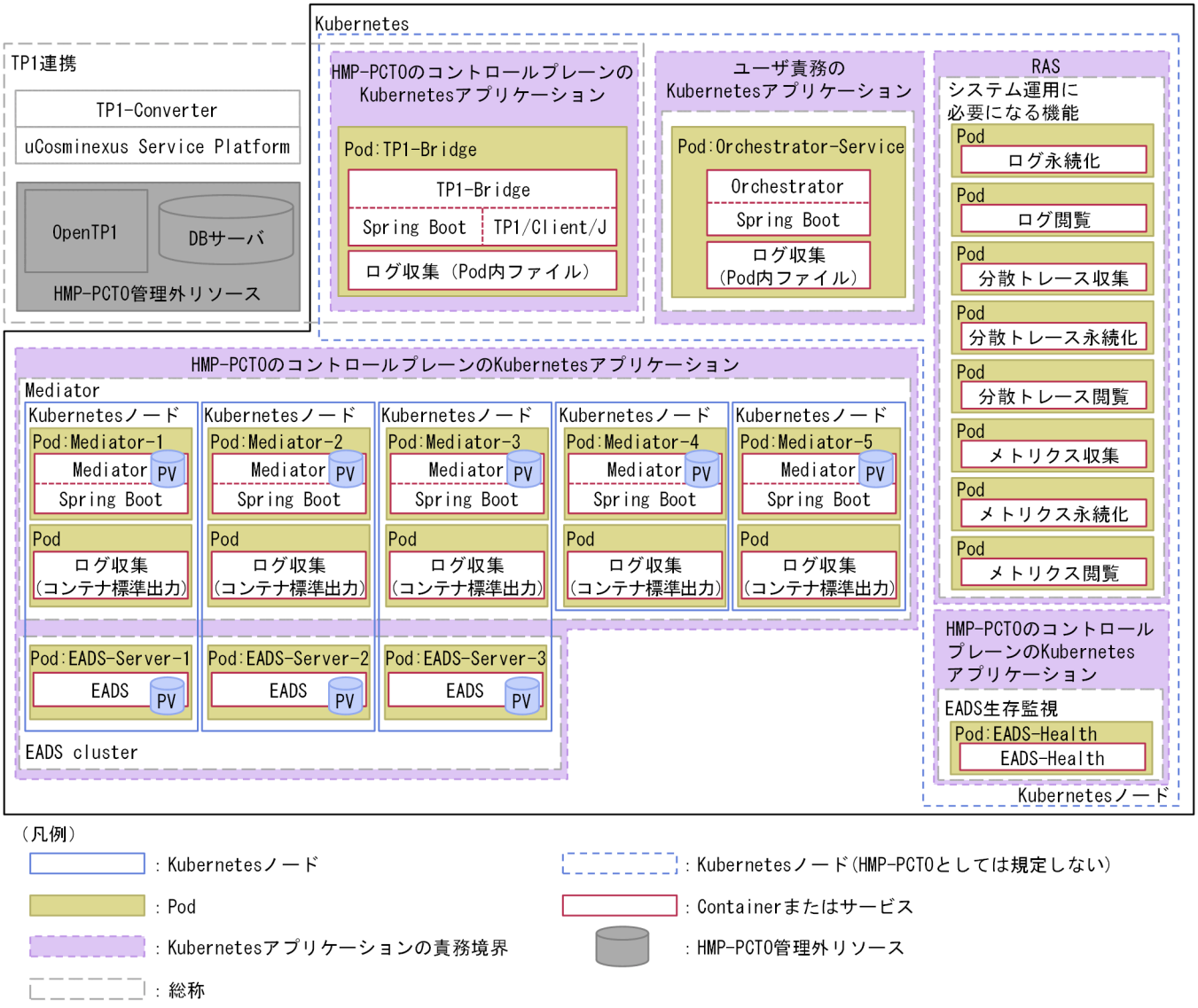
なお、各 Mediator の Pod と、補助合意用の EADS クラスタ内の Pod（EADS サーバの Pod を除く）は、任意に Kubernetes ワーカーノードに配置できます（HMP-PCTO では規定しません）。

OpenTP1 は、Kubernetes クラスタ外に配置してください。

- 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。

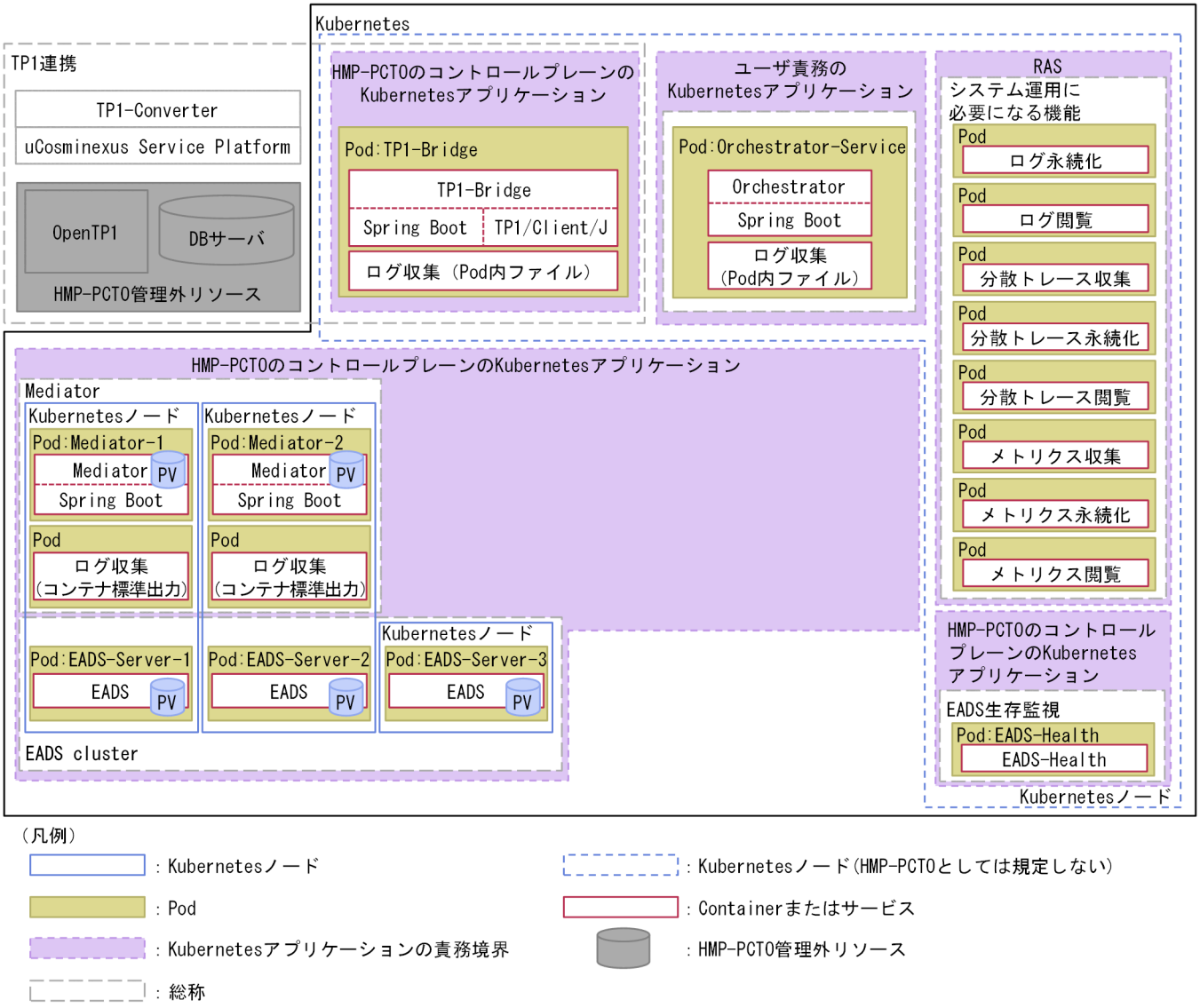
  図 1-10　TP1-Bridge を使用する場合の最小機器構成（ThreeQuarters 合意方式（通常版））



- 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（通常版））



1.4　前提条件

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。 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
5	クラスタ運用モード （Google Cloud Platform 環境を使用する場合）	Standard モード
6	クラスタ運用モード （Amazon Web Services 環境を使用する場合）	• Amazon EKS • EKS Auto Mode

(1)　Kubernetes クラスタのワークロード数

ワークロードとは、クラスタ上で実行される Kubernetes アプリケーションを実行するためのリソース（deployment、statefulset など）を指す総称です。ワークロード数は deployment や statefulset の数を意味します。次の例の場合、レプリカ数はカウントしないため、ワークロード数は 3 になります。

例：

Relay-Service として Relay-Service1、Relay-Service2、Relay-Service3 の 3 つをデプロイする。Relay-Service1 はレプリカ数 2、Relay-Service2 はレプリカ数 1、Relay-Service3 はレプリカ数 4 でデプロイする。

HMP-PCTO の Kubernetes アプリケーションごとに使用できるワークロード数と、1 ワークロードで使用できるレプリカ数について説明します。Kubernetes クラスタにデプロイできる数と、1 トランザクションで使用できる数に分けて説明します。

(a) Kubernetes クラスタにデプロイできる数

Kubernetes アプリケーションで使用する HMP-PCTO の機能に応じて、デプロイできるワークロード数およびレプリカ数が異なります。

Orchestrator の機能を使用する Kubernetes アプリケーション

Orchestrator の機能を使用する Kubernetes アプリケーションの一覧と、デプロイできるワークロード数およびレプリカ数を次の表に示します。Kubernetes アプリケーションごとに、少なくとも 1 つのワークロードが必要です。

表 1-8 Orchestrator の機能を使用する Kubernetes アプリケーションごとにデプロイできるワークロード数およびレプリカ数

項番	Kubernetes アプリケーション	ワークロード数 下限	ワークロード数 上限	レプリカ数 下限	レプリカ数 上限
1	Orchestrator-Service	1	なし	1	• スケーリング対応機能を使用する場合：なし • スケーリング対応機能を使用しない場合：1
2	Orchestrator-Service(SQL)	1		1	
3	Alternate-Service	1		1	
4	Alternate-Service(SQL)	1		1	

Participant の機能を使用する Kubernetes アプリケーション

Participant の機能を使用する Kubernetes アプリケーションの一覧と、デプロイできるワークロード数およびレプリカ数を次の表に示します。これらの Kubernetes アプリケーションごとに、少なくとも 1 つのワークロードが必要です。

ただし、HMP-PCTO は分散トランザクション機能を提供しており、複数のリソース間で一貫性のある処理を保証します。そのため、HMP-PCTO を十分に活用するには、Participant の機能を使用する Kubernetes アプリケーションを 2 つ以上のワークロードで構成することを推奨します。

表 1-9 Participant の機能を使用する Kubernetes アプリケーションごとにデプロイできるワークロード数およびレプリカ数

項番	Kubernetes アプリケーション	ワークロード数 下限	ワークロード数 上限	レプリカ数 下限	レプリカ数 上限
1	Entity-Service(SQL)	1	なし	1	• スケーリング対応機能を使用する場合：なし※

項番	Kubernetes アプリケーション	ワークロード数 下限	ワークロード数 上限	レプリカ数 下限	レプリカ数 上限
					• スケーリング対応機能を使用しない場合：1
2	Entity-Service(TCC)	1		1	1
3	Orchestrator-Service(SQL)	1		1	• スケーリング対応機能を使用する場合：なし※ • スケーリング対応機能を使用しない場合：1
4	Alternate-Service(SQL)	1		1	• スケーリング対応機能を使用する場合：なし※ • スケーリング対応機能を使用しない場合：1
5	TP1-Bridge	1		1	1

注※

スケーリング対応機能を使用する場合、デプロイするレプリカ数に応じて、Participant の機能を使用する Kubernetes アプリケーションが接続するリソースとの最大接続数を決定してください。詳細については、「4.1　スケーリング対応機能の概要」を参照してください。

Mediator の機能を使用する Kubernetes アプリケーション

Mediator の機能を使用する Kubernetes アプリケーションの一覧と、デプロイできるワークロード数およびレプリカ数を次の表に示します。

表 1-10　Mediator の機能を使用する Kubernetes アプリケーションごとにデプロイできるワークロード数およびレプリカ数

項番	Kubernetes アプリケーション	ワークロード数 下限	ワークロード数 上限	レプリカ数 下限	レプリカ数 上限
1	Mediator	1	1 つの namespace につき、1 つまでです。namespace 数に制限はありません。	2	64

Entity-Module の機能を使用する Kubernetes アプリケーション

Entity-Module の機能を使用する Kubernetes アプリケーションの一覧と、デプロイできるワークロード数およびレプリカ数を次の表に示します。

表 1-11　Entity-Module の機能を使用する Kubernetes アプリケーションごとにデプロイできるワークロード数およびレプリカ数

項番	Kubernetes アプリケーション	ワークロード数 下限	ワークロード数 上限	レプリカ数 下限	レプリカ数 上限
1	Relay-Service	0	なし	1	• スケーリング対応機能を使用する場合：なし • スケーリング対応機能を使用しない場合：1
2	Entity-Service(SQL)※	0		1	• スケーリング対応機能を使用する場合：なし • スケーリング対応機能を使用しない場合：1
3	Entity-Service(TCC)※	0		1	1
4	Alternate-Service	0		1	• スケーリング対応機能を使用する場合：なし • スケーリング対応機能を使用しない場合：1
5	Alternate-Service(SQL)※	0		1	• スケーリング対応機能を使用する場合：なし • スケーリング対応機能を使用しない場合：1

注※

Kubernetes アプリケーションのワークロード数およびレプリカ数は、リソースの参照および更新をしない場合の数を示します。リソースの参照および更新をする場合は「Participant の機能を使用する Kubernetes アプリケーション」を参照してください。

(b)　1 トランザクションで使用できる数

Kubernetes アプリケーションで使用する HMP-PCTO の機能に応じて、1 トランザクションで使用できるワークロード数、レプリカ数が異なります。

Orchestrator の機能を使用する Kubernetes アプリケーション

Orchestrator の機能を使用する Kubernetes アプリケーションの一覧と、使用できるワークロード数およびレプリカ数を次の表に示します。Kubernetes アプリケーションごとに使用できるワークロードは 1 つだけです。また、複数のレプリカがあっても、そのうちの 1 つのレプリカを使用します。

表 1-12　Orchestrator の機能を使用する Kubernetes アプリケーションごとに使用できるワークロード数およびレプリカ数

項番	Kubernetes アプリケーション	ワークロード数上限	レプリカ数上限
1	Orchestrator-Service	1	1
2	Orchestrator-Service(SQL)	1	1
3	Alternate-Service	1	1
4	Alternate-Service(SQL)	1	1

Participant の機能を使用する Kubernetes アプリケーション

Participant の機能を使用する Kubernetes アプリケーションの一覧と、使用できるワークロード数およびレプリカ数を次の表に示します。Kubernetes アプリケーションごとに 1 つ以上のワークロードが必要で、合計で最大 20 まで使用できます。また、複数のレプリカがあっても、そのうちの 1 つのレプリカを使用します。

表 1-13　Participant の機能を使用する Kubernetes アプリケーションごとに使用できるワークロード数およびレプリカ数

項番	Kubernetes アプリケーション	ワークロード数上限	レプリカ数上限
1	Entity-Service(SQL)	20	1
2	Entity-Service(TCC)	20	1
3	Orchestrator-Service(SQL)	20	1
4	Alternate-Service(SQL)	20	1
5	TP1-Bridge	20	1

Mediator の機能を使用する Kubernetes アプリケーション

次の表に、Mediator の機能を使用する Kubernetes アプリケーションの一覧と、使用できるワークロード数およびレプリカ数を次の表に示します。

表 1-14　Mediator の機能を使用する Kubernetes アプリケーションごとに使用できるワークロード数およびレプリカ数

項番	Kubernetes アプリケーション	ワークロード数上限	レプリカ数上限
1	Mediator	1	mediator.size パラメタを参照してください。

Entity-Module の機能を使用する Kubernetes アプリケーション

Entity-Module の機能を使用する Kubernetes アプリケーションの一覧と、使用できるワークロード数およびレプリカ数を次の表に示します。

表 1-15　Entity-Module の機能を使用する Kubernetes アプリケーションごとに使用できるワークロード数およびレプリカ数

項番	Kubernetes アプリケーション	ワークロード数上限	レプリカ数上限
1	Relay-Service	なし	1
2	Entity-Service(SQL)※		1
3	Entity-Service(TCC)※		1
4	Alternate-Service		1
5	Alternate-Service(SQL)※		1

注※

Kubernetes アプリケーションのワークロードおよびレプリカ数は、リソースの参照または更新をしない場合の数を示します。リソースの参照または更新をする場合は「Participant の機能を使用する Kubernetes アプリケーション」を参照してください。

1.4.2　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 1-16　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

1.4.3　前提とする Red Hat OpenShift Container Platform の設定

Red Hat OpenShift Container Platform で、デフォルトで用意されている SCC（Security Context Constraints）ではなく、ユーザがカスタマイズした SCC を使用する場合、HMP-PCTO が前提とする SCC のパラメタ設定を次に示します。

表 1-17　HMP-PCTO が前提とする SCC のパラメタ設定

項番	項目	前提条件
1	readOnlyRootFilesystem	false を指定
2	seccompProfiles	RuntimeDefault を指定
3	volumes	次のボリュームタイプを指定 • configMap • csi • downwardAPI

項番	項目	前提条件
		• emptyDir • ephemeral • persistentVolumeClaim • projected • secret

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。

     スケーリング対応機能を使用する場合、監視に必要な Kubernetes アプリケーションのデプロイ、およびスケーリング対応機能を使用する Kubernetes アプリケーションへのスケーリング対応機能の設定が必要です（4. スケーリング対応機能の運用）。デプロイ後、Blue/Green デプロイメントを使用して環境を切り替えることもできます（取扱説明書「HMP-PCTO テクニカルガイド」）。
   - HMP-PCTO のコントロールプレーンの Kubernetes アプリケーションおよび RAS をデプロイする場合（3. HMP-PCTO のコントロールプレーンの Kubernetes アプリケーションおよび RAS 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。

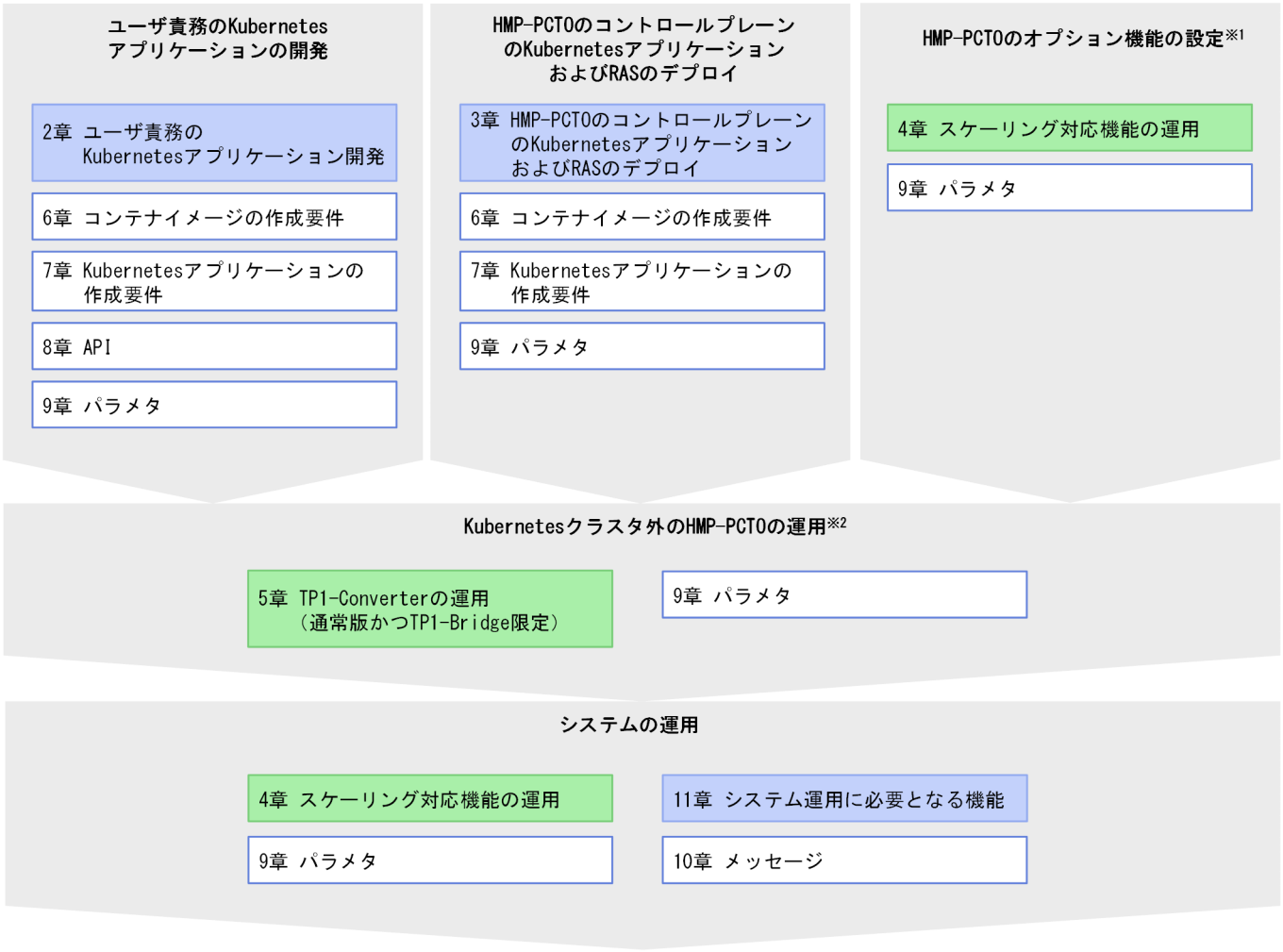
     スケーリング対応機能を使用する場合、監視に必要な Kubernetes アプリケーションのデプロイ、およびスケーリング対応機能を使用する Kubernetes アプリケーションへのスケーリング対応機能の設定が必要です（4. スケーリング対応機能の運用）。デプロイ後、Blue/Green デプロイメントを使用して環境を切り替えることもできます（取扱説明書「HMP-PCTO テクニカルガイド」）。
2. HMP-PCTO を適用したシステムを運用する（11. システム運用に必要となる機能）

   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。また、HMP-PCTO の稼働日を特定する場合に、必要なファイルを取得します。

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（5. 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。特定の機能を使用する場合は「使用する機能に応じて読む章」を、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は「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、機能や環境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　：使用する機能に応じて読む章　：必要に応じて読む章

注※1：ユーザ責務のKubernetesアプリケーションの開発、HMP-PCTOのコントロールプレーンのKubernetesアプリケーションおよびRASのデプロイが完了したあとに、HMP-PCTOのオプション機能の設定を進めることもできます。

注※2：ユーザ責務のKubernetesアプリケーションの開発、HMP-PCTOのコントロールプレーンのKubernetesアプリケーションおよびRASのデプロイ、HMP-PCTOのオプション機能の設定と並行して、Kubernetesクラスタ外のHMP-PCTOの運用を進めることもできます。

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 アプリケーションの開発の流れについて説明します。

デプロイ後のシステム運用時の注意事項については、「付録 E　システム運用時の注意事項」を参照してください。

> **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。

2.1.1　オンプレミス環境の場合

オンプレミス環境の場合、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。スケーリング対応機能を使用する場合は、「4.　スケーリング対応機能の運用」も併せて参照してください。

表 2-1　ユーザ責務の Kubernetes アプリケーションの開発の流れ（オンプレミス環境の場合）

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	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。	2.3　コンテナイメージ作成
3	永続ボリュームとプロビジョニング方法の選定	HMP-PCTO で使用する永続ボリュームのストレージ種別と、そのプロビジョニング方法を選定します。通常版の場合だけ実施します。	2.4　永続ボリュームとプロビジョニング方法の選定（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 および関連製品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

2.1.2 Microsoft Azure 環境の場合

Microsoft Azure 環境の場合、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。スケーリング対応機能を使用する場合は、「4. スケーリング対応機能の運用」も併せて参照してください。

表 2-2 ユーザ責務の Kubernetes アプリケーションの開発の流れ（Microsoft Azure 環境を使用する場合）

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	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。	2.3 コンテナイメージ作成
3	Azure Container Registry の有効化およびコンテナイメージの登録	Azure Kubernetes Service で作成した AKS クラスタから参照可能なコンテナレジストリ Azure Container Registry に、コンテナイメージを登録します。	Microsoft Azure の公式ドキュメント
4	永続ボリュームとプロビジョニング方法の選定	HMP-PCTO で使用する永続ボリュームのストレージ種別と、そのプロビジョニング方法を選定します。通常版の場合だけ実施します。	2.4 永続ボリュームとプロビジョニング方法の選定（通常版限定）
5	Microsoft Azure のマネージドストレージサービスのプロビジョニング	HMP-PCTO および関連製品で使用する共有ファイルシステムをプロビジョニングします。通常版の場合だけ実施します。	• Microsoft Azure の公式ドキュメント • 2.5 共有ファイルシステムでの作業（通常版限定）
6	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。	• Microsoft Azure の公式ドキュメント • 2.6 Kubernetes マニフェスト作成
7	永続ボリュームの Kubernetes マニフェストの作成	共有ファイルシステムを永続ボリュームとして使用するための Kubernetes マニフェストを作成します。	Microsoft Azure の公式ドキュメント
8	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

2.1.3 Google Cloud Platform 環境の場合

Google Cloud Platform 環境の場合、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。スケーリング対応機能を使用する場合は、「4. スケーリング対応機能の運用」も併せて参照してください。

表 2-3　ユーザ責務の Kubernetes アプリケーションの開発の流れ（Google Cloud Platform 環境を使用する場合）

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	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。	2.3　コンテナイメージ作成
3	Artifact Registry の有効化およびコンテナイメージの登録	Google Kubernetes Engine で作成した GKE クラスタから参照可能なコンテナレジストリ Artifact Registry にコンテナイメージを登録します。	Google Cloud Platform の公式ドキュメント
4	永続ボリュームとプロビジョニング方法の選定	HMP-PCTO で使用する永続ボリュームのストレージ種別と、そのプロビジョニング方法を選定します。通常版の場合だけ実施します。	2.4　永続ボリュームとプロビジョニング方法の選定（通常版限定）
5	Google Cloud Platform のマネージドストレージサービスのプロビジョニング	HMP-PCTO および関連製品で使用する共有ファイルシステムをプロビジョニングします。	• Google Cloud Platform の公式ドキュメント • 2.5　共有ファイルシステムでの作業（通常版限定）
6	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
7	永続ボリュームの Kubernetes マニフェストの作成	共有ファイルシステムを永続ボリュームとして使用するための Kubernetes マニフェストを作成します。	Google Cloud Platform の公式ドキュメント
8	Cloud Trace を出力する権限の付与	HMP-PCTO の分散トレースを収集するため、Cloud Trace を出力する権限を付与します。 「2.6　Kubernetes マニフェスト作成」で作成した、各 Pod が使用するサービスアカウントに対して、cloudtrace.agent ロールを付与してください。	Google Cloud Platform の公式ドキュメント
9	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

2.1.4　Amazon Web Services 環境の場合

Amazon Web Services 環境の場合、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。スケーリング対応機能を使用するときは、「4.　スケーリング対応機能の運用」も併せて参照してください。

表 2-4　ユーザ責務の Kubernetes アプリケーションの開発の流れ（Amazon Web Services 環境を使用する場合）

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	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。	2.3　コンテナイメージ作成
3	Amazon ECR の有効化およびコンテナイメージの登録	Amazon Elastic Kubernetes Service で作成した EKS クラスタから参照可能なコンテナレジストリ Amazon ECR に、コンテナイメージを登録します。	Amazon Web Services の公式ドキュメント
4	永続ボリュームとプロビジョニング方法の選定	HMP-PCTO で使用する永続ボリュームのストレージ種別と、そのプロビジョニング方法を選定します。通常版の場合だけ実施します。	2.4　永続ボリュームとプロビジョニング方法の選定（通常版限定）
5	Amazon Web Services のマネージドストレージサービスのプロビジョニング	HMP-PCTO および関連製品で使用する共有ファイルシステムをプロビジョニングします。通常版の場合だけ実施します。	• Amazon Web Services の公式ドキュメント • 2.5　共有ファイルシステムでの作業（通常版限定）
6	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
7	永続ボリュームの Kubernetes マニフェストの作成	共有ファイルシステムを永続ボリュームとして使用するための Kubernetes マニフェストを作成します。	Amazon Web Services の公式ドキュメント
8	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

2.1.5　Red Hat OpenShift Container Platform 環境の場合

Red Hat OpenShift Container Platform 環境の場合、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。スケーリング対応機能を使用する場合は、「4.　スケーリング対応機能の運用」も併せて参照してください。

表 2-5　ユーザ責務の Kubernetes アプリケーションの開発の流れ（Red Hat OpenShift Container Platform 環境の場合）

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	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。	2.3　コンテナイメージ作成

項番	項目	説明	参照先
3	永続ボリュームとプロビジョニング方法の選定	HMP-PCTO で使用する永続ボリュームのストレージ種別と、そのプロビジョニング方法を選定します。通常版の場合だけ実施します。	2.4　永続ボリュームとプロビジョニング方法の選定（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 および関連製品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

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、Gradle8x 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。

作成する Java アプリケーションがどの Kubernetes アプリケーションかどうかによって、作業項目が変わります。

表 2-6　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	Kubernetes アプリケーション	備考	参照先
共通	事前準備	全 Kubernetes アプリケーション共通	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 Kubernetes アプリケーション共通	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-Service • Relay-Service • Entity-Service(SQL) • Entity-Service(TCC) • Orchestrator-Service(SQL) • Alternate-Service • Alternate-Service(SQL)	必ず実施します。	2.2.4　コンポーネントスキャンの追加
	DI に使用するアノテーションの注意事項			2.2.5　DI に使用するアノテーションの注意事項
	アノテーションによるトランザクション境界の設定	• Orchestrator-Service • Orchestrator-Service(SQL) • Alternate-Service		2.2.6　アノテーションによるトランザクション境界の設定

分類	作業項目	Kubernetes アプリケーション	備考	参照先
		• Alternate-Service(SQL)		
	アノテーションによるトランザクションブランチの作成	• Orchestrator-Service(SQL) • Alternate-Service(SQL)		2.2.7　アノテーションによるトランザクションブランチの作成（SQL-Participant 限定）
	通信によるトランザクションブランチの作成（REST 通信-クライアント）	• Orchestrator-Service • Relay-Service • Entity-Service(SQL) • Entity-Service(TCC) • Orchestrator-Service(SQL) • Alternate-Service • Alternate-Service(SQL)		2.2.8　通信によるトランザクションブランチの作成（REST 通信-クライアント）
	通信によるトランザクションブランチの作成（REST 通信-サーバ）	• Relay-Service • Entity-Service(SQL) • Entity-Service(TCC) • Alternate-Service • Alternate-Service(SQL)		2.2.9　通信によるトランザクションブランチの作成（REST 通信-サーバ）
	通信によるトランザクションブランチの作成（gRPC 通信-クライアント）	• Orchestrator-Service • Relay-Service • Entity-Service(SQL) • Entity-Service(TCC) • Orchestrator-Service(SQL) • Alternate-Service • Alternate-Service(SQL)		2.2.10　通信によるトランザクションブランチの作成（gRPC 通信-クライアント）

分類	作業項目	Kubernetes アプリケーション	備考	参照先
	通信によるトランザクションブランチの作成（gRPC 通信-サーバ）	• Relay-Service • Entity-Service(SQL) • Entity-Service(TCC) • Alternate-Service • Alternate-Service(SQL)		2.2.11　通信によるトランザクションブランチの作成（gRPC 通信-サーバ）
	JDBC Proxy Driver の使用（SQL-Participant 限定）	• Entity-Service(SQL) • Orchestrator-Service(SQL) • Alternate-Service(SQL)	SQL-Participant を使用する場合だけ実施します。	2.2.12　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）	Entity-Service(TCC)	TCC-Participant を使用する場合だけ実施します。	2.2.13　TccService インタフェースの実装クラスの作成（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）	Entity-Service(TCC)		2.2.14　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-Service • Relay-Service • Entity-Service(SQL) • Entity-Service(TCC) • Orchestrator-Service(SQL) • Alternate-Service • Alternate-Service(SQL)	TP1-Bridge を使用する場合だけ実施します。	2.2.15　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		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。	2.2.16　ロギングライブラリの設定ファイルのカスタマイズ
	分散トレース機能の設定		分散トレース機能を使用する場合だけ実施します。	2.2.17　分散トレース機能の設定
	Quartz スケジューラ機能の使用		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 を使用する場合だけ実施します。	2.2.18　Quartz スケジューラ機能の利用
	アプリケーションが出力するメッセージの文字コード		必ず実施します。	2.2.19　アプリケーションが出力するメッセージの文字コード
	Spring Boot の組み込み HTTP サーバの設定			2.2.20　Spring Boot の組み込み HTTP サーバの設定

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、次のアプリケーションで実装してください。

- Orchestrator-Service
- Relay-Service
- Entity-Service(SQL)
- Entity-Service(TCC)
- Orchestrator-Service(SQL)
- Alternate-Service
- Alternate-Service(SQL)

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 などほか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　アノテーションによる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、次のアプリケーションで実装してください。

- Orchestrator-Service
- Orchestrator-Service(SQL)
- Alternate-Service
- Alternate-Service(SQL)

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、「8.　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 で組み込む記述例を次に示します。

```
dependencies {
    implementation 'jp.co.Hitachi.soft.hmppcto:hmppcto-orchestrator:<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。

(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 2-7　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	なし
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、または Mediator が決着結果不明を返却し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、「11.　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　アノテーションによるトランザクションブランチの作成（SQL-Participant 限定）

アノテーションによってトランザクションブランチを作成するアプリケーションの開発方法について説明します。アノテーションによるトランザクションブランチの作成は、次のアプリケーションで実装してください。

- Orchestrator-Service(SQL)
- Alternate-Service(SQL)

トランザクションブランチの作成には、@PctoTransactionBranch アノテーションを使用します。また、@PctoTransactionBranch アノテーションに属性を設定することで、先行プリペア機能を使用できます。@PctoTransactionBranch アノテーションの API 仕様の詳細については、「8.2.16 @PctoTransactionBranch アノテーション」を参照してください。@PctoTransactionBranch 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。同一のクラスの別メソッドから呼び出した場合は、HMP-PCTO の機能は正しく動作しません。

(1)　@PctoTransactionBranch アノテーションをメソッドに指定

(a)　ライブラリの組み込み

@PctoTransactionBranch アノテーションを使用するには、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 で組み込む記述例を次に示します。

```
dependencies {
    implementation 'jp.co.Hitachi.soft.hmppcto:hmppcto-orchestrator:<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。

(b)　コーディング例

@PctoTransactionBranch 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。
@PctoTransactionBranch 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。

```
@PctoTransactionBranch
public void service() {

    // 業務処理

}
```

(c)　エラーハンドリング

トランザクションではない呼び出し元で@PctoTransactionBranch アノテーションを指定したメソッドを呼び出した場合、メソッドの呼び出し元に TransactionBranchException 例外が送出されます。

(2)　先行プリペア機能の使用

先行プリペア機能を使用するには、@PctoTransactionBranch アノテーションを指定する際に、アノテーションの prePreparedOn 属性を設定してください。

(a)　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、@PctoTransactionBranch アノテーションをメソッドに指定する際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、「(a)　ライブラリの組み込み」を参照してください。

(b)　コーディング例

先行プリペア機能を使用する場合、@PctoTransactionBranch アノテーションの prePreparedOn 属性に true を設定します。prePreparedOn 属性の詳細については、「8.2.16　@PctoTransactionBranch アノテーション」を参照してください。

```
@PctoTransactionBranch(prePreparedOn = true)
public void service() {

    // 業務処理

}
```

(c)　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は発生しません。

(3)　注意事項

先行プリペアを実行した@PctoTransactionBranch アノテーションを指定したメソッドの実行後に、@PctoTransactionBranch アノテーションを指定した別のメソッドを実行した場合、SQL は実行できません。先行プリペア実行後に SQL を実行するために JDBC Proxy Driver のメソッドを実行した場合は、PctoPrePreparedException 例外を送出して、SQL を実行しないで無視します。トランザクションがコミットするかロールバックするかには影響しません。アプリケーションで PctoPrePreparedException 例外をキャッチしてロールバックするかどうかを判断してください。

2.2.8　通信によるトランザクションブランチの作成（REST 通信-クライアント）

REST 通信によるトランザクションブランチを作成するクライアントの開発方法について説明します。

Spring Framework の RestTemplate を使用してトランザクションブランチを作成します。REST 通信によるトランザクションブランチの作成は、次のアプリケーションで実装してください。

- Orchestrator-Service
- Relay-Service
- Entity-Service(SQL)
- Entity-Service(TCC)
- Orchestrator-Service(SQL)
- Alternate-Service
- Alternate-Service(SQL)

クライアントでは、HMP-PCTO のインタセプタを登録してトランザクションを伝搬する必要があります。また、トランザクションブランチを作成することで、先行プリペア機能を使用できます。

インタセプタの登録方法と、先行プリペア機能の使用方法を次に示します。

(1) Spring Framework の RestTemplate を使用したインタセプタの登録

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、次の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。

- hmppcto-propagation-rest-template
- hmppcto-common

Gradle で組み込む記述例を次に示します。

```
dependencies {
    implementation 'jp.co.Hitachi.soft.hmppcto:hmppcto-propagation-rest-template:<V.R.S>※'
    implementation 'jp.co.Hitachi.soft.hmppcto:hmppcto-common:<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。

(b) 宛先のホスト部分の形式

トランザクションの伝搬時、HTTP 通信に指定する宛先のホスト部分は、宛先の Service を示すドメイン名で指定してください。

スケーリング対応機能を使用し、HMP-PCTO が提供するルーティング機能を使用する（set-routing パラメタに「inner」を指定する）場合は、次の形式で指定する必要があります。

Service を示すドメイン名の指定形式を次に示します。

- <Service 名>
- <Service 名>.<Namespace 名>
- <Service 名>.<Namespace 名>.svc

上記以外の形式で宛先を指定した場合は、HMP-PCTO の機能は正しく動作しません。

以降に、宛先の Service への HTTP 通信に指定するドメイン名の例を示します。

```
RequestEntity<MyRequest> request = RequestEntity    // エンティティ作成
     .post("https://my-svc.my-namespace.svc:8080/{foo}", "bar")          // 宛先のServiceを示すドメイン名で指定
     .body(body);
```

この例では、宛先の Service の Service を示すドメイン名を"my-svc.my-namespace.svc"（「*<Service 名>*.*<Namespace 名>*.svc」形式）としています。

(c) 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
```

(d) エラーハンドリング

RestTemplate を利用した通信時に動作する TxPropagationRestTemplateInterceptor クラスの前提処理で例外が発生したとき、RestClientException が送出されます。必要に応じて例外を catch して後処理をしてください。

この場合、例外の Caused には次の例外の情報が出力されます。

- jp.co.Hitachi.soft.hmppcto.lb.exception.ServiceNotFoundException
- jp.co.Hitachi.soft.hmppcto.lb.exception.PodNotFoundException

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(AbstractClientHt
tpResponse.java:33)
at jp.co.Hitachi.soft.hmppcto.filter.propagation.TxPropagationRestTemplateInterceptor.interc
ept(TxPropagationRestTemplateInterceptor.java:93)
:
```

(2) Spring Framework の WebClient を使用したインタセプタの登録

TxPropagationWebClientFactory クラスの webClientFilter()メソッドまたはそのメソッドの戻り値を、org.springframework.web.reactive.function.client.WebClient.Builder クラスの filter() メソッドの引数として指定して、org.springframework.web.reactive.function.client.WebClient クラスのインスタンスを生成してください。

(a) ライブラリの組み込み

TxPropagationWebClientFactory クラスを登録するには、次の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。

- hmppcto-propagation-webclient
- hmppcto-common

Gradle で組み込む記述例を次に示します。

```
dependencies {
    implementation 'jp.co.Hitachi.soft.hmppcto:hmppcto-propagation-webclient:<V.R.S>※'
    implementation 'jp.co.Hitachi.soft.hmppcto:hmppcto-common:<V.R.S>※'
}
```

上記のように、hmppcto-propagation-webclient をアプリケーションに組み込むと、リアクティブ分散トランザクションを処理できます。リアクティブ分散トランザクションを使用しない場合は、hmppcto-propagation-webclient を dependencies から除外してください。

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。

(b)　宛先のホスト部分の形式

宛先のホスト部分の形式は、「(b)　宛先のホスト部分の形式 」に記載しています。

以降に、宛先の Service への HTTP 通信に指定するドメイン名の例を示します。

```
Mono<ResponseDto> result = webClient
     .post()
     .uri("https://my-svc.my-namespace.svc:8080/{foo}", "bar")          // 宛先のServiceを示すドメイン名で指定
     .bodyValue(body);
```

(c)　コーディング例

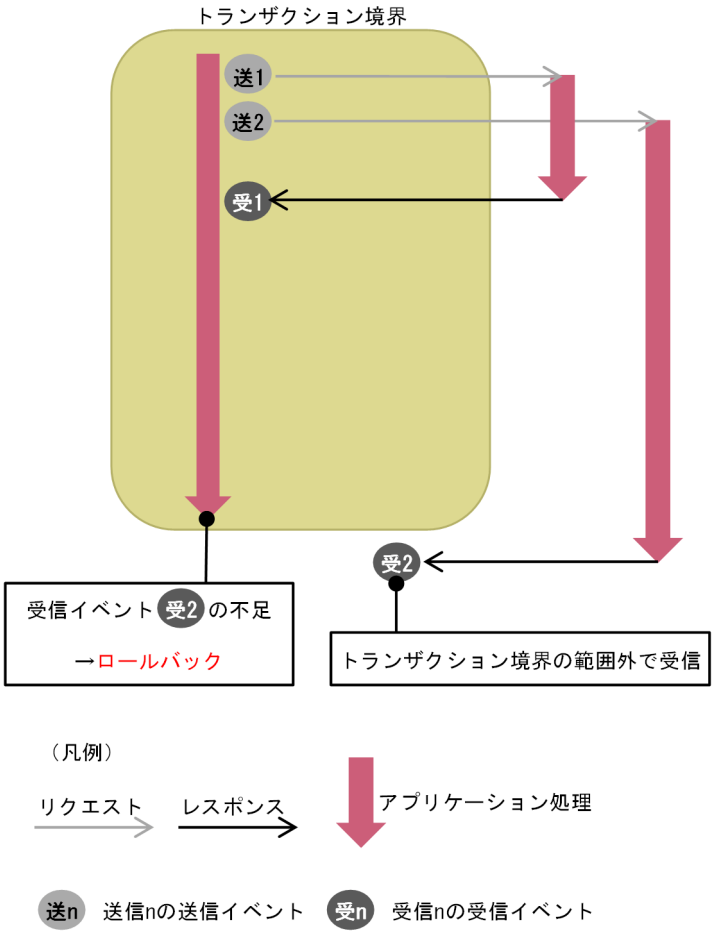
org.springframework.web.reactive.function.client.WebClient.Builder クラスの filter()メソッドの引数として TxPropagationWebClientFactory クラスの webClientFilter()メソッドを指定し、WebClient のインスタンスを生成する例を記載します。アプリケーションが使用する他のインタセプタとの登録の順番に制約はありません。

TxPropagationWebClientFactory 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
TxPropagationWebClientFactory txIc;

WebClient webClient = WebClient.builder()          // WebClientのビルダーを取得
    .filter(txIc.webClientFilter())                // HMP-PCTOのインタセプタを登録
    .build();                                      // WebClientインスタンスを生成
```

WebClient を使用したトランザクションブランチの呼び出しでは、呼び出し元はすべてのトランザクションブランチの応答をトランザクション境界内で受信する必要があります。次の図のように、トランザクション境界内で、トランザクションブランチの呼び出し元が受信していない応答がある場合、トランザクションをロールバックします。

図 2-1　トランザクション境界内で受信していないトランザクションブランチの応答がある場合の動作



図のようにロールバックすることを避けるために、アプリケーションはすべてのトランザクションブランチからの応答を受信するまでスレッドをブロックする必要があります。次に応答受信のコードの一例を示します。この例では Flux.collectList().block()を使用して、トランザクションブランチからの応答を受信するまでスレッドをブロックします。

```
List<String> urls = new ArrayList();
urls.add("https://my-svc.my-namespace-A.svc:8080/bar");
urls.add("https://my-svc.my-namespace-B.svc:8080/foo");

List<Flux<String>> requests = urls.stream()
            .map(uri -> webClient.post().uri(uri).retrieve().bodyToFlux(String.class))
            .toList();

Flux<String> responses = Flux.merge(requests);                  //リクエストのパラレル実行を指定
List<String> resultList = responses.collectList().block();      //リクエストを送信しスレッドをブロック
```

(d)　エラーハンドリング

WebClient を利用した通信時に動作する TxPropagationWebClientFactory クラスの前処理で例外が発生したとき、WebClientException が送出されます。

この場合、例外の Caused には次の例外の情報が出力されます。

- jp.co.Hitachi.soft.hmppcto.lb.exception.ServiceNotFoundException
- jp.co.Hitachi.soft.hmppcto.lb.exception.PodNotFoundException

WebClient 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 TxPropagationWebClientFactory 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(AbstractClientH
ttpResponse.java:33)
 at jp.co.Hitachi.soft.hmppcto.filter.propagation.TxPropagationWebClientFactory.webClientFil
ter(TxPropagationWebClientFactory.java:93)
:
```

(3)　先行プリペア機能の使用

先行プリペア機能を使用するには、トランザクションを伝搬する際にクライアントアプリケーションで 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。

(a)　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、「(a)　ライブラリの組み込み」を参照してください。

(b)　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。ヘッダ名と値の詳細については、「8.2.4　XidHeaders インタフェース（SQL-Participant 限定）」を参照してください。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。

■ Spring Framework の RestTemplate を使用する場合

```
RequestEntity<MyRequest> request = RequestEntity                    // RequestEntity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);
```

■ Spring Framework の WebClient を使用する場合

```
Mono<ResponseDto> result = webClient
     .post()
.uri("https://example.com/{foo}", "bar")
.header(XidHeaders.IS_LAST_CALL, XidHeaders.TRUE)              // 先行プリペアヘッダ追加
     .bodyValue(body);
```

(c) 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。

(4) 注意事項

- 通信先のサーバに HMP-PCTO のフィルタを登録していない場合は、サービスを呼び出した結果にトランザクション制御用ヘッダが存在しない状態になるため、トランザクションをロールバックにマークします。トランザクション外のサービスを呼び出したい場合は、HMP-PCTO のインタセプタを利用しないでサービスを呼び出してください。
- 先行プリペア実行後に、再度同じサービスを呼び出しても SQL は実行できません。
- 先行プリペア実行後に SQL を実行するために JDBC Proxy Driver のメソッドを実行した場合は、PctoPrePreparedException 例外を送出して、SQL を実行しないで無視します。トランザクションがコミットするかロールバックするかには影響しません。アプリケーションで PctoPrePreparedException 例外をキャッチしてロールバックするかどうかを判断してください。
- スケーリング対応機能を使用し、ユーザが配置するロードバランサでルーティングする（set-routing パラメタに「user」を指定する）場合、同一トランザクション内で同一サービス（URI が異なる場合を含む）を複数回呼び出すときは、どの通信でも同一 Pod に対して通信をするようにアプリケーションを実装し、ロードバランサを設定してください。
- スケーリング対応機能を使用する場合、次のどちらのときでも、トランザクションの伝搬時の HTTP 通信に指定する宛先のホストには「localhost」を指定しないでください。
  - HMP-PCTO が提供するルーティング機能を利用する（set-routing パラメタに「inner」を指定する）とき
  - ユーザが配置するロードバランサでルーティングする（set-routing パラメタに「user」を指定する）とき
- グローバルトランザクションの範囲内での WebClient による同一サービスのパラレルな呼び出しをしないでください。

2.2.9 通信によるトランザクションブランチの作成（REST 通信-サーバ）

REST 通信によるトランザクションブランチを作成するサーバの開発方法について説明します。

REST 通信によるトランザクションブランチの作成は、次のアプリケーションで実装してください。

- Relay-Service
- Entity-Service(SQL)
- Entity-Service(TCC)
- Alternate-Service

- Alternate-Service(SQL)

サーバでは、HMP-PCTO のフィルタを登録してトランザクションを伝搬する必要があります。フィルタの登録はライブラリの組み込みを行うことで、自動的に行われます。

また、トランザクションブランチを作成することで、通信によるトランザクションブランチのロールバック条件を設定できます。

ライブラリの組み込み方法とトランザクションブランチのロールバック条件の設定方法を次に示します。

(1) フィルタの登録

(a) ライブラリの組み込み

フィルタを登録するには次の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。

- hmppcto-entity-service
- hmppcto-server-http-filter
- jakarta.ws.rs-api

Gradle で組み込む記述例を次に示します。

```
dependencies {
    implementation 'jp.co.Hitachi.soft.hmppcto:hmppcto-entity-service:<V.R.S>※1'
    implementation 'jp.co.Hitachi.soft.hmppcto:hmppcto-server-http-filter:<V.R.S>※1'
    implementation 'jakarta.ws.rs:jakarta.ws.rs-api:<rs-apiのV.R.S>※2'
}
```

注※1

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

注※2

*<rs-api の V.R.S>*に指定するバージョンは、リリースノートの「Java ライブラリ」の「jakarta.ws.rs:jakarta.ws.rs-api」を参照してください。

(2) トランザクションブランチのロールバック条件の設定

@PctoBranchRollbackFor アノテーションを使用して、通信によるトランザクションブランチのロールバック条件を設定する方法を説明します。@PctoBranchRollbackFor アノテーションの API 仕様の詳細は「8.2.13　@PctoBranchRollbackFor アノテーション」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は、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

トランザクションブランチのロールバック条件を設定する場合に組み込むライブラリは、トランザクションブランチの作成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、「(a)　ライブラリの組み込み」を参照してください。

(b)　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ロールバック条件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

}
```

(c)　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。

トランザクションブランチ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、トランザクションブランチでトランザクションをロールバックしたい場合は、トランザクション境界を設定した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。

- トランザクションブランチ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

(d)　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 アノテーションを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 レスポンスを設定してください。

2.2.10　通信によるトランザクションブランチの作成（gRPC 通信-クライアント）

gRPC 通信によるトランザクションブランチを作成するクライアントの開発方法について説明します。

gRPC 通信によるトランザクションブランチの作成は、次のアプリケーションで実装してください。

- Orchestrator-Service
- Relay-Service
- Entity-Service(SQL)
- Entity-Service(TCC)
- Orchestrator-Service(SQL)
- Alternate-Service
- Alternate-Service(SQL)

クライアントでは、インタセプタを追加してトランザクションを伝搬する必要があります。また、トランザクションブランチを作成することで、先行プリペア機能を使用できます。

インタセプタの追加方法と、先行プリペア機能の使用方法を次に示します。

(1)　gRPC の ManagedChannel を使用したインタセプタの追加

ManagedChannelBuilder、またはそのサブクラスに HMP-PCTO が提供する TxPropagationGrpcClientInterceptor クラスを追加して使用します。

また、分散トレースを取得するため OpenTelemetry のインタセプタを HMP-PCTO の GrpcTraceInterceptorFactory クラスを使用して取得し、ManagedChannelBuilder またはそのサブクラスに追加してください。OpenTelemetry のインタセプタは、OpenTelemetry のクラスを直接使用して取得しないでください。直接取得した場合は、HMP-PCTO の初期化処理と処理順番が不定となり、Spring Boot の初期化処理中に例外が発生して、アプリケーションの起動に失敗することがあります。

スケーリング対応機能を使用する場合、スキーム名を HMP-PCTO の TxPropagationGrpcNameResolverProvider クラスから、ロードバランシングポリシーを HMP-PCTO の TxPropagationGrpcLoadBalancerProvider クラスからそれぞれ取得して ManagedChannelBuilder、またはそのサブクラスに設定してください。

(a)　ライブラリの組み込み

ManagedChannelWrapper クラスを使用するには、次の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。

- hmppcto-propagation-grpc
- hmppcto-common

Gradle で組み込む記述例を次に示します。

```
dependencies {
    implementation 'jp.co.Hitachi.soft.hmppcto:hmppcto-propagation-grpc:<V.R.S>※'
    implementation 'jp.co.Hitachi.soft.hmppcto:hmppcto-common:<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。

(b)　宛先のホスト部分の形式

トランザクションの伝搬時、gRPC 通信に指定する宛先のホスト部分は、宛先の Service を示すドメイン名で指定してください。詳細については「(b)　宛先のホスト部分の形式 」の形式を参照してください。

(c)　コーディング例

io.grpc.ManagedChannelBuilder クラスを使用した例を記載します。

■ スケーリング対応機能を使用しない場合のコーディング例

```
@Autowired
TxPropagationGrpcClientInterceptor grpcClientInterceptor;

@Autowired
GrpcTraceInterceptorFactory interceptorFactory;

String host = "grpc-service:";  // 送信先ホスト
int port = 50051;               // 送信先ポート

ManagedChannelBuilder<?> chBuilder = ManagedChannelBuilder.forAddress(host, port); // Channe
lBuilder作成
ManagedChannel ch = chBuilder.usePlaintext()    // ManagedChannel作成
                    // トランザクション伝搬用インタセプタ追加
                    .intercept(grpcClientInterceptor)
                    // 分散トレース取得用インタセプタ追加
                    .intercept(interceptorFactory.getClientInterceptor())
.build();
```

■ スケーリング対応機能を使用する場合のコーディング例

```
@Autowired
TxPropagationGrpcClientInterceptor grpcClientInterceptor;

@Autowired
GrpcTraceInterceptorFactory interceptorFactory;

String host = "grpc-service";  // 送信先ホスト
String port = "50051";           // 送信先ポート
String scheme = TxPropagationGrpcNameResolverProvider.SCHEME_SCALING;    // スキーム名
String targetStr = scheme + ":///" + host + ":" + port;  // スキーム名 + 送信先ホスト + 送信
先ポート
```

```
ManagedChannelBuilder<?> chBuilder = ManagedChannelBuilder.forTarget(targetStr) // ChannelBu
ilder作成
                            // ロードバランシングポリシー設定
                            .defaultLoadBalancingPolicy(TxPropagationGrpcLoadBalancerProvide
r.POLICY_SCALING);
ManagedChannel ch = chBuilder.usePlaintext()    // ManagedChannel作成
                    // トランザクション伝搬用インタセプタ追加
                    .intercept(grpcClientInterceptor)
                    // 分散トレース取得用インタセプタ追加
                    .intercept(interceptorFactory.getClientInterceptor())
                    .build();
```

(d) エラーハンドリング

チャネル作成時に HMP-PCTO が障害を検知すると、IllegalArgumentException または IllegalStateException を送出します。

また、gRPC 通信時に HMP-PCTO が障害を検知すると、gRPC ステータス INTERNAL を設定した StatusRuntimeException を送出します。必要に応じて例外を catch して後処理をしてください。

- 通信障害以外の障害の原因として、次のことが考えられます。
  - スキーム名の指定誤り
  - ロードバランシングポリシー名の指定誤り
  - アプリケーションが指定した通信先 Service の誤り
  - 通信先 Service 名の指定誤り
  - 通信先 Service 名と Pod の対応の指定誤り

スケーリング対応機能を使用する場合に、誤ったスキーム名とロードバランシングポリシー名を指定したときの動作を次の表に示します。

表 2-8 スケーリング対応機能を使用する場合に誤ったスキーム名とロードバランシングポリシー名を指定したときの動作

項番	障害を検知する契機	スキーム名	ロードバランシングポリシー名	動作
1	チャネル作成時	誤ったスキーム名を指定する	TxPropagationGrpcLoadBalancerProvider クラスから取得したロードバランシングポリシー名を指定する	IllegalArgumentException を送出します。
2		TxPropagationGrpcNameResolverProvider クラスから取得したスキーム名を指定する	誤ったロードバランシングポリシー名を指定する	IllegalStateException を送出します。
3	gRPC 通信時	指定しない	TxPropagationGrpcLoadBalancerProvider クラスから取得したロードバランシングポリシー名を指定する	gRPC ステータス INTERNAL を設定した StatusRuntimeExc

項番	障害を検知する契機	スキーム名	ロードバランシングポリシー名	動作
				eption を送出します。
4		TxPropagationGrpcNameResolverProvider クラスから取得したスキーム名を指定する	指定しない	gRPC ステータス INTERNAL を設定した StatusRuntimeException を送出します。

スケーリング対応機能を使用しない場合に、誤ったスキーム名とロードバランシングポリシー名を指定したときの動作を次の表に示します。

表 2-9　スケーリング対応機能を使用しないでスキーム名とロードバランシングポリシー名を指定したときの動作

項番	障害を検知する契機	スキーム名	ロードバランシングポリシー名	動作
1	チャネル作成時	TxPropagationGrpcNameResolverProvider クラスから取得したスキーム名を指定する	TxPropagationGrpcLoadBalancerProvider クラスから取得したロードバランシングポリシー名を指定する	IllegalArgumentException を送出します。
2	gRPC 通信時	指定しない	TxPropagationGrpcLoadBalancerProvider クラスから取得したロードバランシングポリシー名を指定する	gRPC ステータス INTERNAL を設定した StatusRuntimeException を送出します。

アプリケーションが指定した通信先 Service の誤り、通信先 Service 名の指定誤り、または通信先 Service 名と Pod の対応の指定の誤りの場合、次の動作をします。

- gRPC ステータス INTERNAL を設定した StatusRuntimeException を送出します。

アプリケーションが指定した通信先 Service の誤り、通信先 Service 名の指定誤りの場合、gRPC の ManagedChannel の制御によってエラーメッセージが繰り返し出力されることがあります。

(2)　先行プリペア機能の使用

先行プリペア機能を使用するには、トランザクションを伝搬する際にクライアントアプリケーションで HMP-PCTO が提供するインタフェースを使用して、先行プリペア機能を使用するかどうかを切り替えます。

(a) 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、「(a) ライブラリの組み込み」を参照してください。

(b) コーディング例

.proto ファイルの設定が次の場合に、同じサービスを複数回呼び出す際の例を示します。

.proto ファイルに定義したサービス名	.proto ファイルに定義したメソッド名
SampleService	sampleMethod1
	sampleMethod2

同じサービスを複数回呼び出す際に、サービスの呼び出し直前に先行プリペア機能の使用有無を切り替える方法と、サービスの呼び出し前後で先行プリペア機能の使用有無を切り替える方法があります。

ここでは、サービス呼び出し前後で先行プリペア機能の使用有無を切り替える場合の例を記載します。メソッドの詳細については、「8.2.4 XidHeaders インタフェース（SQL-Participant 限定）」を参照してください。

```
SampleServiceBlockingStub serviceStub = SampleServiceGrpc.newBlockingStub(ch);   // Stub作成

serviceStub.sampleMethod1();                                  // 先行プリペア機能を使用しないで
サービス呼び出し

TxPropagationOptionalSettings.setPrePrepared(true);      // 先行プリペア機能を「使用する」に
設定
serviceStub.sampleMethod2();                                  // 先行プリペア機能を使用してサービ
ス呼び出し
TxPropagationOptionalSettings.setPrePrepared(false);     // 先行プリペア機能を「使用しない」
に設定
```

(c) エラーハンドリング

先行プリペア機能を使用するかどうかを切り替えれば、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。

(3) 注意事項

- 通信先のサーバに HMP-PCTO のインタセプタを追加していない場合は、サービスを呼び出した結果にトランザクション制御用ヘッダが存在しない状態になるため、トランザクションをロールバックにマークします。トランザクション外のサービスを呼び出したい場合は、HMP-PCTO のインタセプタを利用しないでサービスを呼び出してください。
- 先行プリペア機能を使用してサービスを呼び出したあとに、同じトランザクションブランチのサービスを呼び出す場合、呼び出し先のサービスで SQL は実行できません。SQL を実行するために JDBC Proxy Driver のメソッドを実行した場合は、PctoPrePreparedException 例外を送出して、SQL を実行しないで無視します。トランザクションがコミットするかロールバックするかには影響しません。呼び出し

先のサービスで PctoPrePreparedException 例外をキャッチしてロールバックするかどうかを判断してください。

2.2.11　通信によるトランザクションブランチの作成（gRPC 通信-サーバ）

gRPC 通信によるトランザクションブランチを作成するサーバの開発方法について説明します。

gRPC 通信によるトランザクションブランチの作成は、次のアプリケーションで実装してください。

- Relay-Service
- Entity-Service(SQL)
- Entity-Service(TCC)
- Alternate-Service
- Alternate-Service(SQL)

サーバでは、HMP-PCTO のインタセプタを追加してトランザクションを伝搬する必要があります。また、トランザクションブランチを作成することで、通信によるトランザクションブランチのロールバック条件を設定できます。

グレースフルシャットダウンをサポートするアプリケーションでは、グレースフルにシャットダウンする実装にしてください。

インタセプタの追加方法、トランザクションブランチのロールバック条件の設定方法、およびグレースフルシャットダウンの実装例を次に示します。

(1)　インタセプタの追加

(a)　ライブラリの組み込み

TxPropagationGrpcServerInterceptor クラスを追加するには、次の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。

- hmppcto-entity-service
- hmppcto-server-grpc-interceptor
- hmppcto-common

Gradle で組み込む記述例を次に示します。

```
dependencies {
    implementation 'jp.co.Hitachi.soft.hmppcto:hmppcto-entity-service:<V.R.S>※'
    implementation 'jp.co.Hitachi.soft.hmppcto:hmppcto-server-grpc-interceptor:<V.R.S>※'
    implementation 'jp.co.Hitachi.soft.hmppcto:hmppcto-common:<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。

また、分散トレースを取得するため OpenTelemetry のインタセプタを HMP-PCTO の GrpcTraceInterceptorFactory クラスを使用して取得し、gRPC サーバに追加してください。OpenTelemetry のインタセプタを OpenTelemetry のクラスを使用して直接取得しないでください。直接取得した場合は、HMP-PCTO の初期化処理と処理の順番が不定となり、Spring Boot の初期化処理中に例外が発生して、アプリケーションの起動に失敗することがあります。

(b) コーディング例

TxPropagationGrpcServerInterceptor 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。

次に示す条件の場合に io.grpc.ServerBuilder クラスの intercept()メソッドを使用して追加する例を記載します。

設定項目	設定値
サービスクラス名	sampleService
サービス受信ポート番号	50051

```
@Autowired
TxPropagationGrpcServerInterceptor grpcServerInterceptor;

@Autowired
GrpcTraceInterceptorFactory  interceptorFactory;

int port = 50051;                    // 受信ポート

ServerBuilder<?> serverBuilder = ServerBuilder.forPort(port)
                                     // サービスの追加
                                     .addService(new sampleService())
                                     // トランザクション伝搬用インタセプタ追加
                                     .intercept(grpcServerInterceptor)
                                     // 分散トレース取得用インタセプタ追加
                                     .intercept(interceptorFactory.getServerInterceptor());

Server grpcServer = serverBuilder.build();  // サーバ作成
grpcServer.start();                         // サーバ起動
```

(c) エラーハンドリング

追加するインタセプタでは、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)　トランザクションブランチのロールバック条件の設定

@PctoGrpcBranchRollbackFor アノテーションを使用して、通信によるトランザクションブランチのロールバック条件を設定する方法を説明します。@PctoGrpcBranchRollbackFor アノテーションの API 仕様の詳細は「8.2.14　@PctoGrpcBranchRollbackFor アノテーション」を参照してください。
@PctoGrpcBranchRollbackFor 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は、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

トランザクションブランチのロールバック条件を設定する場合に組み込むライブラリは、インタセプタの追加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、「(a)　ライブラリの組み込み」を参照してください。

(b)　コーディング例

@PctoGrpcBranchRollbackFor 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ロールバック条件を設定するコーディング例を次に示します。@PctoGrpcBranchRollbackFor 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。

```
@PctoGrpcBranchRollbackFor
public void service() {

    // 業務処理

}
```

(c)　エラーハンドリング

@PctoGrpcBranchRollbackFor アノテーションを設定すれば、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。

(3)　グレースフルシャットダウンの実装例

gRPC サーバとして動作するアプリケーションをグレースフルにシャットダウンする実装例を次に示します。

```
@PreDestroy
public void shutdown() {
    server.shutdown();
    try {
        if (!server.awaitTermination(15, TimeUnit.SECONDS)) {
            server.shutdownNow();
        }
    } catch (InterruptedException e) {
        Thread.currentThread().interrupt();
        server.shutdownNow();
    }
}
```

2.2.12　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、次のアプリケーションで実装してください。

- Entity-Service(SQL)
- Orchestrator-Service(SQL)
- Alternate-Service(SQL)

JDBC Driver のメソッドの実行結果が java.sql のインタフェースを実装したクラスである場合、JDBC Proxy Driver のメソッドを実行する型の定義は、java.sql のインタフェースを使用することをおすすめします。理由としては、java.sql を実装した HMP-PCTO のクラスには独自のメソッドがないため、HMP-PCTO のクラスを使用するよりも、java.sql のインタフェースを使用する方が移植性が高い実装ができるからです。

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、「9.　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 で組み込む記述例を次に示します。

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

また、通常版の場合、HMP-PCTO が 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 形式を作成する記述例を次に示します。

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

トライアル版の場合は、Launcher クラスの変更は不要です。

(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 からアプリケーションに送出します。

Participant のトランザクションブランチを開始できない場合など、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、「8. 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）。

2.2.13 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。

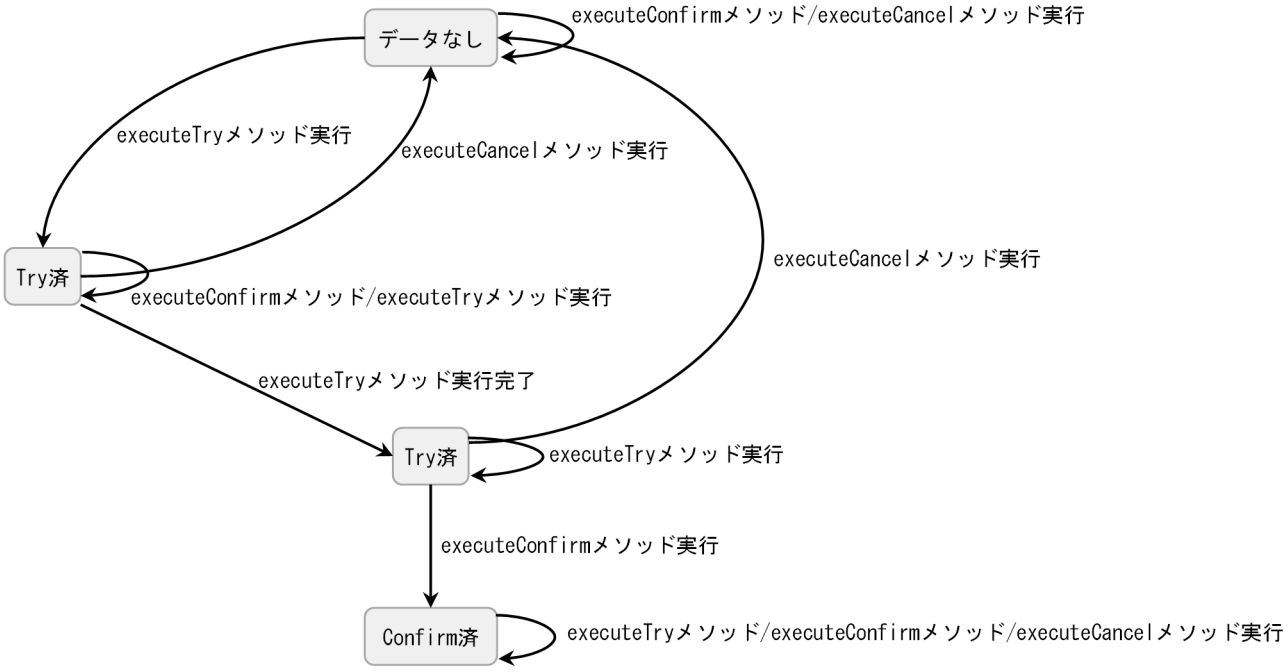
TccService インタフェースの実装クラスの作成は、Entity-Service(TCC)のアプリケーションで実装してください。

TccService インタフェースの詳細については、「8.4.5　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 2-10　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 2-2　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。

ライブラリの組み込み方法は、トライアル版と通常版で差異があります。トライアル版で使用したアプリケーションを通常版でも使用する場合は、通常版の説明に従って修正してください。

Gradle で組み込む記述例を次に示します。

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

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 形式を作成する記述例を次に示します。

```
bootJar {
    manifest {
        attributes 'Main-Class' : 'org.springframework.boot.loader.launch.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、「8.4.4　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 内のほかの 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。

2.2.14　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、Entity-Service(TCC)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 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 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 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。

2.2.15　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、次のアプリケーションで実装してください。

- Orchestrator-Service
- Relay-Service
- Entity-Service(SQL)
- Entity-Service(TCC)
- Orchestrator-Service(SQL)
- Alternate-Service
- Alternate-Service(SQL)

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<String> 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。

2.2.16　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。詳細については、「(4)　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。

次にカスタマイズ例を示します。

(1)　メッセージログをファイルに出力するためのカスタマイズ例

メッセージログをファイルに出力するためのカスタマイズ例および出力手順を次に示します。

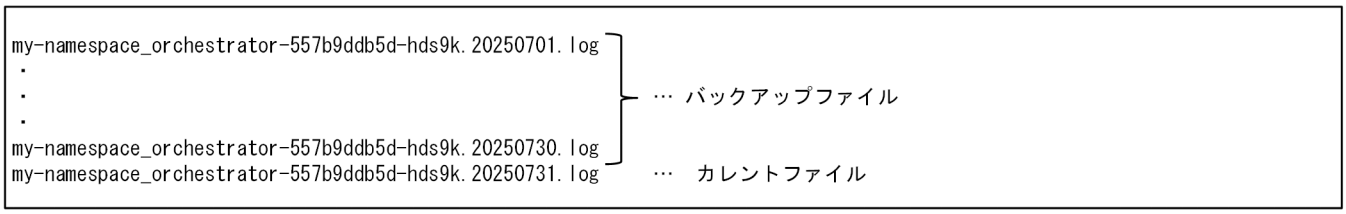
(a)　logback-spring.xml の作成

メッセージログをファイルとして出力する場合、logback-spring.xml として次の設定で定義を作成する必要があります。設定の詳細や変更方法については、Logback の公式ドキュメントを参照してください。

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE logback>
<configuration>
<conversionRule conversionWord="hostname" class="jp.co.Hitachi.soft.hmppcto.common.log.HostN
ameConverter"/>

<!-- rootロガー用 変更可 -->
<appender name="STDOUT" class="ch.qos.logback.core.ConsoleAppender">
    <target>System.out</target>
    <encoder>
        <pattern>%d{yyyy-MM-dd HH:mm:ss.SSS} %-5level %hostname %thread %class{1} ${PID:- }
%msg%n</pattern>
        <charset>UTF-8</charset>
    </encoder>
</appender>

<!-- ファイル出力 -->
<appender name="FILE" class="ch.qos.logback.core.rolling.RollingFileAppender">
    <rollingPolicy class="ch.qos.logback.core.rolling.TimeBasedRollingPolicy">
        <fileNamePattern>/deployments/logs/${HMPPCTO_POD_NAMESPACE}_${HMPPCTO_POD_NAME}.%d{y
yyyMMdd}.log</fileNamePattern>
        <maxHistory>30</maxHistory>
```

```
    </rollingPolicy>
    <encoder>
        <pattern>%d{yyyy-MM-dd HH:mm:ss.SSS} %-5level %hostname %thread %class{1} ${PID:- }
%msg%n</pattern>
        <charset>UTF-8</charset>
    </encoder>
</appender>

<!-- ロガー名PCTO用 の設定-->
<include resource="hmppcto-include.xml"/>
<logger name="HMPPCTO_STDOUT" level="INFO" additivity="false">
    <appender-ref ref="HMPPCTO_STDOUT" />
    <appender-ref ref="FILE" />
</logger>
<logger name="HMPPCTO_STDERR" level="ERROR" additivity="false">
    <appender-ref ref="HMPPCTO_STDERR" />
    <appender-ref ref="FILE" />
</logger>

<!-- rootロガー -->
<root level="INFO">
    <appender-ref ref="STDOUT" />
    <appender-ref ref="FILE" />
</root>
</configuration>
```

この設定によって、HMP-PCTO のメッセージログおよび Logback を使用したアプリケーションのメッセージがコンテナ内の/deployments/logs 配下にログファイル「<namespace 名>_<Pod 名>.yyyymmdd.log」として出力されます。また、日単位でログファイルがローテーションされ、現在日から最大で過去 30 日分のバックアップファイル「<namespace 名>_<Pod 名>.yyyymmdd.log」が作成されます。作成されるログファイルの例を次示します。



重要

- この設定をした場合、必ず Logback を使用してログを出力してください。Logback を使用しないでログを出力した場合、ログファイルに保存されません。(例：java.lang.Throwable.printStackTrace()による例外情報およびスタックトレースの出力)
- ログファイルはログの出力された時刻を判定し、カレントファイルの作成およびファイルのローテーションを実施します。そのため、日単位でログの出力がない場合、該当日のファイルは作成されず、ファイルのローテーションは実施されません。そのため、バックアップファイルの日付が連続していない場合があります。
- この設定をした場合、メッセージログファイル名は namespace 名および Pod 名を使用した、Pod ごとに一意のファイル名になります。出力ファイル名が Pod ごとに一意になると、

Pod が停止または再起動した場合、停止前に使用されていた出力ファイルの上書きや削除がされません。そのため、定期的に外部ツールなどで移動、削除などの運用をしてください。

(b) logback-spring.xml の配置

作成した logback-spring.xml を HMP-PCTO の各 Kubernetes アプリケーションのリソースディレクトリ「src/main/resources」の直下に格納してください。Mediator については、「Mediator 用 Java プロジェクトディレクトリ」（java-projects/mediator）配下にリソースディレクトリ「src/main/resources」を作成し、logback-spring.xml を格納してください。

また、Hitachi Microservices Platform - Bridge Option for OpenTP1 を使用している場合は、「TP1-Bridge 用 Java プロジェクトディレクトリ」（java-projects/tp1-bridge）配下にリソースディレクトリ「src/main/resources」を作成し、logback-spring.xml を格納してください。

(c) メッセージログのファイル出力用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

＜永続ボリュームの作成要領＞

- 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.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)ログファイル1つ当たりのファイルサイズ
(b)ログファイルの世代数（TimeBasedRollingPolicyのmaxHistoryプロパティ設定値＋1）
(c)メッセージログをNFSサーバに出力するPod数
(d)エラーなどでメッセージログが継続して出力される場合を想定した安全率（200%）
```

- 例として、(a)が「10Mi」の場合、「10Mi×31×10×200%」＝「6200Mi」以上となるため、.spec.capacity.storage フィールドには「7Gi」（7 ギビバイト）を指定します。この計算式による値は目安としていただき、実際に必要とするサイズについては、実測するようお願いします。
- 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 Pod が、どのノードからでも永続ボリュームにメッセージログをログファイルに出力できるようになります。
- 永続ボリュームの種類が NFS ストレージの場合は、永続ボリュームの.spec.nfs フィールドに、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.persistentVolumeReclaimPolicy フィールドに、「Retain」を指定してください。

＜永続ボリューム要求の作成要領＞

- 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。

- 各永続ボリューム要求の Namespace（.metadata.namespace フィールド）に、Pod で使用している Namespace 名を指定してください。
- 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.resources.requests.storage フィールドに、前述の永続ボリュームで指定したサイズを指定してください。

<永続ボリュームおよび永続ボリューム要求の記述例>

メッセージログのファイル出力用永続ボリュームおよび永続ボリューム要求の Kubernetes マニフェストの記述例を次に示します。

- 永続ボリュームを NFS ストレージとする場合

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-pod-logs
  namespace: my-namespace
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 7Gi
---
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-pod-logs
  namespace: my-namespace
spec:
  capacity:
    storage: 7Gi
  accessModes:
    - ReadWriteMany
  nfs:
    server: nfs-server.example.com
    path: /nfs/logs
  persistentVolumeReclaimPolicy: Retain
```

(d) メッセージログのファイル出力用永続ボリュームおよび永続ボリューム要求のデプロイ

メッセージログのファイル出力用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。

コマンド例を次に示します。

```
kubectl apply -f ./ pod-logs-volume.yaml
```

この例では、Kubernetes マニフェストのファイル名を「pod-logs-volume.yaml」としています。

＜コマンドの実行結果の例＞

```
persistentvolume/pv-pod-logs 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。

メッセージログのファイル出力用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、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
pv-pod-logs    7Gi        RWX            Retain           Available                                   9s
```

上記の例では、メッセージログのファイル出力用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。

(e) HMP-PCTO の Kubernetes マニフェストのカスタマイズ

HMP-PCTO の各 Kubernetes マニフェストに次の内容を追加してください。

- メッセージログのファイル出力用永続ボリュームおよび永続ボリューム要求の設定

  次の設定によって、メッセージログのファイル出力用永続ボリュームが/deployments/logs としてマウントされます。

  - ベースディレクトリとして/deployments/logs をマウントするため、.spec.template.spec.containers.volumeMounts に次の値を設定してください。

    ```
            - name: nfs-logs-volume
              mountPath: /deployments/logs
    ```

  - 永続ボリュームを/deployments/logs としてマウントするため、.spec.template.spec.volumes に次の値を設定してください。

    ```
          - name: nfs-logs-volume
            persistentVolumeClaim:
              claimName: pvc-pod-logs
    ```

- メッセージログのファイル名に使用する環境変数の設定

  logback-spring.xml ファイルでメッセージログのファイル名に使用する環境変数「HMPPCTO_POD_NAMESPACE」および「HMPPCTO_POD_NAME」を設定します。

- Pod が動作しているネームスペースを環境変数として設定するため、.spec.template.spec.containers.env に次の値を設定してください。

```
        - name: HMPPCTO_POD_NAMESPACE
          valueFrom:
            fieldRef:
              fieldPath: metadata.namespace
```

- Pod 名称を環境変数として設定するために.spec.template.spec.containers.env に次の値を設定してください。

```
        - name: HMPPCTO_POD_NAME
          valueFrom:
            fieldRef:
              fieldPath: metadata.name
```

(f) HMP-PCTO の Helm チャートの修正

Mediator の Helm チャートへメッセージログのファイル出力用永続ボリュームおよびストレージクラスの設定および、メッセージログのファイル名に使用する環境変数の設定する際の手順を記載します。また、Hitachi Microservices Platform - Bridge Option for OpenTP1 を使用している場合は、TP1-Bridge の Helm チャートを同様に修正してください。

1. Helm チャートの展開

   作業ディレクトリで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 mediator-<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。

2. statefulset.yaml ファイルの修正

   Helm チャートは次の構成で展開されます。templates フォルダ内の statefulset.yaml ファイルを開きます。

ファイル／ディレクトリ構成			
<作業ディレクトリ>/			
	mediator		
		.helmignore	
		Chart.yaml	
		values.schema.json	
		values.yaml	
		templates	
			includes
			mediator-config-map.yaml

			service.yaml
			serviceaccount.yaml
			statefulset.yaml
			ucars-config-map.yaml

3. メッセージログのファイル出力用永続ボリュームおよびストレージクラスの設定

   statefulset.yaml ファイルに次の設定を追加します。これによって、メッセージログのファイル出力用永続ボリュームが/deployments/logs としてマウントされます。

   - ベースディレクトリとして/deployments/logs をマウントするため、.spec.template.spec.containers.volumeMounts に次の値を設定してください。

     ```
             - name: nfs-logs-volume
               mountPath: /deployments/logs
     ```

   - 永続ボリュームを/deployments/logs としてマウントするため、.spec.template.spec.volumes に次の値を設定してください。

     ```
           - name: nfs-logs-volume
             persistentVolumeClaim:
               claimName: pvc-pod-logs
     ```

4. メッセージログのファイル名に使用する環境変数の設定

   statefulset.yaml ファイルに次の設定を追加します。これによって logback-spring.xml ファイルで、メッセージログのファイル名に使用する環境変数「HMPPCTO_POD_NAMESPACE」および「HMPPCTO_POD_NAME」を設定します。

   - Pod が動作しているネームスペースを環境変数として設定するため、.spec.template.spec.containers.env に次の値を設定してください。

     ```
             - name: HMPPCTO_POD_NAMESPACE
               valueFrom:
                 fieldRef:
                   fieldPath: metadata.namespace
     ```

   - Pod 名称を環境変数として設定するために.spec.template.spec.containers.env に次の値を設定してください。

     ```
             - name: HMPPCTO_POD_NAME
               valueFrom:
                 fieldRef:
                   fieldPath: metadata.name
     ```

5. パッケージの再作成

   statefulset.yaml ファイルの修正が完了したら、作業ディレクトリへ移動します。次のコマンドを実行し、修正した Helm チャートを再度パッケージ化します。作成された Helm チャートを使用して Mediator をデプロイしてください。

   ```
   helm package ./mediator
   ```

2.2.17　分散トレース機能の設定

分散トレース機能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。

分散トレース機能の設定は、次のアプリケーションで実装してください。

- Orchestrator-Service
- Relay-Service
- Entity-Service(SQL)
- Entity-Service(TCC)
- Orchestrator-Service(SQL)
- Alternate-Service
- Alternate-Service(SQL)

(1)　ライブラリの組み込み

分散トレース機能を使用する場合、使用する分散トレース収集アプリケーションまたはサービスによって異なるライブラリを組み込む必要があります。

(a)　Jaeger および OpenTelemetry Collector を使用する場合

分散トレース収集サービスに Jaeger を使用する場合、分散トレースを OTLP（HTTP）で送信するため、hmppcto-observability-modules-otlp-http 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。また、分散トレースの永続化サービスに New Relic を使用する場合や、確率的サンプリング、テイルベースサンプリングなどを実施する場合も、OpenTelemetry Collector を使用して分散トレースを収集・転送する構成となるため、同様に hmppcto-observability-modules-otlp-http ライブラリを組み込みます。Gradle で組み込む記述例を次に示します。

```
dependencies {
    implementation 'jp.co.Hitachi.soft.hmppcto:hmppcto-observability-modules-otlp-http:<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。

(b)　Google Cloud Platform の Cloud Trace を使用する場合

分散トレース収集サービスに Cloud Trace を使用する場合、hmppcto-observability-modules-gcp 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 で組み込む記述例を次に示します。

```
dependencies {
    implementation 'jp.co.Hitachi.soft.hmppcto:hmppcto-observability-modules-gcp:<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。

(2) ライブラリの使用優先順位

分散トレースは 1 つの分散トレース収集アプリケーションまたはサービスだけ使用できます。複数のライブラリがアプリケーションに組み込まれた場合、次のライブラリの優先順位によって、使用するライブラリを決定します。

表 2-11 分散トレース機能で使用されるライブラリの優先順位

優先順位	ライブラリ
1	hmppcto-observability-modules-otlp-http
2	hmppcto-observability-modules-gcp

2.2.18 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。

2.2.19 アプリケーションが出力するメッセージの文字コード

アプリケーションで標準出力および標準エラー出力にメッセージを出力する場合、文字コードは UTF-8 を指定してください。標準出力および標準エラー出力には、アプリケーションが出力するメッセージと、文字コードが UTF-8 の HMP-PCTO のメッセージが出力されるため、アプリケーションでそのほかの文字コードを使用した場合に複数の文字コードが混在してしまい、ASCII の範囲外の文字が文字化けします。

コンテナイメージに環境変数 LANG および LC_ALL を設定することで、コンテナ内のロケールを UTF-8 に変更します。利用するアプリケーションが必要とするロケールに応じて、適切な値（例：ja_JP.UTF-8）に変更してください。

2.2.20 Spring Boot の組み込み HTTP サーバの設定

(1) ライブラリの組み込み

アプリケーションで Spring Boot の組み込み HTTP サーバを設定してください。使用する組み込み HTTP サーバによって異なるライブラリを組み込む必要があります。

Spring Boot の組み込み HTTP サーバはデフォルトでは Tomcat が使用されます。

(a) Tomcat を使用する場合

アプリケーションで Spring Boot の組み込み HTTP サーバに Tomcat を使用する場合、hmppcto-server-tomcat 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 で組み込む記述例を次に示します。

```
dependencies {
    implementation 'org.springframework.boot:spring-boot-starter-web:<v.r.s>※1'
    implementation 'jp.co.Hitachi.soft.hmppcto:hmppcto-server-tomcat:<V.R.S>※2'
}
```

注※1

v.r.s は使用する Spring Boot 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。

(b) Undertow を使用する場合

アプリケーションで Spring Boot の組み込み HTTP サーバに Undertow を使用する場合、Undertow の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 で組み込む記述例を次に示します。

```
// 依存関係からtomcatを除外
configurations.all {
        exclude group: 'org.springframework.boot', module: 'spring-boot-starter-tomcat'
}

dependencies {
    implementation 'org.springframework.boot:spring-boot-starter-web:<v.r.s>※'
    implementation 'org.springframework.boot:spring-boot-starter-undertow:<v.r.s>※' // Unde
rtowの追加
}
```

注※

v.r.s は使用する Spring Boot のバージョンに合わせて読み替えてください。

(2) メトリクスの設定

Spring Boot の組み込み HTTP サーバのメトリクスを確認することで、HTTP サーバのリクエストやスレッドの状態を確認できます。必要に応じて設定してください。

重要

Spring Boot Actuator を使用する場合

Spring Boot Actuator を使用した場合、Spring Boot Actuator の Prometheus 形式でメトリクスを公開しているエンドポイントに、HMP-PCTO で取得している独自メトリクスも含まれます。そのため、HMP-PCTO としてトラブルシュートに必要な次のメトリクスは、フィルタリングしないで収集してください。

- hmppcto_で始まる名称のメトリクス
- 「表 D-3　HMP-PCTO の性能を評価するための指標」に記載されているメトリクス

(a)　Tomcat を使用する場合

Spring Boot Actuator を使用して、Spring Boot の組み込み HTTP サーバである Tomcat について、リクエストの状態を確認できます。必要に応じて設定してください。

また、HMP-PCTO では hmppcto-server-tomcat ライブラリをアプリケーションに組み込むことで、Spring Boot の組み込み HTTP サーバである Tomcat についての詳細なメトリクス（接続数、スレッド数など）を収集できます。Tomcat のメトリクスを収集する場合は jp.co.Hitachi.soft.hmppcto.metrics.collection.tomcat パラメタに true を設定してください。

(b)　Undertow を使用する場合

Spring Boot Actuator を使用して、Spring Boot の組み込み HTTP サーバである Undertow について、リクエストの状態を確認できます。必要に応じて設定してください。

HMP-PCTO では、Spring Boot の組み込み HTTP サーバである Undertow についての詳細なメトリクス（接続数、スレッド数など）を収集できません。

2.3　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。Red Hat OpenShift Container Platform を使用する場合は、適宜「docker」コマンドを「podman」コマンドに読み替えてください。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。

重要

コンテナイメージのビルドに出力されるメッセージについて

Docker を使用してコンテナイメージのビルドを行った場合、次のメッセージが出力されることがあります。このメッセージは出力されても問題ないため、無視してください。

「java コマンド」の引数に「環境変数 HMPPCTO_JAVA_OPTIONS の値」を設定するために、Containerfile に shell 形式（例：CMD java -jar /deployments/orchestrator-V.R.S.jar）を指定していることから、このメッセージが出力されます。

```
JSONArgsRecommended: JSON arguments recommended for ENTRYPOINT/CMD to prevent unin
tended behavior related to OS signals
```

2.3.1　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 2-12　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。	2.3.2　事前準備
uCosminexus Application Runtime for Spring Boot コンテナベースイメージの作成（通常版限定）	uCosminexus Application Runtime for Spring Boot を導入するコンテナイメージのベースイメージに利用します。	2.3.3　uCosminexus Application Runtime 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 for Spring Boot を導入しないコンテナイメージのベースイメージに利用します。	2.3.4　JDK コンテナベースイメージの作成
Orchestrator-Service コンテナイメージの作成	－	2.3.5　Orchestrator-Service コンテナイメージの作成
Relay-Service コンテナイメージの作成	－	2.3.6　Relay-Service コンテナイメージの作成

作業項目	備考	参照先
Entity-Service(SQL)コンテナイメージの作成	－	2.3.7　Entity-Service(SQL)コンテナイメージの作成（SQL-Participant 限定）
Entity-Service(TCC)コンテナイメージの作成	－	2.3.8　Entity-Service(TCC)コンテナイメージの作成（TCC-Participant 限定）
Orchestrator-Service(SQL)コンテナイメージ	－	2.3.9　Orchestrator-Service(SQL)コンテナイメージの作成（SQL-Participant 限定）
Alternate-Service コンテナイメージ	－	2.3.10　Alternate-Service コンテナイメージの作成
Alternate-Service(SQL)コンテナイメージ	－	2.3.11　Alternate-Service(SQL)コンテナイメージの作成（SQL-Participant 限定）

（凡例）－：該当しません。

2.3.2　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。

2.3.3　uCosminexus Application Runtime 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 for Spring Boot を導入する場合だけ作成します。

uCosminexus Application Runtime for Spring Boot コンテナベースイメージは、uCosminexus Application Runtime 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 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。

重要

Containerfile は FROM 句で指定したコンテナイメージの状態（作業ディレクトリ、環境変数、USER など）を引き継ぎます。そのため、uCosminexus Application Runtime for Spring Boot コンテナベースイメージを使用する場合は、引き継がれる状態に注意してください。

例として、uCosminexus Application Runtime for Spring Boot コンテナベースイメージで「USER 1001:0」が指定されている場合、uCosminexus Application Runtime for Spring Boot コンテナベースイメージを使用したコンテナイメージでは、Containerfile で明示的に USER を上書きしないかぎり、実行ユーザは 1001 のままとなります。この場合、root 権限が必要な処理を実行すると、権限が不足し処理が失敗します。

uCosminexus Application Runtime 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

「6.3.1 uCosminexus Application Runtime 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 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/ubi9/ubi-minimal:9.5-1742914212 AS builder     …1
RUN microdnf install ¥     …2
        -y ¥
        --setopt=install_weak_deps=0 ¥
        --nodocs ¥
        ncurses ¥
        tar ¥
        cpio ¥
        gzip
WORKDIR /tmp
RUN mkdir ./X64LIN     …3
COPY X64LIN ./X64LIN
RUN chmod +x ./X64LIN/setup
RUN ./X64LIN/setup -f -c HL0R ./
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/ubi9/ubi-minimal:9.5-1742914212     …5
COPY --from=builder /opt /opt     …6
RUN microdnf install ¥     …7
        -y ¥
        --setopt=install_weak_deps=0 ¥
        --nodocs ¥
        dnf ¥
    && dnf install ¥
```

```
        -y ¥
        --setopt=install_weak_deps=0 ¥
        --nodocs ¥
        --allowerasing ¥
        java-21-openjdk-headless ¥     …8
        curl-minimal ¥
        jq ¥
        <uCosminexus Application Runtime for Spring Bootの前提RPMパッケージ>     …9
        :
    && dnf clean all ¥
    && rm -rf /var/cache/dnf/*
ENV PATH=/usr/lib/jvm/java-21-openjdk-21.0.7.0.6-1.el9.x86_64/bin:$PATH     …8
ENV LANG=C.UTF-8     …10
ENV LC_ALL=C.UTF-8

RUN chgrp -R 1001 /opt/hitachi/ucars     …11

USER 1001:0     …12
WORKDIR /deployments     …13
```

記述例の番号は、説明の番号と対応しています。

＜説明＞

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 または 9 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 9 Minimal 9.5」を使用しています。
2. 「RUN microdnf install ¥」の行から「gzip」の行で、uCosminexus Application Runtime 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。uCosminexus Application Runtime for Spring Boot のリリースノートを参照し、uCosminexus Application Runtime for Spring Boot に必要な前提 RPM パッケージを記述してください。
3. 「RUN mkdir ./X64LIN」の行から「RUN ./X64LIN/setup -f -c HL0R ./」の行で、uCosminexus Application Runtime 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「HL0R」（uCosminexus Application Runtime for Spring Boot）を指定して実行してください）。
4. このオレンジ色の背景で示す行は、uCosminexus Application Runtime 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 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 または 9 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 9 Minimal 9.5」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 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 for Spring Boot をインストールした状態となります。
7. 「RUN microdnf install ¥」の行から「jq ¥」の行で、HMP-PCTO の前提 RPM パッケージをインストールします。この例では、「Red Hat Universal Base Image 9 Minimal 9.5」の場合を記載しています。コンテナベースイメージが UBI 8 の場合は、curl-minimal を curl に変更してください。

8. 他社製の Java VM を指定します。この例では、Java SE 21 準拠の JDK として「OpenJDK 21」を使用しています。環境変数 PATH の先頭に、他社製の Java VM 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。
9. uCosminexus Application Runtime for Spring Boot の前提 RPM パッケージをインストールします。uCosminexus Application Runtime for Spring Boot のリリースノートを参照し、uCosminexus Application Runtime for Spring Boot に必要な前提 RPM パッケージを記述してください。
10. 「ENV LANG=C.UTF-8」の行から「ENV LC_ALL=C.UTF-8」の行で、環境変数 LANG および LC_ALL を設定して、コンテナ内のロケールを UTF-8 に変更します。利用するアプリケーションが必要とするロケールに応じて、適切な値（例：ja_JP.UTF-8）に変更してください。
11. このオレンジ色の背景で示す行は、Red Hat OpenShift Container Platform を使用し、SCC（Security Context Constraints）の設定でコンテナ実行時に root ユーザでの実行を許容しない場合に記述してください。uCARS のインストールディレクトリ配下の所有グループを実行ユーザが属するグループに変更してください。
12. 任意のコンテナ実行ユーザ・グループを指定します。
13. 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、事前に割り当てられた一定範囲内の 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	−
	X64LIN	日立	uCosminexus Application Runtime 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。

ファイル／ディレクトリ構成		作成・提供元	備考
	(uCosminexus Application Runtime for Spring Boot のインストーラ)		
	PACK_TAR.Z (uCosminexus Application Runtime for Spring Boot の修正パッチ)	日立	uCosminexus Application Runtime 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。
	GTS Root R1.pem GTS Root R2.pem GTS Root R3.pem GTS Root R4.pem (GTS Root R1、GTS Root R2、GTS Root R3、GTS Root R4 証明書)	ユーザ	分散トレース収集サービスに Google Cloud Platform の Cloud Trace を使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 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 for Spring Bootバージョン>△.
```

(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:latest .
```

この例では、uCosminexus Application Runtime for Spring Boot のイメージ名、タグ名を「ucars:latest」としています。

(4) コンテナイメージの登録

ビルドした uCosminexus Application Runtime 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。

2.3.4　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 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 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。

> **重要**
>
> Containerfile は FROM 句で指定したコンテナイメージの状態（作業ディレクトリ、環境変数、USER など）を引き継ぎます。そのため、JDK コンテナベースイメージを使用する場合は、引き継がれる状態に注意してください。
>
> 例として、JDK コンテナベースイメージで「USER 1001:0」が指定されている場合、JDK コンテナベースイメージを使用したコンテナイメージでは、Containerfile で明示的に USER を上書きしないかぎり、実行ユーザは 1001 のままとなります。この場合、root 権限が必要な処理を実行すると、権限が不足し処理が失敗します。

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

「6.3.2　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/ubi9/ubi-minimal:9.5-1742914212      …1
RUN microdnf install ¥      …2
        -y ¥
        --setopt=install_weak_deps=0 ¥
        --nodocs ¥
        java-21-openjdk-headless ¥
        curl-minimal ¥
        jq ¥
    && microdnf clean all ¥
    && rm -rf /var/cache/dnf/*
ENV LANG=C.UTF-8      …3
ENV LC_ALL=C.UTF-8
USER 1001:0      …4
WORKDIR /deployments      …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 UBI 8 または 9 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 9 Minimal 9.5」を使用しています。
2. 「RUN microdnf install ¥」の行から「&& rm -rf /var/cache/dnf/*」の行で、前提 RPM パッケージをインストールします。この例では、「Red Hat Universal Base Image 9 Minimal 9.5」の場合、かつ Java SE 21 準拠の JDK として「OpenJDK 21」を使用しています。コンテナベースイメージが UBI 8 の場合は、curl-minimal を curl に変更してください。
3. 「ENV LANG=C.UTF-8」の行から「ENV LC_ALL=C.UTF-8」の行で、環境変数 LANG および LC_ALL を設定して、コンテナ内のロケールを UTF-8 に変更します。利用するアプリケーションが必要とするロケールに応じて、適切な値（例：ja_JP.UTF-8）に変更してください。
4. 任意のコンテナ実行ユーザ・グループを指定します。
5. アプリケーション配置用ディレクトリを作成します。

■トライアル版の場合

```
FROM quay.io/almalinuxorg/9-minimal:9.5     …1
RUN microdnf install ¥     …2
        -y ¥
        --setopt=install_weak_deps=0 ¥
        --nodocs ¥
        java-21-openjdk-headless ¥
        curl-minimal ¥
        jq ¥
    && microdnf clean all ¥
    && rm -rf /var/cache/dnf/*
ENV LANG=C.UTF-8     …3
ENV LC_ALL=C.UTF-8
USER 1001:0     …4
WORKDIR /deployments     …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 Almalinux 9 Minimal 9.5 を指定します。
2. 「RUN microdnf install ¥」の行から「&& rm -rf /var/cache/dnf/*」の行で、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 21 準拠の JDK として「OpenJDK 21」を使用しています。
3. 「ENV LANG=C.UTF-8」の行から「ENV LC_ALL=C.UTF-8」の行で、環境変数 LANG および LC_ALL を設定して、コンテナ内のロケールを UTF-8 に変更します。利用するアプリケーションが必要とするロケールに応じて、適切な値（例：ja_JP.UTF-8）に変更してください。
4. 任意のコンテナ実行ユーザ・グループを指定します。
5. 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、事前に割り当てられた一定範囲内の 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 2-14 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:latest .
```

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:latest」としています。

(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。

2.3.5 Orchestrator-Service コンテナイメージの作成

Orchestrator-Service 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

「6.4.1 Orchestrator-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、Orchestrator-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 for Spring Boot 導入あり）の場合

```
FROM myregistry.example.com/myproject/ucars:latest      …1
COPY --chown=1001:0 --chmod=0440 ./orchestrator-V.R.S※.jar /deployments/      …2
COPY --chown=1001:0 --chmod=0550 ./health_check.sh /deployments/      …3
COPY --chown=1001:0 --chmod=0550 ./wait_transactions.sh /deployments/      …4
CMD /opt/hitachi/ucars/bin/starter.sh java ${HMPPCTO_JAVA_OPTIONS} -jar /deployments/orchest
rator-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「2.3.3 uCosminexus Application Runtime 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 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:latest」を指定しています。
2. 「ユーザの Orchestrator-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 Orchestrator-Service 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. 「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。

4. 「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。
5. コマンドを「uCosminexus Application Runtime 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-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 for Spring Boot 導入なし）の場合

```
FROM myregistry.example.com/myproject/jdk:latest      …1
COPY --chown=1001:0 --chmod=0440 ./orchestrator-V.R.S※.jar /deployments/      …2
COPY --chown=1001:0 --chmod=0550 ./health_check.sh /deployments/      …3
COPY --chown=1001:0 --chmod=0550 ./wait_transactions.sh /deployments/      …4
CMD java ${HMPPCTO_JAVA_OPTIONS} -jar /deployments/orchestrator-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「2.3.4　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:latest」を指定しています。
2. 「ユーザの Orchestrator-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 Orchestrator-Service 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. 「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。
4. 「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。
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 Orchestrator-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。

■トライアル版の場合

```
FROM myregistry.example.com/myproject/jdk:latest      …1
COPY --chown=1001:0 --chmod=0440 ./orchestrator-V.R.S※.jar /deployments/      …2
COPY --chown=1001:0 --chmod=0550 ./health_check.sh /deployments/      …3
```

```
COPY --chown=1001:0 --chmod=0550 ./wait_transactions.sh /deployments/      …4
CMD java ${HMPPCTO_JAVA_OPTIONS} -jar /deployments/orchestrator-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「2.3.4　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:latest」を指定しています。
2. 「ユーザの Orchestrator-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 Orchestrator-Service 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. 「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。
4. 「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。
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 Orchestrator-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、事前に割り当てられた一定範囲内の 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 2-15　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-Service 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	−

（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-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>/]<Orchestrator-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/orchestrator:latest .
```

この例では、Orchestrator-Service コンテナイメージのイメージ名・タグ名を「orchestrator:latest」としています。

(4)　コンテナイメージの登録

ビルドした Orchestrator-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。

2.3.6　Relay-Service コンテナイメージの作成

Relay-Service コンテナイメージの作成方法について説明します。

ユーザ責務の Kubernetes アプリケーションの Relay-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

「6.4.2　Relay-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、Relay-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 for Spring Boot 導入あり）の場合

```
FROM myregistry.example.com/myproject/ucars:latest      …1
COPY --chown=1001:0 --chmod=0440 ./relay-service-V.R.S※.jar /deployments/      …2
COPY --chown=1001:0 --chmod=0550 ./health_check.sh /deployments/     …3
CMD /opt/hitachi/ucars/bin/starter.sh java ${HMPPCTO_JAVA_OPTIONS} -jar /deployments/relay-s
ervice-V.R.S※.jar      …4
```

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「2.3.3　uCosminexus Application Runtime 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 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:latest」を指定しています。
2. 「ユーザの Relay-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 Relay-Service アプリケーションの JAR ファイル」の名前を「relay-service-*V.R.S*※.jar」としています。
3. 「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。
4. コマンドを「uCosminexus Application Runtime 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 Relay-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 for Spring Boot 導入なし）の場合

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

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「2.3.4　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:latest」を指定しています。
2. 「ユーザの Relay-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 Relay-Service アプリケーションの JAR ファイル」の名前を「relay-service-*V.R.S*※.jar」としています。
3. 「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。
4. 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 Relay-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。

■トライアル版の場合

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

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「2.3.4　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:latest」を指定しています。
2. 「ユーザの Relay-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 Relay-Service アプリケーションの JAR ファイル」の名前を「relay-service-*V.R.S*※.jar」としています。
3. 「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。

4. 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 Relay-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、事前に割り当てられた一定範囲内の 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 2-16 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 Relay-Service 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	−

（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。

Relay-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>/]<Relay-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/relay-service:latest .
```

この例では、Relay-Service コンテナイメージのイメージ名・タグ名を「relay-service:latest」としています。

(4) コンテナイメージの登録

ビルドした Relay-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。

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

Entity-Service(SQL)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

「6.4.3 Entity-Service(SQL)コンテナイメージ（SQL-Participant 限定）」に記載されている要件を満たすように、Entity-Service(SQL)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 for Spring Boot 導入あり）の場合

```
FROM registry.access.redhat.com/ubi9/ubi-minimal:9.5-1742914212  AS builder     …1
RUN microdnf install ¥     …2
        -y ¥
        --setopt=install_weak_deps=0 ¥
        --nodocs ¥
        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:latest     …5
RUN mkdir /deployments/eads-lib ¥     …6
    && mkdir /deployments/eads-log ¥
    && chown 1001:0 /deployments/eads-log ¥
    && chmod 770 /deployments/eads-log
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.jar /deployments/     …11
COPY --chown=1001:0 --chmod=0440 ./pdjdbc4.jar /deployments/     …12
COPY --chown=1001:0 --chmod=0440 ./xdb-23.6.0.24.10.jar /deployments/     …13
COPY --chown=1001:0 --chmod=0440 ./xmlparserv2_sans_jaxp_services-23.6.0.24.10.jar /deployme
nts/     …14
COPY --chown=1001:0 --chmod=0440 ./HiRDB.ini /deployments/     …15
CMD /opt/hitachi/ucars/bin/starter.sh java ${HMPPCTO_JAVA_OPTIONS} -jar /deployments/sql-par
ticipant.jar     …16
```

記述例の番号は、説明の番号と対応しています。

＜説明＞

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 または 9 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 9 Minimal 9.5」を使用しています。
2. 「RUN microdnf install ¥」の行から「gzip」の行で、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。コンテナベースイメージが UBI 8 の場合は、ncompress もインストールしてください。
3. 「RUN mkdir ./X64LIN」の行から「RUN ./X64LIN/setup -f -c HC13 ./」の行で、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. この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「2.3.3　uCosminexus Application Runtime 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 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:latest」を指定しています。
6. 「RUN mkdir /deployments/eads-lib ¥」の行から「&& chmod 770 /deployments/eads-log」の行で、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。

また、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-Service(SQL)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(SQL)アプリケーションの JAR ファイル」の名前を「sql-participant.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、Entity-Service(SQL)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. この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「OracleDatabase」、かつ XMLType 型を利用する場合だけ記述します。必要な追加ライブラリを配置し、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。ライブラリの詳細については、SQL-Participant の jp.co.Hitachi.soft.hmppcto.sql.jdbc.driver.additional-library-paths パラメタを参照してください。
    ここで格納したライブラリを SQL-Participant の jp.co.Hitachi.soft.hmppcto.sql.jdbc.driver.additional-library-paths パラメタに設定してください。
14. 13.と同じです。
15. この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。
16. コマンドを「uCosminexus Application Runtime 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(SQL)アプリケーションの JAR ファイル」を指定します。

■通常版（uCosminexus Application Runtime for Spring Boot 導入なし）の場合

```
FROM registry.access.redhat.com/ubi9/ubi-minimal:9.5-1742914212  AS builder     …1
RUN microdnf install ¥     …2
        -y ¥
```

```
        --setopt=install_weak_deps=0 ¥
        --nodocs ¥
        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:latest     …5
RUN mkdir /deployments/eads-lib ¥     …6
    && mkdir /deployments/eads-log ¥
    && chown 1001:0 /deployments/eads-log ¥
    && chmod 770 /deployments/eads-log
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.jar /deployments/     …11
COPY --chown=1001:0 --chmod=0440 ./pdjdbc4.jar /deployments/     …12
COPY --chown=1001:0 --chmod=0440 ./xdb-23.6.0.24.10.jar /deployments/     …13
COPY --chown=1001:0 --chmod=0440 ./xmlparserv2_sans_jaxp_services-23.6.0.24.10.jar /deployme
nts/     …14
COPY --chown=1001:0 --chmod=0440 ./HiRDB.ini /deployments/     …15
CMD java ${HMPPCTO_JAVA_OPTIONS} -jar /deployments/sql-participant.jar     …16
```

記述例の番号は、説明の番号と対応しています。

＜説明＞

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 または 9 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 9 Minimal 9.5」を使用しています。
2. 「RUN microdnf install ¥」の行から「gzip」の行で、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。コンテナベースイメージが UBI 8 の場合は、ncompress もインストールしてください。
3. 「RUN mkdir ./X64LIN」の行から「RUN ./X64LIN/setup -f -c HC13 ./」の行で、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. この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「2.3.4　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:latest」を指定しています。
6. 「RUN mkdir /deployments/eads-lib ¥」の行から「&& chmod 770 /deployments/eads-log」の行で、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。

   また、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-Service(SQL)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(SQL)アプリケーションの JAR ファイル」の名前を「sql-participant.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. この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「OracleDatabase」、かつ XMLType 型を利用する場合だけ記述します。必要な追加ライブラリを配置し、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。ライブラリの詳細については、SQL-Participant の jp.co.Hitachi.soft.hmppcto.sql.jdbc.driver.additional-library-paths パラメタを参照してください。

    ここで格納したライブラリを SQL-Participant の jp.co.Hitachi.soft.hmppcto.sql.jdbc.driver.additional-library-paths パラメタに設定してください。
14. 13.と同じです。
15. この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。

16. 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(SQL)アプリケーションの JAR ファイル」を指定します。

■トライアル版の場合

```
FROM myregistry.example.com/myproject/jdk:latest      …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.jar /deployments/      …4
COPY --chown=1001:0 --chmod=0440 ./postgresql-42.2.22.jar /deployments/      …5
COPY --chown=1001:0 --chmod=0440 ./xdb-23.6.0.24.10.jar /deployments/      …6
COPY --chown=1001:0 --chmod=0440 ./xmlparserv2_sans_jaxp_services-23.6.0.24.10.jar /deployme
nts/      …7
COPY --chown=1001:0 --chmod=0440 ./HiRDB.ini /deployments/      …8
CMD java ${HMPPCTO_JAVA_OPTIONS} -jar /deployments/sql-participant.jar      …9
```

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「2.3.4　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:latest」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-Service(SQL)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(SQL)アプリケーションの JAR ファイル」の名前を「sql-participant.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. この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「OracleDatabase」、かつ XMLType 型を利用する場合だけ記述します。必要な追加ライブラリを配置し、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。ライブラリの詳細については、SQL-Participant の jp.co.Hitachi.soft.hmppcto.sql.jdbc.driver.additional-library-paths パラメタを参照してください。ここで格納したライブラリを SQL-Participant の jp.co.Hitachi.soft.hmppcto.sql.jdbc.driver.additional-library-paths パラメタに設定してください。

7. 6.と同じです。

8. 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。

9. 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(SQL)アプリケーションの JAR ファイル」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、事前に割り当てられた一定範囲内の 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 2-17 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(SQL)アプリケーションの JAR ファイル	ユーザ	−
<JDBC Driver 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。

ファイル／ディレクトリ構成		作成・提供元	備考
	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-Service(SQL)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(SQL)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:latest .
```

この例では、Entity-Service(SQL)コンテナイメージのイメージ名・タグ名を「sql-participant:latest」としています。

(4) コンテナイメージの登録

ビルドした Entity-Service(SQL)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。

2.3.8 Entity-Service(TCC)コンテナイメージの作成（TCC-Participant 限定）

Entity-Service(TCC)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-Service(TCC)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

「6.4.4 Entity-Service(TCC)コンテナイメージ（TCC-Participant 限定)」に記載されている要件を満たすように、Entity-Service(TCC)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 for Spring Boot 導入あり）の場合

```
FROM registry.access.redhat.com/ubi9/ubi-minimal:9.5-1742914212  AS builder     …1
RUN microdnf install ¥     …2
        -y ¥
        --setopt=install_weak_deps=0 ¥
        --nodocs ¥
        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:latest     …5
RUN mkdir /deployments/eads-lib ¥     …6
    && mkdir /deployments/eads-log ¥
    && chown 1001:0 /deployments/eads-log ¥
    && chmod 770 /deployments/eads-log
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。

＜説明＞

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 または 9 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 9 Minimal 9.5」を使用しています。

2. 「RUN microdnf install ¥」の行から「gzip」の行で、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。コンテナベースイメージが UBI 8 の場合は、ncompress もインストールしてください。
3. 「RUN mkdir ./X64LIN」の行から「RUN ./X64LIN/setup -f -c HC13 ./」の行で、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. この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「2.3.3　uCosminexus Application Runtime 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 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:latest」を指定しています。
6. 「RUN mkdir /deployments/eads-lib ¥」の行から「&& chmod 770 /deployments/eads-log」の行で、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。

   また、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-Service(TCC)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(TCC)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 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(TCC)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 for Spring Boot 導入なし）の場合

```
FROM registry.access.redhat.com/ubi9/ubi-minimal:9.5-1742914212  AS builder     …1
RUN microdnf install ¥     …2
        -y ¥
        --setopt=install_weak_deps=0 ¥
        --nodocs ¥
        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:latest     …5
RUN mkdir /deployments/eads-lib ¥     …6
    && mkdir /deployments/eads-log ¥
    && chown 1001:0 /deployments/eads-log ¥
    && chmod 770 /deployments/eads-log
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。

＜説明＞

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 または 9 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 9 Minimal 9.5」を使用しています。
2. 「RUN microdnf install ¥」の行から「gzip」の行で、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。コンテナベースイメージが UBI 8 の場合は、ncompress もインストールしてください。
3. 「RUN mkdir ./X64LIN」の行から「RUN ./X64LIN/setup -f -c HC13 ./」の行で、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. この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「2.3.4　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:latest」を指定しています。

6. 「RUN mkdir /deployments/eads-lib ¥」の行から「&& chmod 770 /deployments/eads-log」の行で、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。

   また、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-Service(TCC)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(TCC)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-Service(TCC)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:latest      …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「2.3.4　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:latest」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-Service(TCC)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(TCC)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-Service(TCC)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、事前に割り当てられた一定範囲内の 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 2-18　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(TCC)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-Service(TCC)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(TCC)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:latest .
```

この例では、Entity-Service(TCC)コンテナイメージのイメージ名・タグ名を「entity-module-tcc-participant:latest」としています。

(4) コンテナイメージの登録

ビルドした Entity-Service(TCC)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。

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

このコンテナイメージは、Orchestrator-Service で JDBC Proxy Driver を使用する場合に作成します。

Orchestrator-Service(SQL)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

「6.4.5 Orchestrator-Service(SQL)コンテナイメージ（SQL-Participant 限定)」に記載されている要件を満たすように、Orchestrator-Service(SQL)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 for Spring Boot 導入あり）の場合

```
FROM registry.access.redhat.com/ubi9/ubi-minimal:9.5-1742914212  AS builder     …1
RUN microdnf install ¥     …2
        -y ¥
        --setopt=install_weak_deps=0 ¥
        --nodocs ¥
        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:latest     …5
RUN mkdir /deployments/eads-lib ¥     …6
    && mkdir /deployments/eads-log ¥
```

```
    && chown 1001:0 /deployments/eads-log ¥
    && chmod 770 /deployments/eads-log
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 ./orchestrator-sql.jar /deployments/     …11
COPY --chown=1001:0 --chmod=0440 ./pdjdbc4.jar /deployments/     …12
COPY --chown=1001:0 --chmod=0440 ./xdb-23.6.0.24.10.jar /deployments/     …13
COPY --chown=1001:0 --chmod=0440 ./xmlparserv2_sans_jaxp_services-23.6.0.24.10.jar /deployme
nts/     …14
COPY --chown=1001:0 --chmod=0440 ./HiRDB.ini /deployments/     …15
CMD /opt/hitachi/ucars/bin/starter.sh java ${HMPPCTO_JAVA_OPTIONS} -jar /deployments/orchest
rator-sql.jar     …16
```

記述例の番号は、説明の番号と対応しています。

＜説明＞

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 9 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 9 Minimal 9.5」を使用しています。
2. 「RUN microdnf install ¥」の行から「gzip」の行で、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。コンテナベースイメージが UBI 8 の場合は、ncompress もインストールしてください。
3. 「RUN mkdir ./X64LIN」の行から「RUN ./X64LIN/setup -f -c HC13 ./」の行で、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. この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「2.3.3　uCosminexus Application Runtime 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 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:latest」を指定しています。
6. 「RUN mkdir /deployments/eads-lib ¥」の行から「&& chmod 770 /deployments/eads-log」の行で、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。

   また、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 Orchestrator-Service(SQL)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-Service(SQL)アプリケーションの JAR ファイル」の名前を「orchestrator-sql.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. この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「OracleDatabase」、かつ XMLType 型を利用する場合だけ記述します。必要な追加ライブラリを配置し、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。ライブラリの詳細については、SQL-Participant の jp.co.Hitachi.soft.hmppcto.sql.jdbc.driver.additional-library-paths パラメタを参照してください。
    ここで格納したライブラリを SQL-Participant の jp.co.Hitachi.soft.hmppcto.sql.jdbc.driver.additional-library-paths パラメタに設定してください。
14. 13.と同じです。
15. この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。
16. コマンドを「uCosminexus Application Runtime 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-Service(SQL)アプリケーションの JAR ファイル」を指定します。

■通常版（uCosminexus Application Runtime for Spring Boot 導入なし）の場合

```
FROM registry.access.redhat.com/ubi9/ubi-minimal:9.5-1742914212  AS builder     …1
RUN microdnf install ¥     …2
        -y ¥
        --setopt=install_weak_deps=0 ¥
        --nodocs ¥
        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:latest     …5
RUN mkdir /deployments/eads-lib ¥     …6
    && mkdir /deployments/eads-log ¥
    && chown 1001:0 /deployments/eads-log ¥
    && chmod 770 /deployments/eads-log
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 ./orchestrator-sql.jar /deployments/     …11
COPY --chown=1001:0 --chmod=0440 ./pdjdbc4.jar /deployments/     …12
COPY --chown=1001:0 --chmod=0440 ./xdb-23.6.0.24.10.jar /deployments/     …13
COPY --chown=1001:0 --chmod=0440 ./xmlparserv2_sans_jaxp_services-23.6.0.24.10.jar /deployme
nts/     …14
COPY --chown=1001:0 --chmod=0440 ./HiRDB.ini /deployments/     …15
CMD java ${HMPPCTO_JAVA_OPTIONS} -jar /deployments/orchestrator-sql.jar     …16
```

記述例の番号は、説明の番号と対応しています。

＜説明＞

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 9 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 9 Minimal 9.5」を使用しています。
2. 「RUN microdnf install ¥」の行から「gzip」の行で、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。コンテナベースイメージが UBI 8 の場合は、ncompress もインストールしてください。
3. 「RUN mkdir ./X64LIN」の行から「RUN ./X64LIN/setup -f -c HC13 ./」の行で、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. この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「2.3.4　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:latest」を指定しています。
6. 「RUN mkdir /deployments/eads-lib ¥」の行から「&& chmod 770 /deployments/eads-log」の行で、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。

   また、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 Orchestrator-Service(SQL)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-Service(SQL)アプリケーションの JAR ファイル」の名前を「orchestrator-sql.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. この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「OracleDatabase」、かつ XMLType 型を利用する場合だけ記述します。必要な追加ライブラリを配置し、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。ライブラリの詳細については、SQL-Participant の jp.co.Hitachi.soft.hmppcto.sql.jdbc.driver.additional-library-paths パラメタを参照してください。

    ここで格納したライブラリを SQL-Participant の jp.co.Hitachi.soft.hmppcto.sql.jdbc.driver.additional-library-paths パラメタに設定してください。
14. 13.と同じです。
15. この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。

16. 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-Service(SQL)アプリケーションの JAR ファイル」を指定します。

■トライアル版の場合

```
FROM myregistry.example.com/myproject/jdk:latest     …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 ./orchestrator-sql.jar /deployments/     …4
COPY --chown=1001:0 --chmod=0440 ./postgresql-42.2.22.jar /deployments/     …5
COPY --chown=1001:0 --chmod=0440 ./xdb-23.6.0.24.10.jar /deployments/     …6
COPY --chown=1001:0 --chmod=0440 ./xmlparserv2_sans_jaxp_services-23.6.0.24.10.jar /deployme
nts/     …7
COPY --chown=1001:0 --chmod=0440 ./HiRDB.ini /deployments/     …8
CMD java ${HMPPCTO_JAVA_OPTIONS} -jar /deployments/orchestrator-sql.jar     …9
```

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「2.3.4　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:latest」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 Orchestrator-Service(SQL)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-Service(SQL)アプリケーションの JAR ファイル」の名前を「orchestrator-sql.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. この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「OracleDatabase」、かつ XMLType 型を利用する場合だけ記述します。必要な追加ライブラリを配置し、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。ライブラリの詳細については、SQL-Participant の jp.co.Hitachi.soft.hmppcto.sql.jdbc.driver.additional-library-paths パラメタを参照してください。ここで格納したライブラリを SQL-Participant の jp.co.Hitachi.soft.hmppcto.sql.jdbc.driver.additional-library-paths パラメタに設定してください。
7. 6.と同じです。

8. 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。
9. 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-Service(SQL)アプリケーションの JAR ファイル」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、事前に割り当てられた一定範囲内の 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 2-19 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-Service(SQL)アプリケーションの JAR ファイル	ユーザ	−
<JDBC Driver 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。

ファイル／ディレクトリ構成		作成・提供元	備考
	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。

Orchestrator-Service(SQL)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-Service(SQL)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-sql:latest
.
```

この例では、Orchestrator-Service(SQL)コンテナイメージのイメージ名・タグ名を「orchestrator-sql:latest」としています。

(4) コンテナイメージの登録

ビルドした Orchestrator-Service(SQL)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。

2.3.10 Alternate-Service コンテナイメージの作成

Alternate-Service 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

「6.4.6 Alternate-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、Alternate-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 for Spring Boot 導入あり）の場合

```
FROM myregistry.example.com/myproject/ucars:latest      …1
COPY --chown=1001:0 --chmod=0440 ./alternate-service-V.R.S※.jar /deployments/      …2
COPY --chown=1001:0 --chmod=0550 ./health_check.sh /deployments/      …3
COPY --chown=1001:0 --chmod=0550 ./wait_transactions.sh /deployments/      …4
CMD /opt/hitachi/ucars/bin/starter.sh java ${HMPPCTO_JAVA_OPTIONS} -jar /deployments/alterna
te-service-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「2.3.3 uCosminexus Application Runtime 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 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:latest」を指定しています。
2. 「ユーザの Alternate-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 Alternate-Service アプリケーションの JAR ファイル」の名前を「alternate-service-*V.R.S*※.jar」としています。
3. 「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。
4. 「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。
5. コマンドを「uCosminexus Application Runtime 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 Alternate-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 for Spring Boot 導入なし）の場合

```
FROM myregistry.example.com/myproject/jdk:latest      …1
COPY --chown=1001:0 --chmod=0440 ./alternate-service-V.R.S※.jar /deployments/      …2
COPY --chown=1001:0 --chmod=0550 ./health_check.sh /deployments/      …3
COPY --chown=1001:0 --chmod=0550 ./wait_transactions.sh /deployments/      …4
CMD java ${HMPPCTO_JAVA_OPTIONS} -jar /deployments/alternate-service-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「2.3.4　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:latest」を指定しています。
2. 「ユーザの Alternate-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 Alternate-Service アプリケーションの JAR ファイル」の名前を「alternate-service-*V.R.S*※.jar」としています。
3. 「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。
4. 「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。
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 Alternate-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。

■トライアル版の場合

```
FROM myregistry.example.com/myproject/jdk:latest      …1
COPY --chown=1001:0 --chmod=0440 ./alternate-service-V.R.S※.jar /deployments/      …2
COPY --chown=1001:0 --chmod=0550 ./health_check.sh /deployments/      …3
COPY --chown=1001:0 --chmod=0550 ./wait_transactions.sh /deployments/      …4
CMD java ${HMPPCTO_JAVA_OPTIONS} -jar /deployments/alternate-service-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「2.3.4　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:latest」を指定しています。
2. 「ユーザの Alternate-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 Alternate-Service アプリケーションの JAR ファイル」の名前を「alternate-service-*V.R.S*※.jar」としています。
3. 「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。
4. 「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。

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 Alternate-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、事前に割り当てられた一定範囲内の 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 2-20 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 Alternate-Service アプリケーションの JAR ファイル>	ユーザ	－
	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	－

（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。

Alternate-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>/]<Alternate-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/alternate-service:latest
 .
```

この例では、Alternate-Service コンテナイメージのイメージ名・タグ名を「alternate-service:latest」としています。

(4) コンテナイメージの登録

ビルドした Alternate-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。

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

このコンテナイメージは、Alternate-Service で JDBC Proxy Driver を使用する場合に作成します。

Alternate-Service(SQL)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

「6.4.7 Alternate-Service(SQL)コンテナイメージ（SQL-Participant 限定）」に記載されている要件を満たすように、Alternate-Service(SQL)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 for Spring Boot 導入あり）の場合

```
FROM registry.access.redhat.com/ubi9/ubi-minimal:9.5-1742914212  AS builder     …1
RUN microdnf install ¥     …2
        -y ¥
        --setopt=install_weak_deps=0 ¥
        --nodocs ¥
        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:latest      …5
RUN mkdir /deployments/eads-lib ¥      …6
    && mkdir /deployments/eads-log ¥
    && chown 1001:0 /deployments/eads-log ¥
    && chmod 770 /deployments/eads-log
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 ./alternate-service-sql.jar /deployments/      …11
COPY --chown=1001:0 --chmod=0440 ./pdjdbc4.jar /deployments/      …12
COPY --chown=1001:0 --chmod=0440 ./xdb-23.6.0.24.10.jar /deployments/      …13
COPY --chown=1001:0 --chmod=0440 ./xmlparserv2_sans_jaxp_services-23.6.0.24.10.jar /deployme
nts/      …14
COPY --chown=1001:0 --chmod=0440 ./HiRDB.ini /deployments/      …15
CMD /opt/hitachi/ucars/bin/starter.sh java ${HMPPCTO_JAVA_OPTIONS} -jar /deployments/alterna
te-service-sql.jar      …16
```

記述例の番号は、説明の番号と対応しています。

＜説明＞

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 または 9 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 9 Minimal 9.5」を使用しています。
2. 「RUN microdnf install ¥」の行から「gzip」の行で、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。コンテナベースイメージが UBI 8 の場合は、ncompress もインストールしてください。
3. 「RUN mkdir ./X64LIN」の行から「RUN ./X64LIN/setup -f -c HC13 ./」の行で、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. この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「2.3.3　uCosminexus Application Runtime 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 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:latest」を指定しています。

6.「RUN mkdir /deployments/eads-lib ¥」の行から「&& chmod 770 /deployments/eads-log」の行で、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。

また、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 Alternate-Service(SQL)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 Alternate-Service(SQL)アプリケーションの JAR ファイル」の名前を「alternate-service-sql.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. この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「OracleDatabase」、かつ XMLType 型を利用する場合だけ記述します。必要な追加ライブラリを配置し、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。ライブラリの詳細については、SQL-Participant の jp.co.Hitachi.soft.hmppcto.sql.jdbc.driver.additional-library-paths パラメタを参照してください。

ここで格納したライブラリを SQL-Participant の jp.co.Hitachi.soft.hmppcto.sql.jdbc.driver.additional-library-paths パラメタに設定してください。

14. 13.と同じです。

15. この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。

16. コマンドを「uCosminexus Application Runtime 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 Alternate-Service(SQL)アプリケーションの JAR ファイル」を指定します。

■通常版（uCosminexus Application Runtime for Spring Boot 導入なし）の場合

```
FROM registry.access.redhat.com/ubi9/ubi-minimal:9.5-1742914212  AS builder     …1
RUN microdnf install ¥     …2
        -y ¥
        --setopt=install_weak_deps=0 ¥
        --nodocs ¥
        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:latest     …5
RUN mkdir /deployments/eads-lib ¥     …6
    && mkdir /deployments/eads-log ¥
    && chown 1001:0 /deployments/eads-log ¥
    && chmod 770 /deployments/eads-log
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 ./alternate-service-sql.jar /deployments/     …11
COPY --chown=1001:0 --chmod=0440 ./pdjdbc4.jar /deployments/     …12
COPY --chown=1001:0 --chmod=0440 ./xdb-23.6.0.24.10.jar /deployments/     …13
COPY --chown=1001:0 --chmod=0440 ./xmlparserv2_sans_jaxp_services-23.6.0.24.10.jar /deployme
nts/     …14
COPY --chown=1001:0 --chmod=0440 ./HiRDB.ini /deployments/     …15
CMD java ${HMPPCTO_JAVA_OPTIONS} -jar /deployments/alternate-service-sql.jar     …16
```

記述例の番号は、説明の番号と対応しています。

＜説明＞

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 または 9 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 9 Minimal 9.5」を使用しています。
2. 「RUN microdnf install ¥」の行から「gzip」の行で、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。コンテナベースイメージが UBI 8 の場合は、ncompress もインストールしてください。
3. 「RUN mkdir ./X64LIN」の行から「RUN ./X64LIN/setup -f -c HC13 ./」の行で、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. この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「2.3.4　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:latest」を指定しています。

6. 「RUN mkdir /deployments/eads-lib ¥」の行から「&& chmod 770 /deployments/eads-log」の行で、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。

   また、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 Alternate-Service(SQL)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 Alternate-Service(SQL)アプリケーションの JAR ファイル」の名前を「alternate-service-sql.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. この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「OracleDatabase」、かつ XMLType 型を利用する場合だけ記述します。必要な追加ライブラリを配置し、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。ライブラリの詳細については、SQL-Participant の jp.co.Hitachi.soft.hmppcto.sql.jdbc.driver.additional-library-paths パラメタを参照してください。

    ここで格納したライブラリを SQL-Participant の jp.co.Hitachi.soft.hmppcto.sql.jdbc.driver.additional-library-paths パラメタに設定してください。

14. 13.と同じです。

15. この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。
16. 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 Alternate-Service(SQL)アプリケーションの JAR ファイル」を指定します。

■トライアル版の場合

```
FROM myregistry.example.com/myproject/jdk:latest      …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 ./alternate-service-sql.jar /deployments/      …4
COPY --chown=1001:0 --chmod=0440 ./postgresql-42.2.22.jar /deployments/      …5
COPY --chown=1001:0 --chmod=0440 ./xdb-23.6.0.24.10.jar /deployments/      …6
COPY --chown=1001:0 --chmod=0440 ./xmlparserv2_sans_jaxp_services-23.6.0.24.10.jar /deployme
nts/      …7
COPY --chown=1001:0 --chmod=0440 ./HiRDB.ini /deployments/      …8
CMD java ${HMPPCTO_JAVA_OPTIONS} -jar /deployments/alternate-service-sql.jar      …9
```

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「2.3.4　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:latest」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 Alternate-Service(SQL)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 Alternate-Service(SQL)アプリケーションの JAR ファイル」の名前を「alternate-service-sql.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. この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「OracleDatabase」、かつ XMLType 型を利用する場合だけ記述します。必要な追加ライブラリを配置し、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。ライブラリの詳細については、SQL-Participant の jp.co.Hitachi.soft.hmppcto.sql.jdbc.driver.additional-library-paths パラメタを参照してください。

ここで格納したライブラリを SQL-Participant の jp.co.Hitachi.soft.hmppcto.sql.jdbc.driver.additional-library-paths パラメタに設定してください。

7. 6.と同じです。
8. 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。
9. 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 Alternate-Service(SQL)アプリケーションの JAR ファイル」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、事前に割り当てられた一定範囲内の 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 2-21 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 Alternate-Service(SQL)アプリケーションの JAR ファイル	ユーザ	−
<JDBC Driver 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	−

ファイル／ディレクトリ構成		作成・提供元	備考
	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。

Alternate-Service(SQL)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>/]<Alternate-Service(SQL)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/alternate-service-sql:latest .
```

この例では、Alternate-Service(SQL)コンテナイメージのイメージ名・タグ名を「alternate-service-sql:latest」としています。

(4) コンテナイメージの登録

ビルドした Alternate-Service(SQL)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　永続ボリュームとプロビジョニング方法の選定（通常版限定）

HMP-PCTO が使用する永続ボリュームの選定について説明します。HMP-PCTO がサポートしている永続ボリュームのストレージ種別と、そのプロビジョニング方法の一覧を次の表に示します。この表を参照して、プロビジョニング方法を選定してください。

表 2-22　HMP-PCTO でサポートしている永続ボリュームのストレージ種別と、そのプロビジョニング方法の一覧

ストレージの種別	Kubernetes 環境	マネージドファイルサービス	プロビジョニング方法		説明
			静的	動的	
ローカル永続ボリューム	規定しません	該当しません	○	−	ストレージクラスなどの詳細は用途によって異なります。詳細は「2.6　Kubernetes マニフェスト作成」を参照してください。
NFS	オンプレミス環境	該当しません	○※	−	ストレージクラスなどの詳細は用途によって異なります。詳細は「2.6　Kubernetes マニフェスト作成」を参照してください。 また、作業の詳細は各システムで異なります。HMP-PCTO としての作業については、「2.5　共有ファイルシステムでの作業（通常版限定）」を参照してください。
	Red Hat OpenShift Container Platform 環境	該当しません	○※	−	
	Microsoft Azure 環境	Azure Files	○	−	NFS をサポートしているため、Azure Files および Azure Blob Storage を永続ボリュームとして使用できます。プロビジョニングする方法については、Microsoft Azure の公式ドキュメントを参照してください。 HMP-PCTO として必要な作業は、「2.5　共有ファイルシステムでの作業（通常版限定）」を参照してください。
		Azure Blob Storage	−	○	
	Google Cloud Platform 環境	Filestore	○	○	NFS をサポートしているため、Filestore を永続ボリュームとして使用できます。プロビジョニングする方法については、Google Cloud Platform の公式ドキュメントを参照してください。

ストレージの種別	Kubernetes 環境	マネージドファイルサービス	プロビジョニング方法		説明
			静的	動的	
					HMP-PCTO として必要な作業は、「2.5　共有ファイルシステムでの作業（通常版限定）」を参照してください。
	Amazon Web Services 環境	Amazon EFS	○	○	NFS をサポートしているため、Amazon EFS を永続ボリュームとして使用できます。プロビジョニングする方法については、Amazon Web Services の公式ドキュメントを参照してください。 HMP-PCTO として必要な作業は、「2.5　共有ファイルシステムでの作業（通常版限定）」を参照してください。

（凡例）

○：サポートしています。HMP-PCTO および EADS で動作検証が実施済みの構成です。

－：サポートしていません。HMP-PCTO および EADS で動作検証が未実施の構成です。この構成を使用する場合は、ユーザ責任で動作検証を実施してください。

注※

目的によっては使用できない場合があります。詳細な選定方法は「2.6　Kubernetes マニフェスト作成」を参照してください。

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 2-23　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
マネージドファイルサービスでのプロビジョニング（通常版限定）	Microsoft Azure 環境、Google Cloud Platform 環境、または Amazon Web Services 環境で、uCosminexus Application Runtime for Spring Boot を導入するときだけ、次のどちらか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の Kubernetes アプリケーションおよび RAS	2.5.2　マネージドファイルサービスでのプロビジョニング（通常版限定）
uCosminexus Application Runtime 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）	通常版の場合で、uCosminexus Application Runtime for Spring Boot を導入するときだけ、次のどちらか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の Kubernetes アプリケーションおよび RAS	2.5.3　uCosminexus Application Runtime 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　マネージドファイルサービスでのプロビジョニング（通常版限定）

Microsoft Azure 環境、Google Cloud Platform 環境、または Amazon Web Services 環境の場合で NFS ストレージが必要なときは、次のどちらかをデプロイするための作業時に、マネージドファイルサービスでのプロビジョニングを実施します。

- 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の Kubernetes アプリケーションおよび RAS

詳細については、「3.6.2　マネージドファイルサービスでのプロビジョニング」を参照してください。

2.5.3 uCosminexus Application Runtime 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 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の Kubernetes アプリケーションおよび RAS

詳細については、「3.6.5　uCosminexus Application Runtime 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 Namespace 名を「.」でつないだ文字列が、41 バイト以内となるようにしてください。

TCC-Participant を構成する Pod を定義した Kubernetes の Service の Service 名と Namespace 名を「.」でつないだ文字列が、41 バイト以内となるようにしてください。

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、「3.11.1 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 2-24　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-Service の Kubernetes マニフェストの作成	−	2.6.2　Orchestrator-Service の Kubernetes マニフェストの作成
Relay-Service の Kubernetes マニフェストの作成	−	2.6.3　Relay-Service の Kubernetes マニフェストの作成
Entity-Service(SQL)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　Entity-Service(SQL)の Kubernetes マニフェストの作成（SQL-Participant 限定）
Entity-Service(TCC)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　Entity-Service(TCC)の Kubernetes マニフェストの作成（TCC-Participant 限定）
Orchestrator-Service(SQL)の Kubernetes マニフェストの作成（SQL-Participant 限定）	−	2.6.6　Orchestrator-Service(SQL)の Kubernetes マニフェストの作成（SQL-Participant 限定）
Alternate-Service の Kubernetes マニフェストの作成	−	2.6.7　Alternate-Service の Kubernetes マニフェストの作成
Alternate-Service(SQL)の Kubernetes マニフェストの作成（SQL-Participant 限定）	−	2.6.8　Alternate-Service(SQL)の Kubernetes マニフェストの作成（SQL-Participant 限定）

作業項目	備考	参照先
Kubernetes リソース取得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。	2.6.9　Kubernetes リソース取得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 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の Kubernetes アプリケーションおよび RAS	2.6.10　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 Kubernetes マニフェストの作成
uCosminexus Application Runtime 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）	通常版の場合で、uCosminexus Application Runtime for Spring Boot を導入するときだけ作成します。次のどちらかの 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の Kubernetes アプリケーションおよび RAS	2.6.11　uCosminexus Application Runtime 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）
HPA の Kubernetes マニフェストの作成	スケーリング対応機能を使用する場合で、Metrics Server でメトリクスの監視をしてオートスケールを適用するときだけ作成します。	4.2.4　HPA の Kubernetes マニフェストの作成
	スケーリング対応機能を使用する場合で、Prometheus でメトリクスの監視をしてオートスケールを適用するときだけ作成します。	4.3.6　HPA の Kubernetes マニフェストの作成

（凡例）－：該当しません。

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

「7.6.1　Orchestrator-Service の要件」に記載されている要件を満たすように、Orchestrator-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
kind: Deployment
metadata:
  name: orchestrator
  namespace: my-namespace
spec:
  replicas: 1     …1
  selector:
    matchLabels:
      app.kubernetes.io/name: orchestrator
  template:
    metadata:
      labels:
        app.kubernetes.io/name: orchestrator
        hmppcto-app-group: x-system
```

```
      annotations:
        prometheus.io/scrape: 'true'
        prometheus.io/path: '/hmppcto/metrics'      …2
        prometheus.io/port: '8080'      …3
        prometheus.io/scheme: 'http'      …4
    spec:
      serviceAccount: orchestrator-service-account      …5
      securityContext:      …6
        fsGroup: 0
      imagePullSecrets:      …7
      - name: regcred
      initContainers:
      - name: dependency-checker      …8
        image: myregistry.example.com/myproject/dependency-checker:V.R.S※1
        args:
        - "30"
        - "10"
        - "10"
        - "10"
        - my-namespace/deployments/sql-participant※2
      containers:
      - name: orchestrator      …9
        image: myregistry.example.com/myproject/orchestrator:V.R.S※1
        resources:      …10
          limits:
            cpu: 2
            memory: 1Gi
          requests:
            cpu: 1
            memory: 512Mi
        ports:
        - name: http
          containerPort: 8080
          protocol: TCP
        - name: ucars      …11
          containerPort: 28081
          protocol: TCP
        env:
        - name: TZ      …12
          value: Asia/Tokyo
        - name: HMPPCTO_JAVA_OPTIONS      …13
          value: -XX:+UseG1GC
        volumeMounts:
        - name: config      …14
          mountPath: /deployments/config
        - name: ucars-config      …15
          mountPath: /opt/hitachi/ucars/conf
        - name: snapshot-volume      …16
          mountPath: /ucars-snapshots
        - name: tomcat      …17
          mountPath: /deployments/tomcat
        livenessProbe:      …18
          exec:
            command:
            - /deployments/health_check.sh
            - "0"
            - "0"
            - "10"
```

```
            - "10"
            - "8080"
          initialDelaySeconds: 180
          periodSeconds: 30
          failureThreshold: 3
          timeoutSeconds: 10
        lifecycle:
          postStart:     …19
            exec:
              command:
              - /deployments/health_check.sh
              - "30"
              - "10"
              - "10"
              - "10"
              - "8080"
          preStop:     …20
            exec:
              command:
              - /deployments/wait_transactions.sh
              - "10"
              - "10"
              - "10"
              - "10"
              - "8080"
      - name: filebeat-sidecar      …21
        image: myregistry.example.com/myproject/filebeat:V.R.S※1
        resources:      …22
          limits:
            cpu: 1000m
            memory: 2Gi
          requests:
            cpu: 1000m
            memory: 2Gi
        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
        securityContext:
          runAsUser: 0
        volumeMounts:
         - name: beatconfig
           mountPath: /usr/share/filebeat/filebeat.yml
           readOnly: true
           subPath: filebeat.yml
         - name: tomcat      …17
           mountPath: /deployments/tomcat
      terminationGracePeriodSeconds: 180      …24
      volumes:
      - name: config      …14
        configMap:
          name: orchestrator-config
          items:
          - key: application.properties
            path: application.properties
      - name: ucars-config      …15
        configMap:
          name: orchestrator-ucars-config
          items:
            - key: config.properties
              path: config.properties
      - name: snapshot-volume      …16
        persistentVolumeClaim:
          claimName: snapshot-volume
      - name: beatconfig      …21
        configMap:
          name: orchestrator-filebeat-sidecar-config
      - name: tomcat      …17
---
apiVersion: v1
kind: Service
metadata:
  name: orchestrator
  namespace: my-namespace
  labels:
    hmppcto-app-group: x-system
spec:
  type: ClusterIP
  selector:
```

```
    app.kubernetes.io/name: orchestrator
  ports:
  - name: http     …3
    protocol: TCP
    port: 8080
    targetPort: http
  - name: ucars    …11
    protocol: TCP
    port: 28081
    targetPort: ucars
---
apiVersion: v1    …14
kind: ConfigMap
metadata:
  name: orchestrator-config
  namespace: my-namespace
data:
  application.properties: |
    server.port=8080
    server.tomcat.accesslog.enabled=true
    server.tomcat.basedir=tomcat
    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-xid-transaction-id}i" "
%{x-xid-branch-id}i" "%{x-xid-format-id}i" "%{x-transaction-context}i" "%{traceparent}i"
 …25
    server.tomcat.accesslog.locale=en_US
    server.tomcat.accesslog.max-days=7
    server.shutdown=graceful
    spring.lifecycle.timeout-per-shutdown-phase=30s
    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.instance-id=Orchestrator-Service_001
    jp.co.Hitachi.soft.hmppcto.tracing.url=http://jaeger:4318/v1/traces
    jp.co.Hitachi.soft.hmppcto.use-scaling=true
    jp.co.Hitachi.soft.hmppcto.set-routing=inner
---
apiVersion: v1     …15
kind: ConfigMap
metadata:
  name: orchestrator-ucars-config
  namespace: my-namespace
data:
  config.properties: |
    monitor.rest.bindaddress=0.0.0.0
    monitor.rest.port=28081     …11
    snapshot.log.filepath=/ucars-snapshots/${HOSTNAME}/snapshot
    healthcheck.startdelay.timeout=60000
    healthcheck.startdelay.actions.failure.retrymax=4
---
apiVersion: v1     …21
kind: ConfigMap
metadata:
  name: orchestrator-filebeat-sidecar-config
  namespace: my-namespace
data:
```

```
  filebeat.yml: |
    filebeat.inputs:
      - 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"]
---
apiVersion: v1      …5
kind: ServiceAccount
metadata:
  name: orchestrator-service-account
  namespace: my-namespace
```

注※1

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。

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

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。また、使用する Spring Boot の組み込み HTTP サーバを「Tomcat」としています。

1. オートスケールを使用しない場合だけ、replicas を設定します。
2. 「prometheus.io/path」の値は、コンテキストパスを設定していない場合の指定値です。

コンテキストパスを設定している場合は、次の形式で指定してください。

```
prometheus.io/path: <コンテキストパス>/hmppcto/metrics
```

コンテキストパスが/context-path の場合の指定例を次に示します。

```
prometheus.io/path: /context-path/hmppcto/metrics
```

3. 「prometheus.io/port: '8080'」の行、および「- name: http」の行から「targetPort: http」の行で、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。
4. メトリクスのスクレイプ時に使用する接続スキームを指定します。ユーザアプリケーションの暗号化通信を有効にする場合は、https を設定してください。
5. 「serviceAccount: orchestrator-service-account」の行、および「apiVersion: v1」の行から「namespace: my-namespace」の行で、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。
6. このオレンジ色の背景で示す行は、Google Cloud Platform 環境の場合だけ記述します。securityContext を設定します。fsGroup にはコンテナ実行グループと同等の設定をしてください。
7. このオレンジ色の背景で示す行は、コンテナイメージを Pull する際に資格情報が必要な場合に記述します。Red Hat OpenShift Container Platform 環境で資格情報が不要な場合は imagePullSecrets: [] を指定してください。
8. 「- name: dependency-checker」の行から「- my-namespace/deployments/sql-participant」の行で、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。
9. 「- name: orchestrator」の行から「preStop:」の「- "8080"」の行で、コンテナ構成に Orchestrator-Service コンテナを定義します。「2.3.5　Orchestrator-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。
10. 「resources:」の行から「memory: 512Mi」の行で、Orchestrator-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。

    オートスケールのメトリクスの監視に Metrics Server を使用する場合は必ず指定してください。
11. このオレンジ色の背景で示す行（「- name: ucars」の行から「protocol: TCP」の行、「- name: ucars」の行から「targetPort: ucars」の行、および「monitor.rest.port=28081」の行）は、通常版（uCosminexus Application Runtime for Spring Boot 導入あり）の場合だけ記述します。

    uCosminexus Application Runtime 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。
12. 「- name: TZ」の行から「value: Asia/Tokyo」の行で、Orchestrator-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。
13. 「- name: HMPPCTO_JAVA_OPTIONS」の行から「value: -XX:+UseG1GC」の行で、Orchestrator-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。

14. 「- name: config」の行から「mountPath: /deployments/config」の行、「- name: config」の行から「path: application.properties」の行、および「apiVersion: v1」の行から「jp.co.Hitachi.soft.hmppcto.set-routing=inner」の行で、Orchestrator-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、設定するパラメタは最小限の例としています。

15. このオレンジ色の背景で示す行（「- name: ucars-config」の行から「mountPath: /opt/hitachi/ucars/conf」の行、「- name: ucars-config」の行から「path: config.properties」の行、および「apiVersion: v1」の行から「healthcheck.startdelay.actions.failure.retrymax=4」の行）は、通常版（uCosminexus Application Runtime for Spring Boot 導入あり）の場合だけ記述します。

    uCosminexus Application Runtime for Spring Boot の設定ファイル(config.properties)を Orchestrator-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、「11.6.2　情報の収集方法」を参照して設定してください。

    - healthcheck.startdelay.timeout=60000

      モニタ対象プロセス（Orchestrator-Service）開始完了通知待ちタイムアウト時間を定義します。Orchestrator-Service の起動時に、設定した時間が経過すると snapshot.log.filepath で指定した先にスナップショットを出力します。モニタ対象プロセス開始完了通知待ちタイムアウト時間の詳細は、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。

    - healthcheck.startdelay.actions.failure.retrymax=4

      モニタ対象プロセス（Orchestrator-Service）開始完了通知待ちタイムアウトが発生した場合に、モニタ対象プロセス開始完了待ちをリトライする最大回数を定義します。最大リトライ回数を超えると、デフォルトの設定ではモニタ対象プロセスを停止します。モニタ対象プロセス開始完了待ちリトライ回数の詳細は、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。

      モニタ対象プロセス開始完了通知待ちタイムアウトの合計時間（下記の A）と Orchestrator-Service の開始に掛かる時間（下記の B）の関係を、「A ＞ B」にしてください。

      A：モニタ対象プロセス開始完了通知待ちタイムアウトの合計時間

```
healthcheck.startdelay.timeoutに指定した値×（healthcheck.startdelay.actions.failure.retrymaxに指定した値＋1）
```

B：Orchestrator-Service 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のリトライ時間

```
jp.co.Hitachi.soft.hmppcto.mediator.discovery-interval-in-millisに指定した値×jp.co.Hitachi.soft.hmppcto.mediator.discovery-retry-countに指定した値
```

A を B よりも大きな値とするのは、モニタ対象プロセス開始完了通知待ちタイムアウトによるプロセス停止が、Orchestrator-Service 起動時の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処理を中止することを防ぐためです。

この記述例では、Orchestrator-Service の開始に掛かる時間（180 秒）の間に、スナップショットを複数回（3 回）出力するように設定しています。

16. このオレンジ色の背景で示す行（「- name: snapshot-volume」の行から「mountPath: /ucars-snapshots」の行、および「- name: snapshot-volume」の行から「claimName: snapshot-volume」の行）は、通常版（uCosminexus Application Runtime for Spring Boot 導入あり）の場合だけ記述します。

    uCosminexus Application Runtime for Spring Boot スナップショットログの永続ボリュームを Orchestrator-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、「2.6.11 uCosminexus Application Runtime 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 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。

17. 「- name: tomcat」の行から「mountPath: /deployments/tomcat」の行、および「- name: tomcat」の行で、Tomcat のベースディレクトリとして /deployments/tomcat をマウントします。

18. 「livenessProbe:」の行から「timeoutSeconds: 10」の行で、コンテナライフサイクル設定の Liveness Probe を定義します。

19. 「postStart:」の行から「- "8080"」の行で、コンテナライフサイクル設定の postStart フックを定義します。

20. 「preStop:」の行から「- "8080"」の行で、コンテナライフサイクル設定の preStop フックを定義します。

21. このオレンジ色の背景で示す行（「- name: filebeat-sidecar」の行から「mountPath: /deployments/tomcat」の行、「- name: beatconfig」の行から「name: orchestrator-filebeat-sidecar-config」の行、および「apiVersion: v1」の行から「hosts: ["logstash.my-namespace.svc.cluster.local:5044"]」の行）は、Elastic Stack を使用してトラブルシュート情報を収集する場合だけ記述します。

    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。

22.「resources:」の行から「requests:」の「memory: 2Gi」の行で、Filebeat 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。

オートスケールのメトリクスの監視に Metrics Server を使用する場合は必ず指定してください。

23.「env:」の行から「fieldPath: spec.nodeName」の行、および「processors:」の行から「name: ${HMPPCTO_NODE_NAME}」の行で、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。

24. Pod ライフサイクル設定の terminationGracePeriodSeconds を定義します。

25. このオレンジ色の背景で示す行は、通常版（uCosminexus Application Runtime for Spring Boot 導入あり）の場合だけ記述します。

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 Relay-Service の Kubernetes マニフェストの作成

「7.6.2 Relay-Service の要件」に記載されている要件を満たすように、Relay-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
kind: Deployment
metadata:
  name: relay-service
  namespace: my-namespace
spec:
  replicas: 1     …1
  selector:
    matchLabels:
      app.kubernetes.io/name: relay-service
  template:
    metadata:
      labels:
        app.kubernetes.io/name: relay-service
        hmppcto-app-group: x-system
      annotations:
        prometheus.io/scrape: 'true'
        prometheus.io/path: '/hmppcto/metrics'      …2
```

```
        prometheus.io/port: '8080'      …3
        prometheus.io/scheme: 'http'      …4
    spec:
      serviceAccount: relay-service-service-account      …5
      securityContext:      …6
        fsGroup: 0
      imagePullSecrets:      …7
      - name: regcred
      initContainers:
      - name: dependency-checker      …8
        image: myregistry.example.com/myproject/dependency-checker:V.R.S※1
        args:
        - "30"
        - "10"
        - "10"
        - "10"
        - my-namespace/statefulsets/mediator
        - my-namespace/deployments/sql-participant※2
      containers:
      - name: relay-service      …9
        image: myregistry.example.com/myproject/relay-service:V.R.S※1
        resources:      …10
          limits:
            cpu: 2
            memory: 1Gi
          requests:
            cpu: 1
            memory: 512Mi
        ports:
        - name: http      …3
          containerPort: 8080
          protocol: TCP
        - name: ucars      …11
          containerPort: 28081
          protocol: TCP
        env:
        - name: TZ      …12
          value: Asia/Tokyo
        - name: HMPPCTO_JAVA_OPTIONS      …13
          value: -XX:+UseG1GC
        volumeMounts:
        - name: relay-service-config      …14
          mountPath: /deployments/config
        - name: ucars-config      …15
          mountPath: /opt/hitachi/ucars/conf
        - name: snapshot-volume      …16
          mountPath: /ucars-snapshots
        - name: tomcat      …17
          mountPath: /deployments/tomcat
        livenessProbe:      …18
          exec:
            command:
            - /deployments/health_check.sh
            - "0"
            - "0"
            - "10"
            - "10"
            - "8080"
```

```
          initialDelaySeconds: 180
          periodSeconds: 30
          failureThreshold: 3
          timeoutSeconds: 10
        lifecycle:
          postStart:     …19
            exec:
              command:
              - /deployments/health_check.sh
              - "30"
              - "10"
              - "10"
              - "10"
              - "8080"
          preStop:     …20
            exec:
              command:
              - /bin/sh
              - -c
              - sleep 20
      - name: filebeat-sidecar      …21
        image: myregistry.example.com/myproject/filebeat:V.R.S※1
        resources:      …22
          limits:
            cpu: 1000m
            memory: 2Gi
          requests:
            cpu: 1000m
            memory: 2Gi
        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
        securityContext:
          runAsUser: 0
        volumeMounts:
         - name: beatconfig
           mountPath: /usr/share/filebeat/filebeat.yml
           readOnly: true
           subPath: filebeat.yml
         - name: tomcat     …17
           mountPath: /deployments/tomcat
      terminationGracePeriodSeconds: 180     …24
      volumes:
      - name: relay-service-config     …14
        configMap:
          name: relay-service-config
          items:
          - key: application.properties
            path: application.properties
      - name: ucars-config     …15
        configMap:
          name: relay-service-ucars-config
          items:
            - key: config.properties
              path: config.properties
      - name: snapshot-volume     …16
        persistentVolumeClaim:
          claimName: snapshot-volume
      - name: beatconfig  …21
        configMap:
          name: es-filebeat-sidecar-config
      - name: tomcat     …17
---
apiVersion: v1
kind: Service
metadata:
  name: relay-service
  namespace: my-namespace
  labels:
    hmppcto-app-group: x-system
spec:
  type: ClusterIP
  selector:
    app.kubernetes.io/name: relay-service
  ports:
  - name: http     …3
    protocol: TCP
    port: 8080
```

```
    targetPort: http
  - name: ucars      …11
    protocol: TCP
    port: 28081
    targetPort: ucars
---
apiVersion: v1      …14
kind: ConfigMap
metadata:
  name: relay-service-config
  namespace: my-namespace
data:
  application.properties: |
    server.port=8080
    server.tomcat.accesslog.enabled=true
    server.tomcat.basedir=tomcat
    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-xid-transaction-id}i" "
%{x-xid-branch-id}i" "%{x-xid-format-id}i" "%{x-transaction-context}i" "%{traceparent}i"
 …25
    server.tomcat.accesslog.locale=en_US
    server.tomcat.accesslog.max-days=7
    server.shutdown=graceful
    spring.lifecycle.timeout-per-shutdown-phase=140s
    jp.co.Hitachi.soft.hmppcto.client.proposer.type=SQL
    jp.co.Hitachi.soft.hmppcto.tracing.instance-id=Relay-Service_001
    jp.co.Hitachi.soft.hmppcto.tracing.url=http://jaeger:4318/v1/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
    jp.co.Hitachi.soft.hmppcto.use-scaling=true
    jp.co.Hitachi.soft.hmppcto.set-routing=inner
---
apiVersion: v1  …15
kind: ConfigMap
metadata:
  name: relay-service-ucars-config
  namespace: my-namespace
data:
  config.properties: |
    monitor.rest.bindaddress=0.0.0.0
    monitor.rest.port=28081      …11
    snapshot.log.filepath=/ucars-snapshots/${HOSTNAME}/snapshot
---
apiVersion: v1  …21
kind: ConfigMap
metadata:
  name: es-filebeat-sidecar-config
  namespace: my-namespace
data:
  filebeat.yml: |
    filebeat.inputs:
      - 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"]
---
apiVersion: v1     …5
kind: ServiceAccount
metadata:
  name: relay-service-service-account
  namespace: my-namespace
```

注※1

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。

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

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。また、使用する Spring Boot の組み込み HTTP サーバを「Tomcat」としています。

1. オートスケールを使用しない場合だけ、replicas を設定します。
2. 「prometheus.io/path」の値は、コンテキストパスを設定していない場合の指定値です。
   コンテキストパスを設定している場合は、次の形式で指定してください。

   ```
   prometheus.io/path: <コンテキストパス>/hmppcto/metrics
   ```

   コンテキストパスが/context-path の場合の指定例を次に示します。

```
prometheus.io/path: /context-path/hmppcto/metrics
```

3. 「prometheus.io/port: '8080'」の行、「- name: http」の行から「protocol: TCP」の行、および「- name: http」の行から「targetPort: http」の行で、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。
4. メトリクスのスクレイプ時に使用する接続スキームを指定します。ユーザアプリケーションの暗号化通信を有効にする場合は、https を設定してください。
5. 「serviceAccount: relay-service-service-account」の行、および「apiVersion: v1」の行から「namespace: my-namespace」の行で、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。
6. このオレンジ色の背景で示す行は、Google Cloud Platform 環境の場合だけ記述します。securityContext を設定します。fsGroup にはコンテナ実行グループと同等の設定をしてください。
7. このオレンジ色の背景で示す行は、コンテナイメージを Pull する際に資格情報が必要な場合に記述します。Red Hat OpenShift Container Platform 環境で資格情報が不要な場合は imagePullSecrets: [] を指定してください。
8. 「- name: dependency-checker」の行から「- my-namespace/deployments/sql-participant」の行で、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。
9. 「- name: relay-service」の行から「- sleep 20」の行で、コンテナ構成に Relay-Service コンテナを定義します。「2.3.6　Relay-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。
10. 「resources:」の行から「memory: 512Mi」の行で、Relay-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。

    オートスケールのメトリクスの監視に Metrics Server を使用する場合は必ず指定してください。
11. このオレンジ色の背景で示す行（「- name: ucars」の行から「protocol: TCP」の行、「- name: ucars」の行から「targetPort: ucars」の行、および「monitor.rest.port=28081」の行）は、通常版（uCosminexus Application Runtime for Spring Boot 導入あり）の場合だけ記述します。

    uCosminexus Application Runtime 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。
12. 「- name: TZ」の行から「value: Asia/Tokyo」の行で、Relay-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。
13. 「- name: HMPPCTO_JAVA_OPTIONS」の行から「value: -XX:+UseG1GC」の行で、Relay-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。

14. 「- name: relay-service-config」の行から「mountPath: /deployments/config」の行、「- name: relay-service-config」の行から「path: application.properties」の行、および「apiVersion: v1」の行から「jp.co.Hitachi.soft.hmppcto.set-routing=inner」の行で、Relay-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。

15. このオレンジ色の背景で示す行（「- name: ucars-config」の行から「mountPath: /opt/hitachi/ucars/conf」の行、「- name: ucars-config」の行から「path: config.properties」の行、および「apiVersion: v1」の行から「snapshot.log.filepath=/ucars-snapshots/${HOSTNAME}/snapshot」の行）は、通常版（uCosminexus Application Runtime for Spring Boot 導入あり）の場合だけ記述します。

    uCosminexus Application Runtime for Spring Boot の設定ファイル(config.properties)を Relay-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、「11.6.2　情報の収集方法」を参照して設定してください。

16. このオレンジ色の背景で示す行（「- name: snapshot-volume」の行から「mountPath: /ucars-snapshots」の行、および「- name: snapshot-volume」の行から「claimName: snapshot-volume」の行）は、通常版（uCosminexus Application Runtime for Spring Boot 導入あり）の場合だけ記述します。

    uCosminexus Application Runtime for Spring Boot スナップショットログの永続ボリュームを Relay-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、「2.6.11 uCosminexus Application Runtime 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 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。

17. 「- name: tomcat」の行から「mountPath: /deployments/tomcat」の行、および「- name: tomcat」の行で、Tomcat のベースディレクトリとして /deployments/tomcat をマウントします。

18. 「livenessProbe:」の行から「timeoutSeconds: 10」の行で、コンテナライフサイクル設定の Liveness Probe を定義します。

19. 「postStart:」の行から「- "8080"」の行で、コンテナライフサイクル設定の postStart フックを定義します。
20. 「preStop:」の行から「- sleep 20」の行で、コンテナライフサイクル設定の preStop フックを定義します。
21. このオレンジ色の背景で示す行（「- name: filebeat-sidecar」の行から「mountPath: /deployments/tomcat」の行、「- name: beatconfig」の行から「name: es-filebeat-sidecar-config」の行、および「apiVersion: v1」の行から「hosts: ["logstash.my-namespace.svc.cluster.local:5044"]」の行）は、Elastic Stack を使用してトラブルシュート情報を収集する場合だけ記述します。

    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。
22. 「resources:」の行から「requests:」の「memory: 2Gi」の行で、Filebeat 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。

    オートスケールのメトリクスの監視に Metrics Server を使用する場合は必ず指定してください。
23. 「env:」の行から「fieldPath: spec.nodeName」の行、および「processors:」の行から「name: ${HMPPCTO_NODE_NAME}」の行で、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。
24. Pod ライフサイクル設定の terminationGracePeriodSeconds を定義します。
25. このオレンジ色の背景で示す行は、通常版（uCosminexus Application Runtime for Spring Boot 導入あり）の場合だけ記述します。

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

「7.6.3 Entity-Service(SQL)の要件（SQL-Participant 限定）」に記載されている要件を満たすように、Entity-Service(SQL)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 マニフェストは、通常版とトライアル版で差異があります。トライアル版の Kubernetes マニフェストを基に通常版の Kubernetes マニフェストを作成する場合は、通常版の記述例に従って修正してください。

Kubernetes マニフェストの記述例を次に示します。

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: sql-participant
  namespace: my-namespace
spec:
  replicas: 1     …1
  selector:
    matchLabels:
      app.kubernetes.io/name: sql-participant     …2
  template:
    metadata:
      labels:
        app.kubernetes.io/name: sql-participant
        hmppcto-app-group: x-system
      annotations:
        prometheus.io/scrape: 'true'
        prometheus.io/path: '/hmppcto/metrics'     …3
        prometheus.io/port: '8080'     …4
        prometheus.io/scheme: 'http'     …5
    spec:
      serviceAccount: sql-participant-service-account     …6
      securityContext:     …7
        fsGroup: 0
      imagePullSecrets:     …8
      - name: regcred
      initContainers:
      - name: dependency-checker     …9
        image: myregistry.example.com/myproject/dependency-checker:V.R.S※1
        args:
        - "30"
        - "10"
        - "10"
        - "10"
        - my-namespace/statefulsets/mediator
        - my-namespace/statefulsets/eads-server-eads-server※2
        - my-namespace/statefulsets/postgresql
      containers:
      - name: sql-participant     …10
        image: myregistry.example.com/myproject/sql-participant:V.R.S※1
        resources:     …11
          limits:
            cpu: 2
            memory: 1Gi
          requests:
            cpu: 1
            memory: 512Mi
        ports:
        - name: http     …4
          containerPort: 8080
          protocol: TCP
        - name: ucars-sql  …12
          containerPort: 28081
          protocol: TCP
        env:
```

```
        - name: TZ     …13
          value: Asia/Tokyo
        - name: HMPPCTO_JAVA_OPTIONS     …14
          value: -XX:+UseG1GC
        volumeMounts:
        - name: sql-participant-config
          mountPath: /deployments/config
        - name: sql-participant-ucars-config  …15
          mountPath: /opt/hitachi/ucars/conf
        - name: snapshot-volume  …16
          mountPath: /ucars-snapshots
        - name: eads-config     …17
          mountPath: /eads-properties
        - name: tomcat     …18
          mountPath: /deployments/tomcat
        - name: eads   …19
          mountPath: /deployments/eads-log
        livenessProbe:     …20
          exec:
            command:
            - /deployments/health_check.sh
            - "0"
            - "0"
            - "10"
            - "10"
            - "8080"
          initialDelaySeconds: 180
          periodSeconds: 30
          failureThreshold: 3
          timeoutSeconds: 10
        lifecycle:
          postStart:     …21
            exec:
              command:
              - /deployments/health_check.sh
              - "30"
              - "10"
              - "10"
              - "10"
              - "8080"
          preStop:     …22
            exec:
              command:
              - /deployments/wait_transactions.sh
              - "10"
              - "10"
              - "10"
              - "10"
              - "8080"
      - name: filebeat-sidecar     …23
        image: myregistry.example.com/myproject/filebeat:V.R.S※1
        resources:     …24
          limits:
            cpu: 1000m
            memory: 2Gi
          requests:
            cpu: 1000m
            memory: 2Gi
```

```
        args:
        - "-e"
        - "-E"
        - "http.enabled=true"
        env:      …25
        - 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      …18
            mountPath: /deployments/tomcat
          - name: eads      …19
            mountPath: /deployments/eads-log
      terminationGracePeriodSeconds: 180      …26
      volumes:
      - name: snapshot-volume      …16
        persistentVolumeClaim:
          claimName: snapshot-volume
      - name: sql-participant-config
```

```
        configMap:
          name: sql-participant-config
          items:
          - key: application.properties
            path: application.properties
      - name: sql-participant-ucars-config …15
        configMap:
          name: sql-participant-ucars-config
          items:
            - key: config.properties
              path: config.properties
      - name: eads-config     …17
        configMap:
          name: eads-config
          items:
          - key: hmp-pcto.eads-client.properties
            path: hmp-pcto.eads-client.properties
      - name: beatconfig      …23
        configMap:
          name: sql-p-filebeat-sidecar-config
      - name: tomcat     …18
      - name: eads     …19
---
apiVersion: v1
kind: Service
metadata:
  name: sp
  namespace: my-namespace
  labels:
    hmppcto-app-group: x-system
spec:
  type: ClusterIP
  selector:
    app.kubernetes.io/name: sql-participant
  ports:
  - name: http     …4
    protocol: TCP
    port: 8080
    targetPort: http
  - name: ucars-sql     …12
    protocol: TCP
    port: 28081
    targetPort: ucars-sql
---
apiVersion: v1     …27
kind: ConfigMap
metadata:
  name: sql-participant-config
  namespace: my-namespace
data:
  application.properties: |
    server.port=8080
    server.tomcat.accesslog.enabled=true
    server.tomcat.basedir=tomcat
    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-xid-transaction-id}i" "
%{x-xid-branch-id}i" "%{x-xid-format-id}i" "%{x-transaction-context}i" "%{traceparent}i"
 …28
```

```
    server.tomcat.accesslog.locale=en_US
    server.tomcat.accesslog.max-days=7
    server.shutdown=graceful
    spring.lifecycle.timeout-per-shutdown-phase=30s
    jp.co.Hitachi.soft.hmppcto.client.proposer.type=SQL
    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=pctodriver
    spring.datasource.username=dummy
    spring.datasource.password=dummy
    spring.sql.init.mode=never
    jp.co.Hitachi.soft.hmppcto.sql.pod.service-name=sp.my-namespace
    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     …29
    jp.co.Hitachi.soft.hmppcto.tracing.instance-id=Entity-Service-SQL_001
    jp.co.Hitachi.soft.hmppcto.tracing.url=http://jaeger:4318/v1/traces
    jp.co.Hitachi.soft.hmppcto.use-scaling=true
    jp.co.Hitachi.soft.hmppcto.set-routing=inner
---
apiVersion: v1 …15
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 …12
    snapshot.log.filepath=/ucars-snapshots/${HOSTNAME}/snapshot
    healthcheck.startdelay.timeout=90000
    healthcheck.startdelay.actions.failure.retrymax=4
---
apiVersion: v1     …17
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     …6
kind: ServiceAccount
```

```
metadata:
  name: sql-participant-service-account
  namespace: my-namespace
---
apiVersion: v1      …23
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:      …25
      - 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※1

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。

注※2

通常版の場合は my-namespace/statefulsets/eads-server-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。

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。また、使用する Spring Boot の組み込み HTTP サーバを「Tomcat」としています。

1. オートスケールを使用しない場合だけ、replicas を設定します。
2. 「app.kubernetes.io/name」を必ず指定してください。

   「app.kubernetes.io/name」は、ほかの Kubernetes アプリケーションに設定した値と異なる値を設定してください。

16 バイト以内の文字列を指定してください。

Blue/Green デプロイメントを実施する場合は、Blue と Green の「app.kubernetes.io/name」には同一の値を設定してください。

3. 「prometheus.io/path」の値は、コンテキストパスを設定していない場合の指定値です。

   コンテキストパスを設定している場合は、次の形式で指定してください。

   ```
   prometheus.io/path: <コンテキストパス>/hmppcto/metrics
   ```

   コンテキストパスが/context-path の場合の指定例を次に示します。

   ```
   prometheus.io/path: /context-path/hmppcto/metrics
   ```

4. 「prometheus.io/port: '8080'」の行、「- name: http」の行から「protocol: TCP」の行、および「- name: http」の行から「targetPort: http」の行で、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. メトリクスのスクレイプ時に使用する接続スキームを指定します。ユーザアプリケーションの暗号化通信を有効にする場合は、https を設定してください。

6. 「serviceAccount: sql-participant-service-account」の行、および「apiVersion: v1」の行から「kind: ServiceAccount」の「namespace: my-namespace」の行で、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. このオレンジ色の背景で示す行は、Google Cloud Platform 環境の場合だけ記述します。securityContext を設定します。fsGroup にはコンテナ実行グループと同等の設定をしてください。

8. このオレンジ色の背景で示す行は、コンテナイメージを Pull する際に資格情報が必要な場合に記述します。Red Hat OpenShift Container Platform 環境で資格情報が不要な場合は imagePullSecrets: [] を指定してください。

9. 「- name: dependency-checker」の行から「- my-namespace/statefulsets/postgresql」の行で、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。

   Dependency-Checker の対象として接続する RDBMS の Kubernetes オブジェクトを指定する必要があります。

   この例では、「- my-namespace/statefulsets/postgresql」を指定しています。

10. 「- name: sql-participant」の行から「preStop:」の「- "8080"」の行で、コンテナ構成に Entity-Service(SQL)コンテナを定義します。「2.3.7　Entity-Service(SQL)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。

11. 「resources:」の行から「memory: 512Mi」の行で、Entity-Service(SQL)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。

    オートスケールのメトリクスの監視に Metrics Server を使用する場合は必ず指定してください。

12. このオレンジ色の背景で示す行（「- name: ucars-sql」の行から「protocol: TCP」の行、「- name: ucars-sql」の行から「targetPort: ucars-sql」の行、および「monitor.rest.port=28081」の行）は、通常版（uCosminexus Application Runtime for Spring Boot 導入あり）の場合だけ記述します。

    uCosminexus Application Runtime 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。

13. 「- name: TZ」の行から「value: Asia/Tokyo」の行で、Entity-Service(SQL)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. 「- name: HMPPCTO_JAVA_OPTIONS」の行から「value: -XX:+UseG1GC」の行で、Entity-Service(SQL)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。

15. このオレンジ色の背景で示す行（「- name: sql-participant-ucars-config」の行から「mountPath: /opt/hitachi/ucars/conf」の行、「- name: sql-participant-ucars-config」の行から「path: config.properties」の行、および「apiVersion: v1」の行から「healthcheck.startdelay.actions.failure.retrymax=4」の行）は、通常版（uCosminexus Application Runtime for Spring Boot 導入あり）の場合だけ記述します。

    uCosminexus Application Runtime for Spring Boot の設定ファイル(config.properties)を Entity-Service(SQL)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、「11.6.2　情報の収集方法」を参照して設定してください。

    - healthcheck.startdelay.timeout=90000

      モニタ対象プロセス（Entity-Service(SQL)）開始完了通知待ちタイムアウト時間を定義します。Entity-Service(SQL)の起動時に、設定した時間が経過すると snapshot.log.filepath で指定した先にスナップショットを出力します。モニタ対象プロセス開始完了通知待ちタイムアウト時間の詳細は、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。

    - healthcheck.startdelay.actions.failure.retrymax=4

モニタ対象プロセス（Entity-Service(SQL)）開始完了通知待ちタイムアウトが発生した場合に、モニタ対象プロセス開始完了待ちをリトライする最大回数を定義します。最大リトライ回数を超えると、デフォルトの設定ではモニタ対象プロセスを停止します。モニタ対象プロセス開始完了待ちリトライ回数の詳細は、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。

モニタ対象プロセス開始完了通知待ちタイムアウトの合計時間（下記の A）と Entity-Service(SQL)の開始に掛かる時間（下記の B、および C）の関係を、「A ＞ B ＞ C」にしてください。

A：モニタ対象プロセス開始完了通知待ちタイムアウトの合計時間

```
healthcheck.startdelay.timeoutに指定した値×（healthcheck.startdelay.actions.failure.retrymaxに指定した値＋1）
```

B：postStart フックに指定するトランザクションヘルスチェックスクリプト（health_check.sh）の合計時間

```
リトライ回数×リトライ間隔秒数
```

C：コネクションプールの初期化で、DB とのコネクションの取得に失敗した場合に行うリトライの合計時間

```
sql.pool.connection-retry-interval-in-secondsに指定した値×sql.pool.connection-retry-limitに指定した値
```

A を B よりも大きな値にするのは、モニタ対象プロセス開始完了通知待ちタイムアウトによるプロセス停止が、postStart フックに指定するトランザクションヘルスチェックスクリプトによる Entity-Service(SQL)の開始完了待ちを中止することを防ぐためです。

B を C よりも大きな値とするのは、postStart フックに指定するトランザクションヘルスチェックスクリプトのタイムアウトによる Entity-Service(SQL)の停止が、Entity-Service(SQL)のコネクション接続処理を中止することを防ぐためです。

この記述例では、Entity-Service(SQL)の開始に掛かる時間（300 秒）の間に、スナップショットを複数回（3 回）出力するように設定しています。

16. このオレンジ色の背景で示す行（「- name: snapshot-volume」の行から「mountPath: /ucars-snapshots」の行、および「- name: snapshot-volume」の行から「claimName: snapshot-volume」の行）は、通常版（uCosminexus Application Runtime for Spring Boot 導入あり）の場合だけ記述します。

    uCosminexus Application Runtime for Spring Boot スナップショットログの永続ボリュームを Entity-Service(SQL)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、「2.6.11　uCosminexus Application Runtime 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 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。

17. 「- name: eads-config」の行から「mountPath: /eads-properties」の行、「- name: eads-config」の行から「path: hmp-pcto.eads-client.properties」の行、および「apiVersion: v1」の行から

「eads.client.connectionPool.poolsize=1」の行で、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「9.2.5　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。

18. 「- name: tomcat」の行から「mountPath: /deployments/tomcat」の行、および「- name: tomcat」の行で、Tomcat のベースディレクトリとして /deployments/tomcat をマウントします。
19. 「- name: eads」の行から「mountPath: /deployments/eads-log」の行、および「- name: eads」の行で、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。
20. 「livenessProbe:」の行から「timeoutSeconds: 10」の行で、コンテナライフサイクル設定の Liveness Probe を定義します。
21. 「postStart:」の行から「- "8080"」の行で、コンテナライフサイクル設定の postStart フックを定義します。
22. 「preStop:」の行から「- "8080"」の行で、コンテナライフサイクル設定の preStop フックを定義します。
23. このオレンジ色の背景で示す行（「- name: filebeat-sidecar」の行から「mountPath: /deployments/eads-log」の行、「- name: beatconfig」の行から「name: sql-p-filebeat-sidecar-config」の行、および「apiVersion: v1」の行から「hosts: ["logstash.my-namespace.svc.cluster.local:5044"]」の行）は、Elastic Stack を使用してトラブルシュート情報を収集する場合だけ記述します。

    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。
24. 「resources:」の行から「requests:」の「memory: 2Gi」の行で、Filebeat 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。

    オートスケールのメトリクスの監視に Metrics Server を使用する場合は必ず指定してください。
25. 「env:」の行から「fieldPath: spec.nodeName」の行、および「processors:」の行から「name: ${HMPPCTO_NODE_NAME}」の行で、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。
26. Pod ライフサイクル設定の terminationGracePeriodSeconds を定義します。
27. 「apiVersion: v1」の行から「jp.co.Hitachi.soft.hmppcto.set-routing=inner」の行で、Entity-Service(SQL)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。
28. このオレンジ色の背景で示す行は、通常版（uCosminexus Application Runtime for Spring Boot 導入あり）の場合だけ記述します。

29. Entity-Service(SQL)が接続する RDBMS を指定します。PostgreSQL を指定する場合、「9.4.1 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。
- Entity-Service(SQL)を構成する Pod を定義した Kubernetes の Service の Service 名と Namespace 名を「.」でつないだ文字列が、41 バイト以内となるようにしてください。

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

「7.6.4 Entity-Service(TCC)の要件（TCC-Participant 限定)」に記載されている要件を満たすように、Entity-Service(TCC)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 マニフェストは、通常版とトライアル版で差異があります。トライアル版の Kubernetes マニフェストを基に通常版の Kubernetes マニフェストを作成する場合は、通常版の記述例に従って修正してください。

Kubernetes マニフェストの記述例を次に示します。

```
apiVersion: apps/v1
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
        hmppcto-app-group: x-system
      annotations:
        prometheus.io/scrape: 'true'
        prometheus.io/path: '/hmppcto/metrics'      …1
        prometheus.io/port: '8080'      …2
        prometheus.io/scheme: 'http'      …3
    spec:
      serviceAccount: tcc-participant-service-account      …4
      securityContext:      …5
        fsGroup: 0
```

```
      imagePullSecrets:     …6
      - name: regcred
      initContainers:
      - name: dependency-checker     …7
        image: myregistry.example.com/myproject/dependency-checker:V.R.S※1
        args:
        - "30"
        - "10"
        - "10"
        - "10"
        - my-namespace/statefulsets/mediator
        - my-namespace/statefulsets/eads-server-eads-server※2
      containers:
      - name: entity-module-tcc-participant     …8
        image: myregistry.example.com/myproject/entity-module-tcc-participant:V.R.S※1
        resources:     …9
          limits:
            cpu: 2
            memory: 1Gi
          requests:
            cpu: 1
            memory: 512Mi
        ports:
        - name: http     …2
          containerPort: 8080
          protocol: TCP
        - name: ucars     …10
          containerPort: 28081
          protocol: TCP
        env:
        - name: TZ     …11
          value: Asia/Tokyo
        - name: HMPPCTO_JAVA_OPTIONS     …12
          value: -XX:+UseG1GC
        volumeMounts:
        - name: entity-module-tcc-participant-config     …13
          mountPath: /deployments/config
        - name: ucars-config     …14
          mountPath: /opt/hitachi/ucars/conf
        - name: snapshot-volume     …15
          mountPath: /ucars-snapshots
        - name: eads-config     …16
          mountPath: /eads-properties
        - name: tomcat     …17
          mountPath: /deployments/tomcat
        - name: eads     …18
          mountPath: /deployments/eads-log
        livenessProbe:     …19
          exec:
            command:
            - /deployments/health_check.sh
            - "0"
            - "0"
            - "10"
            - "10"
            - "8080"
          initialDelaySeconds: 180
          periodSeconds: 30
```

```
          failureThreshold: 3
          timeoutSeconds: 10
        lifecycle:
          postStart:     …20
            exec:
              command:
              - /deployments/health_check.sh
              - "30"
              - "10"
              - "10"
              - "10"
              - "8080"
          preStop:     …21
            exec:
              command:
              - /deployments/wait_transactions.sh
              - "10"
              - "10"
              - "10"
              - "10"
              - "8080"
      - name: filebeat-sidecar      …22
        image: myregistry.example.com/myproject/filebeat:V.R.S※1
        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
        securityContext:
          runAsUser: 0
        volumeMounts:
          - name: beatconfig
            mountPath: /usr/share/filebeat/filebeat.yml
            readOnly: true
            subPath: filebeat.yml
          - name: tomcat     …17
            mountPath: /deployments/tomcat
          - name: eads     …18
            mountPath: /deployments/eads-log
      terminationGracePeriodSeconds: 150     …24
      volumes:
      - name: entity-module-tcc-participant-config     …13
        configMap:
          name: entity-module-tcc-participant-config
          items:
          - key: application.properties
            path: application.properties
      - name: ucars-config     …14
        configMap:
          name: entity-module-tcc-participant-ucars-config
          items:
            - key: config.properties
              path: config.properties
      - name: snapshot-volume     …15
        persistentVolumeClaim:
          claimName: snapshot-volume
      - name: eads-config     …16
        configMap:
          name: eads-config
          items:
          - key: hmp-pcto.eads-client.properties
            path: hmp-pcto.eads-client.properties
      - name: beatconfig     …22
        configMap:
          name: tcc-p-filebeat-sidecar-config
      - name: tomcat     …17
      - name: eads     …18
---
apiVersion: v1
kind: Service
metadata:
  name: tp
  namespace: my-namespace
  labels:
    hmppcto-app-group: x-system
spec:
  type: ClusterIP
  selector:
    app.kubernetes.io/name: tcc-participant
  ports:
```

```
  - name: http     …2
    protocol: TCP
    port: 8080
    targetPort: http
  - name: ucars    …10
    protocol: TCP
    port: 28081
    targetPort: ucars
---
apiVersion: v1     …13
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
    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-xid-transaction-id}i" "
%{x-xid-branch-id}i" "%{x-xid-format-id}i" "%{x-transaction-context}i" "%{traceparent}i"
 …25
    server.tomcat.accesslog.locale=en_US
    server.tomcat.accesslog.max-days=7
    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.instance-id=Entity-Service-TCC_001
    jp.co.Hitachi.soft.hmppcto.tracing.url=http://jaeger:4318/v1/traces
    jp.co.Hitachi.soft.hmppcto.use-scaling=false
---
apiVersion: v1     …14
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     …10
    snapshot.log.filepath=/ucars-snapshots/${HOSTNAME}/snapshot
---
apiVersion: v1     …16
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      …4
kind: ServiceAccount
metadata:
  name: tcc-participant-service-account
  namespace: my-namespace
---
apiVersion: v1      …22
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※1

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。

注※2

通常版の場合は my-namespace/statefulsets/eads-server-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。

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。また、使用する Spring Boot の組み込み HTTP サーバを「Tomcat」としています。

1. 「prometheus.io/path」の値は、コンテキストパスを設定していない場合の指定値です。
   コンテキストパスを設定している場合は、次の形式で指定してください。

   ```
   prometheus.io/path: <コンテキストパス>/hmppcto/metrics
   ```

   コンテキストパスが/context-path の場合の指定例を次に示します。

   ```
   prometheus.io/path: /context-path/hmppcto/metrics
   ```

2. 「prometheus.io/port: '8080'」の行、「- name: http」の行から「protocol: TCP」の行、および「- name: http」の行から「targetPort: http」の行で、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。
3. メトリクスのスクレイプ時に使用する接続スキームを指定します。ユーザアプリケーションの暗号化通信を有効にする場合は、https を設定してください。
4. 「serviceAccount: tcc-participant-service-account」の行、および「apiVersion: v1」の行から「kind: ServiceAccount」の「namespace: my-namespace」の行で、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。
5. このオレンジ色の背景で示す行は、Google Cloud Platform 環境の場合だけ記述します。
   securityContext を設定します。fsGroup にはコンテナ実行グループと同等の設定をしてください。
6. このオレンジ色の背景で示す行は、コンテナイメージを Pull する際に資格情報が必要な場合に記述します。Red Hat OpenShift Container Platform 環境で資格情報が不要な場合は imagePullSecrets: [] を指定してください。
7. 「- name: dependency-checker」の行から「image: myregistry.example.com/myproject/dependency-checker:V.R.S」の行で、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。
8. 「- name: entity-module-tcc-participant」の行から「preStop:」の「- "8080"」の行で、コンテナ構成に Entity-Service(TCC)コンテナを定義します。「2.3.8　Entity-Service(TCC)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。
9. 「resources:」の行から「memory: 512Mi」の行で、Entity-Service(TCC)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。
   オートスケールで Metrics-Server を利用する場合は必ず指定してください。
10. このオレンジ色の背景で示す行（「- name: ucars」の行から「protocol: TCP」の行、「- name: ucars」の行から「targetPort: ucars」の行、および「monitor.rest.port=28081」の行）は、通常版（uCosminexus Application Runtime for Spring Boot 導入あり）の場合だけ記述します。
    uCosminexus Application Runtime 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。

11. 「- name: TZ」の行から「value: Asia/Tokyo」の行で、Entity-Service(TCC)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。

12. 「- name: HMPPCTO_JAVA_OPTIONS」の行から「value: -XX:+UseG1GC」の行で、Entity-Service(TCC)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。

13. 「- name: entity-module-tcc-participant-config」の行から「mountPath: /deployments/config」の行、「- name: entity-module-tcc-participant-config」の行から「path: application.properties」の行、および「apiVersion: v1」の行から「jp.co.Hitachi.soft.hmppcto.use-scaling=false」の行で、Entity-Service(TCC)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。

14. このオレンジ色の背景で示す行（「- name: ucars-config」の行から「mountPath: /opt/hitachi/ucars/conf」の行、「- name: ucars-config」の行から「path: config.properties」の行、および「apiVersion: v1」の行から「snapshot.log.filepath=/ucars-snapshots/${HOSTNAME}/snapshot」の行）は、通常版（uCosminexus Application Runtime for Spring Boot 導入あり）の場合だけ記述します。

    uCosminexus Application Runtime for Spring Boot の設定ファイル(config.properties)を Entity-Service(TCC)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、「11.6.2　情報の収集方法」を参照して設定してください。

15. このオレンジ色の背景で示す行（「- name: snapshot-volume」の行から「mountPath: /ucars-snapshots」の行、および「- name: snapshot-volume」の行から「claimName: snapshot-volume」の行）は、通常版（uCosminexus Application Runtime for Spring Boot 導入あり）の場合だけ記述します。

    uCosminexus Application Runtime for Spring Boot スナップショットログの永続ボリュームを Entity-Service(TCC)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、「2.6.11　uCosminexus Application Runtime 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 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。

16. 「- name: eads-config」の行から「mountPath: /eads-properties」の行、「- name: eads-config」の行から「path: hmp-pcto.eads-client.properties」の行、および「apiVersion: v1」の行から「eads.client.connectionPool.poolsize=1」の行で、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「9.2.6　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。

17. 「- name: tomcat」の行から「mountPath: /deployments/tomcat」の行、および「- name: tomcat」の行で、Tomcat のベースディレクトリとして /deployments/tomcat をマウントします。

18. 「- name: eads」の行から「mountPath: /deployments/eads-log」の行、および「- name: eads」の行で、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。

19. 「livenessProbe:」の行から「timeoutSeconds: 10」の行で、コンテナライフサイクル設定の Liveness Probe を定義します。

20. 「postStart:」の行から「- "8080"」の行で、コンテナライフサイクル設定の postStart フックを定義します。

21. 「preStop:」の行から「- "8080"」の行で、コンテナライフサイクル設定の preStop フックを定義します。

22. このオレンジ色の背景で示す行（「- name: filebeat-sidecar」の行から「mountPath: /deployments/eads-log」の行、「- name: beatconfig」の行から「name: tcc-p-filebeat-sidecar-config」の行、および「apiVersion: v1」の行から「hosts: ["logstash.my-namespace.svc.cluster.local:5044"]」の行）は、Elastic Stack を使用してトラブルシュート情報を収集する場合だけ記述します。

    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。

23. 「env:」の行から「fieldPath: spec.nodeName」の行、および「processors:」の行から「name: ${HMPPCTO_NODE_NAME}」の行で、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。

24. Pod ライフサイクル設定の terminationGracePeriodSeconds を定義します。

25. このオレンジ色の背景で示す行は、通常版（uCosminexus Application Runtime for Spring Boot 導入あり）の場合だけ記述します。

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。
- Entity-Service(TCC)を構成する Pod を定義した Kubernetes の Service の Service 名と Namespace 名を「.」でつないだ文字列が、41 バイト以内となるようにしてください。

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

「7.6.5 Orchestrator-Service(SQL)の要件（SQL-Participant 限定)」に記載されている要件を満たすように、Orchestrator-Service(SQL)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 マニフェストは、通常版とトライアル版で差異があります。トライアル版の Kubernetes マニフェストを基に通常版の Kubernetes マニフェストを作成する場合は、通常版の記述例に従って修正してください。

Kubernetes マニフェストの記述例を次に示します。

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: orchestrator-sql
  namespace: my-namespace
spec:
  replicas: 1     …1
  selector:
    matchLabels:
      app.kubernetes.io/name: orchestrator-sql     …2
  template:
    metadata:
      labels:
        app.kubernetes.io/name: orchestrator-sql
        hmppcto-app-group: x-system
      annotations:
        prometheus.io/scrape: 'true'
        prometheus.io/path: '/hmppcto/metrics'     …3
        prometheus.io/port: '8080'     …4
        prometheus.io/scheme: 'http'     …5
    spec:
      serviceAccount: orchestrator-sql-service-account     …6
      securityContext:     …7
        fsGroup: 0
      imagePullSecrets:     …8
      - name: regcred
      initContainers:
      - name: dependency-checker     …9
```

```
        image: myregistry.example.com/myproject/dependency-checker:V.R.S※1
        args:
        - "30"
        - "10"
        - "10"
        - "10"
        - my-namespace/statefulsets/mediator
        - my-namespace/statefulsets/eads-server-eads-server※2
        - my-namespace/statefulsets/postgresql
      containers:
      - name: orchestrator-sql     …10
        image: myregistry.example.com/myproject/orchestrator-sql:V.R.S※1
        resources:     …11
          limits:
            cpu: 2
            memory: 1Gi
          requests:
            cpu: 1
            memory: 512Mi
        ports:
        - name: http     …4
          containerPort: 8080
          protocol: TCP
        - name: ucars     …12
          containerPort: 28081
          protocol: TCP
        env:
        - name: TZ     …13
          value: Asia/Tokyo
        - name: HMPPCTO_JAVA_OPTIONS     …14
          value: -XX:+UseG1GC
        volumeMounts:
        - name: orchestrator-sql-config
          mountPath: /deployments/config
        - name: orchestrator-sql-ucars-config  …15
          mountPath: /opt/hitachi/ucars/conf
        - name: snapshot-volume  …16
          mountPath: /ucars-snapshots
        - name: eads-config     …17
          mountPath: /eads-properties
        - name: tomcat     …18
          mountPath: /deployments/tomcat
        - name: eads   …19
          mountPath: /deployments/eads-log
        livenessProbe:     …20
          exec:
            command:
            - /deployments/health_check.sh
            - "0"
            - "0"
            - "10"
            - "10"
            - "8080"
          initialDelaySeconds: 180
          periodSeconds: 30
          failureThreshold: 3
          timeoutSeconds: 10
        lifecycle:
```

```
          postStart:     …21
            exec:
              command:
              - /deployments/health_check.sh
              - "30"
              - "10"
              - "10"
              - "10"
              - "8080"
          preStop:     …22
            exec:
              command:
              - /deployments/wait_transactions.sh
              - "10"
              - "10"
              - "10"
              - "10"
              - "8080"
      - name: filebeat-sidecar     …23
        image: myregistry.example.com/myproject/filebeat:V.R.S※1
        resources:     …24
          limits:
            cpu: 1000m
            memory: 2Gi
          requests:
            cpu: 1000m
            memory: 2Gi
        args:
        - "-e"
        - "-E"
        - "http.enabled=true"
        env:     …25
        - 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      …18
              mountPath: /deployments/tomcat
            - name: eads      …19
              mountPath: /deployments/eads-log
      terminationGracePeriodSeconds: 180      …26
      volumes:
      - name: snapshot-volume      …16
        persistentVolumeClaim:
          claimName: snapshot-volume
      - name: orchestrator-sql-config
        configMap:
          name: orchestrator-sql-config
          items:
          - key: application.properties
            path: application.properties
      - name: orchestrator-sql-ucars-config      …15
        configMap:
          name: orchestrator-sql-ucars-config
          items:
            - key: config.properties
              path: config.properties
      - name: eads-config      …17
        configMap:
          name: eads-config
          items:
          - key: hmp-pcto.eads-client.properties
            path: hmp-pcto.eads-client.properties
      - name: beatconfig      …23
        configMap:
          name: orchestrator-sql-filebeat-sidecar-config
      - name: tomcat      …18
      - name: eads      …19
---
apiVersion: v1
kind: Service
metadata:
  name: orchestrator-sql
  namespace: my-namespace
  labels:
    hmppcto-app-group: x-system
spec:
```

```
  type: ClusterIP
  selector:
    app.kubernetes.io/name: orchestrator-sql
  ports:
  - name: http     …4
    protocol: TCP
    port: 8080
    targetPort: http
  - name: ucars     …12
    protocol: TCP
    port: 28081
    targetPort: ucars
---
apiVersion: v1     …27
kind: ConfigMap
metadata:
  name: orchestrator-sql-config
  namespace: my-namespace
data:
  application.properties: |
    server.port=8080
    server.tomcat.accesslog.enabled=true
    server.tomcat.basedir=tomcat
    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-xid-transaction-id}i" "
%{x-xid-branch-id}i" "%{x-xid-format-id}i" "%{x-transaction-context}i" "%{traceparent}i"
 …28
    server.tomcat.accesslog.locale=en_US
    server.tomcat.accesslog.max-days=7
    server.shutdown=graceful
    spring.lifecycle.timeout-per-shutdown-phase=30s
    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=orchestratorsql
    jp.co.Hitachi.soft.hmppcto.client.proposer.type=SQL
    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=pctodriver
    spring.datasource.username=dummy
    spring.datasource.password=dummy
    spring.sql.init.mode=never
    jp.co.Hitachi.soft.hmppcto.sql.pod.service-name=sp.my-namespace
    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.port=20551
    jp.co.Hitachi.soft.hmppcto.sql.db-type=PostgreSQL     …29
    jp.co.Hitachi.soft.hmppcto.tracing.instance-id=Orchestrator-Service-SQL_001
    jp.co.Hitachi.soft.hmppcto.tracing.url=http://jaeger:4318/v1/traces
    jp.co.Hitachi.soft.hmppcto.use-scaling=true
    jp.co.Hitachi.soft.hmppcto.set-routing=inner
---
apiVersion: v1 …15
kind: ConfigMap
```

```
metadata:
  name: orchestrator-sql-ucars-config
  namespace: my-namespace
data:
  config.properties: |
    monitor.rest.bindaddress=0.0.0.0
    monitor.rest.port=28081 …12
    snapshot.log.filepath=/ucars-snapshots/${HOSTNAME}/snapshot
    healthcheck.startdelay.timeout=90000
    healthcheck.startdelay.actions.failure.retrymax=4
---
apiVersion: v1     …17
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     …6
kind: ServiceAccount
metadata:
  name: orchestrator-sql-service-account
  namespace: my-namespace
---
apiVersion: v1     …23
kind: ConfigMap
metadata:
  name: orchestrator-sql-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:     …25
      - 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※1

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。

注※2

通常版の場合は my-namespace/statefulsets/eads-server-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。

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。また、使用する Spring Boot の組み込み HTTP サーバを「Tomcat」としています。

1. オートスケールを使用しない場合だけ、replicas を設定します。
2. 「app.kubernetes.io/name」を必ず指定してください。

   「app.kubernetes.io/name」は、ほかの Kubernetes アプリケーションに設定した値と異なる値を設定してください。

   16 バイト以内の文字列を指定してください。

   Blue/Green デプロイメントを実施する場合は、Blue と Green の「app.kubernetes.io/name」には同一の値を設定してください。
3. 「prometheus.io/path」の値は、コンテキストパスを設定していない場合の指定値です。

   コンテキストパスを設定している場合は、次の形式で指定してください。

   ```
   prometheus.io/path: <コンテキストパス>/hmppcto/metrics
   ```

   コンテキストパスが/context-path の場合の指定例を次に示します。

   ```
   prometheus.io/path: /context-path/hmppcto/metrics
   ```
4. 「prometheus.io/port: '8080'」の行、「- name: http」の行から「protocol: TCP」の行、および「- name: http」の行から「targetPort: http」の行で、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. メトリクスのスクレイプ時に使用する接続スキームを指定します。ユーザアプリケーションの暗号化通信を有効にする場合は、https を設定してください。

6. 「serviceAccount: orchestrator-sql-service-account」の行、および「apiVersion: v1」の行から「kind: ServiceAccount」の「namespace: my-namespace」の行で、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. このオレンジ色の背景で示す行は、Google Cloud Platform 環境の場合だけ記述します。securityContext を設定します。fsGroup にはコンテナ実行グループと同等の設定をしてください。
8. このオレンジ色の背景で示す行は、コンテナイメージを Pull する際に資格情報が必要な場合に記述します。Red Hat OpenShift Container Platform 環境で資格情報が不要な場合は imagePullSecrets: [] を指定してください。
9. 「- name: dependency-checker」の行から「- my-namespace/statefulsets/postgresql」の行で、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。

   Dependency-Checker の対象として接続する RDBMS の Kubernetes オブジェクトを指定する必要があります。

   この例では、「- my-namespace/statefulsets/postgresql」を指定しています。
10. 「- name: orchestrator-sql」の行から「preStop:」の「- "8080"」の行で、コンテナ構成に Orchestrator-Service(SQL)コンテナを定義します。「2.3.9　Orchestrator-Service(SQL)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。
11. 「resources:」の行から「memory: 512Mi」の行で、Orchestrator-Service(SQL)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。

    オートスケールのメトリクスの監視に Metrics Server を使用する場合は必ず指定してください。
12. このオレンジ色の背景で示す行（「- name: ucars」の行から「protocol: TCP」の行、「- name: ucars」の行から「targetPort: ucars」の行、および「monitor.rest.port=28081」の行）は、通常版（uCosminexus Application Runtime for Spring Boot 導入あり）の場合だけ記述します。

    uCosminexus Application Runtime 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。
13. 「- name: TZ」の行から「value: Asia/Tokyo」の行で、Orchestrator-Service(SQL)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. 「- name: HMPPCTO_JAVA_OPTIONS」の行から「value: -XX:+UseG1GC」の行で、Orchestrator-Service(SQL)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。
15. このオレンジ色の背景で示す行（「- name: orchestrator-sql-ucars-config」の行から「mountPath: /opt/hitachi/ucars/conf」の行、「- name: orchestrator-sql-ucars-config」の行から「path: config.properties」の行、および「apiVersion: v1」の行から「healthcheck.startdelay.actions.failure.retrymax=4」の行）は、通常版（uCosminexus Application Runtime for Spring Boot 導入あり）の場合だけ記述します。

uCosminexus Application Runtime for Spring Boot の設定ファイル(config.properties)をOrchestrator-Service(SQL)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、「11.6.2　情報の収集方法」を参照して設定してください。

- healthcheck.startdelay.timeout=90000

  モニタ対象プロセス（Orchestrator-Service(SQL)）開始完了通知待ちタイムアウト時間を定義します。Orchestrator-Service(SQL)の起動時に、設定した時間が経過すると snapshot.log.filepath で指定した先にスナップショットを出力します。モニタ対象プロセス開始完了通知待ちタイムアウト時間の詳細は、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。

- healthcheck.startdelay.actions.failure.retrymax=4

  モニタ対象プロセス（Orchestrator-Service(SQL)）開始完了通知待ちタイムアウトが発生した場合に、モニタ対象プロセス開始完了待ちをリトライする最大回数を定義します。最大リトライ回数を超えると、デフォルトの設定ではモニタ対象プロセスを停止します。モニタ対象プロセス開始完了待ちリトライ回数の詳細は、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。

  モニタ対象プロセス開始完了通知待ちタイムアウトの合計時間（下記の A）と Orchestrator-Service(SQL)の開始に掛かる時間（下記の B、および C）の関係を、「A ＞ B ＞ C」にしてください。

  A：モニタ対象プロセス開始完了通知待ちタイムアウトの合計時間

  ```
  healthcheck.startdelay.timeoutに指定した値×（healthcheck.startdelay.actions.failure.retrymaxに指定した値＋1）
  ```

  B：postStart フックに指定するトランザクションヘルスチェックスクリプト（health_check.sh）の合計時間

  ```
  リトライ回数×リトライ間隔秒数
  ```

  C：コネクションプールの初期化で、DB とのコネクションの取得に失敗した場合に行うリトライの合計時間

```
sql.pool.connection-retry-interval-in-secondsに指定した値×sql.pool.connection-retry-limitに指定した値
```

A を B よりも大きな値にするのは、モニタ対象プロセス開始完了通知待ちタイムアウトによるプロセス停止が、postStart フックに指定するトランザクションヘルスチェックスクリプトによる Orchestrator-Service(SQL)の開始完了待ちを中止することを防ぐためです。

B を C よりも大きな値とするのは、postStart フックに指定するトランザクションヘルスチェックスクリプトのタイムアウトによる Orchestrator-Service(SQL)の停止が、Orchestrator-Service(SQL)のコネクション接続処理を中止することを防ぐためです。

この記述例では、Orchestrator-Service(SQL)の開始に掛かる時間（300 秒）の間に、スナップショットを複数回（3 回）出力するように設定しています。

16. このオレンジ色の背景で示す行（「- name: snapshot-volume」の行から「mountPath: /ucars-snapshots」の行、および「- name: snapshot-volume」の行から「claimName: snapshot-volume」の行）は、通常版（uCosminexus Application Runtime for Spring Boot 導入あり）の場合だけ記述します。

    uCosminexus Application Runtime for Spring Boot スナップショットログの永続ボリュームを Orchestrator-Service(SQL)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、「2.6.11　uCosminexus Application Runtime 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 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。

17. 「- name: eads-config」の行から「mountPath: /eads-properties」の行、「- name: eads-config」の行から「path: hmp-pcto.eads-client.properties」の行、および「apiVersion: v1」の行から「eads.client.connectionPool.poolsize=1」の行で、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「9.2.5　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。

18. 「- name: tomcat」の行から「mountPath: /deployments/tomcat」の行、および「- name: tomcat」の行で、Tomcat のベースディレクトリとして /deployments/tomcat をマウントします。

19. 「- name: eads」の行から「mountPath: /deployments/eads-log」の行、および「- name: eads」の行で、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。

20. 「livenessProbe:」の行から「timeoutSeconds: 10」の行で、コンテナライフサイクル設定の Liveness Probe を定義します。

21. 「postStart:」の行から「- "8080"」の行で、コンテナライフサイクル設定の postStart フックを定義します。

22.「preStop:」の行から「- "8080"」の行で、コンテナライフサイクル設定の preStop フックを定義します。

23. このオレンジ色の背景で示す行（「- name: filebeat-sidecar」の行から「mountPath: /deployments/eads-log」の行、「- name: beatconfig」の行から「name: orchestrator-sql-filebeat-sidecar-config」の行、および「apiVersion: v1」の行から「hosts: ["logstash.my-namespace.svc.cluster.local:5044"]」の行）は、Elastic Stack を使用してトラブルシュート情報を収集する場合だけ記述します。

    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。

24.「resources:」の行から「requests:」の「memory: 2Gi」の行で、Filebeat 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。

    オートスケールのメトリクスの監視に Metrics Server を使用する場合は必ず指定してください。

25.「env:」の行から「fieldPath: spec.nodeName」の行、および「processors:」の行から「name: ${HMPPCTO_NODE_NAME}」の行で、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。

26. Pod ライフサイクル設定の terminationGracePeriodSeconds を定義します。

27.「apiVersion: v1」の行から「jp.co.Hitachi.soft.hmppcto.set-routing=inner」の行で、Orchestrator-Service(SQL)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。

28. このオレンジ色の背景で示す行は、通常版（uCosminexus Application Runtime for Spring Boot 導入あり）の場合だけ記述します。

29. Orchestrator-Service(SQL)が接続する RDBMS を指定します。PostgreSQL を指定する場合、「9.4.1　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。
- Orchestrator-Service(SQL)を構成する Pod を定義した Kubernetes の Service の Service 名と Namespace 名を「.」でつないだ文字列が、41 バイト以内となるようにしてください。

2.6.7　Alternate-Service の Kubernetes マニフェストの作成

「7.6.6　Alternate-Service の要件」に記載されている要件を満たすように、Alternate-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
kind: Deployment
metadata:
  name: alternate-service
  namespace: my-namespace
spec:
  replicas: 1     …1
  selector:
    matchLabels:
      app.kubernetes.io/name: alternate-service
  template:
    metadata:
      labels:
        app.kubernetes.io/name: alternate-service
        hmppcto-app-group: x-system
      annotations:
        prometheus.io/scrape: 'true'
        prometheus.io/path: '/hmppcto/metrics'      …2
        prometheus.io/port: '8080'      …3
        prometheus.io/scheme: 'http'      …4
    spec:
      serviceAccount: alternate-service-service-account      …5
      securityContext:      …6
        fsGroup: 0
      imagePullSecrets:      …7
      - name: regcred
      initContainers:
      - name: dependency-checker      …8
        image: myregistry.example.com/myproject/dependency-checker:V.R.S※1
        args:
        - "30"
        - "10"
        - "10"
        - "10"
        - my-namespace/statefulsets/mediator
        - my-namespace/deployments/sql-participant※2
      containers:
      - name: alternate-service      …9
        image: myregistry.example.com/myproject/alternate-service:V.R.S※1
        resources:      …10
          limits:
            cpu: 2
            memory: 1Gi
          requests:
            cpu: 1
            memory: 512Mi
        ports:
        - name: http
          containerPort: 8080
```

```
          protocol: TCP
        - name: ucars      …11
          containerPort: 28081
          protocol: TCP
        env:
        - name: TZ      …12
          value: Asia/Tokyo
        - name: HMPPCTO_JAVA_OPTIONS      …13
          value: -XX:+UseG1GC
        volumeMounts:
        - name: config      …14
          mountPath: /deployments/config
        - name: ucars-config      …15
          mountPath: /opt/hitachi/ucars/conf
        - name: snapshot-volume      …16
          mountPath: /ucars-snapshots
        - name: tomcat      …17
          mountPath: /deployments/tomcat
        livenessProbe:      …18
          exec:
            command:
            - /deployments/health_check.sh
            - "0"
            - "0"
            - "10"
            - "10"
            - "8080"
          initialDelaySeconds: 180
          periodSeconds: 30
          failureThreshold: 3
          timeoutSeconds: 10
        lifecycle:
          postStart:      …19
            exec:
              command:
              - /deployments/health_check.sh
              - "30"
              - "10"
              - "10"
              - "10"
              - "8080"
          preStop:      …20
            exec:
              command:
              - /deployments/wait_transactions.sh
              - "10"
              - "10"
              - "10"
              - "10"
              - "8080"
      - name: filebeat-sidecar      …21
        image: myregistry.example.com/myproject/filebeat:V.R.S※1
        resources:      …22
          limits:
            cpu: 1000m
            memory: 2Gi
          requests:
            cpu: 1000m
```

```
            memory: 2Gi
        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
        securityContext:
          runAsUser: 0
        volumeMounts:
         - name: beatconfig
           mountPath: /usr/share/filebeat/filebeat.yml
           readOnly: true
           subPath: filebeat.yml
         - name: tomcat      …17
           mountPath: /deployments/tomcat
      terminationGracePeriodSeconds: 180      …24
      volumes:
      - name: config      …14
        configMap:
          name: alternate-service-config
          items:
          - key: application.properties
```

```
            path: application.properties
      - name: ucars-config      …15
        configMap:
          name: alternate-service-ucars-config
          items:
            - key: config.properties
              path: config.properties
      - name: snapshot-volume      …16
        persistentVolumeClaim:
          claimName: snapshot-volume
      - name: beatconfig     …21
        configMap:
          name: alternate-service-filebeat-sidecar-config
      - name: tomcat      …17
---
apiVersion: v1
kind: Service
metadata:
  name: alternate-service
  namespace: my-namespace
  labels:
    hmppcto-app-group: x-system
spec:
  type: ClusterIP
  selector:
    app.kubernetes.io/name: alternate-service
  ports:
  - name: http      …3
    protocol: TCP
    port: 8080
    targetPort: http
  - name: ucars     …11
    protocol: TCP
    port: 28081
    targetPort: ucars
---
apiVersion: v1      …14
kind: ConfigMap
metadata:
  name: alternate-service-config
  namespace: my-namespace
data:
  application.properties: |
    server.port=8080
    server.tomcat.accesslog.enabled=true
    server.tomcat.basedir=tomcat
    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-xid-transaction-id}i" "
%{x-xid-branch-id}i" "%{x-xid-format-id}i" "%{x-transaction-context}i" "%{traceparent}i"
 …25
    server.tomcat.accesslog.locale=en_US
    server.tomcat.accesslog.max-days=7
    server.shutdown=graceful
    spring.lifecycle.timeout-per-shutdown-phase=30s
    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=alternate-service
    jp.co.Hitachi.soft.hmppcto.tracing.instance-id=Alternate-Service_001
    jp.co.Hitachi.soft.hmppcto.tracing.url=http://jaeger:4318/v1/traces
    jp.co.Hitachi.soft.hmppcto.client.proposer.type=SQL
    jp.co.Hitachi.soft.hmppcto.use-scaling=true
    jp.co.Hitachi.soft.hmppcto.set-routing=inner
---
apiVersion: v1      …15
kind: ConfigMap
metadata:
  name: alternate-service-ucars-config
  namespace: my-namespace
data:
  config.properties: |
    monitor.rest.bindaddress=0.0.0.0
    monitor.rest.port=28081      …11
    snapshot.log.filepath=/ucars-snapshots/${HOSTNAME}/snapshot
    healthcheck.startdelay.timeout=60000
    healthcheck.startdelay.actions.failure.retrymax=4
---
apiVersion: v1      …21
kind: ConfigMap
metadata:
  name: alternate-service-filebeat-sidecar-config
  namespace: my-namespace
data:
  filebeat.yml: |
    filebeat.inputs:
      - 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"]
---
apiVersion: v1      …5
kind: ServiceAccount
metadata:
  name: alternate-service-service-account
  namespace: my-namespace
```

注※1

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。

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

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。また、使用する Spring Boot の組み込み HTTP サーバを「Tomcat」としています。

1. オートスケールを使用しない場合だけ、replicas を設定します。
2. 「prometheus.io/path」の値は、コンテキストパスを設定していない場合の指定値です。
   コンテキストパスを設定している場合は、次の形式で指定してください。

   ```
   prometheus.io/path: <コンテキストパス>/hmppcto/metrics
   ```

   コンテキストパスが/context-path の場合の指定例を次に示します。

   ```
   prometheus.io/path: /context-path/hmppcto/metrics
   ```

3. 「prometheus.io/port: '8080'」の行、および「- name: http」の行から「targetPort: http」の行で、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。
4. メトリクスのスクレイプ時に使用する接続スキームを指定します。ユーザアプリケーションの暗号化通信を有効にする場合は、https を設定してください。
5. 「serviceAccount: alternate-service-service-account」の行、および「apiVersion: v1」の行から「namespace: my-namespace」の行で、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。
6. このオレンジ色の背景で示す行は、Google Cloud Platform 環境の場合だけ記述します。
   securityContext を設定します。fsGroup にはコンテナ実行グループと同等の設定をしてください。
7. このオレンジ色の背景で示す行は、コンテナイメージを Pull する際に資格情報が必要な場合に記述します。Red Hat OpenShift Container Platform 環境で資格情報が不要な場合は imagePullSecrets: [] を指定してください。

8. 「- name: dependency-checker」の行から「- my-namespace/deployments/sql-participant」の行で、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。
9. 「- name: alternate-service」の行から「preStop:」の「- "8080"」の行で、コンテナ構成に Alternate-Service コンテナを定義します。「2.3.10　Alternate-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。
10. 「resources:」の行から「memory: 512Mi」の行で、Alternate-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。

    オートスケールのメトリクスの監視に Metrics Server を使用する場合は必ず指定してください。
11. このオレンジ色の背景で示す行（「- name: ucars」の行から「protocol: TCP」の行、「- name: ucars」の行から「targetPort: ucars」の行、および「monitor.rest.port=28081」の行）は、通常版（uCosminexus Application Runtime for Spring Boot 導入あり）の場合だけ記述します。

    uCosminexus Application Runtime 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。
12. 「- name: TZ」の行から「value: Asia/Tokyo」の行で、Alternate-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。
13. 「- name: HMPPCTO_JAVA_OPTIONS」の行から「value: -XX:+UseG1GC」の行で、Alternate-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。
14. 「- name: config」の行から「mountPath: /deployments/config」の行、「- name: config」の行から「path: application.properties」の行、および「apiVersion: v1」の行から「jp.co.Hitachi.soft.hmppcto.set-routing=inner」の行で、Alternate-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、設定するパラメタは最小限の例としています。
15. このオレンジ色の背景で示す行（「- name: ucars-config」の行から「mountPath: /opt/hitachi/ucars/conf」の行、「- name: ucars-config」の行から「path: config.properties」の行、および「apiVersion: v1」の行から「healthcheck.startdelay.actions.failure.retrymax=4」の行）は、通常版（uCosminexus Application Runtime for Spring Boot 導入あり）の場合だけ記述します。

    uCosminexus Application Runtime for Spring Boot の設定ファイル(config.properties)を Alternate-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、「11.6.2　情報の収集方法」を参照して設定してください。

- healthcheck.startdelay.timeout=60000

  モニタ対象プロセス（Alternate-Service）開始完了通知待ちタイムアウト時間を定義します。Alternate-Service の起動時に、設定した時間が経過すると snapshot.log.filepath で指定した先にスナップショットを出力します。モニタ対象プロセス開始完了通知待ちタイムアウト時間の詳細は、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。

- healthcheck.startdelay.actions.failure.retrymax=4

  モニタ対象プロセス（Alternate-Service）開始完了通知待ちタイムアウトが発生した場合に、モニタ対象プロセス開始完了待ちをリトライする最大回数を定義します。最大リトライ回数を超えると、デフォルトの設定ではモニタ対象プロセスを停止します。モニタ対象プロセス開始完了待ちリトライ回数の詳細は、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。

  モニタ対象プロセス開始完了通知待ちタイムアウトの合計時間（下記の A）と Alternate-Service の開始に掛かる時間（下記の B）の関係を、「A ＞ B」にしてください。

  A：モニタ対象プロセス開始完了通知待ちタイムアウトの合計時間

  ```
  healthcheck.startdelay.timeoutに指定した値×（healthcheck.startdelay.actions.failure.retrymaxに指定した値＋1）
  ```

  B：Alternate-Service 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のリトライ時間

  ```
  jp.co.Hitachi.soft.hmppcto.mediator.discovery-interval-in-millisに指定した値×jp.co.Hitachi.soft.hmppcto.mediator.discovery-retry-countに指定した値
  ```

  A を B よりも大きな値とするのは、モニタ対象プロセス開始完了通知待ちタイムアウトによるプロセス停止が、Alternate-Service 起動時の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処理を中止することを防ぐためです。

  この記述例では、Alternate-Service の開始に掛かる時間（180 秒）の間に、スナップショットを複数回（3 回）出力するように設定しています。

16. このオレンジ色の背景で示す行（「- name: snapshot-volume」の行から「mountPath: /ucars-snapshots」の行、および「- name: snapshot-volume」の行から「claimName: snapshot-volume」の行）は、通常版（uCosminexus Application Runtime for Spring Boot 導入あり）の場合だけ記述します。

    uCosminexus Application Runtime for Spring Boot スナップショットログの永続ボリュームを Alternate-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、「2.6.11 uCosminexus Application Runtime 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 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。

17. 「- name: tomcat」の行から「mountPath: /deployments/tomcat」の行、および「- name: tomcat」の行で、Tomcat のベースディレクトリとして /deployments/tomcat をマウントします。
18. 「livenessProbe:」の行から「timeoutSeconds: 10」の行で、コンテナライフサイクル設定の Liveness Probe を定義します。
19. 「postStart:」の行から「- "8080"」の行で、コンテナライフサイクル設定の postStart フックを定義します。
20. 「preStop:」の行から「- "8080"」の行で、コンテナライフサイクル設定の preStop フックを定義します。
21. このオレンジ色の背景で示す行（「- name: filebeat-sidecar」の行から「mountPath: /deployments/tomcat」の行、「- name: beatconfig」の行から「name: alternate-service-filebeat-sidecar-config」の行、および「apiVersion: v1」の行から「hosts: ["logstash.my-namespace.svc.cluster.local:5044"]」の行）は、Elastic Stack を使用してトラブルシュート情報を収集する場合だけ記述します。

    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。
22. 「resources:」の行から「requests:」の「memory: 2Gi」の行で、Filebeat 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。

    オートスケールのメトリクスの監視に Metrics Server を使用する場合は必ず指定してください。
23. 「env:」の行から「fieldPath: spec.nodeName」の行、および「processors:」の行から「name: ${HMPPCTO_NODE_NAME}」の行で、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。
24. Pod ライフサイクル設定の terminationGracePeriodSeconds を定義します。
25. このオレンジ色の背景で示す行は、通常版（uCosminexus Application Runtime for Spring Boot 導入あり）の場合だけ記述します。

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。

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

「7.6.7 Alternate-Service(SQL)の要件（SQL-Participant 限定）」に記載されている要件を満たすように、Alternate-Service(SQL)）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 マニフェストは、通常版とトライアル版で差異があります。トライアル版の Kubernetes マニフェストを基に通常版の Kubernetes マニフェストを作成する場合は、通常版の記述例に従って修正してください。

Kubernetes マニフェストの記述例を次に示します。

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: alternate-service-sql
  namespace: my-namespace
spec:
  replicas: 1     …1
  selector:
    matchLabels:
      app.kubernetes.io/name: alt-service-sql     …2
  template:
    metadata:
      labels:
        app.kubernetes.io/name: alt-service-sql
        hmppcto-app-group: x-system
      annotations:
        prometheus.io/scrape: 'true'
        prometheus.io/path: '/hmppcto/metrics'      …3
        prometheus.io/port: '8080'      …4
        prometheus.io/scheme: 'http'      …5
    spec:
      serviceAccount: alternate-service-sql-service-account      …6
      securityContext:      …7
        fsGroup: 0
      imagePullSecrets:      …8
      - name: regcred
      initContainers:
      - name: dependency-checker      …9
        image: myregistry.example.com/myproject/dependency-checker:V.R.S※1
        args:
        - "30"
        - "10"
        - "10"
        - "10"
        - my-namespace/statefulsets/mediator
        - my-namespace/statefulsets/eads-server-eads-server※2
        - my-namespace/statefulsets/postgresql
      containers:
      - name: alternate-service-sql      …10
        image: myregistry.example.com/myproject/alternate-service-sql:V.R.S※1
        resources:      …11
          limits:
```

```
            cpu: 2
            memory: 1Gi
          requests:
            cpu: 1
            memory: 512Mi
        ports:
        - name: http     …4
          containerPort: 8080
          protocol: TCP
        - name: ucars     …12
          containerPort: 28081
          protocol: TCP
        env:
        - name: TZ     …13
          value: Asia/Tokyo
        - name: HMPPCTO_JAVA_OPTIONS     …14
          value: -XX:+UseG1GC
        volumeMounts:
        - name: alternate-service-sql-config
          mountPath: /deployments/config
        - name: alternate-service-sql-ucars-config  …15
          mountPath: /opt/hitachi/ucars/conf
        - name: snapshot-volume  …16
          mountPath: /ucars-snapshots
        - name: eads-config     …17
          mountPath: /eads-properties
        - name: tomcat     …18
          mountPath: /deployments/tomcat
        - name: eads   …19
          mountPath: /deployments/eads-log
        livenessProbe:     …20
          exec:
            command:
            - /deployments/health_check.sh
            - "0"
            - "0"
            - "10"
            - "10"
            - "8080"
          initialDelaySeconds: 180
          periodSeconds: 30
          failureThreshold: 3
          timeoutSeconds: 10
        lifecycle:
          postStart:     …21
            exec:
              command:
              - /deployments/health_check.sh
              - "30"
              - "10"
              - "10"
              - "10"
              - "8080"
          preStop:     …22
            exec:
              command:
              - /deployments/wait_transactions.sh
              - "10"
```

```
            - "10"
            - "10"
            - "10"
            - "8080"
      - name: filebeat-sidecar      …23
        image: myregistry.example.com/myproject/filebeat:V.R.S※1
        resources:      …24
          limits:
            cpu: 1000m
            memory: 2Gi
          requests:
            cpu: 1000m
            memory: 2Gi
        args:
        - "-e"
        - "-E"
        - "http.enabled=true"
        env:      …25
        - 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      …18
            mountPath: /deployments/tomcat
          - name: eads      …19
            mountPath: /deployments/eads-log
      terminationGracePeriodSeconds: 180      …26
      volumes:
      - name: snapshot-volume      …16
        persistentVolumeClaim:
          claimName: snapshot-volume
      - name: alternate-service-sql-config
        configMap:
          name: alternate-service-sql-config
          items:
          - key: application.properties
            path: application.properties
      - name: alternate-service-sql-ucars-config …15
        configMap:
          name: alternate-service-sql-ucars-config
          items:
            - key: config.properties
              path: config.properties
      - name: eads-config      …17
        configMap:
          name: eads-config
          items:
          - key: hmp-pcto.eads-client.properties
            path: hmp-pcto.eads-client.properties
      - name: beatconfig      …23
        configMap:
          name: alternate-service-sql-filebeat-sidecar-config
      - name: tomcat      …18
      - name: eads      …19
---
apiVersion: v1
kind: Service
metadata:
  name: alternate-service-sql
  namespace: my-namespace
  labels:
    hmppcto-app-group: x-system
spec:
  type: ClusterIP
  selector:
    app.kubernetes.io/name: alt-service-sql
  ports:
  - name: http      …4
    protocol: TCP
    port: 8080
    targetPort: http
  - name: ucars      …12
    protocol: TCP
    port: 28081
    targetPort: ucars
---
apiVersion: v1      …27
```

```
kind: ConfigMap
metadata:
  name: alternate-service-sql-config
  namespace: my-namespace
data:
  application.properties: |
    server.port=8080
    server.tomcat.accesslog.enabled=true
    server.tomcat.basedir=tomcat
    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-xid-transaction-id}i" "
%{x-xid-branch-id}i" "%{x-xid-format-id}i" "%{x-transaction-context}i" "%{traceparent}i"
 …28
    server.tomcat.accesslog.locale=en_US
    server.tomcat.accesslog.max-days=7
    server.shutdown=graceful
    spring.lifecycle.timeout-per-shutdown-phase=30s
    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=alternate-service-sql
    jp.co.Hitachi.soft.hmppcto.client.proposer.type=SQL
    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=pctodriver
    spring.datasource.username=dummy
    spring.datasource.password=dummy
    spring.sql.init.mode=never
    jp.co.Hitachi.soft.hmppcto.sql.pod.service-name=sp.my-namespace
    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.port=20551
    jp.co.Hitachi.soft.hmppcto.sql.db-type=PostgreSQL     …29
    jp.co.Hitachi.soft.hmppcto.tracing.instance-id=Alternate-Service-SQL_001
    jp.co.Hitachi.soft.hmppcto.tracing.url=http://jaeger:4318/v1/traces
    jp.co.Hitachi.soft.hmppcto.use-scaling=true
    jp.co.Hitachi.soft.hmppcto.set-routing=inner
---
apiVersion: v1 …15
kind: ConfigMap
metadata:
  name: alternate-service-sql-ucars-config
  namespace: my-namespace
data:
  config.properties: |
    monitor.rest.bindaddress=0.0.0.0
    monitor.rest.port=28081 …12
    snapshot.log.filepath=/ucars-snapshots/${HOSTNAME}/snapshot
    healthcheck.startdelay.timeout=90000
    healthcheck.startdelay.actions.failure.retrymax=4
---
apiVersion: v1     …17
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     …6
kind: ServiceAccount
metadata:
  name: alternate-service-sql-service-account
  namespace: my-namespace
---
apiVersion: v1     …23
kind: ConfigMap
metadata:
  name: alternate-service-sql-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:     …25
      - 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※1

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。

注※2

通常版の場合は my-namespace/statefulsets/eads-server-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。

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。また、使用する Spring Boot の組み込み HTTP サーバを「Tomcat」としています。

1. オートスケールを使用しない場合だけ、replicas を設定します。
2. 「app.kubernetes.io/name」を必ず指定してください。

   「app.kubernetes.io/name」は、ほかの Kubernetes アプリケーションに設定した値と異なる値を設定してください。

   16 バイト以内の文字列を指定してください。

   Blue/Green デプロイメントを実施する場合は、Blue と Green の「app.kubernetes.io/name」には同一の値を設定してください。
3. 「prometheus.io/path」の値は、コンテキストパスを設定していない場合の指定値です。

   コンテキストパスを設定している場合は、次の形式で指定してください。

   ```
   prometheus.io/path: <コンテキストパス>/hmppcto/metrics
   ```

   コンテキストパスが/context-path の場合の指定例を次に示します。

   ```
   prometheus.io/path: /context-path/hmppcto/metrics
   ```
4. 「prometheus.io/port: '8080'」の行、「- name: http」の行から「protocol: TCP」の行、および「- name: http」の行から「targetPort: http」の行で、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. メトリクスのスクレイプ時に使用する接続スキームを指定します。ユーザアプリケーションの暗号化通信を有効にする場合は、https を設定してください。
6. 「serviceAccount: alternate-service-sql-service-account」の行、および「apiVersion: v1」の行から「kind: ServiceAccount」の「namespace: my-namespace」の行で、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. このオレンジ色の背景で示す行は、Google Cloud Platform 環境の場合だけ記述します。
   securityContext を設定します。fsGroup にはコンテナ実行グループと同等の設定をしてください。
8. このオレンジ色の背景で示す行は、コンテナイメージを Pull する際に資格情報が必要な場合に記述します。Red Hat OpenShift Container Platform 環境で資格情報が不要な場合は imagePullSecrets: [] を指定してください。
9. 「- name: dependency-checker」の行から「- my-namespace/statefulsets/postgresql」の行で、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。

Dependency-Checker の対象として接続する RDBMS の Kubernetes オブジェクトを指定する必要があります。

この例では、「- my-namespace/statefulsets/postgresql」を指定しています。

10. 「- name: alternate-service-sql」の行から「preStop:」の「- "8080"」の行で、コンテナ構成に Alternate-Service(SQL)コンテナを定義します。「2.3.11　Alternate-Service(SQL)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。

11. 「resources:」の行から「memory: 512Mi」の行で、Alternate-Service(SQL)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。

    オートスケールのメトリクスの監視に Metrics Server を使用する場合は必ず指定してください。

12. このオレンジ色の背景で示す行（「- name: ucars」の行から「protocol: TCP」の行、「- name: ucars」の行から「targetPort: ucars」の行、および「monitor.rest.port=28081」の行）は、通常版（uCosminexus Application Runtime for Spring Boot 導入あり）の場合だけ記述します。

    uCosminexus Application Runtime 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。

13. 「- name: TZ」の行から「value: Asia/Tokyo」の行で、Alternate-Service(SQL)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. 「- name: HMPPCTO_JAVA_OPTIONS」の行から「value: -XX:+UseG1GC」の行で、Alternate-Service(SQL)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。

15. このオレンジ色の背景で示す行（「- name: alternate-service-sql-ucars-config」の行から「mountPath: /opt/hitachi/ucars/conf」の行、「- name: alternate-service-sql-ucars-config」の行から「path: config.properties」の行、および「apiVersion: v1」の行から「healthcheck.startdelay.actions.failure.retrymax=4」の行）は、通常版（uCosminexus Application Runtime for Spring Boot 導入あり）の場合だけ記述します。

    uCosminexus Application Runtime for Spring Boot の設定ファイル(config.properties)を Alternate-Service(SQL)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、「11.6.2　情報の収集方法」を参照して設定してください。

- healthcheck.startdelay.timeout=90000

  モニタ対象プロセス（Alternate-Service(SQL)）開始完了通知待ちタイムアウト時間を定義します。Alternate-Service(SQL)の起動時に、設定した時間が経過すると snapshot.log.filepath で指定した先にスナップショットを出力します。モニタ対象プロセス開始完了通知待ちタイムアウト時間の詳細は、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。

- healthcheck.startdelay.actions.failure.retrymax=4

  モニタ対象プロセス（Alternate-Service(SQL)）開始完了通知待ちタイムアウトが発生した場合に、モニタ対象プロセス開始完了待ちをリトライする最大回数を定義します。最大リトライ回数を超えると、デフォルトの設定ではモニタ対象プロセスを停止します。モニタ対象プロセス開始完了待ちリトライ回数の詳細は、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。

  モニタ対象プロセス開始完了通知待ちタイムアウトの合計時間（下記の A）と Alternate-Service(SQL)の開始に掛かる時間（下記の B、および C）の関係を、「A ＞ B ＞ C」にしてください。

  A：モニタ対象プロセス開始完了通知待ちタイムアウトの合計時間

healthcheck.startdelay.timeoutに指定した値×（healthcheck.startdelay.actions.failure.retrymaxに指定した値＋1）

  B：postStart フックに指定するトランザクションヘルスチェックスクリプト（health_check.sh）の合計時間

リトライ回数×リトライ間隔秒数

  C：コネクションプールの初期化で、DB とのコネクションの取得に失敗した場合に行うリトライの合計時間

sql.pool.connection-retry-interval-in-secondsに指定した値×sql.pool.connection-retry-limitに指定した値

  A を B よりも大きな値にするのは、モニタ対象プロセス開始完了通知待ちタイムアウトによるプロセス停止が、postStart フックに指定するトランザクションヘルスチェックスクリプトによる Alternate-Service(SQL)の開始完了待ちを中止することを防ぐためです。

  B を C よりも大きな値とするのは、postStart フックに指定するトランザクションヘルスチェックスクリプトのタイムアウトによる Alternate-Service(SQL)の停止が、Alternate-Service(SQL)のコネクション接続処理を中止することを防ぐためです。

  この記述例では、Alternate-Service(SQL)の開始に掛かる時間（300 秒）の間に、スナップショットを複数回（3 回）出力するように設定しています。

16. このオレンジ色の背景で示す行（「- name: snapshot-volume」の行から「mountPath: /ucars-snapshots」の行、および「- name: snapshot-volume」の行から「claimName: snapshot-volume」の行）は、通常版（uCosminexus Application Runtime for Spring Boot 導入あり）の場合だけ記述します。

uCosminexus Application Runtime for Spring Boot スナップショットログの永続ボリュームを Alternate-Service(SQL)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、「2.6.11　uCosminexus Application Runtime 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 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。

17. 「- name: eads-config」の行から「mountPath: /eads-properties」の行、「- name: eads-config」の行から「path: hmp-pcto.eads-client.properties」の行、および「apiVersion: v1」の行から「eads.client.connectionPool.poolsize=1」の行で、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「9.2.5　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。

18. 「- name: tomcat」の行から「mountPath: /deployments/tomcat」の行、および「- name: tomcat」の行で、Tomcat のベースディレクトリとして /deployments/tomcat をマウントします。

19. 「- name: eads」の行から「mountPath: /deployments/eads-log」の行、および「- name: eads」の行で、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。

20. 「livenessProbe:」の行から「timeoutSeconds: 10」の行で、コンテナライフサイクル設定の Liveness Probe を定義します。

21. 「postStart:」の行から「- "8080"」の行で、コンテナライフサイクル設定の postStart フックを定義します。

22. 「preStop:」の行から「- "8080"」の行で、コンテナライフサイクル設定の preStop フックを定義します。

23. このオレンジ色の背景で示す行（「- name: filebeat-sidecar」の行から「mountPath: /deployments/eads-log」の行、「- name: beatconfig」の行から「name: alternate-service-sql-filebeat-sidecar-config」の行、および「apiVersion: v1」の行から「hosts: ["logstash.my-namespace.svc.cluster.local:5044"]」の行）は、Elastic Stack を使用してトラブルシュート情報を収集する場合だけ記述します。

    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。

24. 「resources:」の行から「requests:」の「memory: 2Gi」の行で、Filebeat 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。

    オートスケールのメトリクスの監視に Metrics Server を使用する場合は必ず指定してください。

25.「env:」の行から「fieldPath: spec.nodeName」の行、および「processors:」の行から「name: ${HMPPCTO_NODE_NAME}」の行で、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。

26. Pod ライフサイクル設定の terminationGracePeriodSeconds を定義します。

27.「apiVersion: v1」の行から「jp.co.Hitachi.soft.hmppcto.set-routing=inner」の行で、Alternate-Service(SQL)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。

28. このオレンジ色の背景で示す行は、通常版（uCosminexus Application Runtime for Spring Boot 導入あり）の場合だけ記述します。

29. Alternate-Service(SQL)が接続する RDBMS を指定します。PostgreSQL を指定する場合、「9.4.1 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。
- Alternate-Service(SQL)を構成する Pod を定義した Kubernetes の Service の Service 名と Namespace 名を「.」でつないだ文字列が、41 バイト以内となるようにしてください。

2.6.9 Kubernetes リソース取得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、Kubernetes リソース取得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 Kubernetes リソース取得時に必要な Kubernetes オブジェクトにアクセスできるようにします。

Kubernetes リソース取得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、pods
- 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、次に示す Kubernetes リソースの取得を実施する Kubernetes アプリケーションの中で、利用する Kubernetes アプリケーション数と同じ数にします。
    ＜ Kubernetes リソースの取得を実施する Kubernetes アプリケーション＞
    ・Orchestrator-Service
    ・Relay-Service
    ・Entity-Service(SQL)
    ・Entity-Service(TCC)
    ・Orchestrator-Service(SQL)
    ・Alternate-Service
    ・Alternate-Service(SQL)
    ・TP1-Bridge
  - .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、Kubernetes リソースの取得を実施する 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。
  - .subjects[].namespace フィールドに、Kubernetes リソースの取得を実施する 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 Namespace を指定してください。

Kubernetes リソース取得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:
  - pods
  - 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: relay-service-service-account
  namespace: my-namespace
- kind: ServiceAccount
  name: sql-participant-service-account
  namespace: my-namespace
- kind: ServiceAccount
  name: tcc-participant-service-account
  namespace: my-namespace
- kind: ServiceAccount
  name: orchestrator-sql-service-account
  namespace: my-namespace
- kind: ServiceAccount
  name: alternate-service-service-account
  namespace: my-namespace
- kind: ServiceAccount
  name: alternate-service-sql-service-account
  namespace: my-namespace
- kind: ServiceAccount
  name: <tp1-bridgeのservice-account>※
  namespace: my-namespace
```

注※

<tp1-bridge の service-account>は、「name: <リリース名>-service-account」で指定してください。

<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

2.6.10 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 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 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の Kubernetes アプリケーションおよび RAS

詳細については、「3.7.6 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。

2.6.11 uCosminexus Application Runtime 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 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 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の Kubernetes アプリケーションおよび RAS

詳細については、「3.7.10 uCosminexus Application Runtime 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 2-25　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の Kubernetes アプリケーションおよび RAS	(2)　Namespace の作成
Orchestrator および 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 および Participant の gRPC 通信機能の暗号化通信用シークレットの作成
Kubernetes リソース取得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)　Kubernetes リソース取得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の Kubernetes アプリケーションおよび RAS	(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 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）	通常版の場合で、uCosminexus Application Runtime for Spring Boot を導入するときだけ作成します。次のどちらか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の Kubernetes アプリケーションおよび RAS	(6)　uCosminexus Application Runtime 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
HPA の設定のカスタマイズ	スケーリング対応機能を使用する場合で、Metrics Server でメトリクスの監視をしてオートスケールを適用するときだけ実施します。	4.2.5　HPA の設定のカスタマイズ

作業項目	備考	参照先
	スケーリング対応機能を使用する場合で、Prometheus でメトリクスの監視をしてオートスケールを適用するときだけ実施します。	4.3.7　HPA の設定のカスタマイズ
HPA のデプロイ	スケーリング対応機能を使用する場合で、Metrics Server でメトリクスの監視をしてオートスケールを適用するときだけ実施します。	4.2.6　HPA のデプロイ
	スケーリング対応機能を使用する場合で、Prometheus でメトリクスの監視をしてオートスケールを適用するときだけ実施します。	4.3.8　HPA 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 2-26　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
ユーザ責務の Kubernetes アプリケーションの Kubernetes マニフェスト	• 「2.6.2　Orchestrator-Service の Kubernetes マニフェストの作成」で作成したもの • 「2.6.3　Relay-Service の Kubernetes マニフェストの作成」で作成したもの • 「2.6.4　Entity-Service(SQL)の Kubernetes マニフェストの作成（SQL-Participant 限定）」で作成したもの • 「2.6.5　Entity-Service(TCC)の Kubernetes マニフェストの作成（TCC-Participant 限定）」で作成したもの • 「2.6.6　Orchestrator-Service(SQL)の Kubernetes マニフェストの作成（SQL-Participant 限定）」で作成したもの • 「2.6.7　Alternate-Service の Kubernetes マニフェストの作成」で作成したもの • 「2.6.8　Alternate-Service(SQL)の Kubernetes マニフェストの作成（SQL-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

資材	備考
Kubernetes リソース取得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	「2.6.9　Kubernetes リソース取得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	「2.6.10　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 Kubernetes マニフェストの作成」で作成したもの 次のどちらか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の Kubernetes アプリケーションおよび RAS
uCosminexus Application Runtime 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）	「2.6.11　uCosminexus Application Runtime 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 通常版の場合で、uCosminexus Application Runtime for Spring Boot を導入するときだけ実施します。次のどちらか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の Kubernetes アプリケーションおよび RAS
HPA の Kubernetes マニフェスト	• 「4.2.4　HPA の Kubernetes マニフェストの作成」で作成したもの • 「4.3.6　HPA の Kubernetes マニフェストの作成」で作成したもの スケーリング対応機能を使用している場合で、オートスケールを適用するときだけ配置します。

(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の Kubernetes アプリケーションおよび RAS

詳細については、「(2)　Namespace の作成」を参照してください。

(3)　Orchestrator および 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 2-27　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	対応する HMP-PCTO の機能	ファイル	備考
• Orchestrator-Service • Orchestrator-Service(SQL) • Alternate-Service • Alternate-Service(SQL)	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。
• Entity-Service(SQL) • Entity-Service(TCC) • Orchestrator-Service(SQL) • Alternate-Service(SQL)	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。

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
```

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/
        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
```

準備したファイルを使用して Orchestrator および 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、および 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 ¥
  -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=client.crt=./participant/client.crt ¥
  --from-file=client.key=./participant/client.key ¥
  -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) Kubernetes リソース取得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 Kubernetes リソース取得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の Kubernetes アプリケーションおよび RAS

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 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 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の Kubernetes アプリケーションおよび RAS

詳細については、「(14)　uCosminexus Application Runtime 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※1、※2
2. Logstash※1、Jaeger※3、Prometheus※4、Metricbeat※1、Filebeat※1
3. Mediator、EADS サーバ（通常版だけ）※5、Ext-Cons（トライアル版だけ)、Participant が使用する DB/外部サービス
4. Relay-Service、Entity-Service(SQL)、Entity-Service(TCC)、TP1-Bridge※6
5. Orchestrator-Service、Orchestrator-Service(SQL)、Alternate-Service、Alternate-Service(SQL)

注※1

これらの Kubernetes アプリケーションは、Elastic Stack を使用してトラブルシュート情報を収集する場合だけデプロイしてください。

注※2

クラウド環境（例：Microsoft Azure 環境）で Elasticsearch を使用する場合もデプロイしてください。

注※3

Jaeger を使用してトラブルシュート情報を収集する場合だけデプロイしてください。クラウド環境（例：Amazon Web Services 環境）で Jaeger を使用する場合もデプロイしてください。

注※4

Prometheus を使用してトラブルシュート情報を収集する場合だけデプロイしてください。

注※5

EADS サーバをデプロイしたあとに「(23)　EADS のセットアップ（通常版限定)」を実施してから、Participant の機能を使用する Kubernetes アプリケーションをデプロイしてください。

注※6

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。

メモ

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。

そのあとで、EADS サーバと Participant の機能を使用する Kubernetes アプリケーションをアンデプロイして、正しい順序でデプロイし直してください。

注※ 「(8)　ユーザ責務の Kubernetes アプリケーションのアンデプロイ」を実施しないでクラスタを再起動する場合も含みます。

デプロイした Kubernetes アプリケーションの起動に失敗したとき、Kubernetes の ERROR メッセージが出力されることがあります。詳細については、「(8)　ユーザ責務の Kubernetes アプリケーションのアンデプロイ」の「Kubernetes アプリケーションの停止時に出力される Kubernetes の ERROR メッセージについて」を参照してください。

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-Service、Orchestrator-Service(SQL)、Alternate-Service、Alternate-Service(SQL)
2. Relay-Service、Entity-Service(SQL)、Entity-Service(TCC)、TP1-Bridge※1
3. Mediator、EADS サーバ（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※2、Jaeger※3、Prometheus※4、Metricbeat※2、Filebeat※2
5. Elasticsearch※2、※5

注※1

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。

注※2

これらの Kubernetes アプリケーションは、Elastic Stack を使用してトラブルシュート情報を収集する場合だけアンデプロイしてください。

注※3

Jaeger を使用してトラブルシュート情報を収集する場合だけアンデプロイしてください。クラウド環境（例：Amazon Web Services 環境）で Jaeger を使用する場合もアンデプロイしてください。

注※4

Prometheus を使用してトラブルシュート情報を収集する場合だけアンデプロイしてください。

注※5

クラウド環境（例：Microsoft Azure 環境）で Elasticsearch を使用する場合もアンデプロイしてください。

クラスタを停止する前には、上記の順序でアンデプロイしてください。アンデプロイする前にクラスタを停止した場合は、クラスタを再起動したあとに、アンデプロイしてから「(7)　ユーザ責務の Kubernetes アプリケーションのデプロイ」で示す順序で再デプロイしてください。

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。

注※　アンデプロイする前にクラスタを停止する場合も含みます。

Kubernetes アプリケーションの停止時に出力される Kubernetes の ERROR メッセージについて

Kubernetes アプリケーションの停止時（起動失敗時を含む）に、Kubernetes アプリケーションのメッセージログに Kubernetes の client で java.lang.InterruptedException が発生したことを示す

ERROR メッセージが出力されることがあります。このメッセージは出力されても問題ないため、無視してください。

ERROR メッセージの出力例の一部を次に示します。

■メッセージ出力例 1

```
ERROR sql-participant-dc47fd869-khlgp pool-13-thread-1 io.kubernetes.client.informer.cache.ProcessorListener 87 processor interrupted: {}
```

■メッセージ出力例 2

```
ERROR sql-participant-dc47fd869-khlgp informer-controller-V1Endpoints io.kubernetes.client.informer.cache.Controller 87 DefaultController#processLoop get interrupted null
```

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の Kubernetes アプリケーションおよび RAS のデプロイ

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

HMP-PCTO のコントロールプレーンの Kubernetes アプリケーションおよび RAS のデプロイの流れについて説明します。

デプロイ後のシステム運用時の注意事項については、「付録 E　システム運用時の注意事項」を参照してください。

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、「(27)　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の Kubernetes アプリケーションおよび RAS 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。

3.1.1 オンプレミス環境の場合

オンプレミス環境の場合、HMP-PCTO のコントロールプレーンの Kubernetes アプリケーションおよび RAS は次の流れでデプロイします。スケーリング対応機能を使用する場合は、「4.　スケーリング対応機能の運用」も併せて参照してください。

表 3-1　HMP-PCTO のコントロールプレーンの Kubernetes アプリケーションおよび RAS のデプロイの流れ（オンプレミス環境の場合）

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	永続ボリュームとプロビジョニング方法の選定	HMP-PCTO で使用する永続ボリュームのストレージ種別と、そのプロビジョニング方法を選定します。	3.5　永続ボリュームとプロビジョニング方法の選定

項番	項目	説明	参照先
5	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 および関連製品で必要なディレクトリを用意します。	3.6　共有ファイルシステムでの作業
6	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。	3.7　Kubernetes マニフェスト作成
7	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。	3.8　Helm チャートのカスタマイズ
8	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。	3.9　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

3.1.2　Microsoft Azure 環境の場合

Microsoft Azure 環境の場合、HMP-PCTO のコントロールプレーンの Kubernetes アプリケーションおよび RAS は次の流れでデプロイします。スケーリング対応機能を使用する場合は、「4.　スケーリング対応機能の運用」も併せて参照してください。

表 3-2　HMP-PCTO のコントロールプレーンの Kubernetes アプリケーションおよび RAS のデプロイの流れ（Microsoft Azure 環境を使用する場合）

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	Azure Container Registry の有効化およびコンテナイメージの登録	Azure Kubernetes Service で作成した AKS クラスタから参照可能なコンテナレジストリ Azure Container Registry に、コンテナイメージを登録します。	Microsoft Azure の公式ドキュメント
5	永続ボリュームとプロビジョニング方法の選定	HMP-PCTO で使用する永続ボリュームのストレージ種別と、そのプロビジョニング方法を選定します。	3.5　永続ボリュームとプロビジョニング方法の選定
6	Microsoft Azure のマネージドストレージサービスのプロビジョニング	HMP-PCTO および関連製品で使用する共有ファイルシステムをプロビジョニングします。	• Microsoft Azure の公式ドキュメント • 3.6　共有ファイルシステムでの作業

項番	項目	説明	参照先
7	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。	• Microsoft Azure の公式ドキュメント • 3.7　Kubernetes マニフェスト作成
8	永続ボリュームの Kubernetes マニフェストの作成	共有ファイルシステムを永続ボリュームとして使用するための Kubernetes マニフェストを作成します。	Microsoft Azure の公式ドキュメント
9	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。	3.8　Helm チャートのカスタマイズ
10	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。	3.9　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
11	Azure Monitor Log Analytics ワークスペースの作成	HMP-PCTO のメッセージログを収集および閲覧するために Azure Monitor Log Analytics ワークスペースを作成します。	Microsoft Azure の公式ドキュメント
12	Azure Monitor ワークスペースの作成	HMP-PCTO のメトリクスを保管するために Azure Monitor ワークスペースを作成します。	
13	マネージド Prometheus の有効化	HMP-PCTO のメトリクスを収集するためにマネージド Prometheus を有効化します。	
14	Azure Managed Grafana の有効化	HMP-PCTO のメトリクスを閲覧するために Azure Managed Grafana を有効化します。	

3.1.3　Google Cloud Platform 環境の場合

Google Cloud Platform 環境の場合、HMP-PCTO のコントロールプレーンの Kubernetes アプリケーションおよび RAS は次の流れでデプロイします。スケーリング対応機能を使用する場合は、「4.　スケーリング対応機能の運用」も併せて参照してください。

表 3-3　HMP-PCTO のコントロールプレーンの Kubernetes アプリケーションおよび RAS のデプロイの流れ（Google Cloud Platform 環境を使用する場合）

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）

項番	項目	説明	参照先
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	Artifact Registry の有効化およびコンテナイメージの登録	Google Kubernetes Engine で作成した GKE クラスタから参照可能なコンテナレジストリ Artifact Registry にコンテナイメージを登録します。	Google Cloud Platform の公式ドキュメント
5	永続ボリュームとプロビジョニング方法の選定	HMP-PCTO で使用する永続ボリュームのストレージ種別と、そのプロビジョニング方法を選定します。通常版の場合だけ実施します。	3.5　永続ボリュームとプロビジョニング方法の選定
6	Google Cloud Platform のマネージドストレージサービスのプロビジョニング	HMP-PCTO および関連製品で使用する共有ファイルシステムをプロビジョニングします。	• Google Cloud Platform の公式ドキュメント • 3.6　共有ファイルシステムでの作業
7	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。	• Google Cloud Platform の公式ドキュメント • 3.7　Kubernetes マニフェスト作成
8	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。	3.8　Helm チャートのカスタマイズ
9	永続ボリュームの Kubernetes マニフェストの作成	共有ファイルシステムを永続ボリュームとして使用するための Kubernetes マニフェストを作成します。	Google Cloud Platform の公式ドキュメント
10	Cloud Trace を出力する権限の付与	HMP-PCTO の分散トレースを収集するため、Cloud Trace を出力する権限を付与します。 「3.7　Kubernetes マニフェスト作成」で作成した、各 Pod が使用するサービスアカウントに対して、cloudtrace.agent ロールを付与してください。	Google Cloud Platform の公式ドキュメント
11	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。	3.9　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
12	Google Cloud Managed Service for Prometheus の有効化	HMP-PCTO のメトリクスを収集するために有効化します。	• Google Cloud Platform の公式ドキュメント • 3.7　Kubernetes マニフェスト作成

3.1.4 Amazon Web Services 環境の場合

Amazon Web Services 環境の場合、HMP-PCTO のコントロールプレーンの Kubernetes アプリケーションおよび RAS は次の流れでデプロイします。スケーリング対応機能を使用する場合は、「4. スケーリング対応機能の運用」も併せて参照してください。

表 3-4 HMP-PCTO のコントロールプレーンの Kubernetes アプリケーションおよび RAS のデプロイの流れ（Amazon Web Services 環境を使用する場合）

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	Amazon ECR の有効化およびコンテナイメージの登録	Amazon Elastic Kubernetes Service で作成した EKS クラスタから参照可能なコンテナレジストリ Amazon ECR にコンテナイメージを登録します。	Amazon Web Services の公式ドキュメント
5	永続ボリュームとプロビジョニング方法の選定	HMP-PCTO で使用する永続ボリュームのストレージ種別と、そのプロビジョニング方法を選定します。通常版の場合だけ実施します。	3.5 永続ボリュームとプロビジョニング方法の選定
6	Amazon Web Services のマネージドストレージサービスのプロビジョニング	HMP-PCTO および関連製品で使用する共有ファイルシステムをプロビジョニングします。	• Amazon Web Services の公式ドキュメント • 3.6 共有ファイルシステムでの作業
7	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。	• Amazon Web Services の公式ドキュメント • 3.7 Kubernetes マニフェスト作成
8	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。	3.8 Helm チャートのカスタマイズ
9	永続ボリュームの Kubernetes マニフェストの作成	共有ファイルシステムを永続ボリュームとして使用するための Kubernetes マニフェストを作成します。	Amazon Web Services の公式ドキュメント
10	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。	3.9 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

項番	項目	説明	参照先
11	Amazon Managed Service for Prometheus の有効化	HMP-PCTO のメトリクスを収集するために有効化します。	• Amazon Web Services の公式ドキュメント • 3.7　Kubernetes マニフェスト作成

3.1.5　Red Hat OpenShift Container Platform 環境の場合

Red Hat OpenShift Container Platform 環境の場合、HMP-PCTO のコントロールプレーンの Kubernetes アプリケーションおよび RAS は次の流れでデプロイします。スケーリング対応機能を使用する場合は、「4.　スケーリング対応機能の運用」も併せて参照してください。

表 3-5　HMP-PCTO のコントロールプレーンの Kubernetes アプリケーションおよび RAS のデプロイの流れ（Red Hat OpenShift Container Platform 環境の場合）

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	永続ボリュームとプロビジョニング方法の選定	HMP-PCTO で使用する永続ボリュームのストレージ種別と、そのプロビジョニング方法を選定します。	3.5　永続ボリュームとプロビジョニング方法の選定
5	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 および関連製品で必要なディレクトリを用意します。	3.6　共有ファイルシステムでの作業
6	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。	3.7　Kubernetes マニフェスト作成
7	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。	3.8　Helm チャートのカスタマイズ
8	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。	3.9　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

3.2 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、Gradle 8x 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 3-6 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）		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）		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 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。

重要

使用可能な Gradle のバージョンについて

使用可能な Gradle のバージョンは、Spring Boot のシステム要件に従って決定されます。

使用する Spring Boot のバージョンによって要件は異なるため、詳細は Spring Boot の公式ドキュメントを参照してください。

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 およびオプション製品の提供媒体の 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 およびオプション製品の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 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。

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 が提供するトラブルシュート機能を使用して分散トレースを出力する場合、「Mediator 用 Java プロジェクトディレクトリ」（java-projects/mediator）の「Mediator 用 build.gradle ファイル」

（build.gradle）を「2.2.17　分散トレース機能の設定」に従って修正し、ライブラリをアプリケーションに組み込んでください。

そのあと、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、Gradle 8x 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 3-7　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）		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）		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 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。

重要

使用可能な Gradle のバージョンについて

使用可能な Gradle のバージョンは、Spring Boot のシステム要件に従って決定されます。

使用する Spring Boot のバージョンによって要件は異なるため、詳細は Spring Boot の公式ドキュメントを参照してください。

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、Hitachi Microservices Platform - Bridge Option for OpenTP1 の提供物を使用してビルドします。

Hitachi Microservices Platform - Bridge Option for OpenTP1 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。

HMP-PCTO が提供するトラブルシュート機能を使用して分散トレースを出力する場合、「TP1-Bridge 用 Java プロジェクトディレクトリ」（java-projects/tp1-bridge）の「TP1-Bridge 用 build.gradle ファイル」（build.gradle）を「2.2.17　分散トレース機能の設定」に従って修正し、ライブラリをアプリケーションに組み込んでください。

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-bridge-*V.R.S*※.jar）」が生成されます。

```
java-projects/tp1-bridge/build/libs/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。Red Hat OpenShift Container Platform を使用する場合は、適宜「docker」コマンドを「podman」コマンドに読み替えてください。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 3-8　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）		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。
- uCosminexus Application Runtime with Java for Spring Boot のリリースノートを参照し、オレンジ色の背景の 1.（「<uCosminexus Application Runtime with Java for Spring Boot の前提 RPM パッケージ>」の行から「:」の行）に、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
- オレンジ色の背景の 2.の行は、Red Hat OpenShift Container Platform を使用し、SCC（Security Context Constraints）の設定で、コンテナ実行時に root ユーザでの実行を許容しない場合だけ記述します。Cosminexus Application Runtime with Java for Spring Boot のインストールディレクトリ配下の所有グループを実行ユーザが属するグループに変更してください。

- オレンジ色の背景の 3.（「COPY ./gtsrootr1.pem /opt/Cosminexus/jdk17/lib/security/」の行から「RUN keytool -import -alias gtsrootr4 -trustcacerts ¥」の「-noprompt」の行）は、分散トレース収集サービスに Google Cloud Platform の Cloud Trace を使用する場合だけ記述します。

  Cloud Trace に接続するために、Java のキーストアにルート CA 証明書として、GTS Root R1、GTS Root R2、GTS Root R3、GTS Root R4 をインポートします。

  keytool コマンドの-storepass オプションに設定するパスワードは環境に合わせて変更してください。

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。

```
FROM registry.access.redhat.com/ubi9/ubi-minimal:9.5-1742914212 AS builder
RUN microdnf install ¥
        -y ¥
        --setopt=install_weak_deps=0 ¥
        --nodocs ¥
        alsa-lib ¥
        cups-libs ¥
        findutils ¥
        lksctp-tools ¥
        nss ¥
        ncurses ¥
        tar ¥
        cpio ¥
        gzip
WORKDIR /tmp
RUN mkdir ./X64LIN
COPY X64LIN ./X64LIN
RUN chmod +x ./X64LIN/setup
RUN ./X64LIN/setup -f -c HL0S ./

FROM registry.access.redhat.com/ubi9/ubi-minimal:9.5-1742914212
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-minimal ¥
        jq ¥
        glibc ¥
        <uCosminexus Application Runtime with Java for Spring Bootの前提RPMパッケージ>※1
  …1
        :
    && dnf clean all ¥
    && rm -rf /var/cache/dnf/*
```

```
ENV PATH=/opt/Cosminexus/jdk17/bin:$PATH
ENV LANG=C.UTF-8
ENV LC_ALL=C.UTF-8

RUN chgrp -R 1001 /opt/hitachi/ucars      …2

COPY ./gtsrootr1.pem /opt/Cosminexus/jdk17/lib/security/      …3
COPY ./gtsrootr2.pem /opt/Cosminexus/jdk17/lib/security/
COPY ./gtsrootr3.pem /opt/Cosminexus/jdk17/lib/security/
COPY ./gtsrootr4.pem /opt/Cosminexus/jdk17/lib/security/
RUN keytool -import -alias gtsrootr1 -trustcacerts ¥
    -file /opt/Cosminexus/jdk17/lib/security/gtsrootr1.pem ¥
    -keystore /opt/Cosminexus/jdk17/lib/security/cacerts ¥
    -storepass changeit ¥
    -noprompt
RUN keytool -import -alias gtsrootr2 -trustcacerts ¥
    -file /opt/Cosminexus/jdk17/lib/security/gtsrootr2.pem ¥
    -keystore /opt/Cosminexus/jdk17/lib/security/cacerts ¥
    -storepass changeit ¥
    -noprompt
RUN keytool -import -alias gtsrootr3 -trustcacerts ¥
    -file /opt/Cosminexus/jdk17/lib/security/gtsrootr3.pem ¥
    -keystore /opt/Cosminexus/jdk17/lib/security/cacerts ¥
    -storepass changeit ¥
    -noprompt
RUN keytool -import -alias gtsrootr4 -trustcacerts ¥
    -file /opt/Cosminexus/jdk17/lib/security/gtsrootr4.pem ¥
    -keystore /opt/Cosminexus/jdk17/lib/security/cacerts ¥
    -storepass changeit ¥
    -noprompt
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 /opt/Cosminexus/jdk17/bin/java ${HMPPCTO_JAVA_OPTIONS}
 -jar /deployments/mediator-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 3-9　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	■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/<RHEL バージョン>/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/<RHEL バージョン>/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。
GTS Root R1.pem GTS Root R2.pem GTS Root R3.pem GTS Root R4.pem （GTS Root R1、GTS Root R2、GTS Root R3、GTS Root R4 証明書）	ユーザ	分散トレース収集サービスに Google Cloud Platform の Cloud Trace を使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 を使用するときに作成します。

(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 Hitachi Microservices Platform - Bridge Option for OpenTP1 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 のリリースノートを参照し、オレンジ色の背景の 1.（「<uCosminexus Application Runtime with Java for Spring Boot の前提 RPM パッケージ>」の行から「:」の行）に、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。
- オレンジ色の背景の 2.の行は、Red Hat OpenShift Container Platform を使用し、SCC（Security Context Constraints）の設定で、コンテナ実行時に root ユーザでの実行を許容しない場合だけ記述します。Cosminexus Application Runtime with Java for Spring Boot のインストールディレクトリ配下の所有グループを実行ユーザが属するグループに変更してください。
- オレンジ色の背景の 3.（「COPY ./gtsrootr1.pem /opt/Cosminexus/jdk17/lib/security/」の行から「RUN keytool -import -alias gtsrootr4 -trustcacerts ¥」の「-noprompt」の行）は、分散トレース収集サービスに Google Cloud Platform の Cloud Trace を使用する場合だけ記述します。

  Cloud Trace に接続するために Java のキーストアにルート CA 証明書として、GTS Root R1、GTS Root R2、GTS Root R3、GTS Root R4 をインポートします。

  keytool コマンドの-storepass オプションに設定するパスワードは環境に合わせて変更してください。

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/ubi9/ubi-minimal:9.5-1742914212 AS builder
RUN microdnf install ¥
        -y ¥
        --setopt=install_weak_deps=0 ¥
        --nodocs ¥
        alsa-lib ¥
        cups-libs ¥
        findutils ¥
        lksctp-tools ¥
        nss ¥
        ncurses ¥
        tar ¥
        cpio ¥
        gzip
WORKDIR /tmp
RUN mkdir ./X64LIN
COPY ucars/X64LIN ./X64LIN
RUN chmod +x ./X64LIN/setup
RUN ./X64LIN/setup -f -c HL0S ./
RUN mkdir patch
WORKDIR patch
COPY ucars/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/ubi9/ubi-minimal:9.5-1742914212 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-minimal ¥
        jq ¥
        glibc ¥
        <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/jdk17/bin:$PATH

RUN chgrp -R 1001 /opt/hitachi/ucars      …2

COPY ./gtsrootr1.pem /opt/Cosminexus/jdk17/lib/security/      …3
COPY ./gtsrootr2.pem /opt/Cosminexus/jdk17/lib/security/
COPY ./gtsrootr3.pem /opt/Cosminexus/jdk17/lib/security/
COPY ./gtsrootr4.pem /opt/Cosminexus/jdk17/lib/security/
RUN keytool -import -alias gtsrootr1 -trustcacerts ¥
    -file /opt/Cosminexus/jdk17/lib/security/gtsrootr1.pem ¥
    -keystore /opt/Cosminexus/jdk17/lib/security/cacerts ¥
    -storepass changeit ¥
    -noprompt
RUN keytool -import -alias gtsrootr2 -trustcacerts ¥
    -file /opt/Cosminexus/jdk17/lib/security/gtsrootr2.pem ¥
    -keystore /opt/Cosminexus/jdk17/lib/security/cacerts ¥
    -storepass changeit ¥
    -noprompt
RUN keytool -import -alias gtsrootr3 -trustcacerts ¥
    -file /opt/Cosminexus/jdk17/lib/security/gtsrootr3.pem ¥
    -keystore /opt/Cosminexus/jdk17/lib/security/cacerts ¥
    -storepass changeit ¥
    -noprompt
RUN keytool -import -alias gtsrootr4 -trustcacerts ¥
    -file /opt/Cosminexus/jdk17/lib/security/gtsrootr4.pem ¥
    -keystore /opt/Cosminexus/jdk17/lib/security/cacerts ¥
    -storepass changeit ¥
    -noprompt

USER 1001:0

FROM registry.access.redhat.com/ubi9/ubi-minimal:9.5-1742914212 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-minimal ¥
        jq ¥
```

```
        glibc ¥
    && dnf clean all ¥
    && rm -rf /var/cache/dnf/*

RUN microdnf install ¥
        -y ¥
        --setopt=install_weak_deps=0 ¥
        --nodocs ¥
        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/ubi9/ubi-minimal:9.5-1742914212
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/jdk17/bin:$PATH
ENV LANG=C.UTF-8
ENV LC_ALL=C.UTF-8
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-minimal ¥
        jq ¥
        glibc ¥
    && 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 /opt/Cosminexus/jdk17/bin/java ${HMPPCTO_JAVA_OPTIONS}
 -jar /deployments/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。

(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 3-10 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	■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/<RHEL バージョン>/tp1-bridge/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/<RHEL バージョン>/container-files/tp1-bridge/patch/Containerfile」を使用してください。
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。
ucars	ユーザ	－

ファイル／ディレクトリ構成	作成・提供元	備考
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 の修正パッチがある場合だけ格納してください。
GTS Root R1.pem GTS Root R2.pem GTS Root R3.pem GTS Root R4.pem （GTS Root R1、GTS Root R2、GTS Root R3、GTS Root R4 証明書）	ユーザ	分散トレース収集サービスに Google Cloud Platform の Cloud Trace を使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。

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＞

```
cd build/
docker build -f ./Containerfile -t myregistry.example.com/myproject/tp1-bridge:latest .
```

この例では、TP1-Bridge コンテナイメージのイメージ名・タグ名を「tp1-bridge:latest」としています。

(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 3-11　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	－
	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を作成し、コンテナレジストリに登録してください。詳細については、取扱説明書「EADS コンテナオーケストレーションツール対応機能」を参照してください。

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を作成し、コンテナレジストリに登録してください。詳細については、取扱説明書「EADS コンテナオーケストレーションツール対応機能」を参照してください。

3.5　永続ボリュームとプロビジョニング方法の選定

HMP-PCTO が使用する永続ボリュームの選定について説明します。HMP-PCTO がサポートしている永続ボリュームのストレージ種別と、そのプロビジョニング方法の一覧を次の表に示します。この表を参照して、プロビジョニング方法を選定してください。

表 3-12　HMP-PCTO でサポートしている永続ボリュームのストレージ種別と、そのプロビジョニング方法の一覧

ストレージの種別	Kubernetes 環境	マネージドファイルサービス	プロビジョニング方法		説明
			静的	動的	
ローカル永続ボリューム	規定しません	該当しません	○	−	ストレージクラスなどの詳細は用途によって異なります。詳細は「3.7　Kubernetes マニフェスト作成」を参照してください。
NFS	オンプレミス環境	該当しません	○※	−	ストレージクラスなどの詳細は用途によって異なります。詳細は「3.7　Kubernetes マニフェスト作成」を参照してください。 また、作業の詳細は各システムで異なります。HMP-PCTO としての作業については、「3.6　共有ファイルシステムでの作業」を参照してください。
	Red Hat OpenShift Container Platform 環境	該当しません	○※	−	
	Microsoft Azure 環境	Azure Files	○	−	NFS をサポートしているため、Azure Files および Azure Blob Storage を永続ボリュームとして使用できます。プロビジョニングする方法については、Microsoft Azure の公式ドキュメントを参照してください。 HMP-PCTO として必要な作業は、「3.6　共有ファイルシステムでの作業」を参照してください。
		Azure Blob Storage	−	○	
	Google Cloud Platform 環境	Filestore	○	○	NFS をサポートしているため、Filestore を永続ボリュームとして使用できます。プロビジョニングする方法については、Google Cloud Platform の公式ドキュメントを参照してください。

ストレージの種別	Kubernetes 環境	マネージドファイルサービス	プロビジョニング方法		説明
			静的	動的	
					HMP-PCTO として必要な作業は、「3.6　共有ファイルシステムでの作業」を参照してください。
	Amazon Web Services 環境	Amazon EFS	○	○	NFS をサポートしているため、Amazon EFS を永続ボリュームとして使用できます。プロビジョニングする方法については、Amazon Web Services の公式ドキュメントを参照してください。 HMP-PCTO として必要な作業は、「3.6　共有ファイルシステムでの作業」を参照してください。

（凡例）

○：サポートしています。HMP-PCTO および EADS で動作検証が実施済みの構成です。

－：サポートしていません。HMP-PCTO および EADS で動作検証が未実施の構成です。この構成を使用する場合は、ユーザ責任で動作検証を実施してください。

注※

目的によっては使用できない場合があります。詳細な選定方法は「3.7　Kubernetes マニフェスト作成」を参照してください。

3.6　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。

3.6.1　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 3-13　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
マネージドファイルサービスでのプロビジョニング	Microsoft Azure 環境、Google Cloud Platform 環境、または Amazon Web Services 環境の場合で、NFS ストレージが必要なときだけ実施します。	3.6.2　マネージドファイルサービスでのプロビジョニング
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だけ作成します。	3.6.3　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	Elastic Stack を使用してトラブルシュート情報を収集する場合だけ作成します。	3.6.4　Elasticsearch リポジトリディレクトリの永続ボリュームマウント用ディレクトリの作成
uCosminexus Application Runtime 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）	通常版の場合で、uCosminexus Application Runtime for Spring Boot を導入するときだけ作成します。	3.6.5　uCosminexus Application Runtime 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）
EADS 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）	通常版の場合で、EADS の永続ボリュームを NFS ストレージとするときだけ作成します。	3.6.6　EADS 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）

3.6.2　マネージドファイルサービスでのプロビジョニング

Microsoft Azure 環境、Google Cloud Platform 環境、または Amazon Web Services 環境の場合で NFS ストレージが必要なときは、次のどちらかをデプロイするための作業時に、マネージドファイルサービスでのプロビジョニングを実施します。

- 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の Kubernetes アプリケーションおよび RAS

HMP-PCTO は共有ファイルシステムとして NFS を前提としています。このため、NFS をサポートしている次のマネージドファイルサービスを使用できます。

- Microsoft Azure 環境
  - Azure Files
  - Azure Blob Storage
- Google Cloud Platform 環境
  - Filestore
- Amazon Web Services 環境
  - Amazon EFS

(1) Azure Files を NFS ストレージとしてプロビジョニングする

Azure Files を永続ボリュームとして使用する場合は、次の項目を実施してください。

- NFS プロトコルを使用してマウントしてください。NFS プロトコルを使用する場合は、Premium プランを選択する必要があります。
- ストレージクラスには、「azurefile-csi-premium」を指定してください。
- 永続ボリュームとして使用する Azure Files は Azure Files CSI ドライバの PersistentVolumes の次のパラメタを設定することで決定されます。
  - spec.csi.volumeAttributes.storageAccount：使用する Azure ストレージアカウントを指定
  - spec.csi.volumeAttributes.shareName：使用する Azure Files 名を指定
  - spec.csi.volumeAttributes.folderName：任意指定。Azure Files 内の共有ディレクトリ名を指定
  - spec.csi.volumeAttributes.protocol：使用するプロトコルとして「nfs」を指定

  上記のパラメタを設定することで、永続ボリュームとして次がマウントされます。

```
/<Azureストレージアカウント>/<Azure Files名>/<共有ディレクトリ名>
```

Azure Files を NFS ストレージとしてプロビジョニングする方法については、Microsoft Azure の公式ドキュメントを参照してください。

また、Azure Files を NFS ストレージとしてプロビジョニングする際は、次の点を考慮してください。

- Azure Files の動的プロビジョニングおよび静的プロビジョニングによる永続ボリュームの作成が使用できます。

  動的プロビジョニングを使用することで、永続ボリュームクレームが発行された時点で Kubernetes によって自動で「pvc-」から始まる名前が付いた永続ボリュームがプロビジョニングされるため、事前に永続ボリュームを作成する必要がなくなります。

  静的プロビジョニングを使用する場合は、事前に永続ボリュームを手動で作成する必要があります。

(2) Azure Blob Storage を NFS ストレージとしてプロビジョニングする

Azure Blob Storage を永続ボリュームとして使用する場合は、次の項目を実施してください。

- NFS プロトコルを使用してマウントしてください。NFS プロトコルを使用する場合は Premium プランを選択する必要があります。
- ストレージクラスには、「azureblob-nfs-premium」を指定してください。
- 永続ボリュームとして使用する BLOB コンテナは Azure Blob Storage CSI ドライバの PersistentVolumes の次のパラメタを設定することで決定されます。
  - spec.csi.volumeAttributes.storageAccount：使用する Azure ストレージアカウントを指定
  - spec.csi.volumeAttributes.containerName：使用する BLOB コンテナ名を指定
  - spec.csi.volumeAttributes.protocol：使用するプロトコルとして「nfs」を指定

  上記のパラメタを設定することで、永続ボリュームとして次がマウントされます。

```
/<Azureストレージアカウント>/<BLOBコンテナ>
```

Azure Blob Storage を NFS ストレージとしてプロビジョニングする方法については、Microsoft Azure の公式ドキュメントを参照してください。

また、Azure Blob Storage を NFS ストレージとしてプロビジョニングする際は、次の点を考慮してください。

- BLOB コンテナがマウントされるディレクトリとなるため、目的に応じた BLOB コンテナを作成することでファイルの混在を防止できます。
- Azure Blob Storage の動的プロビジョニングおよび静的プロビジョニングによる永続ボリュームの作成が使用できます。

  動的プロビジョニングを使用することで、永続ボリュームクレームが発行された時点で Kubernetes によって自動で「pvc-」から始まる名前が付いた永続ボリュームがプロビジョニングされるため、事前に永続ボリュームを作成する必要がなくなります。

  静的プロビジョニングを使用する場合は、事前に永続ボリュームを手動で作成する必要があります。

(3) Google Cloud Platform の Filestore を NFS ストレージとしてプロビジョニングする

Filestore を NFS ストレージとしてプロビジョニングする方法については、Google Cloud Platform の公式ドキュメントを参照してください。

(4) Amazon EFS を NFS ストレージとしてプロビジョニングする

Amazon EFS を NFS ストレージとしてプロビジョニングする方法については、Amazon Web Services の公式ドキュメントを参照してください。

3.6.3 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、「3.7.2　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。

各 Mediator で使用する ConsensusLog を、Mediator 単位で異なる NFS サーバに配置することを推奨します。これによって、障害時の影響範囲を最小化できます。

Mediator 数分の NFS サーバを用意することが難しい場合は、1 台以上の NFS サーバに複数の Mediator の ConsensusLog を分散して配置してください。1 台の NFS サーバに複数の Mediator の ConsensusLog を配置した場合、NFS サーバが単一障害点となるおそれがあるため、NFS サーバの冗長化などの対策を検討してください。

3.6.4 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

Elastic Stack を使用してトラブルシュート情報を収集する場合は、次の手順を実施してください。

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。

3.6.5 uCosminexus Application Runtime 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 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の Kubernetes アプリケーションおよび RAS

＜ uCosminexus Application Runtime 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。

3.6.6 EADS 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）

EADS の永続ボリュームを NFS ストレージとする場合、次の作業を実施してください。

EADS の永続ボリュームマウント用ディレクトリを、EADS が存在する各ワーカーノードでアクセスできる NFS に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、「3.7.5　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）」の例では、ログファイル用のディレクトリ名を「/eads-logs-volume」、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のディレクトリ名を「/eads-cache-volume」、ストアデータファイル用のディレクトリ名を「/eads-store-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。

3.7 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、「3.11.1 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。

3.7.1 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 3-14 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。	3.7.2 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	Elastic Stack を使用してトラブルシュート情報を収集する場合だけ作成します。	3.7.3 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		3.7.4 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）	EADS の永続ボリュームの種類を選択する判断基準は、「3.7.5 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。	3.7.5 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 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の Kubernetes アプリケーションおよび RAS	3.7.6 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	Elastic Stack を使用してトラブルシュート情報を収集する場合だけ作成します。	3.7.7 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	Prometheus を使用してトラブルシュート情報を収集する場合だけ作成します。	3.7.8　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		3.7.9　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 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）	通常版の場合で、uCosminexus Application Runtime for Spring Boot を導入するときだけ作成します。次のどちらかの 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の Kubernetes アプリケーションおよび RAS	3.7.10　uCosminexus Application Runtime 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）
HPA の Kubernetes マニフェストの作成	スケーリング対応機能を使用する場合で、Metrics Server でメトリクスの監視をしてオートスケールを適用するときだけ作成します。	4.2.4　HPA の Kubernetes マニフェストの作成
	スケーリング対応機能を使用する場合で、Prometheus でメトリクスの監視をしてオートスケールを適用するときだけ作成します。	4.3.6　HPA の Kubernetes マニフェストの作成
Filebeat の Kubernetes マニフェスト作成	Elastic Stack を使用してトラブルシュート情報を収集する場合だけ作成します。	3.7.11　Filebeat の Kubernetes マニフェスト作成
Jaeger の Kubernetes マニフェスト作成	Jaeger を使用してトラブルシュート情報を収集する場合だけ作成します。この項目は、クラウド環境（例：Amazon Web Services 環境）で Jaeger を使用する場合にも適用されます。	3.7.12　Jaeger の Kubernetes マニフェスト作成
Prometheus の Kubernetes マニフェスト作成	Prometheus を使用してトラブルシュート情報を収集する場合だけ作成します。	3.7.13　Prometheus の Kubernetes マニフェスト作成
Microsoft Azure のマネージド Prometheus によるメトリクス収集の Kubernetes マニフェストの作成	Microsoft Azure 環境でマネージド Prometheus を使用して HMP-PCTO のメトリクスを収集するときだけ作成します。	3.7.14　Microsoft Azure のマネージド Prometheus によるメトリクス収集の Kubernetes マニフェストの作成
Google Cloud Platform のマネージド Prometheus によるメトリクス	Google Cloud Platform 環境でマネージド Prometheus を使用して	3.7.15　Google Cloud Platform のマネージド Prometheus によるメトリクス収集の Kubernetes マニフェスト作成

作業項目	備考	参照先
収集の Kubernetes マニフェスト作成	HMP-PCTO のメトリクスを収集するときだけ作成します。	
Amazon Web Services のマネージド Prometheus によるメトリクス収集の Kubernetes マニフェスト作成	Amazon Web Services 環境でマネージド Prometheus を使用して HMP-PCTO のメトリクスを収集する場合だけ作成します。	3.7.16　Amazon Web Services のマネージド Prometheus によるメトリクス収集の Kubernetes マニフェスト作成
New Relic によるメトリクス収集の Kubernetes マニフェスト作成	New Relic を使用して HMP-PCTO のメトリクスを収集するときだけ作成します。	3.7.17　New Relic によるメトリクス収集の Kubernetes マニフェスト作成
Red Hat OpenShift Container Platform モニタリングの Prometheus によるメトリクス収集の Kubernetes マニフェスト作成	Red Hat OpenShift Container Platform 環境でメトリクスを収集する場合だけ作成します。	3.7.18　Red Hat OpenShift Container Platform モニタリングの Prometheus によるメトリクス収集の Kubernetes マニフェスト作成
Metricbeat の Kubernetes マニフェスト作成	Elastic Stack を使用してトラブルシュート情報を収集する場合だけ作成します。	3.7.19　Metricbeat の Kubernetes マニフェスト作成
Elasticsearch の Kubernetes マニフェスト作成	Elastic Stack を使用してトラブルシュート情報を収集する場合だけ作成します。この項目は、クラウド環境（例：Microsoft Azure 環境）で Elasticsearch を使用する場合にも適用されます。	3.7.20　Elasticsearch の Kubernetes マニフェスト作成
Logstash の Kubernetes マニフェスト作成	Elastic Stack を使用してトラブルシュート情報を収集する場合だけ作成します。	3.7.21　Logstash の Kubernetes マニフェスト作成

3.7.2　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

「7.6.8　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。

マネージドファイルサービスを使用する場合、次のマネージドファイルサービスを永続ボリュームとしてマウントするために必要な Kubernetes マニフェストについては、それぞれの公式ドキュメントを参照してください。

- Microsoft Azure 環境
  - Azure Files
  - Azure Blob Storage

- Google Cloud Platform 環境
  - Filestore
- Amazon Web Services 環境
  - Amazon EFS

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 デプロイメントのように、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＞

- オートスケールを使用しない場合は、Kubernetes クラスタに起動する Mediator の数と同じ数だけ、永続ボリュームを作成するよう定義した Kubernetes マニフェストを作成してください。

  オートスケールを使用する場合は、「4.2.4　HPA の Kubernetes マニフェストの作成」で作成した Mediator 用の HPA のマニフェストファイルに指定した「スケールアウトできる最大の Pod 数」と同じ数だけ、永続ボリュームを作成するよう定義した 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、「3.6.3　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 の最大台数が「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) ＋ (d)」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))
(d) 1,572,864(稼働日ファイルのサイズ(1.5MB))
```

  例として、(a)が「256」の場合、「256×1,024 ＋ 1,048,576 ＋ 1,572,864」＝「2,883,584」以上となるため、capacity.storage フィールドには「3Mi」（3 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。

＜ストレージクラスおよび永続ボリュームの記述例＞

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: 3Mi
  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: 3Mi
  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: 3Mi
  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: 3Mi
  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: 3Mi
  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 の数を「5」、ワーカーノードを 5 台としています。

＜永続ボリュームを 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: 3Mi
  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: 3Mi
```

```
  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: 3Mi
  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: 3Mi
  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: 3Mi
  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 の数を「5」、ワーカーノードを 5 台としています。

3.7.3 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

Elastic Stack を使用してトラブルシュート情報を収集する場合は、次の手順を実施してください。

「7.6.15 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))) × (h)
(a) Orchestrator-Service、Alternate-Service、Relay-ServiceおよびMediatorが出力する分散トレースのサイズ：300MB
(b) Entity-Service(SQL)、Orchestrator-Service(SQL)、Alternate-Service(SQL)、Entity-Service(TCC)、TP1-Bridgeが出力する分散トレースのサイズ：100MB
(c) Entity-Service(SQL)、Orchestrator-Service(SQL)、Alternate-Service(SQL)、Entity-Service(TCC)、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))) × 15 / (h)
(a) Orchestrator-Service、Alternate-Service、Relay-ServiceおよびMediatorが出力するメトリクスのサイズ：1000MB
(b) Entity-Service(SQL)、Orchestrator-Service(SQL)、Alternate-Service(SQL)、Entity-Service(TCC)、TP1-Bridgeが出力するメトリクスのサイズ：100MB
(c) Entity-Service(SQL)、Orchestrator-Service(SQL)、Alternate-Service(SQL)、Entity-Service(TCC)、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。

注　製品調査時のディスク使用量

障害発生時など製品の調査を行う場合、当社のサポート部署から次のパラメタを true に設定するよう依頼することがあります。その場合、「(A)分散トレースで使用する 1 時間当たりの index サイズ」の見積もり式よりもディスク使用量が増加するため、保存期間が短くなることがあります。

・jp.co.Hitachi.soft.hmppcto.tracing.debug パラメタ

・jp.co.Hitachi.soft.hmppcto.tracing.jdbc.debug パラメタ

・jp.co.Hitachi.soft.hmppcto.tracing.grpc.debug パラメタ

- 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を「3」、データノードの数を「3」、ワーカーノードを 3 台としています。

3.7.4 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

Elastic Stack を使用してトラブルシュート情報を収集する場合は、次の手順を実施してください。

「7.6.15 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「3.6.4　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
```

3.7.5 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）

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。詳細については、取扱説明書「EADS コンテナオーケストレーションツール対応機能」を参照してください。

マネージドファイルサービスを使用する場合、次のマネージドファイルサービスを永続ボリュームとしてマウントするために必要な Kubernetes マニフェストについては、それぞれの公式ドキュメントを参照してください。

- Microsoft Azure 環境
  - Azure Files
  - Azure Blob Storage
- Google Cloud Platform 環境
  - Filestore
- Amazon Web Services 環境
  - Amazon EFS

永続ボリュームの種類を選択する判断基準の例を、次に示します。

- Blue/Green デプロイメントのように、EADS の Pod をデプロイするノードを固定したくない場合には、NFS ストレージを選択してください。
- NFS サーバを用意せず、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時の稼働率が安定します。
  EADS の Pod をデプロイするノードを固定することで、EADS 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 NFS ストレージの場合は、EADS 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、EADS の Pod を不特定のノード上に割り当てます。

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、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 3-15　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

「3.8.4　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

「3.8.4　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

「3.8.4　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。
- 永続ボリュームの種類が NFS ストレージの場合は、StorageClass を定義し、各永続ボリュームの.spec.storageClassName フィールドに、定義した StorageClass 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、次のどちらかを実施してください。
  - 永続ボリュームの種類が NFS ストレージの場合は、各永続ボリュームの.spec.nfs フィールドに、「3.6.6　EADS 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）の場合は、各永続ボリュームの.spec.local フィールドに、「(3)　EADS 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 ストレージの場合は、各永続ボリュームの.spec.nodeAffinity フィールドで、EADS を配置しないノード（コントロールプレーンなど）を決めて、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。例えば、.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。
- 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、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
```

- 永続ボリュームの種類が NFS ストレージの場合は、ログファイル用ボリューム、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の.spec.accessModes フィールドに「ReadWriteMany」を、ストアデータファイル用ボリュームの.spec.accessModes フィールドに「ReadWriteOnce」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）の場合は、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 3-15　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

永続ボリュームを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、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。

■永続ボリュームをローカル永続ボリューム（local）とする場合

```
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
```

■永続ボリュームを NFS ストレージとする場合

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: eads-storage
provisioner: kubernetes.io/no-provisioner
---
apiVersion: v1
kind: PersistentVolume
metadata:
  name: eads-cache-0
spec:
  capacity:
    storage: "514Mi"
  accessModes:
  - ReadWriteMany
  persistentVolumeReclaimPolicy: Retain
  storageClassName: eads-storage
  nfs:
    server: nfs-server.example.com
    path: /eads-cache-volume
  nodeAffinity:
    required:
      nodeSelectorTerms:
```

```
      - matchExpressions:
        - key: kubernetes.io/hostname
          operator: NotIn
          values:
          - control-plane-0
  claimRef:
    name: cache-pvc-0-eads-server-eads-server-0
    namespace: my-namespace
---
apiVersion: v1
kind: PersistentVolume
metadata:
  name: eads-logs-0
spec:
  capacity:
    storage: "1Gi"
  accessModes:
    - ReadWriteMany
  persistentVolumeReclaimPolicy: Retain
  storageClassName: eads-storage
  nfs:
    server: nfs-server.example.com
    path: /eads-logs-volume
  nodeAffinity:
    required:
      nodeSelectorTerms:
      - matchExpressions:
        - key: kubernetes.io/hostname
          operator: NotIn
          values:
          - control-plane-0
  claimRef:
    name: logs-pvc-eads-server-eads-server-0
    namespace: my-namespace
---
apiVersion: v1
kind: PersistentVolume
metadata:
  name: eads-store-0
spec:
  capacity:
    storage: "1Gi"
  accessModes:
    - ReadWriteOnce
  persistentVolumeReclaimPolicy: Retain
  storageClassName: eads-storage
  nfs:
    server: nfs-server.example.com
    path: /eads-store-volume
  nodeAffinity:
    required:
      nodeSelectorTerms:
      - matchExpressions:
        - key: kubernetes.io/hostname
          operator: NotIn
          values:
          - control-plane-0
  claimRef:
```

```
    name: store-pvc-eads-server-eads-server-0
    namespace: my-namespace
---
apiVersion: v1
kind: PersistentVolume
metadata:
  name: eads-cache-1
spec:
  capacity:
    storage: "514Mi"
  accessModes:
  - ReadWriteMany
  persistentVolumeReclaimPolicy: Retain
  storageClassName: eads-storage
  nfs:
    server: nfs-server.example.com
    path: /eads-cache-volume
  nodeAffinity:
    required:
      nodeSelectorTerms:
      - matchExpressions:
        - key: kubernetes.io/hostname
          operator: NotIn
          values:
          - control-plane-0
  claimRef:
    name: cache-pvc-0-eads-server-eads-server-1
    namespace: my-namespace
---
apiVersion: v1
kind: PersistentVolume
metadata:
  name: eads-logs-1
spec:
  capacity:
    storage: "1Gi"
  accessModes:
    - ReadWriteMany
  persistentVolumeReclaimPolicy: Retain
  storageClassName: eads-storage
  nfs:
    server: nfs-server.example.com
    path: /eads-logs-volume
  nodeAffinity:
    required:
      nodeSelectorTerms:
      - matchExpressions:
        - key: kubernetes.io/hostname
          operator: NotIn
          values:
          - control-plane-0
  claimRef:
    name: logs-pvc-eads-server-eads-server-1
    namespace: my-namespace
---
apiVersion: v1
kind: PersistentVolume
metadata:
```

```
  name: eads-store-1
spec:
  capacity:
    storage: "1Gi"
  accessModes:
    - ReadWriteOnce
  persistentVolumeReclaimPolicy: Retain
  storageClassName: eads-storage
  nfs:
    server: nfs-server.example.com
    path: /eads-store-volume
  nodeAffinity:
    required:
      nodeSelectorTerms:
      - matchExpressions:
        - key: kubernetes.io/hostname
          operator: NotIn
          values:
          - control-plane-0
  claimRef:
    name: store-pvc-eads-server-eads-server-1
    namespace: my-namespace
---
apiVersion: v1
kind: PersistentVolume
metadata:
  name: eads-cache-2
spec:
  capacity:
    storage: "514Mi"
  accessModes:
  - ReadWriteMany
  persistentVolumeReclaimPolicy: Retain
  storageClassName: eads-storage
  nfs:
    server: nfs-server.example.com
    path: /eads-cache-volume
  nodeAffinity:
    required:
      nodeSelectorTerms:
      - matchExpressions:
        - key: kubernetes.io/hostname
          operator: NotIn
          values:
          - control-plane-0
  claimRef:
    name: cache-pvc-0-eads-server-eads-server-2
    namespace: my-namespace
---
apiVersion: v1
kind: PersistentVolume
metadata:
  name: eads-logs-2
spec:
  capacity:
    storage: "1Gi"
  accessModes:
    - ReadWriteMany
```

```
  persistentVolumeReclaimPolicy: Retain
  storageClassName: eads-storage
  nfs:
    server: nfs-server.example.com
    path: /eads-logs-volume
  nodeAffinity:
    required:
      nodeSelectorTerms:
      - matchExpressions:
        - key: kubernetes.io/hostname
          operator: NotIn
          values:
          - control-plane-0
  claimRef:
    name: logs-pvc-eads-server-eads-server-2
    namespace: my-namespace
---
apiVersion: v1
kind: PersistentVolume
metadata:
  name: eads-store-2
spec:
  capacity:
    storage: ”1Gi”
  accessModes:
    - ReadWriteOnce
  persistentVolumeReclaimPolicy: Retain
  storageClassName: eads-storage
  nfs:
    server: nfs-server.example.com
    path: /eads-store-volume
  nodeAffinity:
    required:
      nodeSelectorTerms:
      - matchExpressions:
        - key: kubernetes.io/hostname
          operator: NotIn
          values:
          - control-plane-0
  claimRef:
    name: store-pvc-eads-server-eads-server-2
    namespace: my-namespace
```

3.7.6 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 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の Kubernetes アプリケーションおよび RAS

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、「7.6　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、「7.6　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 のコントロールプレーンの 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	備考	ServiceAccount の名称
TP1-Bridge	TP1 連携機能を使用する場合だけ作成します。	<リリース名>-service-account※
Mediator	必ず作成します。	mediator-service-account
Ext-Cons	トライアル版の場合だけ作成します。	ext-cons-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: relay-service-service-account
  namespace: my-namespace
- kind: ServiceAccount
  name: sql-participant-service-account
  namespace: my-namespace
- kind: ServiceAccount
  name: tcc-participant-service-account
  namespace: my-namespace
- kind: ServiceAccount
  name: orchestrator-sql-service-account
  namespace: my-namespace
- kind: ServiceAccount
```

```
  name: alternate-service-service-account
  namespace: my-namespace
- kind: ServiceAccount
  name: alternate-service-sql-service-account
  namespace: my-namespace
- kind: ServiceAccount
  name: <tp1-bridgeのservice-account>※2
  namespace: my-namespace
- kind: ServiceAccount
  name: mediator-service-account
  namespace: my-namespace
- kind: ServiceAccount
  name: ext-cons-service-account※3
  namespace: my-namespace
- kind: ServiceAccount
  name: filebeat-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、「7.6 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

<tp1-bridge の service-account>は、「name: <リリース名>-service-account」で指定してください。

<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

注※3

トライアル版の場合だけ記述します。

■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 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: relay-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: ext-cons-service-account※3
  namespace: my-namespace
- kind: ServiceAccount
  name: filebeat-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、「7.6 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

注※3

トライアル版の場合だけ記述します。

3.7.7　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

Elastic Stack を使用してトラブルシュート情報を収集する場合は、次の手順を実施してください。

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
```

3.7.8 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

Prometheus を使用してトラブルシュート情報を収集する場合は、次の手順を実施してください。

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
```

3.7.9 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

Prometheus を使用してトラブルシュート情報を収集する場合は、次の手順を実施してください。

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
```

3.7.10 uCosminexus Application Runtime 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 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 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の Kubernetes アプリケーションおよび RAS

uCosminexus Application Runtime 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 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。

マネージドファイルサービスを使用する場合、次のマネージドファイルサービスを永続ボリュームとしてマウントするために必要な Kubernetes マニフェストについては、それぞれの公式ドキュメントを参照してください。

- Microsoft Azure 環境
  - Azure Files
  - Azure Blob Storage
- Google Cloud Platform 環境
  - Filestore
- Amazon Web Services 環境
  - Amazon EFS

uCosminexus Application Runtime 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 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-Service/Entity-Service(SQL)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-Service/Entity-Service(SQL)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、「3.6.5　uCosminexus Application Runtime 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 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 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 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。

uCosminexus Application Runtime for Spring Boot スナップショットログ 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 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 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
```

3.7.11　Filebeat の Kubernetes マニフェスト作成

Elastic Stack を使用してトラブルシュート情報を収集する場合は、次の手順を実施してください。

重要

Elasticsearch の TLS による暗号化通信について

Elasticsearch の TLS による暗号化通信を有効にする場合、Filebeat で TLS による暗号化通信の設定が必要です。詳細については、Elasticsearch の公式ドキュメントを参照してください。

(1)　Filebeat のマニフェスト作成

Elasticsearch の公式ドキュメントを参考に、ConfigMap および DaemonSet 以外の Kubernetes マニフェストを作成してください。パラメタおよび設定値の詳細については、Elasticsearch の公式ドキュメントを参照してください。

(2) ConfigMap の作成

メッセージログ収集を設定し、Elasticsearch をメッセージログの永続化サービスとして利用するため、Kubernetes マニフェストの ConfigMap を作成してください。次に記述例を示します。

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: filebeat-config
  namespace: my-namespace
data:
  filebeat.yml: |-
    filebeat.autodiscover:
      providers:
        - type: kubernetes
          templates:
            - condition:                                                    …1
                or:
                  - equals:
                      kubernetes.namespace: my-namespace
              config:
                - type: container
                  paths:
                    - /var/log/containers/*-${data.kubernetes.container.id}.log
    queue:
      disk:
        max_size: 10GB
        read_ahead: 512
    processors:
      - add_kubernetes_metadata:
          in_cluster: true
    output.logstash:                                                        …2
      hosts: ["logstash:5044"]
```

記述例の番号は、説明の番号と対応しています。

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。

＜説明＞

1. 「- condition:」の行から「kubernetes.namespace: my-namespace」の行で、メッセージログの収集対象とする Namespace 名を設定します。

   複数の Namespace からメッセージログを収集する場合は、次のように設定を追記します。

   ```
              - condition:
                  or:
                    - equals:
                        kubernetes.namespace: my-namespace
                    - equals:
                        kubernetes.namespace: my-namespace-2
   ```

2.「output.logstash:」の行から「hosts: ["logstash:5044"]」の行で、Logstash へメッセージログを送信するための設定をします。

(3) DaemonSet の作成

メッセージログ収集の設定のため、Kubernetes マニフェストの DaemonSet を作成してください。DaemonSet を用いることで、クラスタ内のすべてのノード上に Filebeat の Pod が 1 台ずつ起動し、各ノードのメッセージログを収集します。次に記述例を示します。

```
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: filebeat
  namespace: my-namespace
  labels:
    app.kubernetes.io/name: filebeat
spec:
  selector:
    matchLabels:
      app.kubernetes.io/name: filebeat
  template:
    metadata:
      labels:
        app.kubernetes.io/name: filebeat
    spec:
      serviceAccountName: filebeat
      affinity:                                                    …1
        nodeAffinity:
          requiredDuringSchedulingIgnoredDuringExecution:
            nodeSelectorTerms:
              - matchExpressions:
                  - key: kubernetes.io/hostname
                    operator: In
                    values:
                      - node-1
                      - node-2
                      - node-3
      containers:
        - name: filebeat
          image: myregistry.example.com/myproject/filebeat:V.R.S
          args:
            - "-e"
            - "-E"
            - "http.enabled=true"
          env:
            - name: NODE_NAME
              valueFrom:
                fieldRef:
                  fieldPath: spec.nodeName
          securityContext:
            runAsUser: 0
          resources:
            requests:
              cpu: 100m
              memory: 200Mi
```

```
          limits:
            memory: 200Mi
        volumeMounts:
          - name: config
            mountPath: /usr/share/filebeat/filebeat.yml
            subPath: filebeat.yml
            readOnly: true
          - name: varlogcontainers
            mountPath: /var/log/containers
            readOnly: true
          - name: varlogpods
            mountPath: /var/log/pods
            readOnly: true
          - name: varrundockersock
            mountPath: /var/run/docker.sock
            readOnly: true
      volumes:
        - name: config
          configMap:
            name: filebeat-config
            defaultMode: 0600
        - name: varlogcontainers
          hostPath:
            path: /var/log/containers
        - name: varlogpods
          hostPath:
            path: /var/log/pods
        - name: varrundockersock
          hostPath:
            path: /var/run/docker.sock
```

記述例の番号は、説明の番号と対応しています。

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。

＜説明＞

1.「affinity:」の行から「- node-3」の行の設定は省略できます。Filebeat を起動させたいノードを明示的に指定する場合に設定します。この設定を省略した場合、DaemonSet の動作によって、クラスタ内のすべてのノードで Filebeat の Pod が自動的に作成されます。

3.7.12　Jaeger の Kubernetes マニフェスト作成

Jaeger を使用してトラブルシュート情報を収集する場合は、次の手順を実施してください。

この項目は、クラウド環境（例：Amazon Web Services 環境）で Jaeger を使用する場合にも適用されます。

重要

Elasticsearch の TLS による暗号化通信について

Elasticsearch の TLS による暗号化通信を有効にする場合、Jaeger で TLS による暗号化通信の設定が必要です。詳細については、Elasticsearch および Jaeger の公式ドキュメントを参照してください。

(1) Jaeger のマニフェスト作成

Jaeger の公式ドキュメントを参考に、ConfigMap 以外の Kubernetes マニフェストを作成してください。パラメタおよび設定値の詳細については、Jaeger の公式ドキュメントを参照してください。

(2) ConfigMap の作成

分散トレースの収集・閲覧の設定、および Elasticsearch を分散トレースの永続化サービスとして利用する設定をするため、Kubernetes マニフェストの ConfigMap を作成してください。次に記述例を示します。

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: jaeger-config
  namespace: my-namespace
data:
  jaeger.yaml: |
    extensions:                                                    …1
      jaeger_storage:
        backends:
          primary_store:
            elasticsearch:
              server_urls: ["http://elasticsearch-master:9200"]
              auth:
                basic:
                  username: "${env:ES_USERNAME}"
                  password: "${env:ES_PASSWORD}"
      jaeger_query:                                                …2
        storage:
          traces: primary_store
        http:
          endpoint: 0.0.0.0:16686
    receivers:
      otlp:
        protocols:
          http:
            endpoint: 0.0.0.0:4318
    processors:                                                    …3
      batch:
        timeout: 200ms
        send_batch_size: 8192
        send_batch_max_size: 0
    exporters:                                                     …4
      jaeger_storage_exporter:
```

```
      trace_storage: primary_store
      queue:
        num_consumers: 10
        queue_size: 2000
  service:                                                    …5
    extensions: [jaeger_storage, jaeger_query]
    pipelines:
      traces:
        receivers: [otlp]
        processors: [batch]
        exporters: [jaeger_storage_exporter]
    telemetry:                                                …6
      metrics:
        readers:
          - pull:
              exporter:
                prometheus:
                  host: "0.0.0.0"
                  port: 8888
```

記述例の番号は、説明の番号と対応しています。

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。

＜説明＞

1. Elasticsearch との接続を行います。

   「extensions:」の行から「server_urls: ["http://elasticsearch-master:9200"]」の行で、Jaeger のストレージ（保存先）に Elasticsearch を設定します。

   「auth:」の行から「password: "${env:ES_PASSWORD}"」の行で、Elasticsearch への接続時に使用する認証方式を設定します。Elasticsearch の認証に必要な username と password を、環境変数 ES_USERNAME、ES_PASSWORD を使用して設定します。

2. 「jaeger_query:」の行から「endpoint: 0.0.0.0:16686」の行で、分散トレースを閲覧するための設定をします。

   「receivers:」の行から「endpoint: 0.0.0.0:4318」の行で、分散トレースを収集するための設定をします。

3. 「processors:」の行から「send_batch_max_size: 0」の行で、トレースを一定時間または一定件数でまとめて処理する（バッチ処理）ためのプロセッサを設定します。OpenTelemetry Collector の推奨設定で、処理効率向上とバックエンド負荷の平準化を目的として有効化します。

4. 「exporters:」の行から「queue_size: 2000」の行で、Jaeger のストレージに書き込む Exporter の同時実行するワーカ数とキューの大きさを設定します。

5. 「service:」の行から「exporters: [jaeger_storage_exporter]」の行で、Jaeger のストレージと UI を有効化して、OTLP で受け取ったトレースをストレージに保存する設定をします。

6. 「telemetry:」の行から「port: 8888」の行で、Jaeger（OpenTelemetry Collector）の内部メトリクスを Prometheus がスクレイプするために、Prometheus 形式でポート 8888 に/metrics を公開します。

3.7.13　Prometheus の Kubernetes マニフェスト作成

Prometheus を使用してトラブルシュート情報を収集する場合は、次の手順を実施してください。

> **重要**
>
> メトリクスのスクレイプ時の通信について
>
> ユーザアプリケーションの暗号化通信を使用する場合、Prometheus のメトリクス収集の設定についても暗号化の設定が必要です。詳細については、New Relic および Prometheus の公式ドキュメントを参照してください。

(1)　Prometheus のマニフェスト作成

Prometheus の公式ドキュメントを参考に、ConfigMap 以外の Kubernetes マニフェストを作成してください。パラメタの内容および設定値の詳細については、Prometheus の公式ドキュメントを参照してください。

(2)　ConfigMap の作成

メトリクス収集対象の定義、およびに Metricbeat へのメトリクス送信のため、Kubernetes マニフェストの ConfigMap を作成してください。次に記述例を示します。

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: prometheus-config
  namespace: my-namespace
data:
  prometheus.yml: |
    global:
      scrape_interval: 15s
      evaluation_interval: 30s
    scrape_configs:                          …1
      - job_name: 'hmppcto-metrics-scraper'
        kubernetes_sd_configs:
          - role: pod
        relabel_configs:
          - source_labels: [__meta_kubernetes_pod_annotation_prometheus_io_scrape]
            action: keep
            regex: "true"
          - source_labels: [__meta_kubernetes_pod_annotation_prometheus_io_path]
            action: replace
            target_label: __metrics_path__
            regex: ([^¥n]+)
          - source_labels: [__address__, __meta_kubernetes_pod_annotation_prometheus_io_port
]
            action: replace
            regex: (.+):(?:¥d+);(¥d+)
            replacement: ${1}:${2}
            target_label: __address__
```

```
        - source_labels: [__meta_kubernetes_pod_annotation_prometheus_io_scheme]
          regex: (https|http)
          target_label: __scheme__
        - source_labels: [__meta_kubernetes_namespace]
          action: replace
          target_label: namespace
        - source_labels: [__meta_kubernetes_pod_name]
          action: replace
          target_label: pod_name
        - source_labels: [__meta_kubernetes_pod_label_app_kubernetes_io_name]
          action: replace
          target_label: label_app_kubernetes_io_name
        - source_labels: [__meta_kubernetes_pod_container_name]
          regex: "dependency-checker"
          action: drop
        - source_labels: [__meta_kubernetes_pod_container_name]
          regex: "filebeat-sidecar"
          action: drop
    - job_name: 'jaeger-scraper'  …2
      kubernetes_sd_configs:
      - role: endpoints
      metrics_path: /metrics
      relabel_configs:
      - source_labels: [__address__]
        regex: (.+):¥d+
        replacement: ${1}:8888
        target_label: __address__
        action: replace
      - source_labels: [__meta_kubernetes_service_name]
        regex: jaeger
        action: keep
      - source_labels: [__meta_kubernetes_namespace]
        action: replace
        target_label: namespace
      - source_labels: [__meta_kubernetes_pod_name]
        action: replace
        target_label: pod_name
      - source_labels: [__meta_kubernetes_pod_label_app_kubernetes_io_name]
        action: replace
        target_label: label_app_kubernetes_io_name
  remote_write:                          …3
  - url: http://metricbeat:9201/write
    queue_config:
      capacity: 10000
      max_shards: 50
      max_samples_per_send: 2000
    enable_http2: true                    …4
```

記述例の番号は、説明の番号と対応しています。

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。

＜説明＞

1.「scrape_configs:」の行から「- job_name: 'jaeger-scraper'」の 1 つ上の「action: drop」の行で、HMP-PCTO のメトリクスを収集します。

次の設定で、Kubernetes マニフェストの.spec.template.metadata.annotations に設定されている 3 つのキーの値を参照して、メトリクスを収集します。この設定は変更しないでください。

- prometheus.io/scrape
- prometheus.io/path
- prometheus.io/port

```
- job_name: 'hmppcto-metrics-scraper'
   kubernetes_sd_configs:
     - role: pod
   relabel_configs:
    - source_labels: [__meta_kubernetes_pod_annotation_prometheus_io_scrape]
      action: keep
      regex: "true"
    - source_labels: [__meta_kubernetes_pod_annotation_prometheus_io_path]
      action: replace
      target_label: __metrics_path__
      regex: (.+)
    - source_labels: [__address__, __meta_kubernetes_pod_annotation_prometheus_io_port]
      action: replace
      regex: (.+):(?:¥d+);(¥d+)
      replacement: ${1}:${2}
      target_label: __address__
```

次の設定は、HMP-PCTO の Pod の接続スキームを正しく認識するためのものです。これによって、http および https の両方の接続スキームでメトリクスが取得できます。

```
  - source_labels: [__meta_kubernetes_pod_annotation_prometheus_io_scheme]
    regex: (https|http)
    target_label: __scheme__
```

次の設定で、Pod に設定されている Kubernetes のメタデータをラベルとして設定します。追加されるラベルを次に示します。ラベルに関する設定は必要に応じて追加してください。

- namespace：Pod の名前空間
- pod_name：Pod 名
- label_app_kubernetes_io_name：Pod の.spec.template.metadata.labels の「app.kubernetes.io/name」に設定されている値

```
  - source_labels: [__meta_kubernetes_namespace]
    action: replace
    target_label: namespace
  - source_labels: [__meta_kubernetes_pod_name]
    action: replace
    target_label: pod_name
  - source_labels: [__meta_kubernetes_pod_label_app_kubernetes_io_name]
    action: replace
    target_label: label_app_kubernetes_io_name
```

次の設定で、HMP-PCTO の Pod 内に存在するコンテナを、メトリクスの収集対象から除く設定をします。この設定は変更しないでください。

```
  - source_labels: [__meta_kubernetes_pod_container_name]
    regex: "dependency-checker"
    action: drop
  - source_labels: [__meta_kubernetes_pod_container_name]
    regex: "filebeat-sidecar"
    action: drop
```

2. 「- job_name: 'jaeger-scraper'」の行から「target_label: label_app_kubernetes_io_name」の行で、Jaeger のメトリクスを収集します。Jaeger のメトリクスを収集することで、受信した Span を格納する内部キューの使用状況を確認し、チューニングができます。Jaeger を使用する場合に設定してください。

   次の設定で、Jaeger のメトリクスを収集します。この設定は変更しないでください。

```
  - job_name: 'jaeger-scraper'
    kubernetes_sd_configs:
    - role: endpoints
    metrics_path: /metrics
    relabel_configs:
    - source_labels: [__address__]
      regex: (.+):¥¥d+
      replacement: ${1}:8888
      target_label: __address__
      action: replace
    - source_labels: [__meta_kubernetes_service_name]
      regex: jaeger
      action: keep
```

   次の設定で、Pod に設定されている Kubernetes のメタデータをラベルとして設定します。追加されるラベルを次に示します。ラベルに関する設定は必要に応じて追加してください。

   - namespace：Pod の名前空間
   - pod_name：Pod 名
   - label_app_kubernetes_io_name：Pod の.spec.template.metadata.labels の「app.kubernetes.io/name」に設定されている値

```
    - source_labels: [__meta_kubernetes_namespace]
      action: replace
      target_label: namespace
    - source_labels: [__meta_kubernetes_pod_name]
      action: replace
      target_label: pod_name
    - source_labels: [__meta_kubernetes_pod_label_app_kubernetes_io_name]
      action: replace
      target_label: label_app_kubernetes_io_name
```

3. 「remote_write:」の行から「max_samples_per_send: 2000」の行で、Metricbeat にメトリクスを送信するための設定を定義します。

4. Metricbeat はデータを送信する際の通信プロトコルとして HTTP/2 に対応しているため、HTTP/2 の通信を利用できるように明示的に設定します。

3.7.14 Microsoft Azure のマネージド Prometheus によるメトリクス収集の Kubernetes マニフェストの作成

Microsoft Azure 環境でマネージド Prometheus を使用して HMP-PCTO のメトリクスを収集する場合、次の手順を実施してください。

Microsoft Azure 環境のマネージド Prometheus では、サービスを起動しただけでは HMP-PCTO のメトリクスを収集できないため、次のどちらかの定義を設定してください。詳細については、Microsoft Azure の公式ドキュメントを参照してください。

- configmap
- カスタムリソース定義

設定例として、configmap による設定方法の一つである ama-metrics-prometheus-config の記述例およびデプロイ手順を示します。

Kubernetes マニフェストに設定されたアノテーション prometheus.io/*の値を使用してメトリクスを収集します。HMP-PCTO の各 Kubernetes マニフェストの.spec.template.metadata.annotations に次の内容を指定してください。Mediator および TP1-Bridge については、Helm チャート内で各アノテーションが設定されています。

キー名	値	備考
prometheus.io/scrape	true	値が true の場合にメトリクスの収集対象となります。
prometheus.io/path	/hmppcto/metrics※	メトリクス収集用のエンドポイントです。
prometheus.io/port	http ポート番号（Prometheus メトリクス収集用）に指定した値	メトリクス収集用のポートです。
prometheus.io/scheme	http または https	メトリクス収集用のスキームです。

注※

「prometheus.io/path」の値は、コンテキストパスを設定していない場合の指定値です。

コンテキストパスを設定している場合は、次の形式で指定してください。

```
prometheus.io/path: <コンテキストパス>/hmppcto/metrics
```

コンテキストパスが/context-path の場合の指定例を次に示します。

```
prometheus.io/path: /context-path/hmppcto/metrics
```

ama-metrics-prometheus-config に設定するパラメタ内容および設定値の詳細については、Microsoft Azure および Prometheus の公式ドキュメントを参照してください。

重要

メトリクスのスクレイプ時の通信について

ユーザアプリケーションの暗号化通信を使用する場合、Prometheus のメトリクス収集の設定についても暗号化の設定が必要です。詳細については Microsoft Azure および Prometheus の公式ドキュメントを参照してください。

(1) Kubernetes マニフェストの記述例

Kubernetes マニフェストの記述例を次に示します。この例では、Kubernetes マニフェストのファイル名を「ama-metrics-prometheus-config.yaml」としています。

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: ama-metrics-prometheus-config
  namespace: kube-system
data:
  prometheus-config: |
    global:
      scrape_interval: 30s
    scrape_configs:
      - job_name: 'hmppcto-metrics-scraper'
        kubernetes_sd_configs:
          - role: pod
        relabel_configs:
          - source_labels: [__meta_kubernetes_pod_annotation_prometheus_io_scrape]
            action: keep
            regex: "true"
          - source_labels: [__meta_kubernetes_pod_annotation_prometheus_io_path]
            action: replace
            target_label: __metrics_path__
            regex: (.+)
          - source_labels: [__address__, __meta_kubernetes_pod_annotation_prometheus_io_port
]
            action: replace
            regex: (.+):(?:¥d+);(¥d+)
            replacement: ${1}:${2}
            target_label: __address__
          - source_labels: [__meta_kubernetes_pod_annotation_prometheus_io_scheme]
            regex: (https|http)
            target_label: __scheme__
          - source_labels: [__meta_kubernetes_namespace]
            action: replace
            target_label: namespace
          - source_labels: [__meta_kubernetes_pod_name]
            action: replace
            target_label: pod_name
          - source_labels: [__meta_kubernetes_pod_label_app_kubernetes_io_name]
            action: replace
            target_label: label_app_kubernetes_io_name
          - source_labels: [__meta_kubernetes_pod_container_name]
```

```
        regex: ”dependency-checker”
        action: drop
```

重要

metadata.namespace パラメタの設定値を「kube-system」から変更しないでください。

(2) Kubernetes マニフェストのデプロイ

作成した 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 ama-metrics-prometheus-config.yaml
```

3.7.15 Google Cloud Platform のマネージド Prometheus によるメトリクス収集の Kubernetes マニフェスト作成

Google Cloud Platform 環境でマネージド Prometheus を使用して HMP-PCTO のメトリクスを収集する場合、次の定義を設定してください。詳細については、Google Cloud Platform の公式ドキュメントを参照してください。

- カスタムリソース定義

設定例として、カスタムリソースによる設定方法の一つである gcp-metrics-prometheus-podmonitoring.yaml の記述例およびデプロイ手順を示します。

収集対象を設定する matchLabels パラメタには HMP-PCTO の Pod が起動している spec.selector.matchLabels を設定してください。

重要

メトリクスのスクレイプ時の通信について

ユーザアプリケーションの暗号化通信を使用する場合、Prometheus のメトリクス収集の設定についても暗号化の設定が必要です。詳細については Google および Prometheus の公式ドキュメントを参照してください。

(1) Kubernetes マニフェストの記述例

Kubernetes マニフェストの記述例を次に示します。この例では、Kubernetes マニフェストのファイル名を「gcp-metrics-prometheus-podmonitoring.yaml」としています。

```
apiVersion: monitoring.googleapis.com/v1
kind: PodMonitoring
```

```
metadata:
  name: mediator-podmonitoring
  namespace: my-namespace
spec:
  selector:
    matchLabels:
      app.kubernetes.io/name: mediator
  endpoints:
  - port: http
    path: /hmppcto/metrics
---
apiVersion: monitoring.googleapis.com/v1
kind: PodMonitoring
metadata:
  name: sqlparticipant-podmonitoring
  namespace: my-namespace
spec:
  selector:
    matchLabels:
      app.kubernetes.io/name: sql-participant
  endpoints:
  - port: http
    path: /hmppcto/metrics     …1
---
apiVersion: monitoring.googleapis.com/v1
kind: PodMonitoring
metadata:
  name: tccparticipant-podmonitoring
  namespace: my-namespace
spec:
  selector:
    matchLabels:
      app.kubernetes.io/name: tcc-participant
  endpoints:
  - port: http
    path: /hmppcto/metrics     …1
---
apiVersion: monitoring.googleapis.com/v1
kind: PodMonitoring
metadata:
  name: entity-podmonitoring
  namespace: my-namespace
spec:
  selector:
    matchLabels:
      app.kubernetes.io/name: relay-service
  endpoints:
  - port: http
    path: /hmppcto/metrics     …1
---
apiVersion: monitoring.googleapis.com/v1
kind: PodMonitoring
metadata:
  name: tp1bridge-podmonitoring
  namespace: my-namespace
spec:
  selector:
    matchLabels:
```

```
      app.kubernetes.io/name: tp1-bridge
  endpoints:
  - port: http
    path: /hmppcto/metrics
---
apiVersion: monitoring.googleapis.com/v1
kind: PodMonitoring
metadata:
  name: orchestrator-podmonitoring
  namespace: my-namespace
spec:
  selector:
    matchLabels:
      app.kubernetes.io/name: orchestrator
  endpoints:
  - port: http
    path: /hmppcto/metrics     …1
```

記述例の番号は、説明の番号と対応しています。

＜説明＞

1.「spec.endpoints.path」の説明は、コンテキストパスを設定していない場合の記述です。
コンテキストパスを設定している場合は、次の形式で指定してください。

```
spec.endpoints.path: <コンテキストパス>/hmppcto/metrics
```

コンテキストパスが/context-path の場合の設定例を次に示します。

```
spec.endpoints.path: /context-path/hmppcto/metrics
```

(2) Kubernetes マニフェストのデプロイ

作成した 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 gcp-metrics-prometheus-podmonitoring.yaml
```

3.7.16 Amazon Web Services のマネージド Prometheus によるメトリクス収集の Kubernetes マニフェスト作成

Amazon Web Services 環境でマネージド Prometheus を使用して HMP-PCTO のメトリクスを収集する場合、次の手順を実施してください。詳細については、Amazon Web Services の公式ドキュメントを参照してください。

設定例として、設定方法の一つである Kubernetes マニフェストに設定されたアノテーションによるメトリクスの収集の実施例を示します。

重要

メトリクスのスクレイプ時の通信について

ユーザアプリケーションの暗号化通信を使用する場合、Prometheus のメトリクス収集の設定についても暗号化の設定が必要です。詳細については、Amazon Web Services および Prometheus の公式ドキュメントを参照してください。

(1) Kubernetes マニフェストへのアノテーションの追加

Amazon Web Services では、Kubernetes マニフェストに設定されたアノテーション prometheus.io/* の値を使用してメトリクスを収集します。HMP-PCTO の各 Kubernetes マニフェストの.spec.template.metadata.annotations に次の内容を指定してください。Mediator および TP1-Bridge については、Helm チャート内で各アノテーションが設定されています。

表 3-16 各 Kubernetes マニフェストでの.spec.template.metadata.annotations の設定 (Amazon Web Services 環境)

キー名	値	備考
prometheus.io/scrape	true	値が true の場合にメトリクスの収集対象になります。
prometheus.io/path	/hmppcto/metrics※	メトリクス収集用のエンドポイントです。
prometheus.io/port	http ポート番号（Prometheus メトリクス収集用）に指定した値	メトリクス収集用のポートです。
prometheus.io/scheme	http または https	メトリクス収集用のスキームです。

注※

「prometheus.io/path」の値は、コンテキストパスを設定していない場合の指定値です。

コンテキストパスを設定している場合は、次の形式で指定してください。

```
prometheus.io/path: <コンテキストパス>/hmppcto/metrics
```

コンテキストパスが/context-path の場合の指定例を次に示します。

```
prometheus.io/path: /context-path/hmppcto/metrics
```

3.7.17 New Relic によるメトリクス収集の Kubernetes マニフェスト作成

New Relic を使用して HMP-PCTO のメトリクスを収集する場合、次の手順を実施してください。詳細については、New Relic の公式ドキュメントを参照してください。

設定例として、設定方法の一つである Kubernetes マニフェストに設定されたアノテーションによるメトリクスの収集の実施例を示します。

重要

メトリクスのスクレイプ時の通信について

ユーザアプリケーションの暗号化通信を使用する場合、New Relic のメトリクス収集の設定についても暗号化の設定が必要です。詳細については、New Relic の公式ドキュメントを参照してください。

(1) Kubernetes マニフェストへのアノテーションの追加

New Relic では、Kubernetes マニフェストに設定されたアノテーション prometheus.io/*の値を使用してメトリクスを収集します。HMP-PCTO の各 Kubernetes マニフェストの.spec.template.metadata.annotations に次の内容を指定してください。Mediator および TP1-Bridge については、Helm チャート内で各アノテーションが設定されています。

表 3-17 各 Kubernetes マニフェストでの.spec.template.metadata.annotations の設定 (New Relic を使用する場合)

キー名	値	備考
prometheus.io/scrape	true	値が true の場合にメトリクスの収集対象となります。
prometheus.io/path	/hmppcto/metrics※	メトリクス収集用のエンドポイントです。
prometheus.io/port	http ポート番号（Prometheus を使用したメトリクスの収集用）に指定した値	メトリクス収集用のポートです。
prometheus.io/scheme	http または https	メトリクス収集用のスキームです。

注※

「prometheus.io/path」の値は、コンテキストパスを設定していない場合の指定値です。

コンテキストパスを設定している場合は、次の形式で指定してください。

```
prometheus.io/path: <コンテキストパス>/hmppcto/metrics
```

コンテキストパスが/context-path の場合の指定値を次に示します。

```
prometheus.io/path: /context-path/hmppcto/metrics
```

3.7.18 Red Hat OpenShift Container Platform モニタリングの Prometheus によるメトリクス収集の Kubernetes マニフェスト作成

Red Hat OpenShift Container Platform 環境で Red Hat OpenShift Container Platform モニタリングの Prometheus を使用して HMP-PCTO のメトリクスを収集する場合、次の定義を設定してください。

- カスタムリソース定義

設定例として、カスタムリソースによる設定方法の一つである ServiceMonitor によるメトリクス収集の記述例およびデプロイ手順を示します。

重要

Kubernetes マニフェストに記載されたアノテーションについて

HMP-PCTO の各 Kubernetes マニフェストの記述例、および Mediator や TP1-Bridge の Helm チャートにはメトリクス収集のためのアノテーションがあらかじめ設定されていますが、Red Hat OpenShift Container Platform モニタリングの Prometheus では.spec.template.metadata.annotations による収集対象の判定をしていません。アノテーションとして設定した次の内容は反映されないため注意してください。

表 3-18　Red Hat OpenShift Container Platform モニタリングの Prometheus に設定が反映されないメトリクス収集用のアノテーション

キー名	値	備考
prometheus.io/scrape	true	値が true の場合にメトリクスの収集対象になります。
prometheus.io/path	/hmppcto/metrics※	メトリクス収集用のエンドポイントです。
prometheus.io/port	http ポート番号（Prometheus を使用したメトリクスの収集用）に指定した値	メトリクス収集用のポートです。
prometheus.io/scheme	http または https	メトリクス収集用のスキームです。

注※

「prometheus.io/path」の値は、コンテキストパスを設定していない場合の指定例です。
コンテキストパスを設定している場合は、次の形式で指定してください。

```
path: <コンテキストパス>/hmppcto/metrics
```

コンテキストパスが/context-path の場合の指定例を次に示します。

```
path: /context-path/hmppcto/metrics
```

重要

メトリクスのスクレイプ時の通信について

ユーザアプリケーションの暗号化通信を使用する場合、Prometheus のメトリクス収集の設定についても暗号化の設定が必要です。詳細については、Red Hat OpenShift Container Platform および Prometheus の公式ドキュメントを参照してください。

(1) ServiceMonitor の設定

メトリクスを収集するには、Red Hat OpenShift Container Platform モニタリングの Prometheus が監視対象のサービスを認識するための ServiceMonitor の設定をする必要があります。

表 3-19 Red Hat OpenShift Container Platform モニタリングの Prometheus が監視対象のサービスを認識するための ServiceMonitor の設定

キー名	値	備考
spec.endpoints.port	http	Prometheus を使用したメトリクス収集で使用するポートを定義している service の名称（spec.ports[].name）です。
spec.endpoints.path	/hmppcto/metrics※	メトリクス収集用のエンドポイントです。
spec.endpoints.interval	15s	メトリクス収集の間隔です。

注※

「spec.endpoints.path」の値は、コンテキストパスを設定していない場合の指定値です。

コンテキストパスを設定している場合は、次の形式で指定してください。

```
path: <コンテキストパス>/hmppcto/metrics
```

コンテキストパスが/context-path の場合の指定例を次に示します。

```
path: /context-path/hmppcto/metrics
```

この例では、Kubernetes マニフェストのファイル名を「my-namespace-metrics-monitor.yaml」としています。Kubernetes マニフェストの記述例を次に示します。

```
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  name: my-namespace-metrics-monitor
  namespace: my-namespace
  labels:
    release: prometheus-user-workload  …1
spec:
  selector:
    namespaceSelector:
      matchNames:
        - my-namespace
  endpoints:
    - port: http
```

```
    path: /hmppcto/metrics
    interval: 15s
```

記述例の番号は、説明の番号と対応しています。

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。

＜説明＞

1. ServiceMonitor の metadata.labels.release として「prometheus-user-workload」を指定します。「prometheus-user-workload」を指定することで、Red Hat OpenShift Container Platform モニタリングの Prometheus が監視対象のサービスをメトリクスの収集対象として認識します。

(2) Kubernetes マニフェストのデプロイ

作成した 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 my-namespace-metrics-monitor.yaml
```

3.7.19 Metricbeat の Kubernetes マニフェスト作成

Elastic Stack を使用してトラブルシュート情報を収集する場合は、次の手順を実施してください。

重要

Elasticsearch の TLS による暗号化通信について

Elasticsearch の TLS による暗号化通信を有効にする場合、Metricbeat で TLS による暗号化通信設定が必要です。詳細については、Elasticsearch の公式ドキュメントを参照してください。

(1) Metricbeat のマニフェスト作成

Elasticsearch の公式ドキュメントを参考に、ConfigMap 以外の Kubernetes マニフェストを作成してください。パラメタおよび設定値の詳細については、Elasticsearch の公式ドキュメントを参照してください。

(2) ConfigMap の作成

メトリクス収集を設定し、Elasticsearch をメトリクスの永続化サービスとして利用するため、Kubernetes マニフェストの ConfigMap を作成してください。次に記述例を示します。

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: metricbeat-config
```

```
  namespace: my-namespace
data:
  metricbeat.yml: |-
    processors:                                                    …1
      - add_kubernetes_metadata:
          in_cluster: true
    metricbeat.modules:                                            …2
      - module: prometheus
        metricsets: [”remote_write”]
        host: ”0.0.0.0”
        port: ”9201”
        enabled: true
        period: 10s
        relabel_configs:
          - source_labels: [’__name__’]
            regex: ’^prometheus_remote_write.*’
            action: drop
    queue:
      disk:
        max_size: 10GB
        read_ahead: 512
    output.elasticsearch:                                           …3
      hosts: [”http://elasticsearch-master:9200”]
```

記述例の番号は、説明の番号と対応しています。

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。

＜説明＞

1. 「processors:」の行から「in_cluster: true」の行で、採取したメトリクスにメタデータを付与します。
2. 「metricbeat.modules:」の行から「action: drop」の行で、Prometheus からメトリクスを受け取るための設定を定義します。
3. 「output.elasticsearch:」の行から「hosts: ["http://elasticsearch-master:9200"]」の行で、Elasticsearch へメトリクスを送信するための設定を定義します。

3.7.20　Elasticsearch の Kubernetes マニフェスト作成

Elastic Stack を使用してトラブルシュート情報を収集する場合は、次の手順を実施してください。この手順は、クラウド環境（例：Microsoft Azure 環境）で Elasticsearch を使用する場合にも適用されます。

(1)　Elasticsearch のマニフェスト作成

Elasticsearch の公式ドキュメントを参考に、StatefulSet 以外の Kubernetes マニフェストを作成してください。クラスタ構成で構築する場合、マスターノード用とデータノード用の 2 つのマニフェストを作成する必要があります。設定できるパラメタおよび設定値の詳細については、Elasticsearch の公式ドキュメントを参照してください。

(a) シングル構成の場合

シングル構成の場合の記述例を示します。

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: elasticsearch
  namespace: my-namespace
spec:
  serviceName: elasticsearch
  replicas: 1
  selector:
    matchLabels:
      app: elasticsearch
  template:
    metadata:
      labels:
        app: elasticsearch
    spec:
      serviceAccountName: elasticsearch-master-service-account                    …1
      securityContext:
        fsGroup: 1000
        fsGroupChangePolicy: "OnRootMismatch"
      initContainers:
        - name: configure-sysctl
          image: myregistry.example.com/myproject/elasticsearch:V.R.S
          imagePullPolicy: IfNotPresent
          securityContext:
            runAsUser: 0
            privileged: true
          command:
            - sh
            - -c
            - |
              set -e
              sysctl -w vm.max_map_count=262144
      containers:
        - name: elasticsearch
          image: myregistry.example.com/myproject/elasticsearch:V.R.S
          imagePullPolicy: IfNotPresent
          securityContext:
            runAsNonRoot: true
            runAsUser: 1000
            capabilities:
              drop: ["ALL"]
          ports:
            - containerPort: 9200
              name: http
          env:
            - name: discovery.type
              value: "single-node"
            - name: xpack.security.enabled
              value: "false"
            - name: ES_JAVA_OPTS
              value: "-Xms1g -Xmx1g"
          volumeMounts:                                                           …2
```

```
          - name: elasticsearch-data
            mountPath: /usr/share/elasticsearch/data
          - name: elasticsearch-repo
            mountPath: /usr/share/elasticsearch/repository
  volumeClaimTemplates:                                                    …3
    - metadata:
        name: elasticsearch-data
      spec:
        storageClassName: elasticsearch-data-masternode
        accessModes:
          - ReadWriteOnce
        resources:
          requests:
            storage: 3Gi
    - metadata:
        name: elasticsearch-repo
      spec:
        storageClassName: elasticsearch-repo
        accessModes:
          - ReadWriteMany
        resources:
          requests:
            storage: 3Gi
```

記述例の番号は、説明の番号と対応しています。

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。

＜説明＞

1. 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。
2. 「volumeMounts:」の行から「mountPath: /usr/share/elasticsearch/repository」の行で、Elasticsearch のデータおよびスナップショットを永続化するため、PVC を Pod 内の所定ディレクトリにマウントします。
3. 「volumeClaimTemplates:」の行から最終行で、各 Pod に自動でプロビジョニングされる永続ボリュームを定義します。永続ボリュームは事前に作成する必要があります。詳細については、「3.7.3 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」、「3.7.4　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」を参照してください。

(b)　クラスタ構成（マスターノード）の場合

クラスタ構成の場合のマスターノード用の記述例を示します。

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: elasticsearch-master
  namespace: my-namespace
spec:
```

```
  serviceName: elasticsearch-master-headless
  replicas: 3
  selector:
    matchLabels:
      app: elasticsearch-master
  template:
    metadata:
      labels:
        app: elasticsearch-master
    spec:
      serviceAccountName: elasticsearch-master-service-account    …1
      securityContext:
        fsGroup: 1000
        fsGroupChangePolicy: "OnRootMismatch"
      initContainers:
        - name: configure-sysctl
          image: myregistry.example.com/myproject/elasticsearch:V.R.S
          imagePullPolicy: IfNotPresent
          securityContext:
            runAsUser: 0
            privileged: true
          command:
            - sh
            - -c
            - |
              set -e
              sysctl -w vm.max_map_count=262144
      containers:
        - name: elasticsearch
          image: myregistry.example.com/myproject/elasticsearch:V.R.S
          securityContext:
            runAsNonRoot: true
            runAsUser: 1000
            capabilities:
              drop: ["ALL"]
          ports:
            - containerPort: 9300
              name: transport
          env:
            - name: node.roles
              value: '["master"]'
            - name: cluster.initial_master_nodes
              value: '["elasticsearch-master-0","elasticsearch-master-1","elasticsearch-mast
er-2"]'
            - name: discovery.seed_hosts
              value: '["elasticsearch-master-0.elasticsearch-master-headless","elasticsearch
-master-1.elasticsearch-master-headless","elasticsearch-master-2.elasticsearch-master-headle
ss"]'
            - name: xpack.security.enabled
              value: "false"
            - name: ES_JAVA_OPTS
              value: "-Xms1g -Xmx1g"
          volumeMounts:                                           …2
            - name: elasticsearch-data
              mountPath: /usr/share/elasticsearch/data
            - name: elasticsearch-repo
              mountPath: /usr/share/elasticsearch/repository
  volumeClaimTemplates:                                           …3
```

```
  - metadata:
      name: elasticsearch-data
    spec:
      storageClassName: elasticsearch-data-masternode
      accessModes:
        - ReadWriteOnce
      resources:
        requests:
          storage: 3Gi
  - metadata:
      name: elasticsearch-repo
    spec:
      storageClassName: elasticsearch-repo
      accessModes:
        - ReadWriteMany
      resources:
        requests:
          storage: 3Gi
```

記述例の番号は、説明の番号と対応しています。

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。

＜説明＞

1. 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。
2. 「volumeMounts:」の行から「mountPath: /usr/share/elasticsearch/repository」の行で、Elasticsearch のデータおよびスナップショットを永続化するため、PVC を Pod 内の所定ディレクトリにマウントします。
3. 「volumeClaimTemplates:」の行から最終行で、各 Pod に自動でプロビジョニングされる永続ボリュームを定義します。永続ボリュームは事前に作成する必要があります。詳細については、「3.7.3 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」、「3.7.4　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」を参照してください。

(c)　クラスタ構成（データノード）の場合

クラスタ構成の場合のデータノード用の記述例を示します。

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: elasticsearch-data
  namespace: my-namespace
spec:
  serviceName: elasticsearch-data-headless                                    …1
  replicas: 3
  selector:
    matchLabels:
      app: elasticsearch-data
```

```
  template:
    metadata:
      labels:
        app: elasticsearch-data
    spec:
      serviceAccountName: elasticsearch-master-service-account
      securityContext:
        fsGroup: 1000
        fsGroupChangePolicy: "OnRootMismatch"
      initContainers:
        - name: configure-sysctl
          image: myregistry.example.com/myproject/elasticsearch:V.R.S
          imagePullPolicy: IfNotPresent
          securityContext:
            runAsUser: 0
            privileged: true
          command:
            - sh
            - -c
            - |
              set -e
              sysctl -w vm.max_map_count=262144
      containers:
        - name: elasticsearch
          image: myregistry.example.com/myproject/elasticsearch:V.R.S
          securityContext:
            runAsNonRoot: true
            runAsUser: 1000
            capabilities:
              drop: ["ALL"]
          ports:
            - containerPort: 9200
              name: http
            - containerPort: 9300
              name: transport
          env:
            - name: node.roles
              value: '["data_hot","data_content","ingest"]'
            - name: discovery.seed_hosts
              value: '["elasticsearch-master-0.elasticsearch-master-headless","elasticsearch
-master-1.elasticsearch-master-headless","elasticsearch-master-2.elasticsearch-master-headle
ss"]'
            - name: xpack.security.enabled
              value: "false"
            - name: ES_JAVA_OPTS
              value: "-Xms1g -Xmx1g"
          volumeMounts:                                                    …2
            - name: elasticsearch-data
              mountPath: /usr/share/elasticsearch/data
            - name: elasticsearch-repo
              mountPath: /usr/share/elasticsearch/repository
  volumeClaimTemplates:                                                    …3
    - metadata:
        name: elasticsearch-data
      spec:
        storageClassName: elasticsearch-data-masternode
        accessModes:
          - ReadWriteOnce
```

```
      resources:
        requests:
          storage: 3Gi
  - metadata:
      name: elasticsearch-repo
    spec:
      storageClassName: elasticsearch-repo
      accessModes:
        - ReadWriteMany
      resources:
        requests:
          storage: 3Gi
```

記述例の番号は、説明の番号と対応しています。

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。

＜説明＞

1. 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。
2. 「volumeMounts:」の行から「mountPath: /usr/share/elasticsearch/repository」の行で、Elasticsearch のデータおよびスナップショットを永続化するため、PVC を Pod 内の所定ディレクトリにマウントします。
3. 「volumeClaimTemplates:」の行から最終行で、各 Pod に自動でプロビジョニングされる永続ボリュームを定義します。永続ボリュームは事前に作成する必要があります。詳細については、「3.7.3 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」、「3.7.4　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」を参照してください。

3.7.21　Logstash の Kubernetes マニフェスト作成

Elastic Stack を使用してトラブルシュート情報を収集する場合は、次の手順を実施してください。

重要

Elasticsearch の TLS による暗号化通信を有効にする場合、Logstash で TLS による暗号化通信の設定が必要です。詳細については、Elasticsearch の公式ドキュメントを参照してください。

(1)　Logstash のマニフェスト作成

Elasticsearch の公式ドキュメントを参考に、ConfigMap 以外の Kubernetes マニフェストを作成してください。パラメタおよび設定値の詳細については、Elasticsearch の公式ドキュメントを参照してください。

(2) ConfigMap の作成

メッセージログ収集を設定し、Elasticsearch をメッセージログの永続化サービスとして利用するため、Kubernetes マニフェストの ConfigMap を作成してください。次に記述例を示します。

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: logstash-config
  namespace: my-namespace
data:
  logstash.yml: |-
    http.host: "0.0.0.0"
    queue:
      type: persisted
      max_events: 0
    pipeline.batch.size: 125
  pipelines.yml: |-                                                    …1
    - pipeline.id: main
      path.config: "/usr/share/logstash/pipeline"
  logstash.conf: |-
    input {
      beats {
        port => 5044
      }
    }
    output {
      elasticsearch {
        hosts => "http://elasticsearch-master:9200"
      }
    }
```

記述例の番号は、説明の番号と対応しています。

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。

＜説明＞

1.「pipelines.yml: |-」の行から最終行で、Elasticsearch へメッセージログを送信するための設定を定義します。

3.8　Helm チャートのカスタマイズ

「7.1　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 Namespace 名を「.」でつないだ文字列が、41 バイト以内になるようにしてください。TP1-Bridge の Service 名は、「(15) Kubernetes アプリケーションのデプロイ」の「表 3-28　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。

3.8.1　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 3-20　Helm チャートのカスタマイズの作業項目の一覧

作業項目	備考	参照先
Mediator の Helm チャートのカスタマイズ	必ず実施します。	3.8.2　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）	通常版の場合で、TP1-Bridge を使用するときだけ実施します。	3.8.3　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）	通常版の場合だけ実施します。	3.8.4　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）	トライアル版の場合だけ実施します。	3.8.5　Ext-Cons の Helm チャートのカスタマイズ（トライアル版限定）

3.8.2　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。

「9.3.1　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     …1
image: myregistry.example.com/myproject/mediator:V.R.S※
imagePullSecrets:
- name: regcred
imagePullPolicy: IfNotPresent
labels:
  hmppctoAppGroup: x-system
dependencyCheck:     …2
  enabled: false
healthCheck:
  enabled: true
livenessProbe:
  initialDelaySeconds: 180
  periodSeconds: 30
  failureThreshold: 3
  timeoutSeconds: 10
  healthCheck:
    retryCount: 0
    retryIntervalSeconds: 0
    connectTimeoutSeconds: 10
    readTimeoutSeconds: 10
readinessProbe:
  initialDelaySeconds: 40
  timeoutSeconds: 80
postStart:
  healthCheck:
    retryCount: 30
    retryIntervalSeconds: 10
    connectTimeoutSeconds: 10
    readTimeoutSeconds: 10
preStop:
  waitTransactions:
    retryCount: 10
    retryIntervalSeconds: 10
    connectTimeoutSeconds: 10
    readTimeoutSeconds: 10
terminationGracePeriodSeconds: 150
affinity:
  podAntiAffinity:
    requiredDuringSchedulingIgnoredDuringExecution:
    - labelSelector:
        matchExpressions:
        - key: app.kubernetes.io/name
          operator: In
          values:
          - mediator
      topologyKey: kubernetes.io/hostname
securityContext: {}
resources:
  limits:
```

```
    cpu: 2
    memory: 1Gi
  requests:
    cpu: 1
    memory: 512Mi
participant:
  sender:
    threadPoolSize: 30
    queueSize: 5120000
consensusLog:
  size: 256
  io:
    threadPoolSize: 12
    queueSize: 100
    watchInSeconds: 240
    blockSize: 4096
  storageClassName: mediator-consensus-log
http:
  port: 20554
grpc:
  port: 20551
  server:
    threadPoolSize: 30
    queueSize: 258000
  useTls: false
  skipAuth: false
  tls:
    secretName: ""
    clientAuth:
      enabled: false
  keepalive:
    intervalInSeconds: 0
    timeoutInSeconds: 60
  deduplication: true
mediator:
  judge:
    threadPoolSize: 30
    queueSize: 256000
    proposalLogLimitNum: 512
    transactionTerminatedLimitNum: 512
    transactionTerminatedConfirmNotifyIntervalInMillis: 60000
    prepareDelayTimeBaseInMillis: 100
    prepareRetryTimeBaseInMillis: 100
    prepareRetryMaxTimeInSeconds: 30
tracing:
  instanceId: Mediator
  url: http://jaeger:4318/v1/traces
  useTrace: true
  debug: false
  queueSize: 20000
  delayTimeInMillis: 1000
  batchSize: 2000
  grpc:
    debug: false
metrics:
  collection:
    intervalInMillis: 1000
timezone: Asia/Tokyo
```

```
java:
  options: -XX:+UseG1GC
logging:
  level:
    hmppctoSTDOUT: INFO
ucars:
  enabled: true
  snapshots:
    claimName: snapshot-volume
use-scaling: true
runningDates:
  enabled: true
  retentionMonths: 3
```

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)。

記述例の番号は、説明の番号と対応しています。

＜説明＞

1. オートスケールを使用しない場合だけ、replicas を設定します。
2. Elastic Stack、Jaeger および Prometheus を使用してトラブルシュート情報を収集する場合、「dependencyCheck:」の行から「enabled: false」の行で、dependencyCheck.enabled パラメタに true を設定し、dependencyCheck.targets パラメタに対象のトラブルシュート情報を収集する際に必要なアプリケーションを指定してください。

   この項目は、クラウド環境（Amazon Web Services 環境で Jaeger を使用する場合、および Microsoft Azure 環境で Jaeger と Elasticsearch を使用する場合）にも適用されます。

   Google Cloud Platform 環境でトラブルシュート情報を収集する場合は、Elastic Stack、Jaeger および Prometheus の Pod のデプロイを待機する必要がありません。その場合は dependencyCheck.enabled パラメタに false を設定してください。

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。

3.8.3 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、Hitachi Microservices Platform - Bridge Option for OpenTP1 が提供する 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。

「9.3.2　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
labels:
  appKubernetesIoName: tp1-bridge
  hmppctoAppGroup: x-system
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-eads-server
filebeat:
  enabled: true
  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: 150
affinity: {}
securityContext: {}
resources:
  limits:
    cpu: 2
    memory: 1Gi
  requests:
    cpu: 1
    memory: 512Mi
server:
  tomcat:
    threads:
      max: 200
    maxSwallowSize: "2097152"
converter:
  host: converter
  port: 8008
  connectTimeInSeconds: 10
  watchTimeInSeconds: 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
  pool:
    allocationWaitTimeInMillis: 60000
    maxPoolSize: 10
  pod:
    serviceName: tp1-bridge1.my-namespace
  transactionTermination:
    timeoutInMillis: 60000
    consensus:
      replayIntervalInMillis: 1000
  participantProcessing:
    threadPoolSize: 12
    queueSize: 4096000
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: 4096000
http:
  port: 20554
grpc:
  port: 20552
  useTls: false
  skipAuth: false
  tls:
    secretName: ""
    clientAuth:
      enabled: false
    session:
      cacheSize: 0
      timeoutInSeconds: 0
  keepalive:
```

```
    intervalInSeconds: 0
    timeoutInSeconds: 60
tracing:
  instanceId: TP1-Bridge_001
  url: http://jaeger:4318/v1/traces
  useTrace: true
  debug: false
  queueSize: 20000
  delayTimeInMillis: 1000
  batchSize: 2000
  grpc:
    debug: false
metrics:
  collection:
    intervalInMillis: 1000
timezone: Asia/Tokyo
java:
  options: -XX:+UseG1GC
logging:
  level:
    hmppctoSTDOUT: 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
use-scaling: false
```

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 Namespace 名を「.」でつないだ文字列が、41 バイト以内になるようにしてください。TP1-Bridge の Service 名は、「(15)　Kubernetes アプリケーションのデプロイ」の「表 3-28　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。

3.8.4　EADS の Helm チャートのカスタマイズ（通常版限定）

EADS の Helm チャートの values.yaml を作成します。

「9.3.3　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。EADS の values.yaml の詳細については、取扱説明書「EADS コンテナオーケストレーションツール対応機能」を参照してください。

3.8.5　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。

「9.3.4　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: false
affinity: {}
http:
  port: 20556
timezone: Asia/Tokyo
java:
  options: -XX:+UseG1GC
```

```
logging:
  level:
    extCons: 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。

3.9 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。

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で oc コマンドでの操作が必要な作業項目については oc コマンドの例を記述しています。kubectl や oc など、コンテナオーケストレーションのコマンドの詳細については、それぞれのマニュアルを参照してください。

3.9.1 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 3-21 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）とする場合だけ作成します。	(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	Elastic Stack を使用してトラブルシュート情報を収集する場合だけ作成します。	(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

作業対象	作業項目	備考	参照先
	EADS 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）	通常版の場合だけ作成します。	(3)　EADS 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 を使用するために必要な追加手順	Elastic Stack を使用してトラブルシュート情報を収集する場合だけ使用します。	(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の Kubernetes アプリケーションおよび RAS	(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

作業対象	作業項目	備考	参照先
			リュームのデプロイ（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	Elastic Stack を使用してトラブルシュート情報を収集する場合で、Elasticsearch の TLS 通信を行うときだけ作成します。	(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		(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	Elastic Stack を使用してトラブルシュート情報を収集する場合だけ実施します。	(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		(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の Kubernetes アプリケーションおよび RAS	(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	Elastic Stack を使用してトラブルシュート情報を収集する場合だけ実施します。	(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	Prometheus を使用してトラブルシュート情報を収集する場合だけ実施します。	(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		(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 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）	通常版の場合で、uCosminexus Application Runtime for Spring Boot を導入するときだけ実施します。次のどちらか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の Kubernetes アプリケーションおよび RAS	(14) uCosminexus Application Runtime 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
	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	Elastic Stack を使用してトラブルシュート情報を収集する場合だけ作成します。	(16) Elasticsearch のスナッ

作業対象	作業項目	備考	参照先
			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		(17) Elasticsearch の index 設定変更および削除
	Azure Monitor の設定	Microsoft Azure 環境を使用してトラブルシュート情報を収集する場合だけ実施します。	(18) Azure Monitor の設定
	Cloud Logging、Cloud Trace および Cloud Monitoring の設定	Google Cloud Platform 環境を使用してトラブルシュート情報を収集する場合だけ実施します。	(19) Cloud Logging、Cloud Trace および Cloud Monitoring の設定
	Amazon CloudWatch の設定	Amazon Web Services 環境を使用してトラブルシュート情報を収集する場合だけ実施します。	(20) Amazon CloudWatch の設定
	New Relic の設定	New Relic を使用してトラブルシュート情報を収集する場合だけ実施します。	(21) New Relic の設定
	Red Hat OpenShift Container Platform モニタリングの Prometheus の設定	Red Hat OpenShift Container Platform 環境を使用してトラブルシュート情報を収集する場合だけ実施します。	(22) Red Hat OpenShift Container Platform モニタリングの Prometheus の設定
	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。	(23) EADS のセットアッ

作業対象	作業項目	備考	参照先
			プ（通常版限定）
	EADS の生存監視ツールの起動		(24) EADS の生存監視ツールの起動
	HPA の設定のカスタマイズ	スケーリング対応機能を使用する場合で、Metrics Server でメトリクスの監視をしてオートスケールを適用するときだけ実施します。	4.2.5 HPA の設定のカスタマイズ
		スケーリング対応機能を使用する場合で、Prometheus でメトリクスの監視をしてオートスケールを適用するときだけ実施します。	4.3.7 HPA の設定のカスタマイズ
	HPA のデプロイ	スケーリング対応機能を使用する場合で、Metrics Server でメトリクスの監視をしてオートスケールを適用するときだけ実施します。	4.2.6 HPA のデプロイ
		スケーリング対応機能を使用する場合で、Prometheus でメトリクスの監視をしてオートスケールを適用するときだけ実施します。	4.3.8 HPA のデプロイ
	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。	(25) EADS のアンセットアップ（通常版限定）
	EADS の生存監視ツールの停止		(26) EADS の生存監視ツールの停止
	Kubernetes アプリケーションのアンデプロイ	必ず実施します。	(27) Kubernetes アプリケーションのアンデプロイ

3.9.2　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。

「3.7.2　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。Red Hat OpenShift Container Platform の場合は、次の要領に記載されたディレクトリを操作する権限を持つユーザで実施してください。

＜ 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、「3.7.2　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

Elastic Stack を使用してトラブルシュート情報を収集する場合、次の手順を実施してください。

「3.7.3　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、「3.7.3　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（ローカル永続ボリューム（local）とする場合）

EADS の永続ボリューム（ログファイル用、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）をローカル永続ボリューム（local）とする場合、次の作業を実施してください。

「3.7.5　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。Red Hat OpenShift Container Platform の場合は、次の要領に記載されたディレクトリを操作する権限を持つユーザで実施してください。

＜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、「3.7.5　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

Elastic Stack を使用してトラブルシュート情報を収集する場合、次の手順を実施してください。

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
```

3.9.3　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 3-22　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	「3.7.2　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	「3.8.2　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 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	「3.8.3　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	「3.7.5　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-health-cronjob-chart.tgz)	EADS 提供物 通常版の場合だけ配置します。
EADS の Helm チャートの values.yaml	「3.8.4　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 通常版の場合だけ配置します。
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 トライアル版の場合だけ配置します。

資材	備考
Ext-Cons の Helm チャートの values.yaml	「3.8.5　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 トライアル版の場合だけ配置します。
Filebeat の Kubernetes マニフェスト	「3.7.11　Filebeat の Kubernetes マニフェスト作成」でカスタマイズしたもの Elastic Stack を使用してトラブルシュート情報を収集する場合だけ配置します。
Jaeger の Kubernetes マニフェスト	「3.7.12　Jaeger の Kubernetes マニフェスト作成」でカスタマイズしたもの Jaeger を使用してトラブルシュート情報を収集する場合だけ配置します。この項目は、クラウド環境（例：Amazon Web Services 環境）で Jaeger を使用する場合にも適用されます。
Prometheus の Kubernetes マニフェスト	「3.7.13　Prometheus の Kubernetes マニフェスト作成」で作成したもの Prometheus を使用してトラブルシュート情報を収集する場合だけ配置します。
Metricbeat の Kubernetes マニフェスト	「3.7.19　Metricbeat の Kubernetes マニフェスト作成」でカスタマイズしたもの Elastic Stack を使用してトラブルシュート情報を収集する場合だけ配置します。
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	「3.7.3　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 Elastic Stack を使用してトラブルシュート情報を収集する場合だけ配置します。
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	「3.7.4　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 Elastic Stack を使用してトラブルシュート情報を収集する場合だけ配置します。
Elasticsearch の Kubernetes マニフェスト	「3.7.20　Elasticsearch の Kubernetes マニフェスト作成」でカスタマイズしたもの Elastic Stack を使用してトラブルシュート情報を収集する場合だけ配置します。この項目は、クラウド環境（例：Microsoft Azure 環境）で Elasticsearch を使用する場合にも適用されます。
Logstash の Kubernetes マニフェスト	「3.7.21　Logstash の Kubernetes マニフェスト作成」でカスタマイズしたもの

資材	備考
	Elastic Stack を使用してトラブルシュート情報を収集する場合だけ配置します。
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	「3.7.6　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の Kubernetes アプリケーションおよび RAS
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	「3.7.7　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 Elastic Stack を使用してトラブルシュート情報を収集する場合だけ配置します。
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	「3.7.8　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 Prometheus を使用してトラブルシュート情報を収集する場合だけ配置します。
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	「3.7.9　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 Prometheus を使用してトラブルシュート情報を収集する場合だけ配置します。
uCosminexus Application Runtime 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	「3.7.10　uCosminexus Application Runtime 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 通常版を使用する場合で、uCosminexus Application Runtime for Spring Boot を導入するときだけ、次のどちらか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の Kubernetes アプリケーションおよび RAS
HPA の Kubernetes マニフェスト	• 「4.2.4　HPA の Kubernetes マニフェストの作成」で作成したもの • 「4.3.6　HPA の Kubernetes マニフェストの作成」で作成したもの スケーリング対応機能を使用する場合で、オートスケールを適用するときだけ配置します。

(2) Namespace の作成

(a) Red Hat OpenShift Container Platform 環境以外の場合

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の Kubernetes アプリケーションおよび RAS

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の実行例や Kubernetes マニフェストの記述例は、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。

(b) Red Hat OpenShift Container Platform 環境の場合

Red Hat OpenShift Container Platform 環境の場合は、Project を作成してください。Project の作成の延長で、Project 名と同名の Namespace が作成されます。Namespace の作成ルール、コマンドの実行例、および Kubernetes マニフェストの記述例の前提については、「(a)　Red Hat OpenShift Container Platform 環境以外の場合」を参照してください。

Red Hat OpenShift Container Platform 環境で、HMP-PCTO の Kubernetes アプリケーションをデプロイする Project を「my-project」とした場合のコマンド例を示します。

```
oc new-project my-project
```

＜コマンドの実行結果の例＞

```
Now using project ”my-project” on server ”https://api.example.openshift.com:6443”.

You can add applications to this project with the ’new-app’ command. For example, try:

    oc new-app rails-postgresql-example

to build a new example application in Ruby. Or use kubectl to deploy a simple Kubernetes app
lication:

    kubectl create deployment hello-node --image=registry.k8s.io/e2e-test-images/agnhost:2.4
3 -- /agnhost serve-hostname
```

このように、Project の作成に成功し、現在使用中の Project が「my-project」に切り替わっ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。

(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。

表 3-23　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	対応する HMP-PCTO の機能	ファイル	備考
TP1-Bridge	Participant（TP1-Bridge）	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	
Mediator	Mediator	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	
		自サーバ証明書を作成した秘密鍵ファイル	
		接続元クライアント（Orchestrator および 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
    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/
        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
    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
        keystore.jks    # Mediatorの接続元クライアント(Orchestratorおよび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 3-24　TP1-Bridge 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
client.crt	自クライアント証明書（PEM 形式）ファイル
client.key	自クライアント証明書を作成した秘密鍵ファイル

表 3-25　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
keystore	接続元クライアント（orchestrator および 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 ¥
  -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=client.crt=./participant/client.crt ¥
  --from-file=client.key=./participant/client.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 ¥
  --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

この作業は、Elastic Stack を使用してトラブルシュート情報を収集する場合で Elasticsearch の TLS 通信を行うときだけ実施します。

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。

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

この作業は、Elastic Stack を使用してトラブルシュート情報を収集する場合で Elasticsearch の TLS 通信を行うときだけ実施します。

Kubernetes コントロールプレーンの任意のパスへ、次のとおり Elasticsearch の TLS 通信に必要なファイルを準備してください。

表 3-26　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	なし
	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	なし

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 3-27　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

Elastic Stack を使用してトラブルシュート情報を収集する場合は、次の手順を実施してください。

「(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

Elastic Stack を使用してトラブルシュート情報を収集する場合は、次の手順を実施してください。

「(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の Kubernetes アプリケーションおよび RAS 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

Elastic Stack を使用してトラブルシュート情報を収集する場合は、次の手順を実施してください。

「(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

Prometheus を使用してトラブルシュート情報を収集する場合は、次の手順を実施してください。

「(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

Prometheus を使用してトラブルシュート情報を収集する場合は、次の手順を実施してください。

「(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 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 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の Kubernetes アプリケーションおよび RAS 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 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 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 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※1、※2
2. Logstash※1、Jaeger※3、Prometheus※4、Metricbeat※1、Filebeat※1

3. Mediator、EADS サーバ（通常版だけ）※5、Ext-Cons（トライアル版だけ）、Participant が使用する DB/外部サービス
4. Relay-Service、Entity-Service(SQL)、Entity-Service(TCC)、TP1-Bridge※6
5. Orchestrator-Service、Orchestrator-Service(SQL)、Alternate-Service、Alternate-Service(SQL)

注※1

これらの Kubernetes アプリケーションは、Elastic Stack を使用してトラブルシュート情報を収集する場合だけデプロイしてください。

注※2

クラウド環境（例：Microsoft Azure 環境）で Elasticsearch を使用する場合もデプロイしてください。

注※3

Jaeger を使用してトラブルシュート情報を収集する場合だけデプロイしてください。クラウド環境（例：Amazon Web Services 環境）で Jaeger を使用する場合もデプロイしてください。

注※4

Prometheus を使用してトラブルシュート情報を収集する場合だけデプロイしてください。

注※5

EADS サーバをデプロイしたあとに「(23)　EADS のセットアップ（通常版限定)」を実施してから、Participant の機能を使用する Kubernetes アプリケーションをデプロイしてください。

注※6

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。

メモ

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。

そのあとで、EADS サーバと Participant の機能を使用する Kubernetes アプリケーションをアンデプロイして、正しい順序でデプロイし直してください。

注※　「(27)　Kubernetes アプリケーションのアンデプロイ」を実施しないでクラスタを再起動する場合も含みます。

デプロイした Kubernetes アプリケーションの起動に失敗したとき、Kubernetes の ERROR メッセージが出力されることがあります。詳細については、「(27)　Kubernetes アプリケーションのアンデプロイ」

の「Kubernetes アプリケーションの停止時に出力される Kubernetes の ERROR メッセージについて」を参照してください。

HMP-PCTO のコントロールプレーンの 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

<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。

<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。

表 3-28　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
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。

Mediator のデプロイ後に、Mediator の ConsensusLog の永続ボリュームマウント用ディレクトリ下に稼働日ファイル用ディレクトリ（running-dates）が作成されていることを確認してください。

RAS 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 ./deployment.yaml
```

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。この手順は Elastic Stack を使用してトラブルシュート情報を収集する場合に実施してください。

- 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 の確認

  Jaeger の起動の際に Jaeger が使用するストレージ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。

  この手順は Jaeger を使用してトラブルシュート情報を収集する場合に実施してください。

  この項目は、オンプレミス環境およびクラウド環境（例：Amazon Web Services 環境）で Jaeger を使用する場合にも適用されます。

```
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-68bfc6fb87-c6tlm	1/1	Running	0	95s	192.168.125.136	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

Elastic Stack を使用してトラブルシュート情報を収集する場合は、次の手順を実施してください。

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-Ty
pe: 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

Elastic Stack を使用してトラブルシュート情報を収集する場合は、次の手順を実施してください。

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 3-29　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。

■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     XXXXXXXXXXXXXXXXXXXX   X   X        252           18     89.1kb         89.1kb
yellow open   jaeger-jaeger-span-XXXX-XX-XX       XXXXXXXXXXXXXXXXXXXX   X   X  180828498            0       7gb            7gb
yellow open   logstash-XXXX.XX.XX-XXXXXX          XXXXXXXXXXXXXXXXXXXX   X   X          0            0      226b           226b
yellow open   metricbeat-X.XX.X-XXXX.XX.XX-XXXXXX XXXXXXXXXXXXXXXXXXXX   X   X   31880255            0       5gb            5gb
```

■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}
```

■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「■情報のロールオーバー設定の変更方法」を参照してください。

index を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) Azure Monitor の設定

Microsoft Azure 環境を使用してトラブルシュート情報を収集する場合は、Azure Monitor の設定をしてください。

HMP-PCTO が Azure Monitor に出力する情報は、Azure Monitor のワークスペースに格納されます。保存量、保持期間、ライフサイクルなどの詳細は Azure Monitor の公式ドキュメントを参照してください。

(19) Cloud Logging、Cloud Trace および Cloud Monitoring の設定

Google Cloud Platform 環境を使用してトラブルシュート情報を収集する場合は、Cloud Logging、Cloud Trace および Cloud Monitoring の設定をしてください。

Cloud Logging、Cloud Trace および Cloud Monitoring の保存量や保存期間の詳細については、Google Cloud Platform の公式ドキュメントを参照してください。

(20) Amazon CloudWatch の設定

Amazon Web Services 環境を使用してトラブルシュート情報を収集する場合は、Amazon CloudWatch の設定をしてください。

Amazon CloudWatch および Amazon OpenSearch Service の保存量や保存期間の詳細については、Amazon Web Services の公式ドキュメントを参照してください。

(21) New Relic の設定

New Relic 使用時のトラブルシュート情報の保存量や保持期間の詳細については、New Relic の公式ドキュメントを参照してください。

(22) Red Hat OpenShift Container Platform モニタリングの Prometheus の設定

Red Hat OpenShift Container Platform モニタリングのトラブルシュート情報は永続化されていません。また、メトリクス情報の保存量や保持期間に上限が設定されています。詳細については、Red Hat OpenShift Container Platform の公式ドキュメントを参照してください。

- 分散トレース情報を永続化する場合は、TempoStack の設定を変更する必要があります。
- メッセージログ情報を永続化する場合は、Loki Operator の設定を変更する必要があります。
- メトリクス情報を永続化する場合は、Prometheus の設定を変更する必要があります。

(23) 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、オレンジ色の背景の「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。

(24)　EADS の生存監視ツールの起動

「(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-health-cronjob-chart.tgz）と EADS の Helm チャートの values.yaml を使用して、EADS の生存監視ツールを起動してください。

生存監視ツールは、縮退状態の EADS サーバを検知して自動で復旧を実行するツールです。生存監視ツールの詳細については、取扱説明書「EADS コンテナオーケストレーションツール対応機能」を参照してください。

コマンド例を次に示します。

```
helm -n my-namespace install eads-health ./eads-health-cronjob-chart.tgz -f ./eads-values.yaml
```

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＞

```
NAME: eads-health
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。

(25)　EADS のアンセットアップ（通常版限定）

「(27)　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。

(26)　EADS の生存監視ツールの停止

「(27)　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、EADS の生存監視ツールを停止してください。

コマンド例を次に示します。

```
helm -n my-namespace uninstall eads-health
```

＜コマンドの実行結果の例＞

```
release "eads-health" 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。

(27)　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-Service、Orchestrator-Service(SQL)、Alternate-Service、Alternate-Service(SQL)

2. Entity-Service(TCC)、Entity-Service(SQL)、Relay-Service、TP1-Bridge※1
3. Mediator、EADS サーバ（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※2、Jaeger※3、Prometheus※4、Metricbeat※2、Filebeat※2
5. Elasticsearch※2、※5

注※1

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。

注※2

これらの Kubernetes アプリケーションは、Elastic Stack を使用してトラブルシュート情報を収集する場合だけアンデプロイしてください。

注※3

Jaeger を使用してトラブルシュート情報を収集する場合だけアンデプロイしてください。クラウド環境（例：Amazon Web Services 環境）で Jaeger を使用する場合もアンデプロイしてください。

注※4

Prometheus を使用してトラブルシュート情報を収集する場合だけアンデプロイしてください。

注※5

クラウド環境（例：Microsoft Azure 環境）で Elasticsearch を使用する場合もアンデプロイしてください。

クラスタを停止する前には、上記の順序でアンデプロイしてください。アンデプロイする前にクラスタを停止した場合は、クラスタを再起動したあとに、アンデプロイしてから「(15)　Kubernetes アプリケーションのデプロイ」で示す順序で再デプロイしてください。

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。

注※　アンデプロイする前にクラスタを停止する場合も含みます。

Kubernetes アプリケーションの停止時に出力される Kubernetes の ERROR メッセージについて

Kubernetes アプリケーションの停止時（起動失敗時を含む）に、Kubernetes アプリケーションのメッセージログに Kubernetes の client で java.lang.InterruptedException が発生したことを示す ERROR メッセージが出力されることがあります。このメッセージは出力されても問題ないため、無視してください。

ERROR メッセージの出力例の一部を次に示します。

■メッセージ出力例 1

```
ERROR sql-participant-dc47fd869-khlgp pool-13-thread-1 io.kubernetes.client.informer.cache.ProcessorListener 87 processor interrupted: {}
```

■メッセージ出力例 2

```
ERROR sql-participant-dc47fd869-khlgp informer-controller-V1Endpoints io.kubernetes.client.informer.cache.Controller 87 DefaultController#processLoop get interrupted null
```

HMP-PCTO のコントロールプレーンの 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。

RAS 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 ./deployment.yaml
```

Kubernetes アプリケーションをユーザ任意の方法でアンデプロイする場合の詳細については、「(8)　ユーザ責務の Kubernetes アプリケーションのアンデプロイ」を参照してください。

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.
```

3.10　RAS のマネージドサービスの運用

クラウド環境が提供する RAS のマネージドサービスの運用について説明します。

3.10.1　New Relic を使用する場合の作業

HMP-PCTO のトラブルシュート情報を収集するために New Relic を使用する場合に必要な作業の流れを次の表に示します。

表 3-30　New Relic を使用する場合に必要な作業の流れ

項番	項目	説明	参照先
1	New Relic のデプロイ	New Relic の Kubernetes integration が提供しているコンポーネントの Pod を Kubernetes クラスタにデプロイします。 Kubernetes integration のコンポーネントは目的に応じて複数提供しています。目的に応じたコンポーネントを有効にしてください。 • HMP-PCTO のメッセージログを収集する場合 newrelic-logging コンポーネントをデプロイおよび有効化してください。 • HMP-PCTO のメトリクスを収集する場合 nri-prometheus コンポーネントをデプロイおよび有効化してください。 有効化の方法については、New Relic の公式ドキュメントを参照してください。	New Relic の公式ドキュメント
2	OpenTelemetry Collector の Kubernetes マニフェスト作成	HMP-PCTO が出力する分散トレースを New Relic に送信するために、OpenTelemetry Collector の Kubernetes マニフェストを作成します。 適切な New Relic OTLP エンドポイントおよび New Relic のライセンスキーを設定してください。	OpenTelemetry Collector および New Relic の公式ドキュメント
3	OpenTelemetry Collector のデプロイ	OpenTelemetry Collector をユーザ任意の方法で Kubernetes クラスタにデプロイします。	OpenTelemetry Collector の公式ドキュメント

重要

クラウド環境で New Relic をデプロイする場合、必要な作業および操作が異なる場合があります。詳細については、New Relic の公式ドキュメントを参照してください。

3.11　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。

3.11.1　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)　Spring Boot の組み込み HTTP サーバのアクセスログの取得方法

トラブルシュートをするために、アクセスログが必要です。

> **重要**
>
> Microsoft Azure 環境、Google Cloud Platform 環境、Amazon Web Services 環境、New Relic、または Red Hat OpenShift Container Platform 環境を使用している場合、各コンテナのボリュームに出力される情報について、サイドカーコンテナの組み込みによる情報の収集ができません。そのため、uCosminexus Application Runtime for Spring Boot のスナップショットログ機能を使用して、HMP-PCTO コンテナのボリュームに出力される情報を収集してください。

- Tomcat の場合
  - アクセスログ取得を有効にする方法

    各 Kubernetes アプリケーションの 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。また、アクセスログ取得時にヘッダ情報を取得するように定義してください。

    設定するパラメタの詳細については、「(1)　server.tomcat のパラメタ」を参照してください。また、TP1-Bridge には、ヘッダ情報を含めたアクセスログを取得する設定が導入されています。
  - Filebeat サイドカーコンテナの組み込みによる Elasticsearch への格納

    Elastic Stack を使用している場合は、各 Kubernetes アプリケーションのマニフェストの作成時に、Filebeat コンテナの filebeat.yml に次の設定をしてください。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。

    また、TP1-Bridge には、Filebeat サイドカーコンテナを組み込む設定が導入されています。

```
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"]
```

- Undertow の場合
  - アクセスログ取得を有効にする方法

    各 Kubernetes アプリケーションの Spring Boot のコンフィグソースに server.undertow.accesslog.enable=true を設定するように定義してください。また、アクセスログ取得時にヘッダ情報を取得するように定義してください。

    設定するパラメタの詳細については、「(2)　server.undertow のパラメタ」を参照してください
  - Filebeat サイドカーコンテナの組み込みによる Elasticsearch への格納

    Elastic Stack を使用している場合は、各 Kubernetes アプリケーションのマニフェストの作成時に、Filebeat コンテナの filebeat.yml に次の設定をしてください。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: "undertow"
  paths:
    - '<Undertowのアクセスログのパス>'

output.logstash:
  hosts: ["logstash.my-namespace.svc.cluster.local:5044"]
```

(2)　PostgreSQL のトラブルシュート情報の取得方法（SQL-Participant 限定）

トラブルシュートを実施するため、ログ情報が必要です。ログ情報の出力方法、取得方法については、PostgreSQL のドキュメントを参照してください。

(3)　HiRDB のトラブルシュート情報の取得方法（SQL-Participant 限定）

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（SQL-Participant 限定）

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) Oracle Database のトラブルシュート情報の取得方法（SQL-Participant 限定）

トラブルシュートをするためには、次の情報が必要です。

- SQL トレース
  トレース、およびファイルを取得してください。
- ログ情報
  アラート、および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、Oracle Database のマニュアルを参照してください。

(6) 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。

(7) uCosminexus Service Platform のトレース・ログ情報の取得方法（TP1-Bridge 限定）

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
- Java VM スタックトレース情報
- Java VM の GC ログ
- Java VM ログ（Java VM 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。

(8) OpenTP1 のトレース・ログ情報の取得方法（TP1-Bridge 限定）

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。

(9) TP1/Client/J のトラブルシュート情報の取得方法（TP1-Bridge 限定）

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 3-31 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、ログ収集アプリケーションまたはサービスで収集します。 • Elastic Stack を使用している場合、Filebeat サイドカーコンテナで収集します。 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。 • Microsoft Azure 環境の場合、Azure Monitor Log Analytics で収集します。Azure Monitor Log Analytics からエクスポートすることで取得します。 • Google Cloud Platform 環境の場合、Cloud Logging のログエクスプローラで収集します。ログエクスプローラを開き、「操作」メニューをクリックしたあと、「ダウンロード」をクリックしてください。 • Amazon Web Services 環境の場合、Amazon CloudWatch で収集します。Amazon CloudWatch コンソールからロググループ、およびログストリームを選択したあと、取得してください。 • New Relic を使用している場合、New Relic の Logs UI または NRQL で収集します。New Relic の Logs UI を使用して、New Relic で一度に出力できる最大行数までを取得できます。詳細については、New Relic の公式ドキュメントを参照してください。 • Red Hat OpenShift Container Platform 環境の場合、Red Hat OpenShift Logging Operator で収集します。Red Hat OpenShift Container Platform の Web UI 上、メッセージログのモニタリングメニュー内で「Export as CSV」をクリックしてください。
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	
4	UAP トレース	

項番	トラブルシュート情報	取得方法
5	データトレース	出力を有効化したトラブルシュート情報は、uCosminexus Application Runtime for Spring Boot のスナップショットログ機能で取得します。

3.11.2　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、そのログをログ収集アプリケーションまたはサービスで収集するために 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
```

この手順でログ収集アプリケーションまたはサービスで収集される EADS サーバのログは監視目的のために使用します。そのため、サポート部署に調査を依頼する際には EADS サーバのログファイルを収集して送付してください。

(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
  Elastic Stack 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
  Elastic Stack 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】

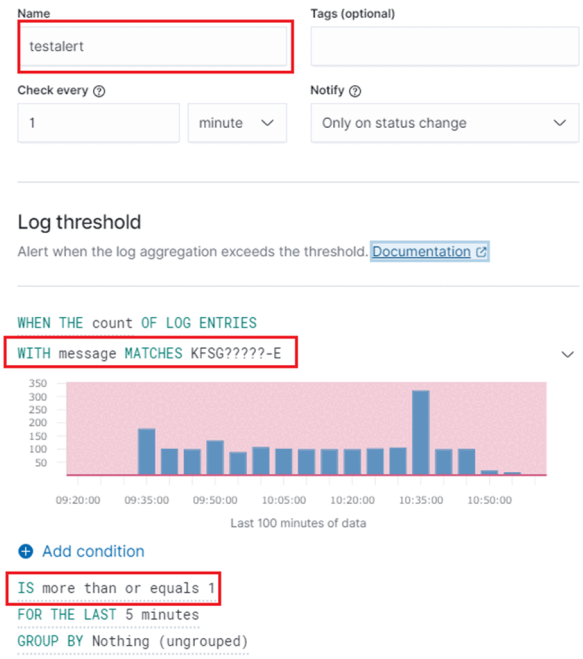
- Kibana と Elasticsearch で通信が暗号化されていること
- Kibana の設定で xpack.encryptedSavedObjects.encryptionKey が設定されていること

【Alert 機能の設定方法】

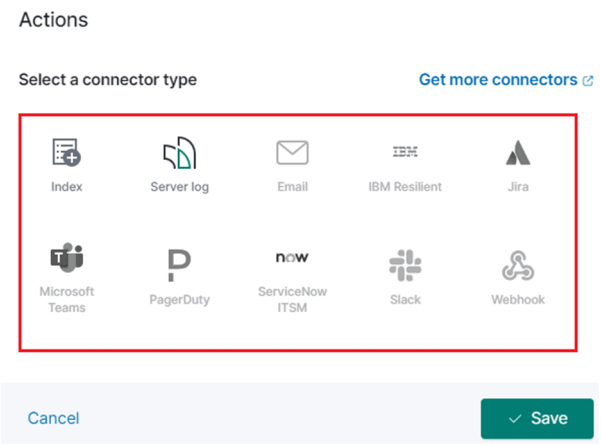
1. Kibana 画面の左上のハンバーガーボタン（三本線）を押し、メニューから Observability を選択する
2. 画面の左カラム内の Logs 内の Stream を選択する
3. 画面の右上の Settings を選択する
4. Log sources 欄内の Log indices に、index 名として「logstash*」をコンマ区切りで追加する
5. Apply を押す
6. 画面の左カラム内の Alerts を選択する
7. 画面の右上の Manage Rules を選択する
8. 画面の右上の Create Alert を選択する

   【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

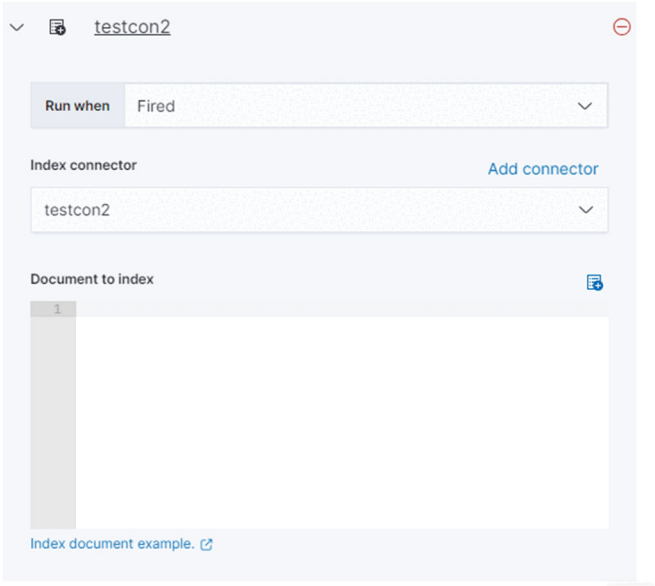
   例として、KFSG の-E メッセージが 1 つ以上出力された場合に、アラートを上げる場合の設定内容を次の図に示します。



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。

3.12　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。オートスケールを使用する場合は、「4.4　オートスケールのパラメタの設計とチューニング」を参照してください。

ヒント

「label_app_kubernetes_io_name」ラベル

説明中の Prometheus のクエリで使用する「label_app_kubernetes_io_name」ラベルは、Kubernetes マニフェストの.spec.template.metadata.labels に「app.kubernetes.io/name」として設定された値です。

Google Cloud Platform のマネージド Prometheus でメトリクスを収集する場合は、「job」ラベルに読み替えてください。

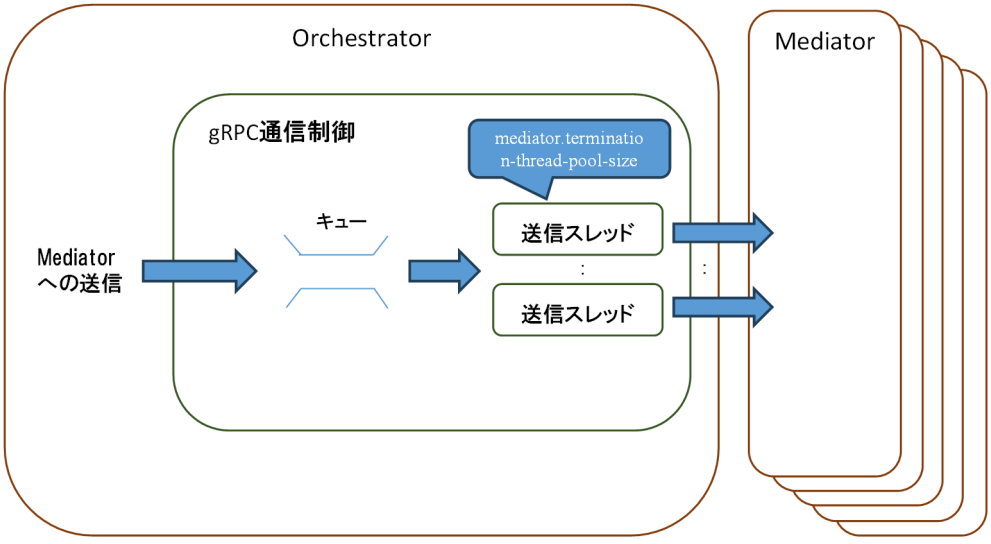
Red Hat OpenShift Container Platform モニタリングの Prometheus でメトリクスを収集する場合は、「service」ラベルに読み替えてください。

3.12.1　スレッドプール

(1)　Orchestrator

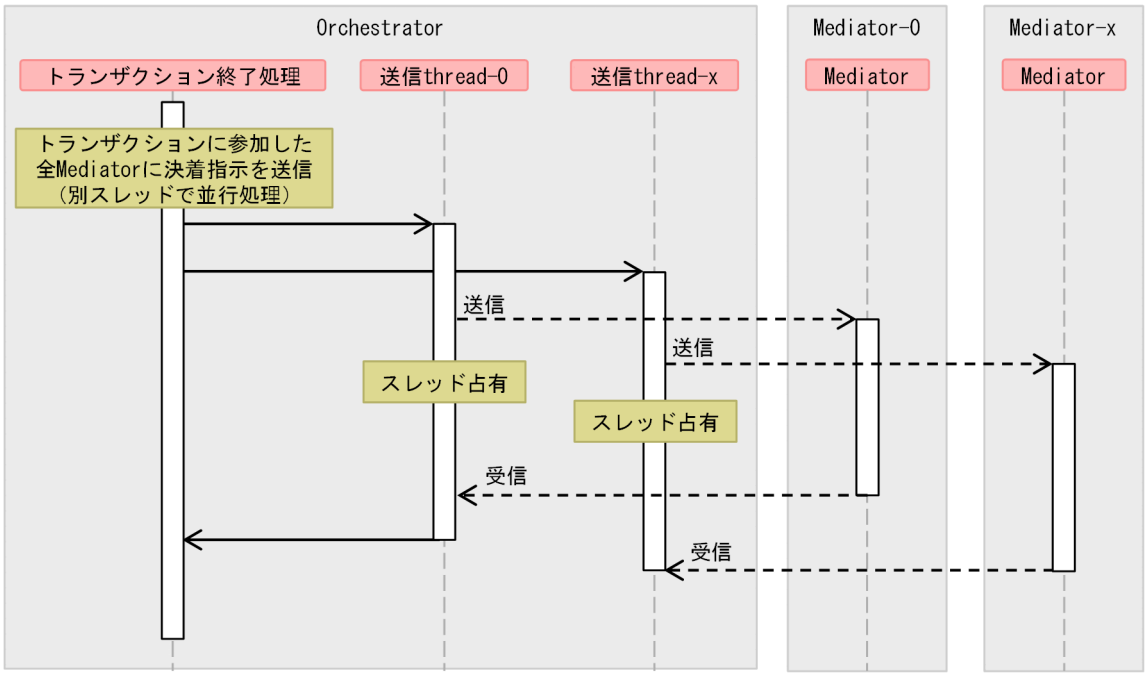
Mediator への送信時（トランザクション決着要求・仕掛中トランザクションのキャンセル要求）の同時実行スレッド数を制御します。同時実行スレッド数は Orchestrator のパラメタ「mediator.termination-thread-pool-size」で指定します。同時実行スレッド数を超えた要求がある場合は、キューに保持します。

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ー一覧

Orchestrator 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 3-32　Orchestrator 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	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、「9.2.3　Orchestrator」を参照してください。

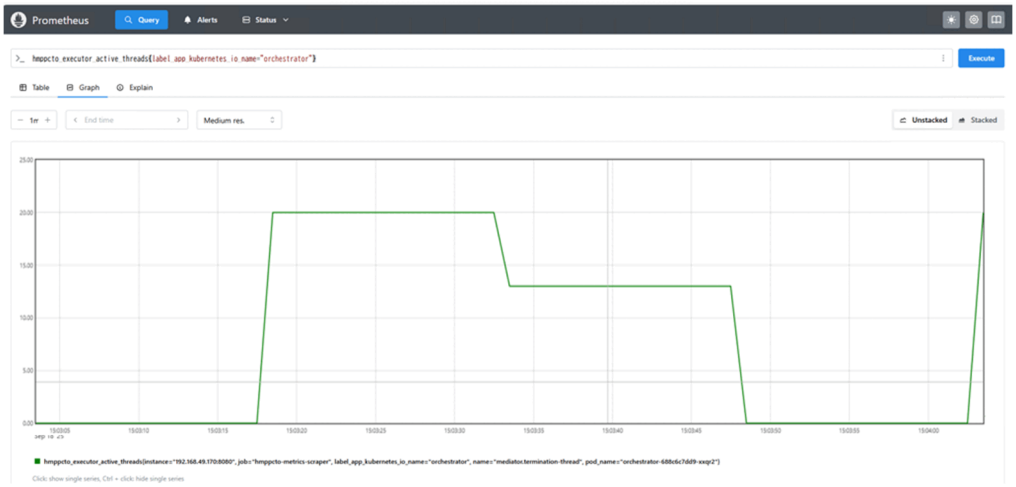
(c)　チューニング方法

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、「11.5.3　メトリクス」を参照してください。

Kubernetes マニフェストの.spec.template.metadata.labels の app.kubernetes.io/name に「orchestrator」が設定されている Orchestrator の機能を使用する Kubernetes アプリケーションの場合、Prometheus の Web UI で「hmppcto_executor_active_threads{label_app_kubernetes_io_name="orchestrator"}」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="Orchestrator の IP アドレス:Orchestrator の http ポート番号"」を追加してください。

入力例：

hmppcto_executor_active_threads{label_app_kubernetes_io_name="orchestrator", instance="192.168.125.166:20554"}

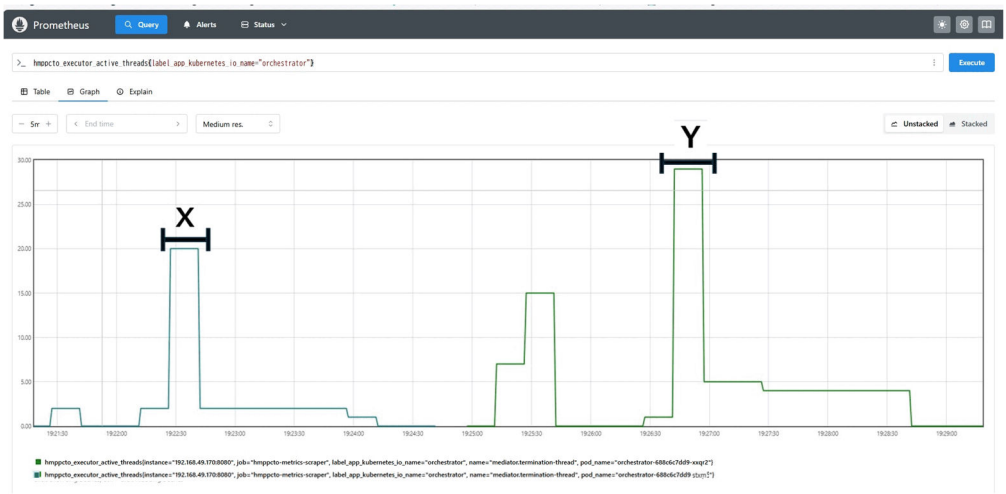
メトリクス名は、クエリを実行した結果によって「name="メトリクス名"」と表示されるため、該当する表示個所からメトリクス名を確認してください。

表 3-33　対象パラメタとメトリクス名の関連づけ

項番	チューニング対象のパラメタ	メトリクス名
1	mediator.termination-thread-pool-size	mediator.termination-thread

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」

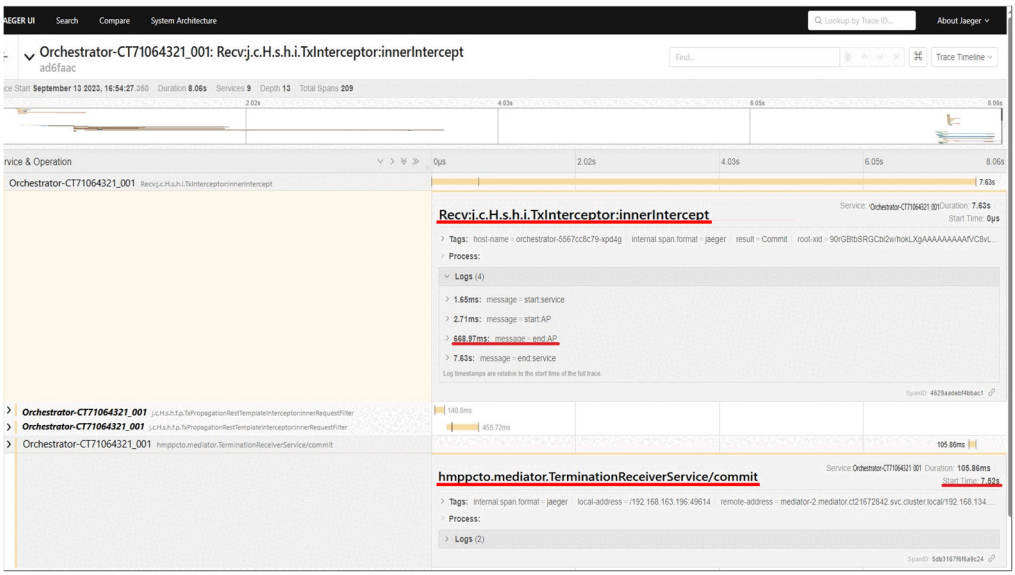
上記 span の Start Time が次の span の Logs「end AP」と差が大きい場合、このパラメタで指定するスレッドが不足してスレッドの空き待ち状態になっているおそれがあります。

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、「11.5.2　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-Module

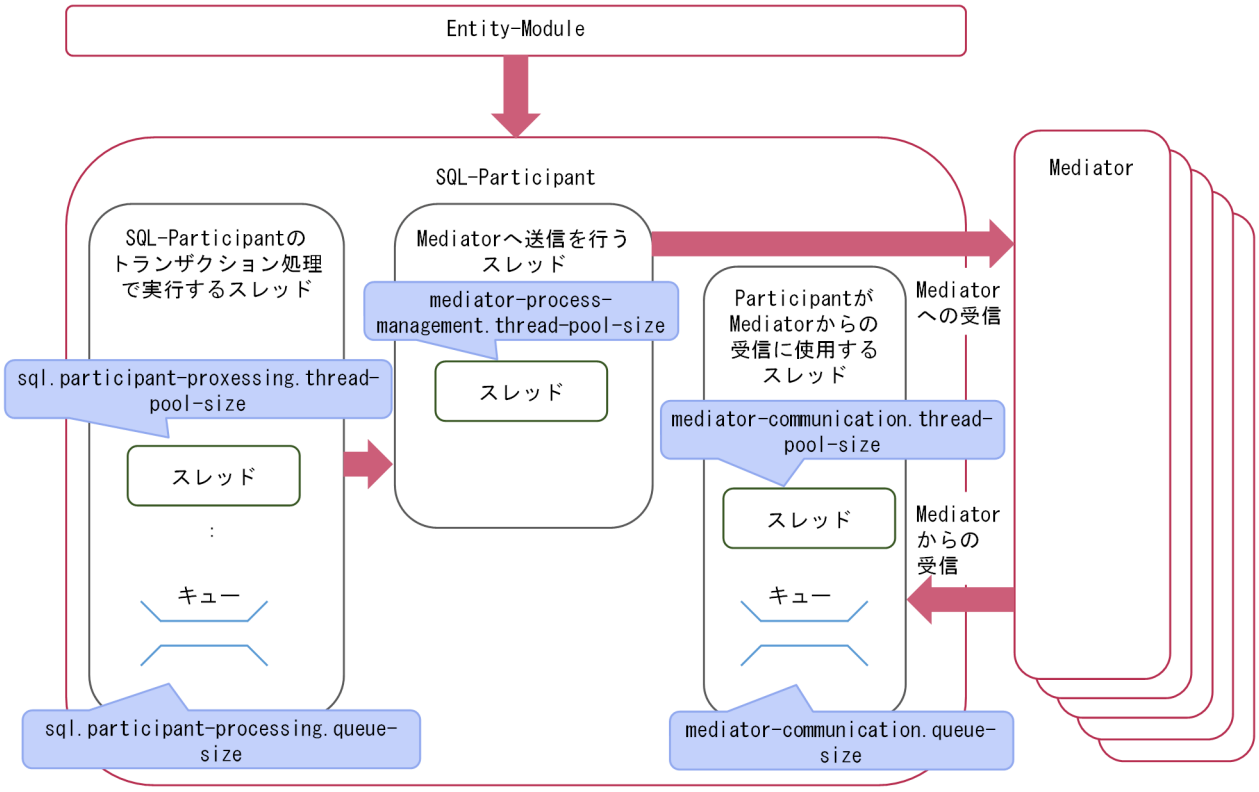
HMP-PCTO が制御するスレッドは存在しないため、HMP-PCTO のスレッドに関するパラメタのチューニングは必要ありません。

(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 がトランザクション処理で実行するスレッドプール
- Mediator へ送信を行うスレッドプール
- Participant が Mediator からの受信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 がトランザクション処理で実行するスレッドプールでは、Entity-Module 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。

- 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プールでは、Mediator へ送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-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。

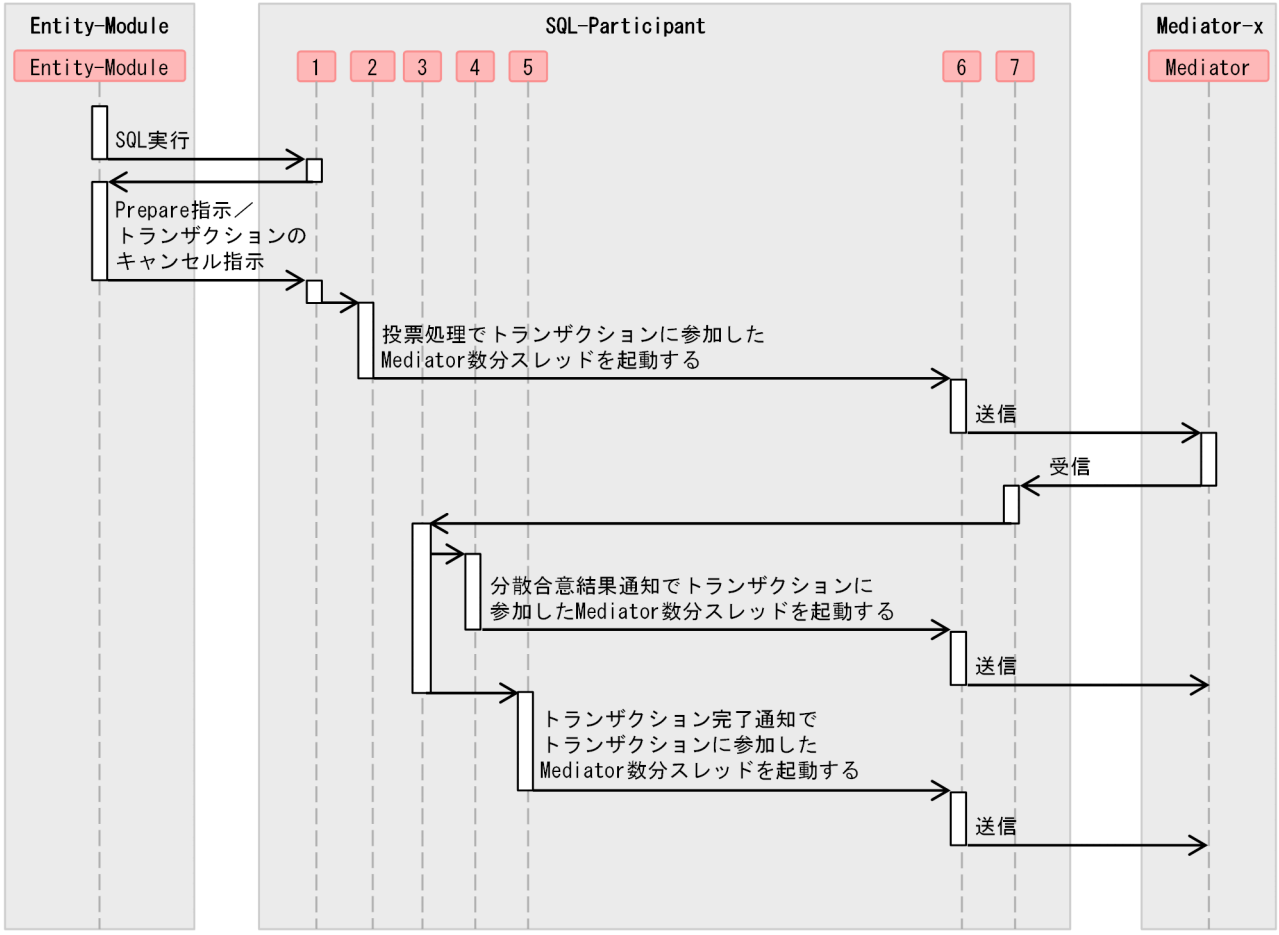
- Mediator との通信処理（投票処理、トランザクション完了通知、分散合意結果通知）が発生した際、トランザクションに参加した Mediator 数分スレッドを使用します。6 のスレッドの生成を開始してから終了するまで占有します。…6

Participant が Mediator からの受信に使用するスレッドプールでは、Participant が Mediator からの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 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。…7

図 3-7　SQL-Participant のスレッド占有期間



(a)　設計指針

■SQL-Participant がトランザクション処理で実行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 がトランザクション処理で実行するキュー

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

■Participant が Mediator からの受信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

■Participant が Mediator からの受信に使用するキュー

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ー覧

SQL-Participant のチューニング対象のパラメター覧を次に示します。

表 3-34　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数を指定します。
4	mediator-communication.thread-pool-size	Participant が Mediator からの受信に使用するスレッド数を指定します。
5	mediator-communication.queue-size	Participant が Mediator からの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、「9.2.5　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、「11.5.3　メトリクス」を参照してください。

Kubernetes マニフェストの.spec.template.metadata.labels の app.kubernetes.io/name に「sql-participant」が設定されている SQL-Participant の機能を使用する Kubernetes アプリケーションの場合、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 3-36　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 3-35　Prometheus で利用するクエリ

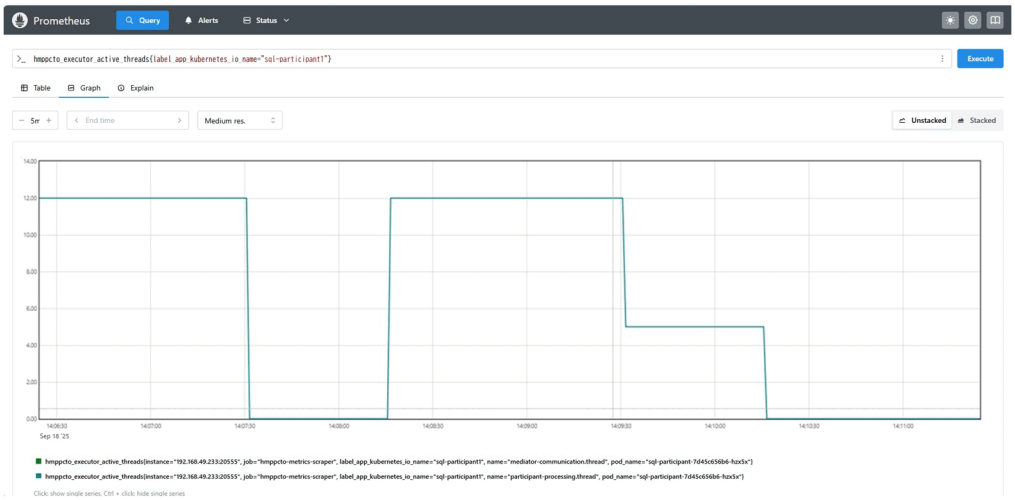
項番	チューニング対象のパラメタ	クエリ
1	sql.participant-processing.thread-pool-size	hmppcto_executor_active_threads{label_app_kubernetes_io_name="sql-participant"}
2	mediator-communication.thread-pool-size	
3	mediator-process-management.thread-pool-size	hmppcto_executor_active{label_app_kubernetes_io_name="sql-participant"}
4	sql.participant-processing.queue-size	hmppcto_executor_queued_tasks{label_app_kubernetes_io_name=" sql-participant"}

項番	チューニング対象のパラメタ	クエリ
5	mediator-communication.queue-size	

表 3-36　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

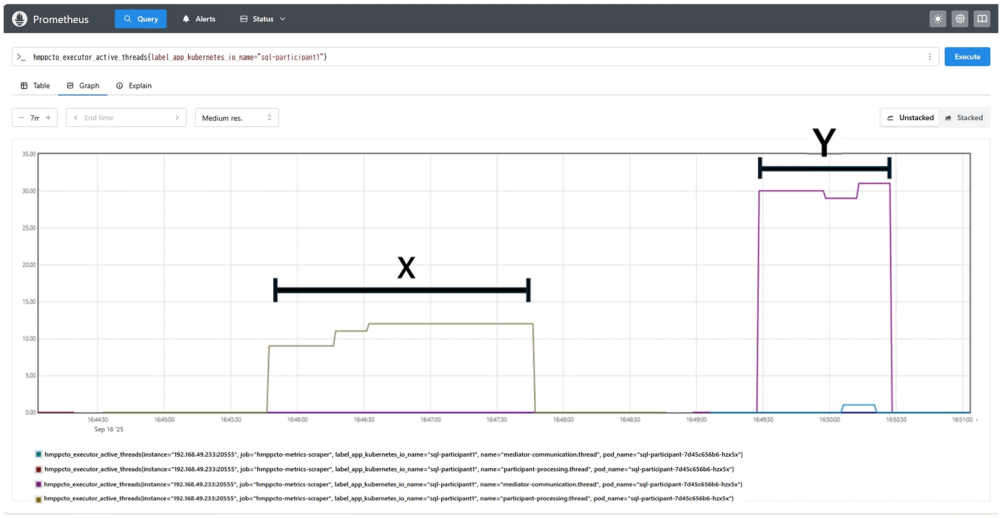
項番	チューニング対象のパラメタ	メトリクス名
1	sql.participant-processing.thread-pool-size	participant-processing.thread
2	sql.participant-processing.queue-size	
3	mediator-process-management.thread-pool-size	mediator-process-management.thread
4	mediator-communication.thread-pool-size	mediator-communication.thread
5	mediator-communication.queue-size	

図 3-8　Prometheus の Web UI



hmppcto_executor_active_threads{label_app_kubernetes_io_name="sql-participant"}の結果が、長い期間 sql.participant-processing.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 3-9　Prometheus の Web UI



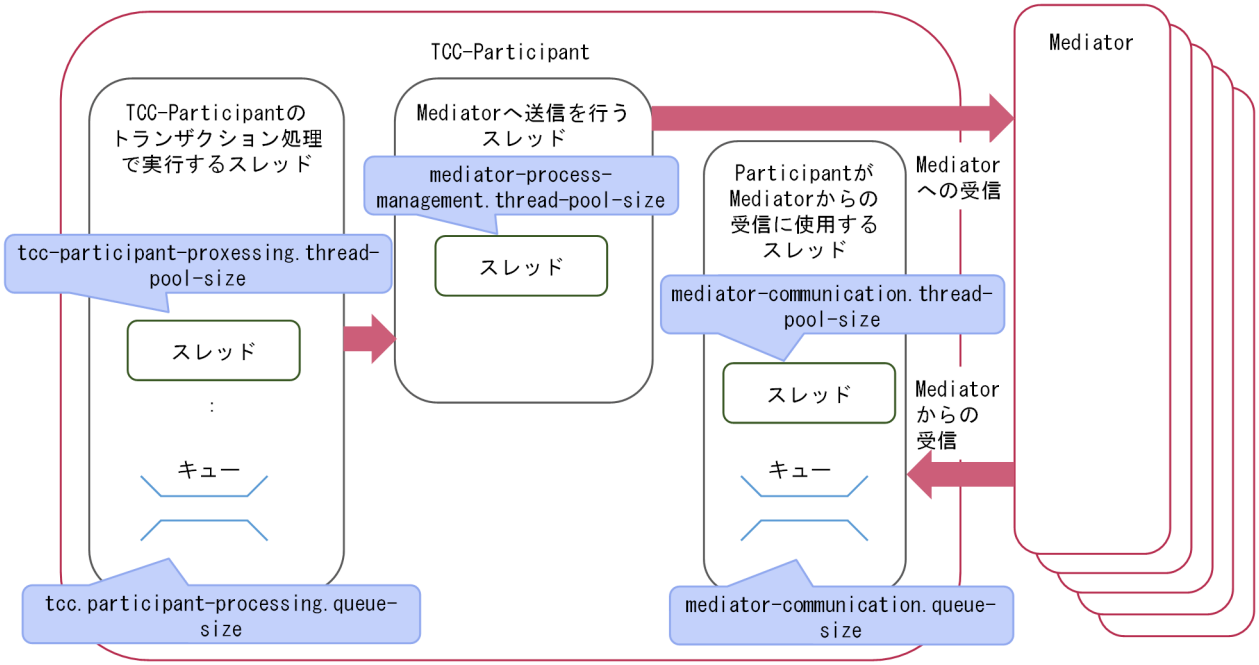
また、hmppcto_executor_queued_tasks{label_app_kubernetes_io_name="sql-participant"}の結果が、sql.participant-processing.queue-size や mediator-communication.queue-size で指定した値で推移している場合、キューサイズ不足が発生しているおそれがあります。これらのパラメタの値を増やすことでキューサイズ不足を防ぐことができます。

(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 がトランザクション処理で実行するスレッドプール
- Mediator へ送信を行うスレッドプール
- Participant が Mediator からの受信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 がトランザクション処理で実行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プールでは、Mediator へ送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-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。

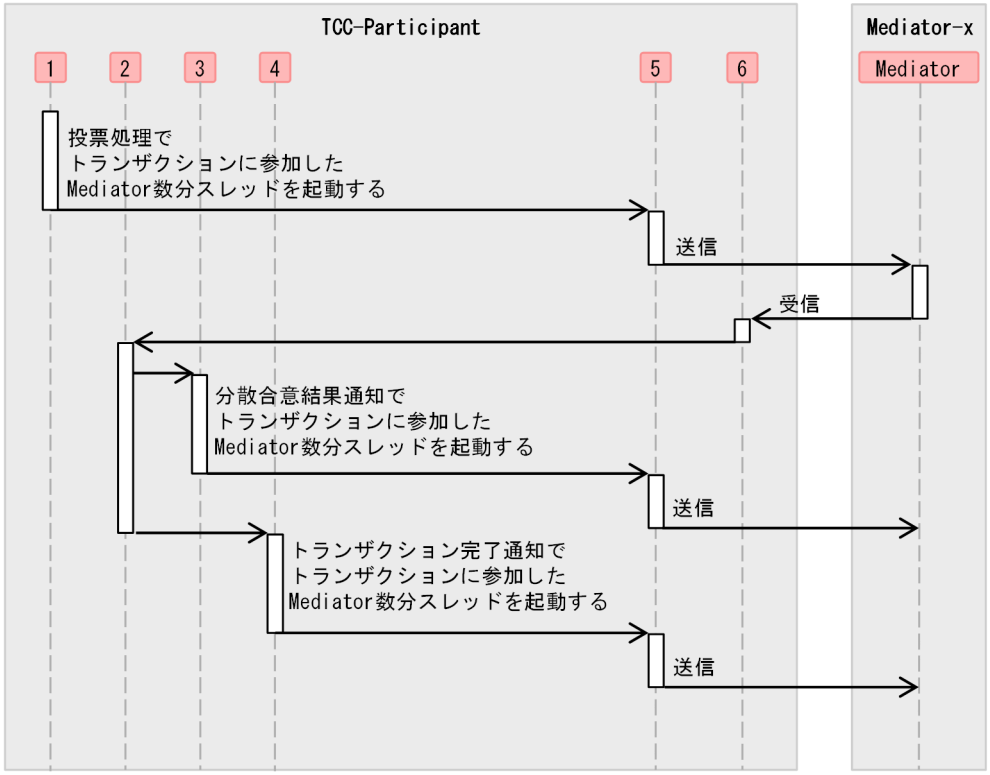
- Mediator との通信処理（投票処理、トランザクション完了通知、分散合意結果通知）が発生した際、トランザクションに参加した Mediator 数分スレッドを使用します。5 のスレッドの生成を開始してから終了するまで占有します。…5

Participant が Mediator からの受信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 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。…6

図 3-11　TCC-Participant のスレッド占有期間



(a)　設計指針

■TCC-Participant がトランザクション処理で実行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 がトランザクション処理で実行するキュー

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

■Participant が Mediator からの受信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

■Participant が Mediator からの受信に使用するキュー

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 × 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ー一覧

TCC-Participant 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 3-37　TCC-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	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で実行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指定します。
4	mediator-communication.thread-pool-size	Participant が Mediator からの受信に使用するスレッド数を指定します。
5	mediator-communication.queue-size	Participant が Mediator 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。

各パラメタの詳細については、「9.2.6　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、「11.5.3　メトリクス」を参照してください。

Kubernetes マニフェストの.spec.template.metadata.labels の app.kubernetes.io/name に「tcc-participant」が設定されている TCC-Participant の機能を使用する Kubernetes アプリケーションの場合、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 3-39　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 3-38　Prometheus で利用するクエリ

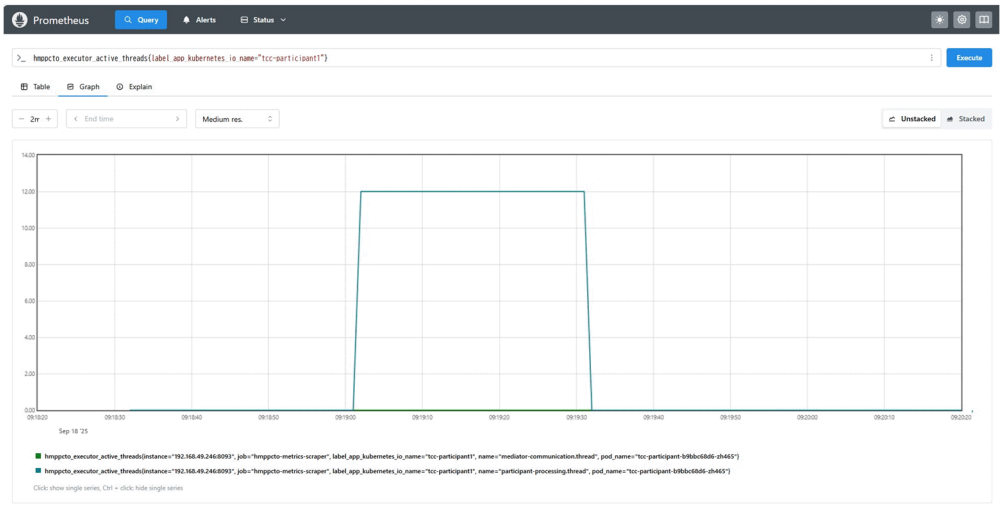
項番	チューニング対象のパラメタ	クエリ
1	tcc.participant-processing.thread-pool-size	hmppcto_executor_active_threads{label_app_kubernetes_io_name="tcc-participant"}
2	mediator-communication.thread-pool-size	
3	mediator-process-management.thread-pool-size	hmppcto_executor_active{label_app_kubernetes_io_name="tcc-participant"}
4	tcc.participant-processing.queue-size	hmppcto_executor_queued_tasks{label_app_kubernetes_io_name="tcc-participant"}

項番	チューニング対象のパラメタ	クエリ
5	mediator-communication.queue-size	

表 3-39　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

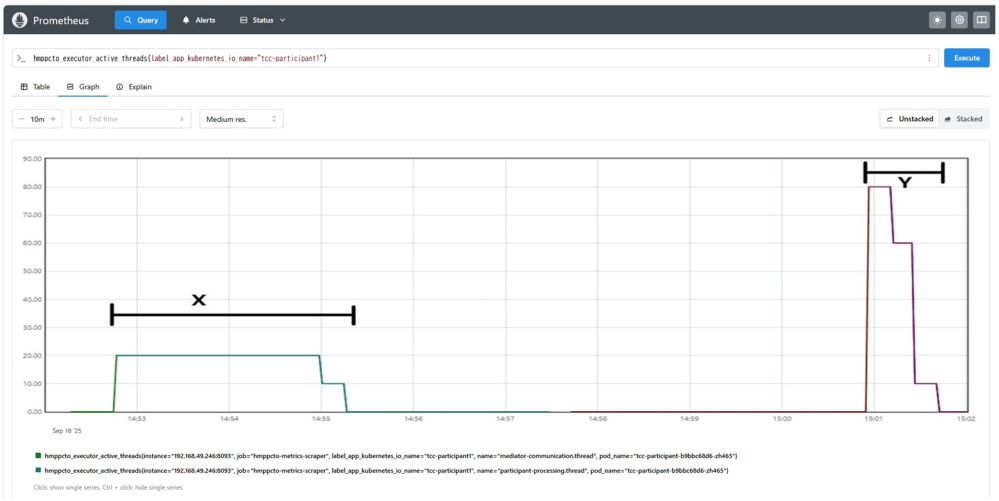
項番	チューニング対象のパラメタ	メトリクス名
1	tcc.participant-processing.thread-pool-size	participant-processing.thread
2	tcc.participant-processing.queue-size	
3	mediator-process-management.thread-pool-size	mediator-process-management.thread
4	mediator-communication.thread-pool-size	mediator-communication.thread
5	mediator-communication.queue-size	

図 3-12　Prometheus の Web UI



hmppcto_executor_active_threads{label_app_kubernetes_io_name=" tcc-participant"}の結果が、長い期間 tcc.participant-processing.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 3-13　Prometheus の Web UI



また、hmppcto_executor_queued_tasks{label_app_kubernetes_io_name=" tcc-participant"}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。

(5)　Mediator

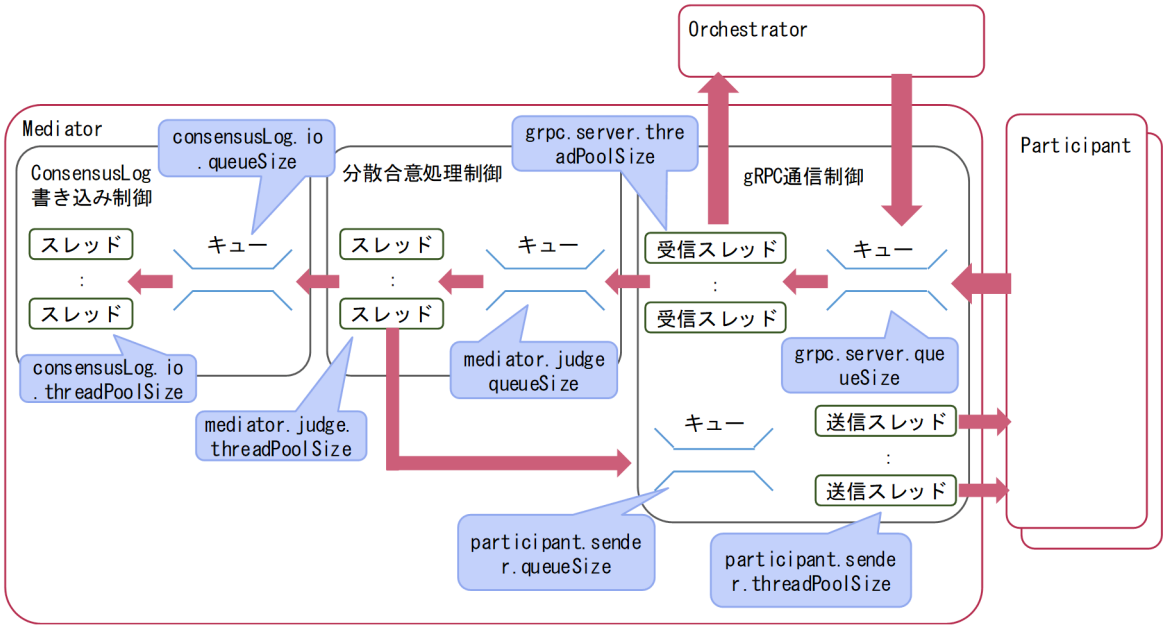
Mediator では次に示す 4 つのスレッドプールがあります。

表 3-40　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。 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。

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「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 × (B ＋ 1) × 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

C: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

C: 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 × B

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

C: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一覧

Mediator のチューニング対象のパラメタ一覧を次に示します。

表 3-41 Mediator 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	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。
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、「9.3.1 Mediator」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、「11.5.3 メトリクス」を参照してください。

Prometheus の Web UI で「表 3-42 Prometheus で利用するクエリ」のクエリを入力し、Execute ボタンを押すことで Graph タブに「図 3-15 Prometheus の Web UI」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、「図 3-15 Prometheus の Web UI」に表示されたメトリクス名の関連づけは、「表 3-43 対象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から「name="メトリクス名"」と表示されるため、該当の表示個所からメトリクス名を確認してください。

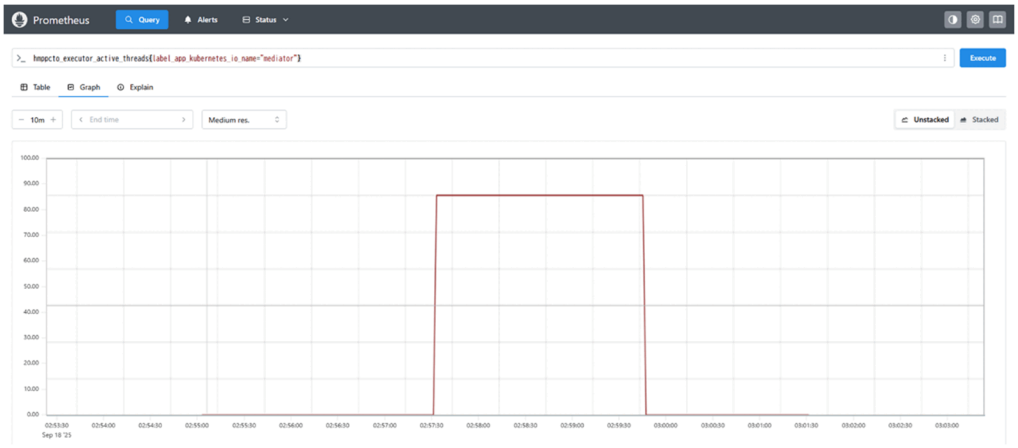
表 3-42　Prometheus で利用するクエリ

項番	チューニング対象のパラメタ	クエリ
1	grpc.server.threadPoolSize	hmppcto_executor_active_threads{label_app_kubernetes_io_name="mediator"}
2	mediator.judge.threadPoolSize	
3	consensusLog.io.threadPoolSize	
4	participant.sender.threadPoolSize	
5	grpc.server.queueSize	hmppcto_executor_queued_tasks{label_app_kubernetes_io_name="mediator"}
6	mediator.judge.queueSize	
7	consensusLog.io.queueSize	
8	participant.sender.queueSize	

表 3-43　対象パラメタとメトリクス名の関連づけ

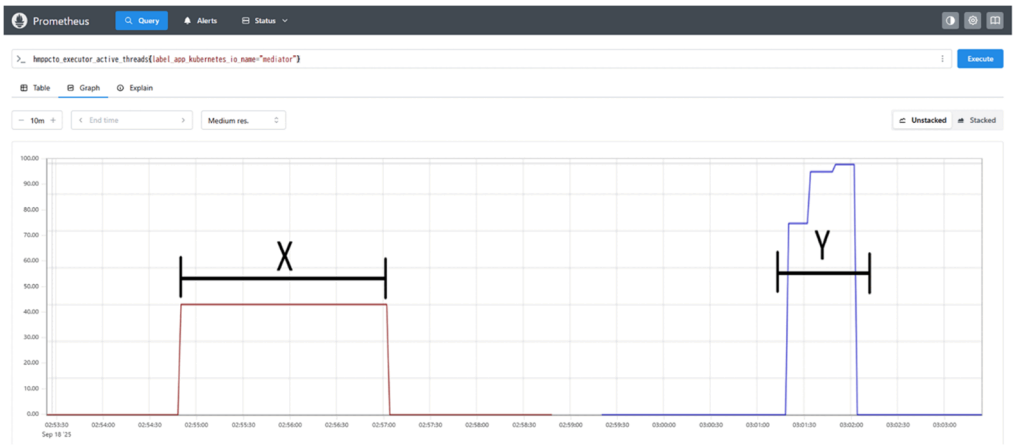
項番	チューニング対象のパラメタ	メトリクス名
1	grpc.server.threadPoolSize	grpc.server.thread
2	grpc.server.queueSize	
3	mediator.judge.threadPoolSize	mediator.judge.thread
4	mediator.judge.queueSize	
5	consensusLog.io.threadPoolSize	consensusLog.io.thread
6	consensusLog.io.queueSize	
7	participant.sender.threadPoolSize	participant.sender.thread
8	participant.sender.queueSize	

図 3-15　Prometheus の Web UI



hmppcto_executor_active_threads{label_app_kubernetes_io_name="mediator"}の結果が、長い期間grpc.server.threadPoolSize、mediator.judge.threadPoolSize、consensusLog.io.threadPoolSize、またはparticipant.sender.threadPoolSize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、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、hmppcto_executor_queued_tasks{label_app_kubernetes_io_name="mediator"}の結果が、grpc.server.queueSize、mediator.judge.queueSize、consensusLog.io.queueSize、またはparticipant.sender.queueSize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げます。

(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 を使用するサービスが複数ある場合は、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 3-44　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。

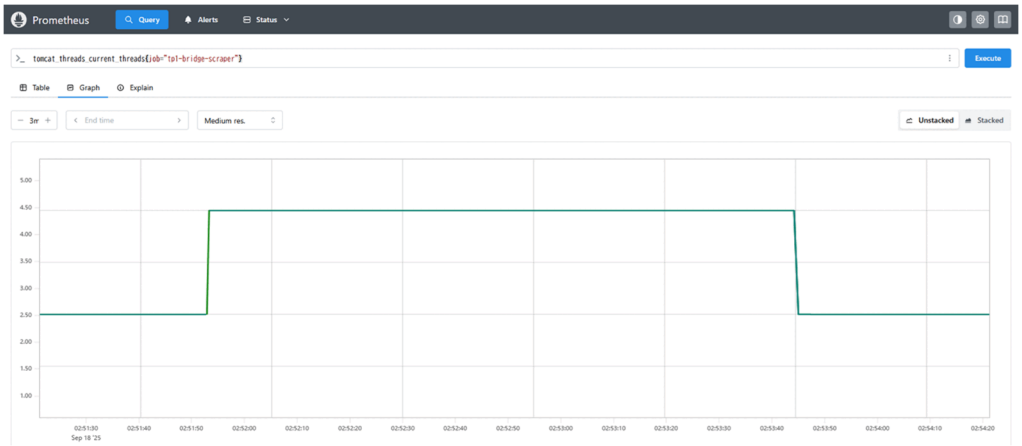
■チューニング方法

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、「11.5.3　メトリクス」を参照してください。

Prometheus の Web UI で「tomcat_threads_current_threads{label_app_kubernetes_io_name="tp1-bridge"}」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



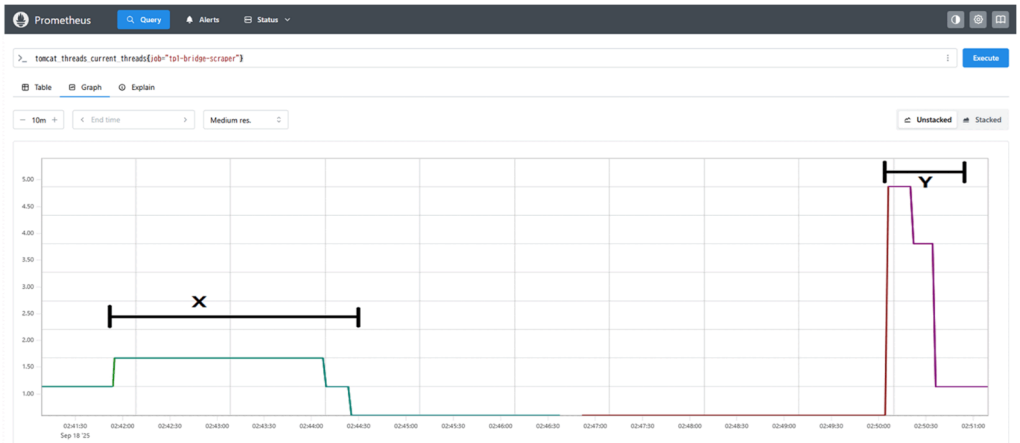
なお、1 つの Prometheus で複数の TP1-Brideg のメトリクスを収集している場合は、クエリに「instance="TP1-Bridge の IP アドレス:TP1-Bridge の http ポート番号"」を追加してください。

入力例：

tomcat_threads_current_threads{label_app_kubernetes_io_name="tp1-bridge", 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。

- TP1-Bridge がトランザクション処理で実行するスレッドプール
- Mediator へ送信を行うスレッドプール
- Participant が Mediator からの受信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

TP1-Bridge のチューニング対象のパラメタ一覧を次に示します。

表 3-45　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で実行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、「9.3.2　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、「11.5.3　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 3-47　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。

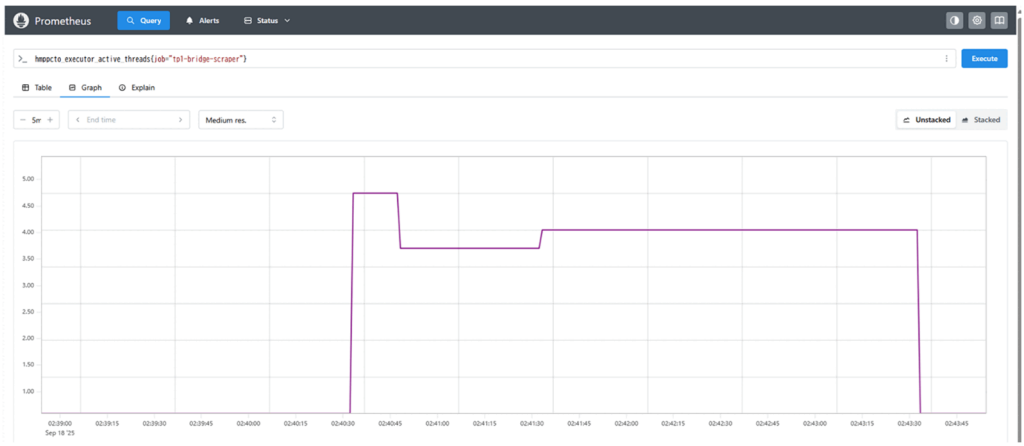
表 3-46　Prometheus で利用するクエリ

項番	チューニング対象のパラメタ	クエリ
1	tp1.participant-processing.thread-pool-size	hmppcto_executor_active_threads{label_app_kubernetes_io_name="tp1-bridge"}
2	mediator-communication.thread-pool-size	
3	mediator-process-management.thread-pool-size	hmppcto_executor_active{label_app_kubernetes_io_name="tp1-bridge"}
4	tp1.participant-processing.queue-size	hmppcto_executor_queued_tasks{label_app_kubernetes_io_name=" tp1-bridge"}
5	mediator-communication.queue-size	

表 3-47　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

項番	チューニング対象のパラメタ	メトリクス名
1	tp1.participant-processing.thread-pool-size	participant-processing.thread
2	tp1.participant-processing.queue-size	
3	mediator-process-management.thread-pool-size	mediator-process-management.thread
4	mediator-communication.thread-pool-size	mediator-communication.thread
5	mediator-communication.queue-size	

図 3-19　Prometheus の Web UI



hmppcto_executor_active_threads{label_app_kubernetes_io_name="tp1-bridge"}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、hmppcto_executor_queued_tasks{label_app_kubernetes_io_name="tp1-bridge"}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。

(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 3-48　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。

3.12.2 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。

表 3-49 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.orchestrator.grpc.watchtime-in-seconds	＜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。 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、「9.2.3 Orchestrator」を参照してください。 ＜設定方法＞

項番	パラメタ設定の主体	パラメタ名	説明
			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。その上でユーザアプリケーションの業務要件に従い 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.transaction-termination.timeout-in-millis
2	• SQL-Participant • TCC-Participant • TP1-Bridge	• SQL-Participant、TCC-Participant の場合 jp.co.Hitachi.soft.hmppcto.external-consensus.interval-in-millis • TP1-Bridge の場合 externalConsensus.intervalInMillis	＜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、「9.2.5　SQL-Participant」、「9.2.6 TCC-Participant」、および「9.3.2　TP1-Bridge（通常版かつ TP1-Bridge 限定）」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

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.consensus.replay-interval-in-millis • TCC-Participant のパラメタ jp.co.Hitachi.soft.hmppcto.tcc.transaction-termination.timeout-in-millis • TCC-Participant のパラメタ jp.co.Hitachi.soft.hmppcto.tcc.consensus.interval-in-millis • TP1-Bridge のパラメタ tp1.transactionTermination.timeoutInMillis • TP1-Bridge のパラメタ tp1.transactionTermination.consensus.replayIntervalInMillis
3	SQL-Participant	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、「9.2.5　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、長めに調整するか次の値を大きくしてください。 • SQL-Participant のパラメタ jp.co.Hitachi.soft.hmppcto.sql.pool.max-pool-size 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。
4		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、「9.2.5　SQL-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アプリケーションの業務要件に従い 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
5		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、「9.2.5　SQL-Participant」を参照してください。 ＜設定方法＞ このタイマはユーザアプリケーションの業務要件に従い 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、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

項番	パラメタ設定の主体	パラメタ名	説明
6	TCC-Participant	jp.co.Hitachi.soft.hmppcto.tcc.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。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、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、「9.2.6　TCC-Participant」を参照してください。 ＜設定方法＞ このタイマはユーザアプリケーションの業務要件に従い 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
7		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、「9.2.6　TCC-Participant」を参照してください。 ＜設定方法＞ このタイマはユーザアプリケーションの業務要件に従い 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

項番	パラメタ設定の主体	パラメタ名	説明
			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
8	TP1-Bridge	tp1.pool.allocationWaitTimeInMillis	＜パラメタの説明＞ 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無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、「9.3.2　TP1-Bridge（通常版かつ TP1-Bridge 限定）」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、長めに調整するか次の値を大きくしてください。 • TP1-Bridge のパラメタ tp1.pool.maxPoolSize タイマの値が長いと OpenTP1 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。
9		tp1.transactionTermination.timeoutInMillis	＜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、TP1-Bridge から OpenTP1 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 TP1-Bridge 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、「9.3.2　TP1-Bridge（通常版かつ TP1-Bridge 限定）」を参照してください。 ＜設定方法＞ このタイマはユーザアプリケーションの業務要件に従い 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が障害が発生した場合に Rollback するまでにタイマの指定値分の時間が掛かります。

項番	パラメタ設定の主体	パラメタ名	説明
			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。 • TP1-Bridge のパラメタ externalConsensus.intervalInMillis
10		tp1.transactionTermination.consensus.replayIntervalInMillis	＜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設定します。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 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際に設定し、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、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るタイマです。 このタイマは、TP1-Bridge 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 TP1-Bridge 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、「9.3.2　TP1-Bridge（通常版かつ TP1-Bridge 限定）」を参照してください。 ＜設定方法＞ このタイマはユーザアプリケーションの業務要件に従い 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が障害が発生した場合に Rollback するまでに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。 • TP1-Bridge のパラメタ externalConsensus.intervalInMillis
11	Mediator	consensusLog.io.watchInSeconds	＜パラメタの説明＞ 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。

項番	パラメタ設定の主体	パラメタ名	説明
			パラメタの詳細については、「9.3.1　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。

3.12.3　G1GC

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一覧

G1GC のチューニング対象のパラメタ一覧を次に示します。

表 3-50　G1GC チューニング対象のパラメタ一覧

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

項番	パラメタ	説明
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
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) 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、「3.8.2 Mediator の Helm チャートのカスタマイズ」を参照してください。

(b) 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、「3.8.3 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。

3.12.4　ディスク

(1)　Mediator

「3.7.2　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）

「3.7.5　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)　Elasticsearch

「3.7.4　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」を参照してください。

3.12.5　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。

(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します。

公式ドキュメントを参照して、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。

(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、「3.7.3　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

(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 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 へと送信する動作と分散トレースを参照するための UI を提供します。Exporter が受信した 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。

チューニング対象のパラメタ一覧

Jaeger のチューニング対象のパラメタ一覧を次に示します。

表 3-51　チューニング対象のパラメタ一覧

項番	パラメタ	説明
1	exporters.jaeger_storage_exporter.queue.queue_size	Exporter 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	exporters.jaeger_storage_exporter.queue.num_consumers	Exporter の送信キューからデータを書き込むワーカ数を指定します。
3	processors.batch.timeout	受信した Span をまとめて Exporter に送信する際の待ち時間を指定します。
4	processors.batch.send_batch_size	一度に送信する Span の最小バッチサイズを指定します。 この数に達した時点で、タイムアウトを待たないで即時に送信が行われます。
5	processors.batch.send_batch_max_size	一度に送信できる Span の最大バッチサイズを指定します。 これ以上の数の Span が溜まった場合は、最大サイズを超えない範囲で複数のバッチに分割して送信されます。

注※

Exporter は、内部キューがいっぱいになると新しい Span を拒否する動作をします。

チューニング方法

表 3-52　チューニング方法

項番	パラメタ	チューニングの考え方
1	exporters.jaeger_storage_exporter.queue.queue_size	Exporter がストレージに書き込む前に一時的に保持できるデータ量を示す内部キューのサイズを設定するパラメタです。 チューニングする場合は、Jaeger の otelcol_exporter_send_failed_spans_total や otelcol_exporter_queue_size を確認し、書き込み処理が追いついていない場合は、設定値の見直しが必要です。

項番	パラメタ	チューニングの考え方
2	exporters.jaeger_storage_exporter.queue.num_consumers	デフォルトでは 10 ワーカが割り当てられているため、通常のトラフィックでは、この値で処理できます。 チューニングする場合は、otelcol_exporter_send_failed_spans_total を確認し、Span の送信失敗が増加している場合は、ワーカ数やストレージ性能などの設定値の見直しが必要です。
3	processors.batch.timeout	バッチを送信するまでの最大待機時間を設定するパラメタです。 Span の数が processors.batch.send_batch_size に設定した値に達しなくても、このパラメタに設定した値に達するとバッチが送信されます。 チューニングする場合は、処理遅延や otelcol_exporter_send_failed_spans_total を確認し、遅延が大き過ぎる場合は短くする、逆にバッチ効率を高めたい場合は長くするなどの調整が必要です。
4	processors.batch.send_batch_size	Exporter への送信のトリガとなる最小バッチサイズを設定するパラメタです。 小さ過ぎると送信回数が増え、オーバーヘッドが増加します。大き過ぎると Collector のメモリ負荷や遅延が増えます。このため、トラフィック量に応じて最小バッチサイズを調整します。 チューニングする場合は、送信効率と遅延のバランスを確認しながら otelcol_exporter_send_failed_spans_total や Collector のメモリ使用量を監視するなどの調節が必要です。
5	processors.batch.send_batch_max_size	1 回のエクスポートで許可される最大バッチサイズを設定するパラメタです。0 は無制限を意味します。 otelcol_exporter_send_failed_spans_total の増加が見られる場合は、バッチサイズの上限を設定して分割送信を促す必要があります。

(c) CPU、メモリのチューニング

Jaeger 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 の公式ドキュメントを参照してリソースを設定してください。

(3) 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 3-53　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（バイト）を定義します。

チューニング方法

表 3-54　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)　Logstash 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。	
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。Logstash の公式ドキュメントを参照してリソースを設定してください。

(4) 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 3-55　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 3-56　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	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)　Filebeat 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。	

項番	パラメタ	チューニングの考え方	デフォルト値
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。Filebeat の公式ドキュメントを参照してリソースを設定してください。

(5) 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 3-57　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 3-58　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)　Metricbeat 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）当たりのメトリクス容量×停止時間×安全係数"	
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れ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。Metricbeat の公式ドキュメントを参照してリソースを設定してください。

(6)　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、各コンポーネントからの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 3-59　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	global.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	remote_write.queue_config.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	remote_write.queue_config.max_shards	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しますが、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	remote_write.queue_config.max_samples_per_send	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	--storage.tsdb.retention.time	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 3-60　チューニング方法

項番	パラメタ	チューニングの考え方
1	global.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。
2	remote_write.queue_config.capacity	Metricbeat に送信する処理のメモリ使用量は、「シャード数 ×（remote_write.queue_config.capacity ＋ remote_write.queue_config.max_samples_per_send）」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、remote_write.queue_config.max_samples_per_send の 3～10 倍程度の値に設定することを推奨しています。
3	remote_write.queue_config.max_shards	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	remote_write.queue_config.max_samples_per_send	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、remote_write.queue_config.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。
5	--storage.tsdb.retention.time	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。Prometheus の公式ドキュメントを参照してリソースを設定してください。

3.12.6 分散トレース

(1) チューニング対象のパラメタ一覧

HMP-PCTO の分散トレースの取得に関する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。

表 3-61 チューニング対象のパラメタ一覧

項番	パラメタ設定の主体	パラメタ名	説明
1	• Orchestrator • Entity-Module • SQL-Participant • TCC-Participant • Mediator • TP1-Bridge	• Orchestrator、Entity-Module、SQL-Participant、TCC-Participant の場合 jp.co.Hitachi.soft.hmppcto.tracing.queue-size • Mediator、TP1-Bridge の場合 tracing.queueSize	＜パラメタの説明＞ 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。 パラメタの詳細については、「9.2.1 共通」を参照してください。 ＜設定方法＞ このパラメタは次のパラメタ以上の値を設定してください。 • jp.co.Hitachi.soft.hmppcto.tracing.batch-size この設定値以上の分散トレースが発生した場合、対象の分散トレースは破棄されてしまいます。 基本はデフォルト値を設定した上で、分散トレースの破棄が頻繁に発生するようであれば、大きな値に調整してください。
2	• Orchestrator • Entity-Module • SQL-Participant • TCC-Participant • Mediator • TP1-Bridge	• Orchestrator、Entity-Module、SQL-Participant、TCC-Participant の場合 jp.co.Hitachi.soft.hmppcto.tracing.delay-time-in-millis • Mediator、TP1-Bridge の場合 tracing.delayTimeInMillis	＜パラメタの説明＞ メモリ上のキューに保持した分散トレースを分散トレース収集アプリケーションまたはサービスへ送信する間隔をミリ秒で指定します。 パラメタの詳細については、「9.2.1 共通」を参照してください。 ＜設定方法＞ このパラメタの設定値を変更すると、メモリ上のキューに保持した分散トレースを送信する間隔が変更されます。 大きな値を設定すると、メモリ上のキューに保持した分散トレースを送信する間隔が長くなり、メモリ上のキューからあふれてしまい、分散トレースが破棄されるおそれがあります。 基本はデフォルト値を設定した上で、分散トレースの破棄が頻繁に発生するようであれば、小さな値に調整してください。
3	• Orchestrator • Entity-Module • SQL-Participant	• Orchestrator、Entity-Module、SQL-Participant、TCC-Participant の場合	＜パラメタの説明＞ メモリ上のキューに保持した分散トレースを分散トレース収集アプリケーションまたはサービスへ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。 パラメタの詳細については、「9.2.1 共通」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
	• TCC-Participant • Mediator • TP1-Bridge	jp.co.Hitachi.soft.hmppcto.tracing.batch-size • Mediator、TP1-Bridge の場合 tracing.batchSize	＜設定方法＞ このパラメタは次のパラメタ以下の値を設定してください。 • jp.co.Hitachi.soft.hmppcto.tracing.queue-size この設定値以上の分散トレースが発生した場合、対象の分散トレースは破棄されます。 このパラメタの設定値を変更すると、メモリ上のキューに保持した分散トレースを送信する数が変更されます。 大きな値を設定すると、メモリ上のキューに保持した分散トレースを送信する数が大きくなり、分散トレース収集アプリケーションまたはサービスが受信できなくなり、分散トレースが破棄されるおそれがあります。 基本はデフォルト値を設定した上で、分散トレースの破棄が頻繁に発生するようであれば、小さな値に調整してください。
4	• Orchestrator • Entity-Module • SQL-Participant • TCC-Participant • Mediator • TP1-Bridge	• Orchestrator、Entity-Module、SQL-Participant、TCC-Participant の場合 jp.co.Hitachi.soft.hmppcto.tracing.debug • Mediator、TP1-Bridge の場合 tracing.debug	＜パラメタの説明＞ HMP-PCTO で実施された内部処理について、デバッグ情報を含むスパンの取得有無を設定します。 このパラメタを true に設定すると、デバッグ情報を含むスパンを作成する処理が動作するため、処理時間が長くなることがあります。 このパラメタを true に設定すると、デバッグ情報を含むスパンが出力されるため、分散トレースとして使用するディスク使用量が増加します。 パラメタの詳細については、「9.2.1　共通」を参照してください。 ＜設定方法＞ このパラメタを true に設定することで、HMP-PCTO の内部処理の詳細を確認できます。 このパラメタについては、当社のサポート部署が調査する場合に設定が必要となります。サポート部署から設定依頼があった場合だけ true を設定してください。
5	SQL-Participant	jp.co.Hitachi.soft.hmppcto.tracing.jdbc.debug	＜パラメタの説明＞ SQL-Participant で実行された JDBC Driver メソッドの開始および終了について、スパンイベントの取得有無を設定します。 このパラメタを true に設定することで、JDBC Driver メソッド開始および終了のスパンイベントを作成する処理が動作するため、処理時間が長くなることがあります。 このパラメタを true に設定することで、JDBC Driver メソッド開始および終了のスパンイベントが出力されるため、分散トレースとして使用するディスク使用量が増加します。 パラメタの詳細については、「9.2.1　共通」を参照してください。 ＜設定方法＞ このパラメタを true に設定することで、SQL-Participant の処理時間が長い場合などに、JDBC Driver メソッドまたは HMP-PCTO の製品処理のどちらで時間が掛かっているかを切り分けることができます。

項番	パラメタ設定の主体	パラメタ名	説明
			ユーザが処理時間について調査や、チューニングを行う場合に設定してください。 当社のサポート部署が調査する場合に設定が必要となることがあります。サポート部署から設定依頼があった場合、true を設定してください。
6	• Orchestrator • SQL-Participant • TCC-Participant • Mediator • TP1-Bridge	• Orchestrator、SQL-Participant、TCC-Participant の場合 jp.co.Hitachi.soft.hmppcto.tracing.grpc.debug • Mediator、TP1-Bridge の場合 tracing.grpc.debug	＜パラメタの説明＞ Orchestrator、Participant および Mediator が実施している gRPC 通信の送受信メッセージ内容について、スパンイベントの取得有無を設定します。 このパラメタを true に設定すると、gRPC 通信の送受信メッセージ内容のスパンイベントを作成する処理が動作するため、処理時間が長くなることがあります。 このパラメタを true に設定すると、gRPC 通信の送受信メッセージ内容のスパンイベントが出力されるため、分散トレースとして使用するディスク使用量が増加します。 パラメタの詳細については、「9.2.1　共通」を参照してください。 ＜設定方法＞ このパラメタを true に設定することで、HMP-PCTO の gRPC 通信の送受信メッセージの内容が正しいこと、また、正しいシーケンスとなっているかを確認できます。 このパラメタについては、当社のサポート部署が調査する場合に設定が必要となります。サポート部署から設定依頼があった場合だけ true を設定してください。

(2)　ディスク使用量の削減

HMP-PCTO の分散トレースではヘッドベースサンプリングを使用し、最初のスパンが作成された時点で、トレース全体をサンプリングするかどうかを決定します。このため、一連の処理で作成されるすべてのスパンが保存され、詳細な処理内容を確認できますが、分散トレースの永続化に必要なディスク使用量が増加しやすくなります。

これに対し、次の機能を組み合わせることで、永続化する分散トレースの量を調整し、ディスク使用量を抑制できます。詳細は OpenTelemetry Collector の公式ドキュメントを参照してください。

- 確率的サンプリングによる、永続化する分散トレースの量の調整
- テイルベースサンプリングによる保存条件の設定

> **重要**
>
> Cloud Trace では、確率的サンプリングおよびテイルベースサンプリングをサポートしていません。確率的サンプリングおよびテイルベースサンプリングは、Cloud Trace にデータを送信するコンポーネントで実施する必要があります。そのため、分散トレースの収集に Cloud Trace

を使用する場合は、OpenTelemetry Collector を経由して Cloud Trace へデータを送信できるように、OpenTelemetry Collector を構成してください。

詳細は Cloud Trace の公式ドキュメントを参照してください。

(a) 取得方針

HMP-PCTO では、通常時は確率的サンプリングによって永続化する分散トレースの量を調整し、トランザクションの失敗やレイテンシの閾値超過などの異常時は分散トレースを永続化することで、ディスク使用量と可観測性を両立します。次の取得方針に従って分散トレースの量を調整してください。

■ 通常時の取得方針

次の方針に従って、永続化する分散トレースを決定してください。

- 確率的サンプリングによって、永続化する分散トレースを決定します。サンプリング対象とならなかった分散トレースは永続化されないで破棄されます。
- 確率的サンプリングのレートはユーザが任意で設定できます。ただし、HMP-PCTO の確率的サンプリングの推奨レートは 10%です。そのため、レートは 10%以上を設定してください。

■ 異常時の取得方針

次のどれかに該当する分散トレースは、必ず永続化してください。

- トランザクションの結果が Commit 以外の場合

  Orchestrator が出力するスパンのオペレーション名「Recv:j.c.H.s.h.i.TxInterceptor:innerIntercept」のスパンタグ「result」に Commit 以外の値が設定された場合は、トランザクションの結果がロールバック、またはトランザクション決着結果が不明となっているため、異常と判断して分散トレースを永続化してください。
- スパンステータスにエラーが記録された場合

  スパンのスパンステータスにエラーが設定された場合は、そのスパンが追跡する操作で何らかのエラーが発生したことを意味するため、異常と判断して分散トレースを永続化してください。
- レイテンシの閾値を超過した場合

  ユーザが実測したトランザクションのレイテンシを確認して、通常の負荷状態やピーク時の負荷の傾向を把握し、閾値を設定してください。設定したレイテンシの閾値を超過した場合は、処理の遅延やボトルネックが発生しているおそれがあるため、異常と判断して分散トレースを永続化してください。

(b) チューニングの設定方法

取得方針に従って分散トレースの量を調整するために、OpenTelemetry Collector を設定してください。

ここでは HMP-PCTO として設定に必要な記載だけを示します。そのほかの記載されていない内容については、OpenTelemetry Collector の公式ドキュメントを参照してください。

■ 確率的サンプリングで永続化する分散トレース量を調整する方法

OpenTelemetry Collector の Processer で tail_sampling を使用して設定します。確率的サンプリングのレートを、HMP-PCTO の推奨レートである 10%とした場合の設定内容を次に示します。

```
processors:
  tail_sampling/hmppcto-sampling:
    policies:
      - name: baseline-10pct
        type: probabilistic
        probabilistic:
        sampling_percentage: 10
```

■ トランザクションの結果が Commit 以外の場合にスパンステータスにエラーを設定する方法

OpenTelemetry Collector の Processer で transform を使用して設定します。Orchestrator が出力するスパンのオペレーション名「Recv:j.c.H.s.h.i.TxInterceptor:innerIntercept」のスパンタグ「result」に Commit 以外の値が設定された場合に、スパンステータスにエラーを設定するときの設定内容を次に示します。

この Processer は、スパンステータスにエラーが記録されたかの判定を実施する前に実施してください。

```
processors:
  transform/hmppcto-transction-status:
    error_mode: ignore
    trace_statements:
      - context: span
        statements:
           - set(status.code, 2) where name == "Recv:j.c.H.s.h.i.TxInterceptor:innerIntercep
t" and
                                       attributes["result"] != nil and attributes["result"] !=
 "Commit"
```

■ スパンステータスにエラーが記録された場合に分散トレースを永続化する方法

OpenTelemetry Collector の Processer で tail_sampling を使用して設定します。スパンステータスにエラーが記録されているかどうかを判定し、エラーが記録されていた場合に分散トレースを永続化するときの設定内容を次に示します。

```
processors:
  tail_sampling/hmppcto-sampling:
    policies:
      - name: error-status
        type: status_code
        status_code:
          status_codes: [ERROR]
```

■ レイテンシの閾値を超過した場合に分散トレースを永続化する方法

OpenTelemetry Collector の Processer で tail_sampling を使用して設定します。スパンのレイテンシの閾値を 1,000ms とし、閾値を超えていた場合に分散トレースを永続化するときの設定内容を次に示します。

```
processors:
  tail_sampling/hmppcto-sampling:
    policies:
      - name: high-latency
        type: latency
        latency:
          threshold_ms: 1000
```

4 スケーリング対応機能の運用

4.1　スケーリング対応機能の概要

スケーリング対応機能とは、Kubernetes による Pod のスケーリング（スケールイン、スケールアウト）に HMP-PCTO が対応するための機能です。Kubernetes による Pod のスケーリングには、オートスケールと手動スケーリングがあります。スケーリング対応機能の詳細および Pod のスケーリングの詳細については、取扱説明書「HMP-PCTO テクニカルガイド」を参照してください。

ここでは、スケーリング対応機能を使用するための運用について説明します。

スケーリング対応機能の対象の Kubernetes アプリケーションを次の表に示します。

表 4–1　スケーリング対応機能の対象の Kubernetes アプリケーションの一覧

Kubernetes アプリケーション	スケーリング対応機能の対象
Orchestrator-Service	○
Relay-Service	○
Entity-Service(SQL)	○
Entity-Service(TCC)	−
Orchestrator-Service(SQL)	○
Alternate-Service	○
Alternate-Service(SQL)	○
Mediator	○
TP1-Bridge	−
EADS（通常版の場合）	−
Ext-Cons（トライアル版の場合）	−
Filebeat	−
Jaeger	−
Prometheus	−
Metricbeat	−
Elasticsearch	−
Logstash	−
Metrics Server	−

（凡例）

○：スケーリング対応機能を使用できます。

−：スケーリング対応機能を使用できません。

オートスケールでは、監視方法として Metrics Server または Prometheus のどちらかを使用します。監視方法は Pod ごとに選択でき、例えば Orchestrator-Service では Metrics Server を、Mediator では

Prometheus を使用した監視方法を選択できます。監視方法は要件に合わせて選択してください。メトリクスの監視方法の詳細については、取扱説明書「HMP-PCTO テクニカルガイド」のメトリクスの監視機能の説明を参照してください。Metrics Server を使用する場合は「4.2　Metrics Server でメトリクスを監視する場合のオートスケールのセットアップ」を、Prometheus を使用する場合は「4.3　Prometheus でメトリクスを監視する場合のオートスケールのセットアップ」を参照してセットアップをしてください。Kubernetes アプリケーションごとに異なる監視方法を選択した場合は、両方のセットアップをしてください。

オートスケールの設定前の Kubernetes アプリケーションと Pod の構成図、およびオートスケールの設定後の Kubernetes アプリケーションと Pod の構成図を、メトリクスの監視方法ごとに示します。なお、Google Cloud Platform 環境の場合は、「KEDA」を「カスタム指標 Stackdriver アダプタ」と読み替えてください。

図 4-1　オートスケールの設定前の Kubernetes アプリケーションと Pod の構成図

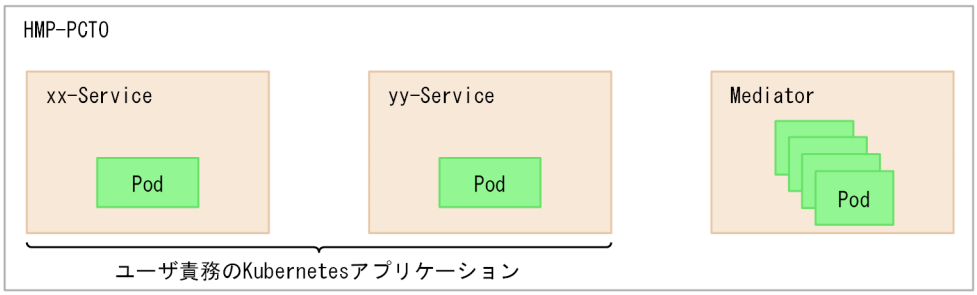


図 4-2　オートスケールの設定後の Kubernetes アプリケーションと Pod の構成図（Metrics Server でメトリクスを監視する方法）

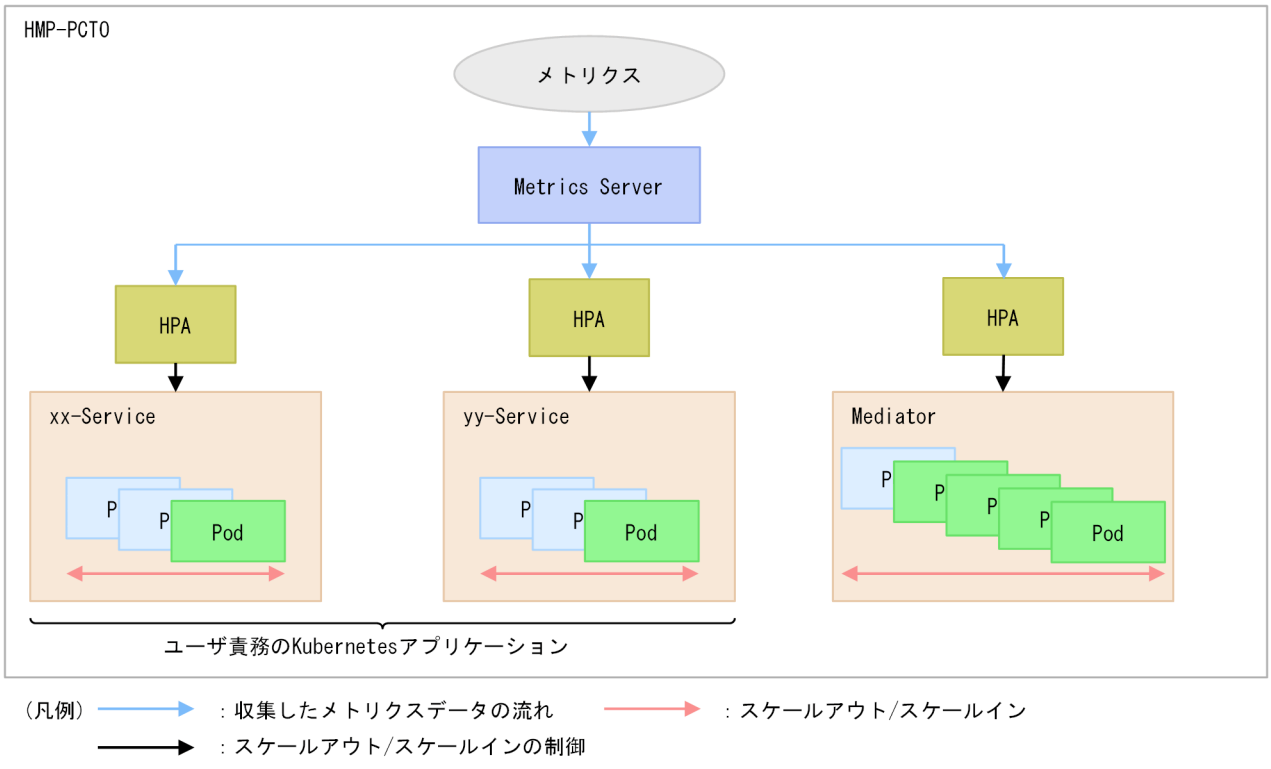
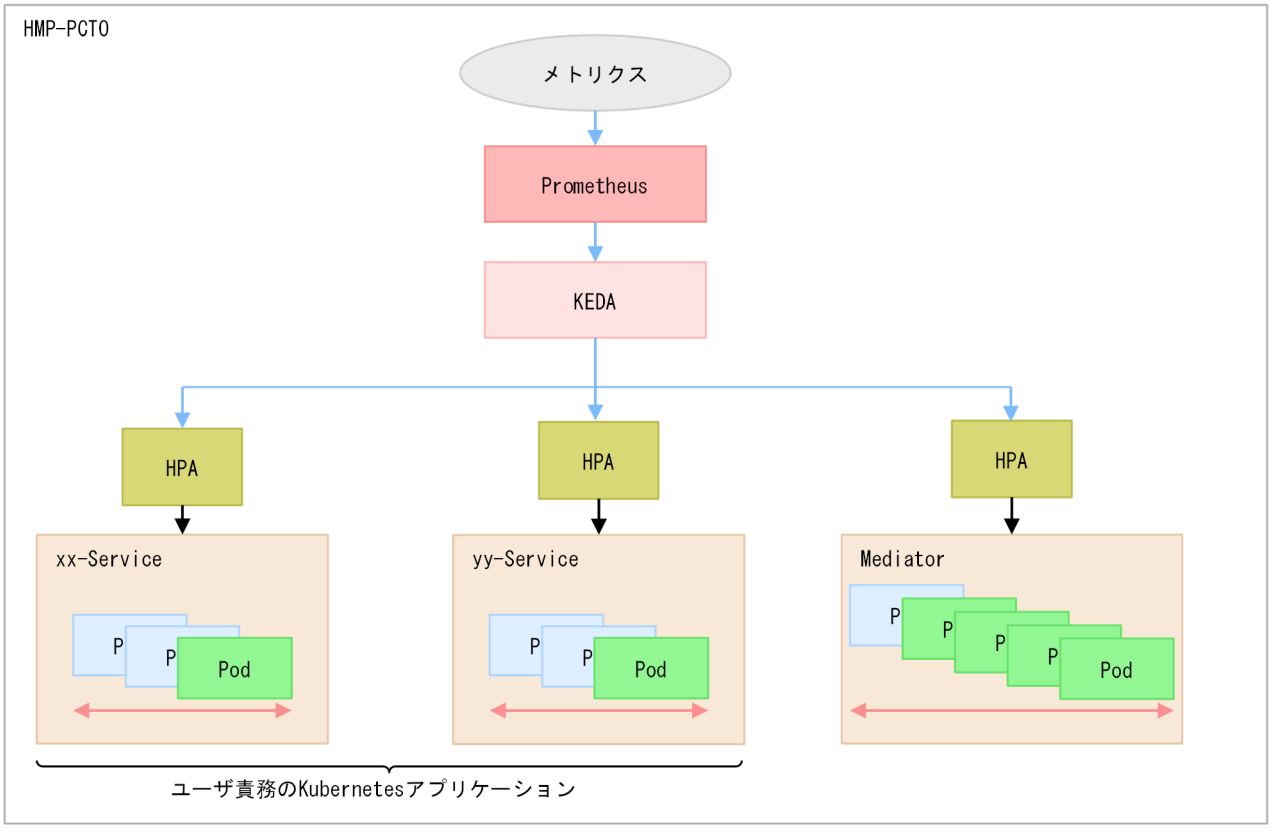


図 4-3　オートスケールの設定後の Kubernetes アプリケーションと Pod の構成図
(Prometheus でメトリクスを監視する方法)



4.2 Metrics Server でメトリクスを監視する場合のオートスケールのセットアップ

Metrics Server でメトリクスを監視する場合のオートスケールのセットアップ手順について説明します。

4.2.1 Metrics Server でメトリクスを監視する場合のオートスケールのセットアップの流れ

新規構築時にオートスケールを適用する場合は、次の手順でオートスケールのセットアップをしてください。オートスケール以外の設定および構築方法については、「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の Kubernetes アプリケーションおよび RAS のデプロイ」を参照してください。

オートスケールを適用しないで構築した環境にオートスケールを適用する場合も同様に、次の手順でオートスケールのセットアップをしてください。ただし、手順 6 と 7 については、オートスケール対象の Kubernetes アプリケーションに対して実施してください。

表 4-2 Metrics Server でメトリクスを監視する場合のオートスケールのセットアップの流れ

手順	項目	備考	参照先
1	Metrics Server をインストールする	オンプレミス環境または Amazon Web Services 環境の場合に必ず実施します。	4.2.2 Metrics Server のインストール（オンプレミス環境または Amazon Web Services 環境限定）
2	Metrics Server をデプロイする	オンプレミス環境の場合に必ず実施します。	4.2.3 Metrics Server のデプロイ（オンプレミス環境限定）
3	HPA の Kubernetes マニフェストを作成する	必ず実施します。	4.2.4 HPA の Kubernetes マニフェストの作成
4	HPA の設定をカスタマイズする	オンプレミス環境、Microsoft Azure 環境、Google Cloud Platform、または Red Hat OpenShift Container Platform の場合で、Kubernetes クラスタ全体の HPA 関連のパラメタの設定を変更したいときに実施します。	4.2.5 HPA の設定のカスタマイズ
5	HPA をデプロイする	必ず実施します。	4.2.6 HPA のデプロイ
6	次の Kubernetes アプリケーションについて、Kubernetes マニフェストを作成、および Helm チャートをカスタマイズする • ユーザ責務の Kubernetes アプリケーション		4.2.7 Kubernetes マニフェストの作成および Helm チャートのカスタマイズ

手順	項目	備考	参照先
	• 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		
7	Kubernetes アプリケーションをデプロイする		4.2.8　Kubernetes アプリケーションのデプロイ

セットアップ後に障害が発生した場合は、「4.5　オートスケールの障害対策」を参照して対応してください。

4.2.2　Metrics Server のインストール（オンプレミス環境または Amazon Web Services 環境限定）

この手順はオンプレミス環境または Amazon Web Services 環境の場合だけ実施します。Metrics Server の公式ドキュメントを参照して、Metrics Server をインストールしてください。

4.2.3　Metrics Server のデプロイ（オンプレミス環境限定）

この手順はオンプレミス環境の場合だけ実施します。Metrics Server をデプロイします。HMP-PCTO として設定が必要なパラメタを次の表に示します。

表 4-3　Metrics Server に HMP-PCTO として設定が必要なパラメタ

パラメタ	説明
image	Metrics Server のコンテナイメージを設定してください。 使用できる Metrics Server のバージョンについては、リリースノートを参照してください。
service.port	Metrics Server 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。

4.2.4　HPA の Kubernetes マニフェストの作成

以降の手順は、Metrics Server が起動していることを前提としています。

以降に示す要件を満たすように、オートスケール用 HPA の Kubernetes マニフェストをオートスケール対象の Kubernetes アプリケーションごとに作成します。このマニフェストファイルを作成することで、オートスケールを使用するために必要な HPA を利用できます。

HPA の Kubernetes マニフェストの記述例を次に示します。HMP-PCTO として apiVersion に、autoscaling/v2 を指定してください。

詳細な設定方法については Kubernetes の公式のドキュメントを参照してください。

```
apiVersion: autoscaling/v2
kind: HorizontalPodAutoscaler
metadata:
  name: metrics-server-hpa     …1
  namespace: my-namespace
spec:
  scaleTargetRef:
    apiVersion: apps/v1       …2
    kind: Deployment          …3
    name: sql-participant     …4
  minReplicas: 1              …5
  maxReplicas: 2              …6
  metrics:                    …7
  - type: Resource
    resource:
      name: cpu
      target:
        type: Utilization
        averageUtilization: 50
  behavior:                     …8
    scaleDown:
      policies:
      - type: Pods
        value: 4
        periodSeconds: 60
    scaleUp:
      policies:
      - type: Percent
        value: 10
        periodSeconds: 60
```

＜説明＞

1. オートスケール対象の Kubernetes アプリケーションの Kubernetes マニフェストに記載する namespace と同一のものを指定してください。
2. オートスケール対象の Kubernetes アプリケーションの Kubernetes マニフェストに記載する apiVersion と同一のものを指定してください。
3. オートスケール対象の Kubernetes アプリケーションの Kubernetes マニフェストに記載する kind と同一のものを指定してください。
4. オートスケール対象の Kubernetes アプリケーションの Kubernetes マニフェストに記載する metadata.name と同一のものを指定してください。この例では sql-participant をオートスケール対象としています。
5. スケールインできる最小の Pod 数を指定します。
6. スケールアウトできる最大の Pod 数を指定します

   5.、6.の Pod 数については、取扱説明書「HMP-PCTO テクニカルガイド」のスケールアウト/スケールインの Pod 数の設定範囲の説明を参照し、その範囲内で指定してください。
7. 「metrics:」の行から「averageUtilization: 50」の行で、評価対象メトリクスと閾値を指定します。

   この例では、CPU、メモリを評価対象メトリクスとしています。

- CPU を監視対象とするため、resource.name に CPU を指定します。
- 監視対象（CPU）の平均利用率を閾値として設定するため、resource.target.type に Utilization を指定します。
- CPU の平均利用率が 50%になるように設定するため、resource.target.averageUtilization に 50 を指定します。

8.「behavior:」の行から「periodSeconds: 60」の行で、スケーリングポリシーを指定します。
スケールアウト/スケールイン時のポリシーをそれぞれ設定します。ポリシーの設定は複数あります。

- 定数による単位秒当たりのスケーリング回数制限の設定
  この例では、スケールイン時のポリシー（scaleDown）を設定しています。scaleDown.policies 以下の type に Pods、value に 4、periodSeconds に 60 を指定しているので、60 秒間当たりにスケールインする最大 Pod 数は 4 台までの設定になります。
- Pod 数の割合による単位秒当たりのスケーリング回数制限の設定
  この例では、スケールアウト時のポリシーの設定（scaleUp）を設定しています。scaleUp.policies 以下の type に Percent、value に 10、periodSeconds に 60 を指定しているので、60 秒間当たりにスケールアウトする台数は全体 Pod 数の 10%までの設定になります。例として全体 Pod 数が 20 台の場合は、60 秒間当たりにスケールアウトする最大 Pod 数は全体の 10%である 2 台までになります。

4.2.5 HPA の設定のカスタマイズ

この手順は、オンプレミス環境、Microsoft Azure 環境、Google Cloud Platform、または Red Hat OpenShift Container Platform の場合で、Kubernetes クラスタ全体の HPA 関連のパラメタの設定を変更したいときに実施します。

kube-controller-manager は、Kubernetes クラスタで、Deployment や Service などの Kubernetes オブジェクトを管理するプロセスを実行する共通コンポーネントです。この設定項目の中に HPA に関する設定項目があります。オートスケール以外の設定項目の詳細については、Kubernetes の公式ドキュメントを参照してください。

kube-controller-manager の設定方法には、マニフェストファイルで設定する方法とコマンドで設定する方法の 2 つの方法があります。ここでは、コマンドでの設定方法を説明します。

コマンドで設定する場合の構文を次に示します。

```
kube-controller-manager [flags]
```

[flags]に設定項目のフラグ名と設定値を指定してコマンドを実行すると、kube-controller-manager の設定が反映されます。

オートスケールに影響のある主なフラグを次に示します。

- --horizontal-pod-autoscaler-cpu-initialization-period

  HPA が CPU のメトリクス評価を始めてから実際にスケーリングをするまでの待機時間です。Pod の起動直後は、起動処理などで通常時よりも CPU 使用率が高くなる場合があります。起動直後の Pod のスケーリングを制御するため、このフラグで待機時間を設定します。

  秒単位で時間を指定します。デフォルト値は 300（5 分）です。

- --horizontal-pod-autoscaler-downscale-stabilization

  HPA がある Pod のスケールインをしてから、次にスケールインをするまでの待機時間です。負荷状況が短い時間で変化する場合に、Pod の稼働台数を安定させるためにこのフラグを使用します。

  秒単位で時間を指定します。デフォルト値は 300（5 分）です。

- --horizontal-pod-autoscaler-initial-readiness-delay

  評価対象の Pod のステータスが Ready に変化してから待機する時間の設定です。Ready になった直後は Pod の負荷状況が安定していない場合があるため、Ready 直後から Pod が評価可能になるまで、最低限待機する時間を設定します。

  秒単位で時間を指定します。デフォルト値は 30（30 秒）です。

- --horizontal-pod-autoscaler-sync-period

  HPA がメトリクスを評価する間隔の設定です。この設定値に従って、メトリクスの評価頻度が変化します。

  秒単位で間隔を指定します。デフォルト値は 15（15 秒）です。

- --horizontal-pod-autoscaler-tolerance

  評価対象メトリクスの値と閾値を比較したときに、スケールアウト/スケールインを実施しない範囲を指定します。

  実数で指定します。デフォルト値は 0.1 です。

メトリクスの評価間隔を 30 秒に設定する場合の、コマンドの実行例を次に示します。

```
kube-controller-manager --horizontal-pod-autoscaler-sync-period=30
```

4.2.6　HPA のデプロイ

Kubernetes コントロールプレーンの任意のパスに、オートスケール対象の Kubernetes アプリケーションごとに作成した HPA の Kubernetes マニフェスト（「4.2.4　HPA の Kubernetes マニフェストの作成」参照）を配置し、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 ./hpa-mediator.yaml
```

この例では、Kubernetes マニフェストのファイル名を「hpa-mediator.yaml」としています。

＜コマンドの実行結果の例＞

```
horizontalpodautoscaler.autoscaling/hpa-mediator 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。

4.2.7　Kubernetes マニフェストの作成および Helm チャートのカスタマイズ

次の Kubernetes アプリケーションについて、Kubernetes マニフェストを作成、および Helm チャートをカスタマイズします。

- 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

Kubernetes マニフェストの作成および Helm チャートのカスタマイズの詳細については、「2.6 Kubernetes マニフェスト作成」または「3.8　Helm チャートのカスタマイズ」を参照してください。

ここでは、スケーリング対応機能に関連する設定について説明します。設定するパラメタについては、「9.2 ユーザのコンポーネントに設定するパラメタ」および「9.3　HMP-PCTO と RAS のコンポーネントに設定するパラメタ」を参照してください。

- スケーリング対応機能を使用するかどうかを設定します。スケーリング対応機能を使用するには、use-scaling パラメタに true（デフォルト値）を指定します。この設定によって、Pod が複数存在する状態でトランザクションが正しく決着するようになります。

  use-scaling パラメタの指定値は、Orchestrator、Entity-Module、SQL-Participant、TCC-Participant、TP1-Bridge、および Mediator で統一する必要があります。
- スケーリング対応機能を使用する場合に必要な設定として、ルーティング機能の実現方法を指定します。set-routing パラメタに inner または user を指定します。
  - inner：HMP-PCTO が提供するルーティング機能を使用してルーティングする場合に指定します。この方法の場合、各種サービスの Pod の前に自前のロードバランサを配置する構成はできません。また、ユーザアプリケーションが実行する HMP-PCTO の Pod 間通信（REST 通信）の暗号化を有効にする場合、サーバ証明書ファイルのホスト名は、次に示す形式で作成します。

    「*.<Pod の Namespace 名>.pod.cluster.local」

    例（Pod の Namespace 名が「test」の場合）：「*.test.pod.cluster.local」
  - user：ユーザがロードバランサを配置し、そのロードバランサでルーティングする場合に指定します。同一トランザクション内で同一のサービス（URI が異なる場合を含む）に対して複数回通信する場合は、同じ Pod に対して通信するように AP を実装し、自前のロードバランサを設定します。同一トランザクション内で異なる Pod に対して通信した場合、トランザクションはロールバックされます。

例として、スケールインなどの理由で Pod が削除されたケースで、同一トランザクション内で異なる Pod に対して通信された場合、HMP-PCTO はトランザクションをロールバックします。

set-routing パラメタの指定値は、Orchestrator、Entity-Module、および SQL-Participant で統一する必要があります。

- HMP-PCTO の各コンテナイメージを識別するための識別子を tracing.instance-id パラメタに設定します。
- 起動する Pod 数を固定しないようにするため、オートスケール対象の Kubernetes アプリケーションでは replicas を設定しないでください。
- Metrics Server を使用する場合に必要な設定として、オートスケール対象の Kubernetes アプリケーションの Kubernetes マニフェストおよび Helm チャートで resources を指定します。

4.2.8 Kubernetes アプリケーションのデプロイ

Kubernetes アプリケーションをデプロイします。デプロイの手順については、「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」、「3.9 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」を参照してください。

デプロイ後、スケーリング対応機能の対象のすべての Kubernetes アプリケーションで、スケーリング対応機能が有効となっていることを示す KFSG82200-I メッセージが起動時のログに出力されていることを確認してください。KFSG82200-I メッセージが出力されていない Kubernetes アプリケーションがある場合、トランザクションが正しく決着しないおそれがあります。

4.3 Prometheus でメトリクスを監視する場合のオートスケールのセットアップ

Prometheus でメトリクスを監視する場合のオートスケールのセットアップ手順について説明します。

4.3.1 Prometheus でメトリクスを監視する場合のオートスケールのセットアップの流れ

このセットアップ手順は、Prometheus が起動している状態で実施する必要があります。

オンプレミス環境の場合

「(15)　Kubernetes アプリケーションのデプロイ」を参照して Prometheus をデプロイしてください。

Microsoft Azure 環境の場合

Microsoft Azure の公式ドキュメントを参照して、マネージド Prometheus を設定およびデプロイしてください。Microsoft Azure 環境のマネージド Prometheus の設定例については、「3.7.14 Microsoft Azure のマネージド Prometheus によるメトリクス収集の Kubernetes マニフェストの作成」を参照してください。

Google Cloud Platform 環境の場合

Google Cloud Platform の公式ドキュメントを参照して、マネージド Prometheus を設定およびデプロイしてください。Google Cloud Platform 環境のマネージド Prometheus の設定例については、「3.7.15　Google Cloud Platform のマネージド Prometheus によるメトリクス収集の Kubernetes マニフェスト作成」を参照してください。

Amazon Web Services 環境の場合

Amazon Web Services の公式ドキュメントを参照して、マネージド Prometheus を設定およびデプロイしてください。Amazon Web Services 環境のマネージド Prometheus の設定例については、「3.7.16　Amazon Web Services のマネージド Prometheus によるメトリクス収集の Kubernetes マニフェスト作成」を参照してください。

Red Hat OpenShift Container Platform 環境の場合

Red Hat OpenShift Container Platform の公式ドキュメントを参照して、マネージド Prometheus を設定およびデプロイしてください。Red Hat OpenShift Container Platform 環境のマネージド Prometheus の設定例については、「3.7.18　Red Hat OpenShift Container Platform モニタリングの Prometheus によるメトリクス収集の Kubernetes マニフェスト作成」を参照してください。

新規構築時にオートスケールを適用する場合は、次の手順でオートスケールのセットアップをしてください。オートスケール以外の設定および構築方法については、「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の Kubernetes アプリケーションおよび RAS のデプロイ」を参照してください。

オートスケールを適用しないで構築した環境にオートスケールを適用する場合も同様に、次の手順でオートスケールのセットアップをしてください。ただし、手順５と６については、オートスケール対象の Kubernetes アプリケーションに対して実施してください。

表 4-4　Prometheus でメトリクスを監視する場合のオートスケールのセットアップの流れ

手順	内容	備考	参照先
1	KEDA をインストールおよびデプロイする	オンプレミス環境または Amazon Web Services 環境の場合に必ず実施します。	4.3.2　KEDA のインストールとデプロイ（オンプレミス環境または Amazon Web Services 環境限定）
	KEDA アドオンをインストールする	Microsoft Azure 環境の場合に必ず実施します。	4.3.3　KEDA アドオンのインストール（Microsoft Azure 環境限定）
	カスタム指標 Stackdriver アダプタをインストールする	Google Cloud Platform 環境の場合に必ず実施します。	4.3.4　カスタム指標 Stackdriver アダプタのインストール（Google Cloud Platform 環境限定）
	Custom Metrics Autoscaler をインストールする	Red Hat OpenShift Container Platform 環境の場合に必ず実施します。	4.3.5　Custom Metrics Autoscaler のインストール（Red Hat OpenShift Container Platform 環境限定）
2	HPA の Kubernetes マニフェストを作成する	オンプレミス環境または Microsoft Azure 環境の場合に必ず実施します。	(1)　オンプレミス環境または Microsoft Azure 環境の場合
		Google Cloud Platform 環境または Amazon Web Services 環境の場合に必ず実施します。	(2)　Google Cloud Platform 環境または Amazon Web Services 環境の場合
		Red Hat OpenShift Container Platform 環境の場合に必ず実施します。	(3)　Red Hat OpenShift Container Platform 環境の場合
3	HPA の設定をカスタマイズする	オンプレミス環境、Microsoft Azure 環境、Google Cloud Platform、または Red Hat OpenShift Container Platform の場合で、Kubernetes クラスタ全体の HPA 関連のパラメタの設定を変更したいときに実施します。	4.3.7　HPA の設定のカスタマイズ
4	HPA をデプロイする	必ず実施します。	4.3.8　HPA のデプロイ
5	次の Kubernetes アプリケーションについて、Kubernetes マニフェストを作成、および Helm チャートをカスタマイズする		4.3.9　Kubernetes マニフェストの作成および Helm チャートのカスタマイズ

手順	内容	備考	参照先
	• 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	Kubernetes アプリケーションをデプロイする		4.3.10　Kubernetes アプリケーションのデプロイ

セットアップ後に障害が発生した場合は、「4.5　オートスケールの障害対策」を参照して対応してください。

4.3.2　KEDA のインストールとデプロイ（オンプレミス環境または Amazon Web Services 環境限定）

この手順はオンプレミス環境または Amazon Web Services 環境の場合だけ実施します。KEDA の公式ドキュメントを参照して、KEDA をインストールおよびデプロイしてください。インストールする KEDA のバージョンについては、リリースノートを参照してください。

以降の手順は、KEDA のデプロイが完了していることを前提としています。

4.3.3　KEDA アドオンのインストール（Microsoft Azure 環境限定）

この手順は Microsoft Azure 環境の場合だけ実施します。Microsoft Azure の公式ドキュメントを参照して、KEDA アドオンをインストールしてください。

以降の手順は、KEDA アドオンのインストールが完了していることを前提としています。

4.3.4　カスタム指標 Stackdriver アダプタのインストール（Google Cloud Platform 環境限定）

この手順は Google Cloud Platform 環境の場合だけ実施します。Google Cloud Platform の公式ドキュメントを参照して、カスタム指標 Stackdriver アダプタをクラスタにインストールしてください。

以降の手順は、カスタム指標 Stackdriver アダプタのインストールが完了していることを前提としています。

4.3.5 Custom Metrics Autoscaler のインストール（Red Hat OpenShift Container Platform 環境限定）

この手順は Red Hat OpenShift Container Platform 環境の場合だけ実施します。Red Hat OpenShift Container Platform の公式ドキュメントを参照して、Red Hat OpenShift Container Platform の Web UI 画面内の OperatorHub から、Custom Metrics Autoscaler をインストールしてください。

以降の手順は、Custom Metrics Autoscaler のインストールが完了していることを前提としています。

4.3.6 HPA の Kubernetes マニフェストの作成

オートスケール用 HPA の Kubernetes マニフェストをオートスケール対象の Kubernetes アプリケーションごとに作成します。

(1) オンプレミス環境または Microsoft Azure 環境の場合

オンプレミス環境または Microsoft Azure 環境の場合、以降に示す要件を満たすように、オートスケール用 HPA の Kubernetes マニフェストをオートスケール対象の Kubernetes アプリケーションごとに作成します。このマニフェストファイルを作成することで、オートスケールを使用するために必要な HPA を利用できます。

HPA の Kubernetes マニフェストの記述例を次に示します。HMP-PCTO として apiVersion に、keda.sh/v1alpha1 を指定してください。

詳細な設定方法については Kubernetes の公式ドキュメントを参照してください。

```
apiVersion: keda.sh/v1alpha1
kind: ScaledObject
metadata:
  name: Prometheus-hpa
  namespace: my-namespace     …1
spec:
  scaleTargetRef:
    apiVersion: apps/v1       …2
    kind: Deployment          …3
    name: sql-participant     …4
  minReplicaCount: 1          …5
  maxReplicaCount: 2          …6
  triggers:                   …7
    - type: prometheus
      metadata:
        serverAddress: 'http://prometheus-server:8080'
        query: 'SUM(hmppcto_participant_transactions_inProcess_operations{id=~"^sql-particip
ant-.*", status=~"NONE|PREPARING|COMMIT_VOTE|ROLLBACK_VOTE|COMMITTING|ROLLING_BACK|READ_ONLY
|COMMITTED|ROLLED_BACK|ROLLED_BACK_BY_RM"})'
        threshold: "3"
  advanced:
    horizontalPodAutoscalerConfig:
      behavior:               …8
```

```
    scaleDown:
      policies:
        - type: Percent
          value: 50
          periodSeconds: 30
    scaleUp:
      policies:
        - type: Percent
          value: 50
          periodSeconds: 10
```

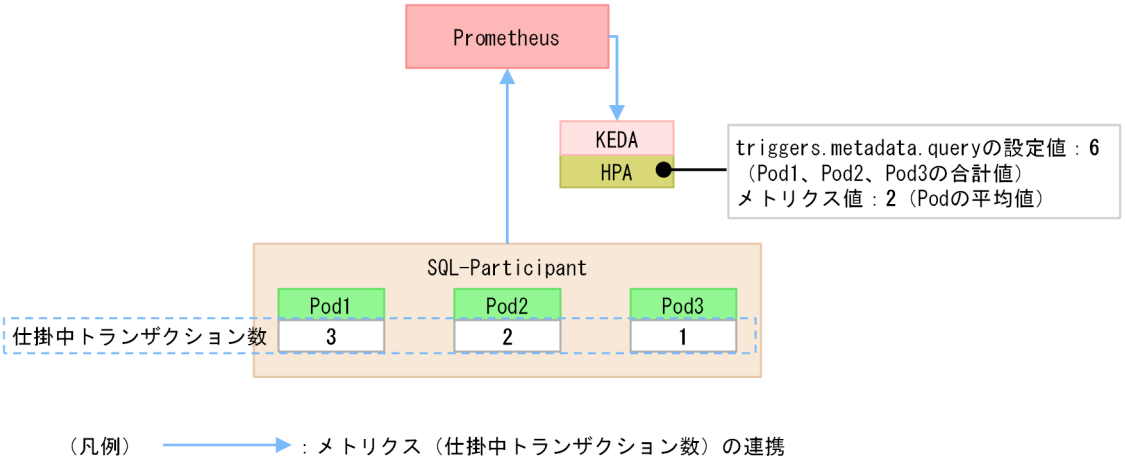
＜説明＞

1. オートスケール対象の Kubernetes アプリケーションの Kubernetes マニフェストに記載する namespace と同一のものを指定してください。
2. オートスケール対象の Kubernetes アプリケーションの Kubernetes マニフェストに記載する apiVersion と同一のものを指定してください。
3. オートスケール対象の Kubernetes アプリケーションの Kubernetes マニフェストに記載する kind と同一のものを指定してください。
4. オートスケール対象の Kubernetes アプリケーションの Kubernetes マニフェストに記載する metadata.name と同一のものを指定してください。この例では sql-participant をオートスケールの対象としています。
5. スケールインできる最小の Pod 数を指定します。
6. スケールアウトできる最大の Pod 数を指定します

   5.、6.の Pod 数については、取扱説明書「HMP-PCTO テクニカルガイド」のスケールアウト/スケールインの Pod 数の設定範囲の説明を参照し、その範囲内で指定してください。
7. 「triggers:」の行から「threshold: "3"」の行で、取得対象のメトリクスと閾値を指定します。
   - triggers.type には、「prometheus」を指定します。
   - triggers.metadata.serverAddress には、使用している Prometheus や環境に応じたメトリクスエンドポイントを指定します。この例ではメトリクスエンドポイントを「http://prometheus-server:8080」としています。
   - triggers.metadata.query には、Prometheus のクエリを指定します。この例では、sql-participant の Pod1 台当たりの仕掛中トランザクション数を取得対象のメトリクスとしています。このクエリは、SUM 関数を使用し、存在する sql-participant のすべての仕掛中トランザクション数を取得しています。

     イメージ図を次に示します。

図 4-4　triggers.metadata.query の設定例イメージ図



HPA は、triggers.metadata.query で設定された値を、オートスケール対象の Kubernetes アプリケーションの Pod 数で除算し平均値を求めます。

「hmppcto_participant_transactions_inProcess_operations」は、HMP-PCTO が提供している仕掛中トランザクション数を取得するメトリクスです。このほかに使用できるメトリクスの一覧は「(3)　出力内容」を参照してください。

「hmppcto_participant_transactions_inProcess_operations」以降の{}で囲まれている範囲はメトリクスの検索条件を指定しています。この例では、正規表現を用いて、sql-participant の仕掛中トランザクションのメトリクスに絞り込んでいます。

クエリの書き方の詳細は、Prometheus の公式ドキュメントを参照してください。

- triggers.metadata.threshold には、閾値を指定します。HPA は、triggers.metadata.query の設定で取得される平均値を、ここで指定された値に近づくように、対象の Pod をオートスケールさせます。数値でもダブルクォーテーションで囲んで指定してください。
- 環境によってはメトリクス取得時に認証が必要となる場合があります。必要に応じて authModes、authenticationRef.name などの認証情報を追加してください。詳細は Prometheus の公式ドキュメントを参照してください。

  Red Hat OpenShift Container Platform モニタリングで提供される Prometheus を使用する場合、Prometheus スケーラーがメトリクスを取得する際に Thanos Querier 経由で通信する必要があります。そのため、Thanos Querier のエンドポイントに対する認証設定が必須です。詳細は Red Hat OpenShift Container Platform の公式ドキュメントを参照してください。

8.「behavior:」の行から「periodSeconds: 10」の行で、スケーリングポリシーを指定します。

設定方法は、「4.2.4　HPA の Kubernetes マニフェストの作成」の項番 8.を参照してください。

(2)　Google Cloud Platform 環境または Amazon Web Services 環境の場合

Google Cloud Platform 環境または Amazon Web Services 環境の場合、オートスケール用 HPA の Kubernetes マニフェストをオートスケール対象の Kubernetes アプリケーションごとに作成します。このマニフェストファイルを作成することで、オートスケールを使用するために必要な HPA を利用できます。

PromQL クエリ（sum など）を使用するときは、マネージドルールを作成し、HPA マニフェストの指標として指定してください。詳細な手順については、Google Cloud Platform または Amazon Web Services の公式ドキュメントを参照してください。

(3) Red Hat OpenShift Container Platform 環境の場合

Red Hat OpenShift Container Platform 環境の場合、オートスケール用 HPA の Kubernetes マニフェストをオートスケール対象の Kubernetes アプリケーションごとに作成します。このマニフェストファイルを作成することで、オートスケールを使用するために必要な HPA を利用できます。

Red Hat OpenShift Container Platform では、Thanos を介して Prometheus メトリクスを公開しているため、Thanos にアクセスするための ServiceAccount、認証情報、および RBAC 権限の設定が必要です。詳細な手順については、Red Hat OpenShift Container Platform の公式ドキュメントを参照してください。

4.3.7 HPA の設定のカスタマイズ

この手順は、オンプレミス環境、Microsoft Azure 環境、Google Cloud Platform、または Red Hat OpenShift Container Platform の場合で、Kubernetes クラスタ全体の HPA 関連のパラメタの設定を変更したいときに実施します。カスタマイズ方法については、「4.2.5 HPA の設定のカスタマイズ」を参照してください。

4.3.8 HPA のデプロイ

Kubernetes コントロールプレーンの任意のパスに、オートスケール対象の Kubernetes アプリケーションごとに作成した HPA の Kubernetes マニフェスト（「4.3.6 HPA の Kubernetes マニフェストの作成」参照）を配置し、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 ./hpa-mediator.yaml
```

この例では、Kubernetes マニフェストのファイル名を「hpa-mediator.yaml」としています。

＜コマンドの実行結果の例＞

```
scaledobject.keda.sh/hpa-mediator 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。

4.3.9　Kubernetes マニフェストの作成および Helm チャートのカスタマイズ

次の Kubernetes アプリケーションについて、Kubernetes マニフェストを作成、および Helm チャートをカスタマイズします。

- 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

Kubernetes マニフェストの作成および Helm チャートのカスタマイズの詳細については、「2.6 Kubernetes マニフェスト作成」または「3.8　Helm チャートのカスタマイズ」を参照してください。

ここでは、スケーリング対応機能に関連する設定について説明します。設定するパラメタについては、「9.2 ユーザのコンポーネントに設定するパラメタ」および「9.3　HMP-PCTO と RAS のコンポーネントに設定するパラメタ」を参照してください。

- スケーリング対応機能を使用するかどうかを設定します。スケーリング対応機能を使用するには、use-scaling パラメタに true（デフォルト値）を指定します。この設定によって、Pod が複数存在する状態でトランザクションが正しく決着するようになります。

  use-scaling パラメタの指定値は、use-scaling パラメタを指定できるすべての Kubernetes アプリケーションで統一する必要があります。
- スケーリング対応機能を使用する場合に必要な設定として、ルーティング機能の実現方法を指定します。set-routing パラメタに inner または user を指定します。
  - inner：HMP-PCTO が提供するルーティング機能を使用してルーティングする場合に指定します。この方法の場合、各種サービスの Pod の前に自前のロードバランサを配置する構成はできません。また、ユーザアプリケーションが実行する HMP-PCTO の Pod 間通信（REST 通信）の暗号化を有効にする場合、サーバ証明書ファイルのホスト名は、次に示す形式で作成します。

    「*.<Pod の Namespace 名>.pod.cluster.local」

    例（Pod の Namespace 名が「test」の場合）：「*.test.pod.cluster.local」
  - user：ユーザがロードバランサを配置し、そのロードバランサでルーティングする場合に指定します。同一トランザクション内で同一のサービス（URI が異なる場合を含む）に対して複数回通信する場合、同じ Pod に対して通信するように AP を実装し、自前のロードバランサを設定します。同一トランザクション内で異なる Pod に対して通信した場合、トランザクションはロールバックされます。

    例として、スケールインなどの理由で Pod が削除されたケースで、同一トランザクション内で異なる Pod に対して通信された場合、HMP-PCTO はトランザクションをロールバックします。

  set-routing パラメタの指定値は、Orchestrator、Entity-Module、および SQL-Participant で統一する必要があります。
- HMP-PCTO の各コンテナイメージを識別するための識別子を tracing.instance-id パラメタに設定します。
- 起動する Pod 数を固定しないようにするため、オートスケール対象の Kubernetes アプリケーションでは replicas を設定しないでください。

4.3.10　Kubernetes アプリケーションのデプロイ

Kubernetes アプリケーションをデプロイします。デプロイの手順については、「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」、「3.9　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」を参照してください。

デプロイ後、スケーリング対応機能の対象のすべての Kubernetes アプリケーションで、スケーリング対応機能が有効となっていることを示す KFSG82200-I メッセージが起動時のログに出力されていることを確認してください。KFSG82200-I メッセージが出力されていない Kubernetes アプリケーションがある場合、トランザクションが正しく決着しないおそれがあります。

4.4　オートスケールのパラメタの設計とチューニング

スケーリング対応機能でオートスケールを使用する場合のパラメタの設計指針、チューニング方法、チューニング例などについて説明します。

4.4.1　設計指針

スケーリング対応機能でオートスケールを使用する場合は、スケールアウト/スケールインに使用する評価対象メトリクスの閾値、評価間隔、スケーリングポリシーなどの設定が必要です。以降で説明する内容に従ってチューニングし、適切な値を設定してください。なお、基本的なチューニングの設計方針に関しては、Kubernetes の公式のドキュメントを参照してください。

ここでは HMP-PCTO として特に注意が必要な観点について説明します。

オートスケールのチューニングの設計指針について、一般的な観点と、HMP-PCTO 固有の観点それぞれに分けて説明します。

(1)　一般的な観点でのチューニングの設計指針

オートスケールをチューニングする上でのチューニング要素は、一般的に次の要素があります。

- 評価対象メトリクスの選定
- 評価対象メトリクスの閾値
- 評価対象メトリクスの評価間隔
- 評価対象メトリクスの許容範囲

これらの要素について、業務アプリケーションの要件に従って、次の観点でチューニングします。

表 4-5　オートスケールの一般的なチューニング要素に対するチューニング観点

項番	チューニング要素	チューニング観点
1	評価対象メトリクスの選定	オートスケールの評価対象の基準となるメトリクスを選定します。業務要件に合わせてシステムの負荷状況を評価するメトリクスを選定する必要があります。詳細は、「4.4.2　評価対象メトリクスの選定」を参照してください。
2	評価対象メトリクスの閾値	システムの過負荷を避けつつ、リソースの無駄を減らす閾値を設定する必要があります。このため、チューニング対象の Kubernetes アプリケーションの過去のメトリクスデータを参照し、通常の負荷状態やピーク時の負荷の傾向を把握し、閾値を決定します。
3	評価対象メトリクスの評価間隔	負荷が急激に変動し、即座にスケーリングをすることが求められるシステムの場合は、短い評価間隔でメトリクスを評価する必要があります。一方、CPU 使用率の瞬間的なスパイクの際に即座にスケーリングをすると、リソースの無駄が発生する場合もあります。

項番	チューニング要素	チューニング観点
		負荷の変動にどれだけ追従してスケーリングをするのかを踏まえて評価間隔を決定します。
4	評価対象メトリクスの許容範囲	評価対象メトリクスの許容範囲が狭いと、スケールするたびに閾値を超えてしまい、スケールアウト/スケールインを頻繁に繰り返してしまいます。そのため、適切な評価対象メトリクスの許容範囲を設定する必要があります。

(2) HMP-PCTO 固有の観点のチューニングの設計指針

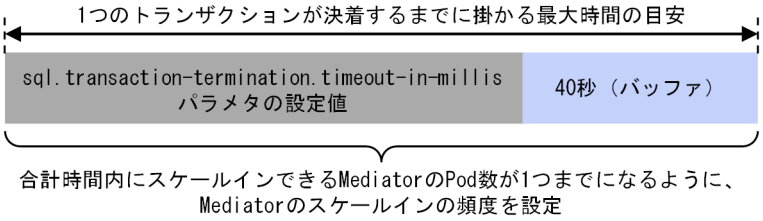
オートスケールをチューニングする上での HMP-PCTO 固有のチューニング観点を次の表に示します。

表 4-6 HMP-PCTO 固有のチューニング観点

項番	チューニング要素	チューニング観点
1	Mediator のスケールインの頻度	Mediator で短時間に多数のスケールインが発生すると、トランザクション決着に必要な Mediator 数を満たせないため、スケールアウトによって十分な Mediator 数になるまでトランザクションが決着しないおそれがあります。そのため、Mediator はスケールインの頻度を制限する必要があります。 具体的には、1 つのトランザクションが決着するまでにスケールインする Mediator 数を 1 つまでに制限する必要があります。1 つのトランザクションが決着するまでに掛かる最大時間の目安は、SQL-Participant の sql.transaction-termination.timeout-in-millis パラメタの設定値＋ 40 秒です。

Mediator のスケールインの頻度について、次の図に示します。

図 4-5 Mediator のスケールインの頻度



4.4.2 評価対象メトリクスの選定

HMP-PCTO では、オートスケールでのメトリクスの収集方法によって、評価対象として利用できるメトリクスが異なります。

(1) Metrics Server でメトリクスを収集する場合

Metrics Server でメトリクスを収集する場合に利用できるメトリクスを次に示します。なお、Pod のリソースは、Pod 内で動作しているコンテナすべてで利用しているリソースの合計値です。

- Pod の CPU リソース

  HMP-PCTO のコンテナの CPU リソースだけに絞り込みたい場合は、「(2)　Prometheus でメトリクスを収集する場合」の process_cpu_usage メトリクスの利用を検討してください。

- Pod のメモリリソース

  Pod のメモリリソースは、java プロセスのガベージコレクションによって変動するため、オートスケールの評価対象としても安定しないおそれがあります。

(2)　Prometheus でメトリクスを収集する場合

Prometheus でメトリクスを収集する場合に利用できるメトリクスは、「(3)　出力内容」を参照してください。利用できるメトリクスのうち、HMP-PCTO の評価対象メトリクスとして推奨するメトリクスを次の表に示します。

表 4-7　HMP-PCTO の評価対象メトリクスとして推奨するメトリクス

メトリクス名	メトリクスの説明	用途	対象の Kubernetes アプリケーション	クエリの例
process_cpu_usage	Java VM プロセスの CPU 使用率	HMP-PCTO の Kubernetes アプリケーションのコンテナの最近の CPU 利用率を評価対象メトリクスとしたい場合に利用します。	• Orchestrator-Service • Relay-Service • Entity-Service(SQL) • Orchestrator-Service(SQL) • Alternate-Service • Alternate-Service(SQL) • Mediator	SUM(process_cpu_usage{label_app_kubernetes_io_name="orchestrator"})※1
hmppcto_participant_transactions_inProcess_operations	処理中のトランザクションブランチ数	Pod 当たりのトランザクションの処理件数が一定値を超えると性能が低下するなどの理由で、Pod 当たりのトランザクションの処理件数を一定の範囲内に収めたいケースで評価対象メトリクスとして利用します。※2	• Entity-Service(SQL) • Orchestrator-Service(SQL) • Alternate-Service(SQL)	SUM(hmppcto_participant_transactions_inProcess_operations{id=~"^sql-participant-.*", status=~"NONE|PREPARING|COMMIT_VOTE|ROLLBACK_VOTE|COMMITTING|ROLLING_BACK|READ_ONLY|COMMITTED|ROLLED_BACK|ROLLED_BACK_BY_RM"})
hmppcto_executor_active_threads	アクティブにタスクを実行しているスレッドのおおよその数	過去のメトリクスデータから、実行中のスレッド数が大きくなった場合に性能低下が見られるような傾向があるとき、評価対象メト	• Orchestrator-Service • Entity-Service(SQL) • Orchestrator-Service(SQL)	SUM(hmppcto_executor_active_threads{label_app_kubernetes_io_name="sql-participant"})※1

メトリクス名	メトリクスの説明	用途	対象の Kubernetes アプリケーション	クエリの例
		リクスとして利用します。	• Alternate-Service • Alternate-Service(SQL) • Mediator	
hmppcto_executor_queued_tasks	実行待ちのタスクのおおよその数	負荷がスレッドの処理能力を超えてタスクがキューに滞留し、性能低下する傾向があるとき、評価対象メトリクスとして利用します。※3		SUM(hmppcto_executor_queued_tasks{label_app_kubernetes_io_name="mediator"})※1

注※1

label_app_kubernetes_io_name ラベルは、Kubernetes マニフェストの.spec.template.metadata.labels の app.kubernetes.io/name に指定した値です。

Google Cloud Platform のマネージド Prometheus でメトリクスを収集する場合は、「job」ラベルに読み替えてください。

Red Hat OpenShift Container Platform モニタリングの Prometheus でメトリクスを収集する場合は、「service」ラベルに読み替えてください。

注※2

評価対象メトリクスとして利用するには、トランザクションが処理中であることを示すステータスに絞り込む必要があります。トランザクションが処理中であることを示すステータスに絞り込むクエリの例は「クエリの例」列を参照してください。

注※3

負荷がスレッドの処理能力を超えていない場合は、実行待ちタスクにならないため、それを踏まえて閾値をチューニングしてください。

4.4.3　チューニング方法

「4.4.1　設計指針」で示したチューニング要素およびチューニング観点について、具体的なチューニング方法を説明します。

(1)　評価対象メトリクスの閾値のチューニング方法

次の流れでチューニングします。

1. チューニング対象の Kubernetes アプリケーションの過去のメトリクスデータを参照して、通常の負荷状態やピーク時の負荷の傾向を把握した上で、閾値を検討します。
2. 検討した閾値を基に、HPA の Kubernetes マニフェストの設定値を修正します。

ただし、オートスケールのメトリクスの収集方法によって、チューニング対象の Kubernetes アプリケーションの過去のメトリクスデータを参照する方法、および HPA の Kubernetes マニフェストの設定項目が異なります。メトリクスの収集方法ごとの差異を次の表に示します。

表 4-8　メトリクスの収集方法ごとの手順の差異

オートスケールのメトリクスの収集方法	チューニング対象の Kubernetes アプリケーションの過去のメトリクスデータを参照する方法	HPA の Kubernetes マニフェストの設定項目
Metrics Server	kubectl top コマンドを利用して、Pod の負荷状況を監視します。	spec.metrics（「4.2.4　HPA の Kubernetes マニフェストの作成」参照）
Prometheus	スケーリングの閾値で利用するクエリを Prometheus の Web UI で確認します。Prometheus の Web UI の確認方法の詳細については、「11.5.3　メトリクス」を参照してください。	spec.triggers（「4.3.6　HPA の Kubernetes マニフェストの作成」参照）

(2)　評価対象メトリクスの評価間隔のチューニング方法

次の流れでチューニングします。

1. チューニング対象の Kubernetes アプリケーションの過去の Pod 数の増減を確認して、負荷の変動とスケーリングの反応速度の関係を把握した上で、評価間隔を検討します。

   チューニング対象の Kubernetes アプリケーションの過去の Pod 数の増減を確認する方法は、「4.4.4 オートスケールによる Pod 数の増減を確認する方法」を参照してください。

2. 検討した閾値を基に、kube-controller-manager の設定値の horizontal-pod-autoscaler-sync-period を修正します。

   Metrics Server でメトリクスを収集する場合の設定方法の詳細については、「4.2.5　HPA の設定のカスタマイズ」を参照してください。Prometheus でメトリクスを収集する場合の設定方法の詳細については、「4.3.7　HPA の設定のカスタマイズ」を参照してください。

3. Prometheus でメトリクスを収集する場合は、Prometheus がメトリクス対象から情報を収集する頻度を設定します。目安として、評価対象メトリクスの評価間隔の半分の時間を設定します。Prometheus の収集頻度は prometheus.scrape.interval パラメタに設定してください。

   prometheus.scrape.interval パラメタの詳細については、「9.3.7　Prometheus」を参照してください。

(3)　評価対象メトリクスの許容範囲のチューニング方法

次の流れでチューニングします。

1. チューニング対象の Kubernetes アプリケーションの過去の Pod 数の増減を確認して、頻繁にスケールイン/スケールアウトを繰り返していないかを確認します。

   チューニング対象の Kubernetes アプリケーションの過去の Pod 数の増減を確認する方法は、「4.4.4 オートスケールによる Pod 数の増減を確認する方法」を参照してください。

2. 1.で問題がある場合、kube-controller-manager の設定値の horizontal-pod-autoscaler-tolerance を修正します。

   Metrics Server でメトリクスを収集する場合の設定方法の詳細については、「4.2.5　HPA の設定のカスタマイズ」を参照してください。Prometheus でメトリクスを収集する場合の設定方法の詳細については、「4.3.7　HPA の設定のカスタマイズ」を参照してください。

(4)　Mediator のスケールインの頻度のチューニング方法

次の流れでチューニングします。

1. チューニング対象の Kubernetes アプリケーションの過去の Pod 数の増減を確認し、「4.4.1　設計指針」で説明した「トランザクションが決着するまでの最大時間の目安」よりも早いタイミングで同時に 2 台以上の Mediator の Pod がスケールインしていないかを確認します。

   確認方法については、「4.4.4　オートスケールによる Pod 数の増減を確認する方法」を参照してください。

2. 1.で問題がある場合、HPA の Kubernetes マニフェストの設定値を修正します。

   オートスケールのメトリクスの収集方法によって、設定項目が異なります。オートスケールのメトリクスの収集方法ごとの設定項目の差異を次の表に示します。

   表 4–9　オートスケールのメトリクスの収集方法ごとの設定項目の差異

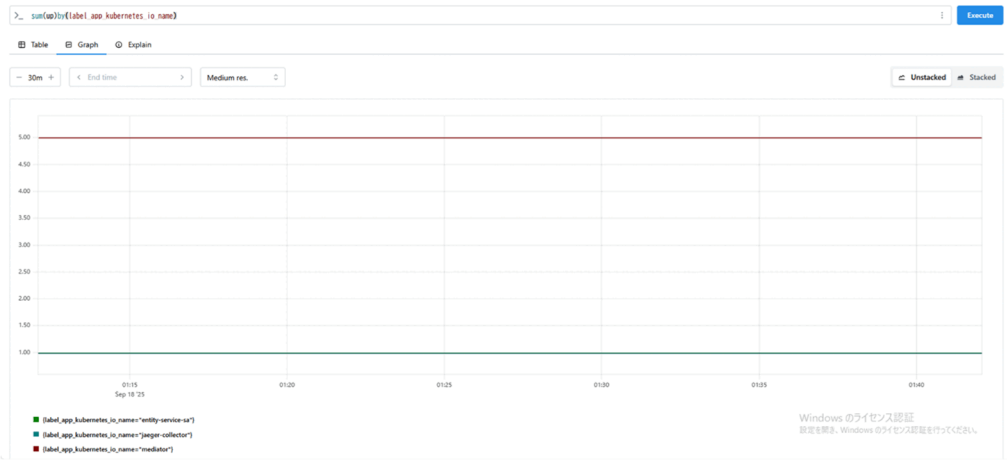
オートスケールのメトリクスの収集方法	HPA の Kubernetes マニフェストの設定項目
Metrics Server	spec.behavior.scaleDown.policies（「4.2.4　HPA の Kubernetes マニフェストの作成」参照）
Prometheus	spec.advanced.horizontalPodAutoscalerConfig.behavior.scaleDown.policies（「4.3.6　HPA の Kubernetes マニフェストの作成」参照）

4.4.4　オートスケールによる Pod 数の増減を確認する方法

オートスケールによる Pod 数の増減に関する情報は、メトリクスデータから確認できます。メトリクスデータの確認方法の詳細については、「11.5.3　メトリクス」を参照してください。

Prometheus の Web UI で「sum(up)by(label_app_kubernetes_io_name)」のクエリを入力し、Execute ボタンを押すことで Graph タブに次の図に示す状態で表示されます。

図 4-6　Prometheus の Web UI



4.4.5　チューニング例

オートスケールを使用する際に、どのような考えに基づいて設定値をチューニングするかの具体例を示します。具体例では、「1.　評価対象とするメトリクスの選定」、「2.　評価対象のメトリクスの閾値の検討」、「3.　評価対象のメトリクスの評価間隔の検討」の順にチューニングする例を説明します。

オートスケール設定前の状態

この例の前提条件として、オートスケールの設定前の状態を次に示します。

- 業務要件として、1 トランザクション当たりの処理時間は 5 秒を上限とする。
- 業務要件として、高負荷状態を許容できる時間は、30 秒までとする。
- 平常時の負荷状況ではユーザ要件を満たせている。
- ピーク時の負荷状況では 1 トランザクション当たりの処理時間 5 秒を超過してしまい、ユーザ要件を満たせていない。
- Pod が起動を開始してからトランザクションが受け付けられるようになるまでに必要な時間は 20 秒である。

オートスケールを適用する前の負荷テストなどでこれらの前提が分かっている場合、オートスケールを使用して、ピーク時でも 1 トランザクション当たりの処理時間 5 秒を実現することを考えます。

1. 評価対象とするメトリクスの選定

過去のメトリクス情報を参照してトランザクションの処理時間と相関のあるメトリクスを調査し、評価対象にするメトリクスを選定します。

調査の結果、次のことが判明したと仮定します。

- ピーク時に 20 秒程度の短い時間で急激に CPU 利用率が増加し、CPU 利用率が 80%を超過するとトランザクションの処理時間が大きく増加する。

これによって、評価対象のメトリクスとして CPU 利用率を使用します。

2. 評価対象のメトリクスの閾値の検討

調査の結果、CPU 利用率が 80%を超過するとトランザクションの平均処理時間が大きく増加することが分かっているため、CPU 利用率 80%を基準に考えます。スケールアウトの場合、ユーザ要件を満たせないタイミング（80%を超過してからスケールアウトによって Pod が起動し安定稼働するまでの間）があるため、10%のバッファを持たせて 70%を基準とします。

3. 評価対象のメトリクスの評価間隔の検討

業務要件から、高負荷状態を許容できる時間を 30 秒までとしたい場合に関して考えます。Pod が起動を開始してからトランザクションが受け付けられるようになるまでの時間が 20 秒のため、最低でも 10 秒間に 1 回、メトリクスの値を評価する必要があります。そのため、メトリクスの評価間隔を 10 秒に設定します。これに伴い、Prometheus がメトリクス対象から情報を取得する頻度を 5 秒に設定します。

これらの手順を踏まえると、次の設定になります。

- kube-controller-manager の設定

```
kube-controller-manager --horizontal-pod-autoscaler-sync-period 10
```

HPA によるメトリクスの評価間隔を 10 秒に変更します（「3.　評価対象のメトリクスの評価間隔の検討」の設定）。

- HPA の Kubernetes マニフェストの記述例

```
apiVersion: autoscaling/v2
kind: HorizontalPodAutoscaler
metadata:
  name: metrics-server-hpa
  namespace: my-namespace
spec:
  （省略）
  metrics:
  - type: Resource
    resource:
      name: cpu                 …1
      target:
        type: Utilization
        averageUtilization: 70 …2
  （以下、省略）
```

＜説明＞

1. 評価対象のメトリクスに CPU を設定します。
2. メトリクスの閾値は平均 CPU 利用率を 70%に設定します（「1.　評価対象とするメトリクスの選定」および「2.　評価対象のメトリクスの閾値の検討」）。

4.5　オートスケールの障害対策

オートスケールのセットアップ後に、HMP-PCTO の Pod が想定どおりにスケーリングしないときの対策について説明します。

4.5.1　Metrics Server でメトリクスを監視する場合の障害対策

Metrics Server でメトリクスを監視する場合に、HMP-PCTO の Pod が想定どおりにスケーリングしないときの主な原因と対策を次の表に示します。

表 4-10　オンプレミス環境の場合の、オートスケールの障害の主な原因と対策（Metrics Server でメトリクスを監視するとき）

項番	現象	主な原因	対策
1	負荷に応じてスケールアウト/スケールインしない※	Metrics Server に障害が発生している	Metrics Server の公式ドキュメントを参照して Metrics Server の障害について対策してください。そのあと、「4.2.3　Metrics Server のデプロイ（オンプレミス環境限定）」を再度実施してください。
2		HPA に障害が発生している	HPA の公式ドキュメントを参照して HPA の障害について対策してください。そのあと、「4.2.6　HPA のデプロイ」を再度実施してください。
3		HPA の Kubernetes マニフェストでのスケーリングに関する設定が誤っている	• 「4.2.4　HPA の Kubernetes マニフェストの作成」の設定内容を見直し、正しく設定されていることを確認してください。 そのあと、「4.2.6　HPA のデプロイ」を再度実施してください。 • 「4.4　オートスケールのパラメタの設計とチューニング」を参照してスケーリングに関する設定をチューニングしてください。 そのあと、「4.2.6　HPA のデプロイ」を再度実施してください。

注※

Prometheus で「sum(up)by(label_app_kubernetes_io_name)」クエリを実行すると、オートスケール対象の各 Kubernetes アプリケーションの Pod 数の推移を確認できます。Prometheus のクエリの実行方法は、「(a)　Prometheus サーバの Web UI による参照」を参照してください。

なお、label_app_kubernetes_io_name ラベルは、Kubernetes マニフェストの.spec.template.metadata.labels の app.kubernetes.io/name に指定した値です。

表 4-11　Microsoft Azure 環境の場合の、オートスケールの障害の主な原因と対策（Metrics Server でメトリクスを監視するとき）

項番	現象	主な原因	対策
1	負荷に応じてスケールアウト/スケールインしない※	Metrics Server に障害が発生している	Microsoft Azure の公式ドキュメント、および Metrics Server の公式ドキュメントを参照して Metrics Server の障害について対策してください。
2		HPA に障害が発生している	Microsoft Azure の公式ドキュメント、および HPA の公式ドキュメントを参照して HPA の障害について対策してください。そのあと、「4.2.6　HPA のデプロイ」を再度実施してください。
3		HPA の Kubernetes マニフェストでのスケーリングに関する設定が誤っている	• 「4.2.4　HPA の Kubernetes マニフェストの作成」の設定内容を見直し、正しく設定されていることを確認してください。 そのあと、「4.2.6　HPA のデプロイ」を再度実施してください。 • 「4.4　オートスケールのパラメタの設計とチューニング」を参照してスケーリングに関する設定をチューニングしてください。 そのあと、「4.2.6　HPA のデプロイ」を再度実施してください。

注※

Prometheus で「sum(up)by(label_app_kubernetes_io_name)」クエリを実行すると、オートスケール対象の各 Kubernetes アプリケーションの Pod 数の推移を確認できます。Prometheus のクエリの実行方法は、「(a)　Prometheus サーバの Web UI による参照」を参照してください。

なお、label_app_kubernetes_io_name ラベルは、Kubernetes マニフェストの.spec.template.metadata.labels の app.kubernetes.io/name に指定した値です。

表 4-12　Google Cloud Platform 環境の場合の、オートスケールの障害の主な原因と対策（Metrics Server でメトリクスを監視するとき）

項番	現象	主な原因	対策
1	負荷に応じてスケールアウト/スケールインしない※	Metrics Server に障害が発生している	Google Cloud Platform の公式ドキュメント、および Metrics Server の公式ドキュメントを参照して Metrics Server の障害について対策してください。
2		HPA に障害が発生している	Google Cloud Platform の公式ドキュメント、および HPA の公式ドキュメントを参照して HPA の障害について対策してください。そのあと、「4.2.6　HPA のデプロイ」を再度実施してください。
3		HPA の Kubernetes マニフェストでのスケーリングに関する設定が誤っている	• 「4.2.4　HPA の Kubernetes マニフェストの作成」の設定内容を見直し、正しく設定されていることを確認してください。 そのあと、「4.2.6　HPA のデプロイ」を再度実施してください。

項番	現象	主な原因	対策
			• 「4.4　オートスケールのパラメタの設計とチューニング」を参照してスケーリングに関する設定をチューニングしてください。 そのあと、「4.2.6　HPA のデプロイ」を再度実施してください。

注※

Prometheus で「sum(up)by(job)」クエリを実行すると、オートスケール対象の各 Kubernetes アプリケーションの Pod 数の推移を確認できます。Prometheus のクエリの実行方法は、「(a)　Prometheus サーバの Web UI による参照」を参照してください。

表 4-13　Amazon Web Services 環境の場合の、オートスケールの障害の主な原因と対策（Metrics Server でメトリクスを監視するとき）

項番	現象	主な原因	対策
1	負荷に応じてスケールアウト/スケールインしない※	Metrics Server に障害が発生している	Amazon Web Services の公式ドキュメント、および Metrics Server の公式ドキュメントを参照して Metrics Server の障害について対策してください。
2		HPA に障害が発生している	Amazon Web Services の公式ドキュメント、および HPA の公式ドキュメントを参照して HPA の障害について対策してください。そのあと、「4.2.6　HPA のデプロイ」を再度実施してください。
3		HPA の Kubernetes マニフェストでのスケーリングに関する設定が誤っている	• 「4.2.4　HPA の Kubernetes マニフェストの作成」の設定内容を見直し、正しく設定されていることを確認してください。 そのあと、「4.2.6　HPA のデプロイ」を再度実施してください。 • 「4.4　オートスケールのパラメタの設計とチューニング」を参照してスケーリングに関する設定をチューニングしてください。 そのあと、「4.2.6　HPA のデプロイ」を再度実施してください。

注※

Prometheus で「sum(up)by(label_app_kubernetes_io_name)」クエリを実行すると、オートスケール対象の各 Kubernetes アプリケーションの Pod 数の推移を確認できます。Prometheus のクエリの実行方法は、「(a)　Prometheus サーバの Web UI による参照」を参照してください。

表 4-14　Red Hat OpenShift Container Platform 環境の場合の、オートスケールの障害の主な原因と対策（Metrics Server でメトリクスを監視するとき）

項番	現象	主な原因	対策
1	負荷に応じてスケールアウト/スケールインしない※	Metrics Server に障害が発生している	Red Hat OpenShift Container Platform の公式ドキュメント、および Metrics Server の公式ドキュメントを参照して Metrics Server の障害について対策してください。

項番	現象	主な原因	対策
2		HPA に障害が発生している	Red Hat OpenShift Container Platform の公式ドキュメント、および HPA の公式ドキュメントを参照して HPA の障害について対策してください。そのあと、「4.2.6 HPA のデプロイ」を再度実施してください。
3		HPA の Kubernetes マニフェストでのスケーリングに関する設定が誤っている	• 「4.2.4　HPA の Kubernetes マニフェストの作成」の設定内容を見直し、正しく設定されていることを確認してください。 そのあと、「4.2.6　HPA のデプロイ」を再度実施してください。 • 「4.4　オートスケールのパラメタの設計とチューニング」を参照してスケーリングに関する設定をチューニングしてください。 そのあと、「4.2.6　HPA のデプロイ」を再度実施してください。

注※

Prometheus で「sum(up)by(service)」クエリを実行すると、オートスケール対象の各 Kubernetes アプリケーションの Pod 数の推移を確認できます。Prometheus のクエリの実行方法は、「(a)　Prometheus サーバの Web UI による参照」を参照してください。

4.5.2　Prometheus でメトリクスを監視する場合の障害対策

Prometheus でメトリクスを監視する場合に、HMP-PCTO の Pod が想定どおりにスケーリングしないときの主な原因と対策を次の表に示します。

表 4-15　オンプレミス環境の場合の、オートスケールの障害の主な原因と対策（Prometheus でメトリクスを監視するとき）

項番	現象	主な原因	対策
1	負荷に応じてスケールアウト/スケールインしない※	KEDA に障害が発生している	KEDA の公式ドキュメントを参照して KEDA の障害について対策してください。そのあと、「4.3.2　KEDA のインストールとデプロイ（オンプレミス環境または Amazon Web Services 環境限定）」を再度実施してください。
2		Prometheus に障害が発生している	「(3)　メトリクス」を参照して Prometheus の障害について対策してください。
3		HPA に障害が発生している	HPA の公式ドキュメントを参照して HPA の障害について対策してください。そのあと、「4.3.8　HPA のデプロイ」を再度実施してください。
4		HPA の Kubernetes マニフェストでのスケーリングに関する設定が誤っている	• 「4.3.6　HPA の Kubernetes マニフェストの作成」の設定内容を見直し、正しく設定されていることを確認してください。 そのあと、「4.3.8　HPA のデプロイ」を再度実施してください。

項番	現象	主な原因	対策
			• 「4.4　オートスケールのパラメタの設計とチューニング」を参照してスケーリングに関する設定をチューニングしてください。 そのあと、「4.3.8　HPA のデプロイ」を再度実施してください。

注※

Prometheus で「sum(up)by(label_app_kubernetes_io_name)」クエリを実行すると、オートスケール対象の各 Kubernetes アプリケーションの Pod 数の推移を確認できます。Prometheus のクエリの実行方法は、「(a)　Prometheus サーバの Web UI による参照」を参照してください。

なお、label_app_kubernetes_io_name ラベルは、Kubernetes マニフェストの.spec.template.metadata.labels の app.kubernetes.io/name に指定した値です。

表 4-16　Microsoft Azure 環境の場合の、オートスケールの障害の主な原因と対策 (Prometheus でメトリクスを監視するとき)

項番	現象	主な原因	対策
1	負荷に応じてスケールアウト/スケールインしない※	KEDA が Prometheus との通信に失敗する	Microsoft Azure の公式ドキュメントを参照して、KEDA アドオンのインストールを実施してください。そのあと、「4.3.6　HPA の Kubernetes マニフェストの作成」を再度実施してください。
2		KEDA に障害が発生している	Microsoft Azure の公式ドキュメントおよび KEDA の公式ドキュメントを参照して、KEDA の障害について対策してください。そのあと、「4.3.3　KEDA アドオンのインストール（Microsoft Azure 環境限定)」を再度実施してください。
3		Prometheus に障害が発生している	「(3)　メトリクス」を参照して Prometheus の障害について対策してください。
4		HPA に障害が発生している	Microsoft Azure の公式ドキュメントおよび HPA の公式ドキュメントを参照して、HPA の障害について対策してください。そのあと、「4.3.8　HPA のデプロイ」を再度実施してください。
5		HPA の Kubernetes マニフェストでのスケーリングに関する設定が誤っている	• 「4.3.6　HPA の Kubernetes マニフェストの作成」の設定内容を見直し、正しく設定されていることを確認してください。 そのあと、「4.3.8　HPA のデプロイ」を再度実施してください。 • 「4.4　オートスケールのパラメタの設計とチューニング」を参照してスケーリングに関する設定をチューニングしてください。 そのあと、「4.3.8　HPA のデプロイ」を再度実施してください。

注※

Prometheus で「sum(up)by(label_app_kubernetes_io_name)」クエリを実行すると、オートスケール対象の各 Kubernetes アプリケーションの Pod 数の推移を確認できます。Prometheus のクエリの実行方法は、「(a)　Prometheus サーバの Web UI による参照」を参照してください。

なお、label_app_kubernetes_io_name ラベルは、Kubernetes マニフェストの.spec.template.metadata.labels の app.kubernetes.io/name に指定した値です。

表 4-17　Google Cloud Platform 環境の場合の、オートスケールの障害の主な原因と対策 (Prometheus でメトリクスを監視するとき)

項番	現象	主な原因	対策
1	負荷に応じてスケールアウト/スケールインしない※	カスタム指標 Stackdriver アダプタが Prometheus との通信に失敗する	Google Cloud Platform の公式ドキュメントを参照してカスタム指標 Stackdriver アダプタをインストールしてください。そのあと、「(2)　Google Cloud Platform 環境または Amazon Web Services 環境の場合」を再度実施してください。
2		Prometheus に障害が発生している	「(3)　メトリクス」を参照して Prometheus の障害について対策してください。
3		HPA に障害が発生している	Google Cloud Platform の公式ドキュメント、および HPA の公式ドキュメントを参照して HPA の障害について対策してください。そのあと、「4.3.8　HPA のデプロイ」を再度実施してください。
4		HPA の Kubernetes マニフェストでのスケーリングに関する設定が誤っている	• 「4.3.6　HPA の Kubernetes マニフェストの作成」の設定内容を見直し、正しく設定されていることを確認してください。 そのあと、「4.3.8　HPA のデプロイ」を再度実施してください。 • 「4.4　オートスケールのパラメタの設計とチューニング」を参照してスケーリングに関する設定をチューニングしてください。 そのあと、「4.3.8　HPA のデプロイ」を再度実施してください。

注※

Prometheus で「sum(up)by(job)」クエリを実行すると、オートスケール対象の各 Kubernetes アプリケーションの Pod 数の推移を確認できます。Prometheus のクエリの実行方法は、「(a)　Prometheus サーバの Web UI による参照」を参照してください。

表 4-18　Amazon Web Services 環境の場合の、オートスケールの障害の主な原因と対策 (Prometheus でメトリクスを監視するとき)

項番	現象	主な原因	対策
1	負荷に応じてスケールアウト/スケールインしない※	KEDA に障害が発生している	KEDA の公式ドキュメントを参照して KEDA の障害について対策してください。そのあと、「4.3.2　KEDA のインストールとデプロイ（オンプレミス環境または Amazon Web Services 環境限定)」を再度実施してください。

項番	現象	主な原因	対策
2		Prometheus に障害が発生している	「(3)　メトリクス」を参照して Prometheus の障害について対策してください。
3		HPA に障害が発生している	Amazon Web Services の公式ドキュメント、および HPA の公式ドキュメントを参照して HPA の障害について対策してください。そのあと、「4.3.8　HPA のデプロイ」を再度実施してください。
4		HPA の Kubernetes マニフェストでのスケーリングに関する設定が誤っている	• 「4.3.6　HPA の Kubernetes マニフェストの作成」の設定内容を見直し、正しく設定されていることを確認してください。 そのあと、「4.3.8　HPA のデプロイ」を再度実施してください。 • 「4.4　オートスケールのパラメタの設計とチューニング」を参照してスケーリングに関する設定をチューニングしてください。 そのあと、「4.3.8　HPA のデプロイ」を再度実施してください。

注※

Prometheus で「sum(up)by(label_app_kubernetes_io_name)」クエリを実行すると、オートスケール対象の各 Kubernetes アプリケーションの Pod 数の推移を確認できます。Prometheus のクエリの実行方法は、「(a)　Prometheus サーバの Web UI による参照」を参照してください。

表 4-19　Red Hat OpenShift Container Platform 環境の場合の、オートスケールの障害の主な原因と対策（Prometheus でメトリクスを監視するとき）

項番	現象	主な原因	対策
1	負荷に応じてスケールアウト/スケールインしない※	Custom Metrics Autoscaler が Prometheus との通信に失敗する	ServiceAccount に Prometheus（Thanos）へのアクセス権限が不足しているおそれがあります。view 権限の付与や、authModes、TriggerAuthentication などの認証情報を確認してください。詳細は Red Hat OpenShift Container Platform の公式ドキュメントを参照してください。
2		Custom Metrics Autoscaler に障害が発生している	Red Hat OpenShift Container Platform の公式ドキュメントを参照して障害について対策してください。または、Custom Metrics Autoscaler は KEDA をベースとしているため、KEDA の公式ドキュメントを参照して障害について対策してください。そのあと、「4.3.5　Custom Metrics Autoscaler のインストール（Red Hat OpenShift Container Platform 環境限定）」を再度実施してください。
3		Prometheus に障害が発生している	「(3)　メトリクス」を参照して Prometheus の障害について対策してください。
4		HPA に障害が発生している	Red Hat OpenShift Container Platform の公式ドキュメント、および HPA の公式ドキュメントを参照して HPA の障害について対策してください。そのあと、「4.3.8 HPA のデプロイ」を再度実施してください。

項番	現象	主な原因	対策
5		HPA の Kubernetes マニフェストでのスケーリングに関する設定が誤っている	• 「4.3.6　HPA の Kubernetes マニフェストの作成」の設定内容を見直し、正しく設定されていることを確認してください。 そのあと、「4.3.8　HPA のデプロイ」を再度実施してください。 • 「4.4　オートスケールのパラメタの設計とチューニング」を参照してスケーリングに関する設定をチューニングしてください。 そのあと、「4.3.8　HPA のデプロイ」を再度実施してください。

注※

Prometheus で「sum(up)by(service)」クエリを実行すると、オートスケール対象の各 Kubernetes アプリケーションの Pod 数の推移を確認できます。Prometheus のクエリの実行方法は、「(a)　Prometheus サーバの Web UI による参照」を参照してください。

5 TP1-Converter の運用（通常版かつ TP1-Bridge 限定）

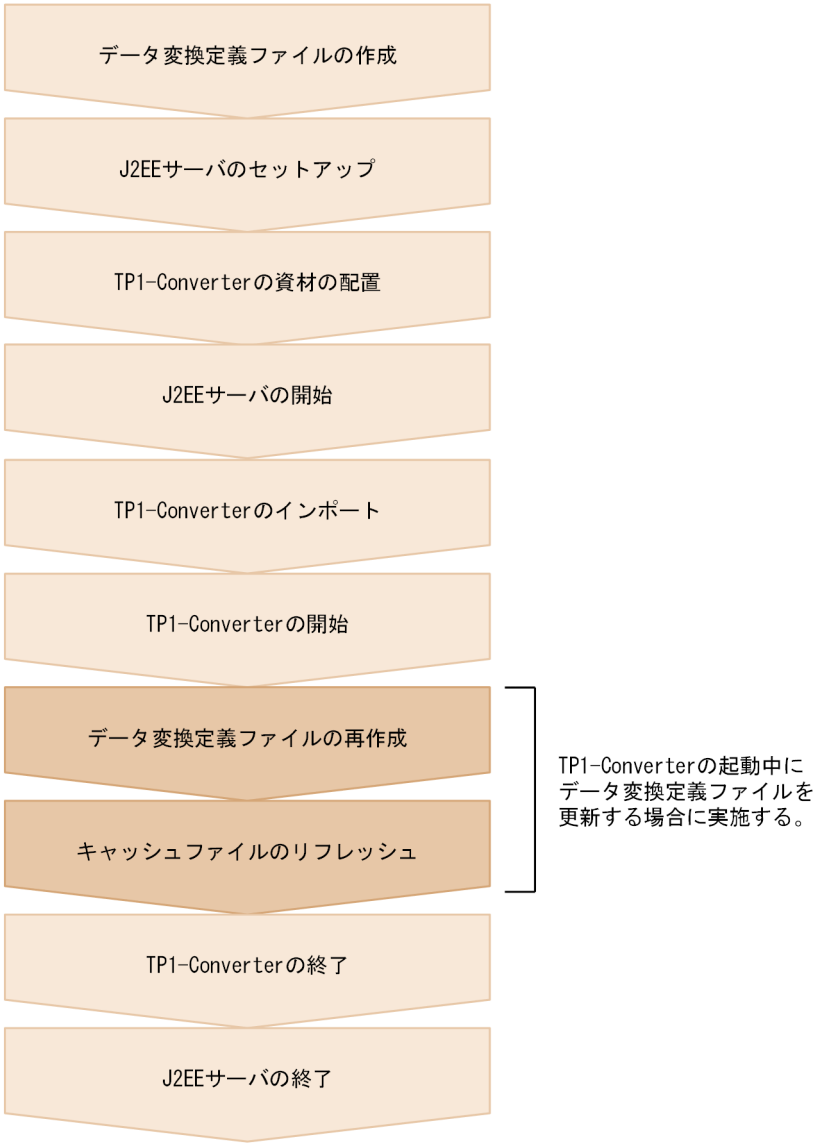
5.1　TP1-Converter の運用の流れ（通常版かつ TP1-Bridge 限定）

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 5-1　TP1-Converter を使用するための作業の流れ



5.2　データ変換定義ファイルの作成（通常版かつ TP1-Bridge 限定）

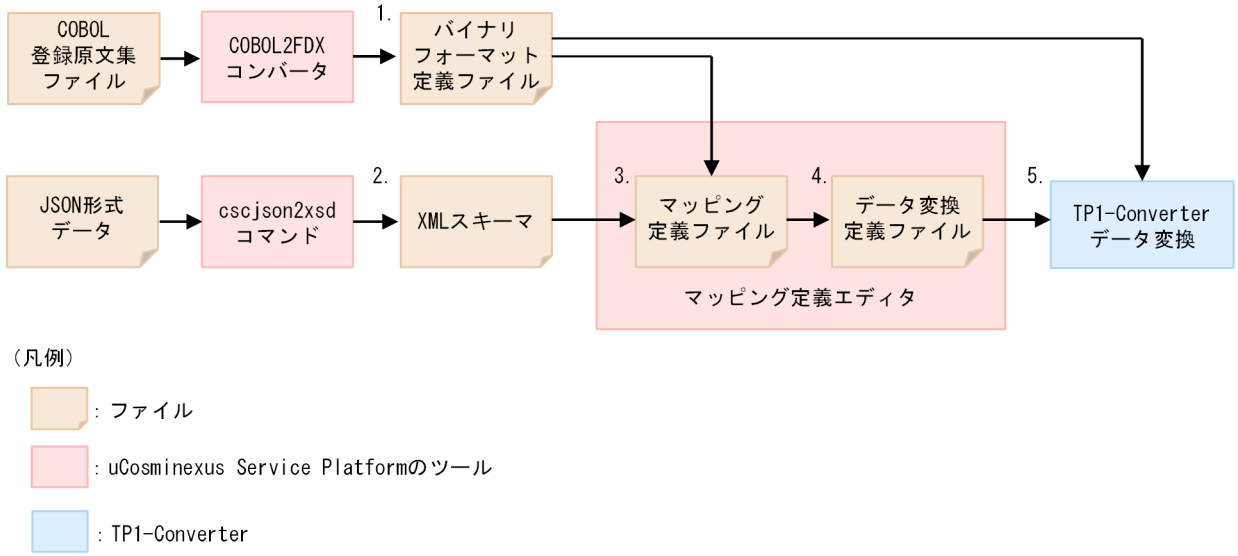
TP1-Converter のデータ変換で使用するデータ変換定義ファイルは、uCosminexus Service Architect の機能を使用して作成します。

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。

5.2.1　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 5-2　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 5-1　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	5.2.2　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	5.2.3　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	5.2.4　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	5.2.5　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。

5.2.2　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。

5.2.3 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-Module 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。

5.2.4 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。

5.2.5 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。

5.3　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。

5.4　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 5-2　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）	<Hitachi Microservices Platform - Bridge Option for OpenTP1 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）	<Hitachi Microservices Platform - Bridge Option for OpenTP1 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）	<Hitachi Microservices Platform - Bridge Option for OpenTP1 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	「5.2　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.directory の設定で変更できます。

5.5　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。

5.6 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。

5.7 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。

5.8　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、「5.4　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 5-3　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。

5.9 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。

5.10　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。

5.11　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。

5.12　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。

5.13 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

5.13.1 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。

5.13.2 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。

6 コンテナイメージの作成要件

6.1　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 for Spring Boot 導入あり
  - uCosminexus Application Runtime 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 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部は、共通のコンテナベースイメージをあらかじめ作成しておき、そのコンテナベースイメージを使用して作成します。例えば、HMP-PCTO のアプリケーションの各コンポーネントの多くは Java アプリケーションのため、各コンポーネントのコンテナイメージの前提プログラムとして Java ランタイムが必要です。このため、Java ランタイムを含んだ共通のコンテナベースイメージをあらかじめ作成しておき、そのコンテナベースイメージを使用して各コンポーネントのコンテナイメージを作成します。

HMP-PCTO のコンテナベースイメージの一覧を次の表に示します。これらのコンテナベースイメージは、「表 6-2　HMP-PCTO のコンテナイメージの一覧」のコンテナイメージを作成する際に使用します。

表 6-1　HMP-PCTO のコンテナベースイメージの一覧

コンテナベースイメージ	説明
uCosminexus Application Runtime for Spring Boot コンテナベースイメージ	通常版で uCosminexus Application Runtime for Spring Boot を導入するユーザ責務のコンテナイメージを作成する際に使用します。

コンテナベースイメージ	説明
JDK コンテナベースイメージ	トライアル版の場合、または通常版で uCosminexus Application Runtime for Spring Boot を導入しないユーザ責務のコンテナイメージを作成する際に使用します。

HMP-PCTO のコンテナイメージの一覧を次の表に示します。これらのコンテナイメージの一部は、「表 6-1　HMP-PCTO のコンテナベースイメージの一覧」のコンテナベースイメージを使用して作成します。

表 6-2　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		トライアル版	
			uCosminexus Application Runtime for Spring Boot 導入あり	uCosminexus Application Runtime for Spring Boot 導入なし		
Orchestrator-Service コンテナイメージ	○	−	○	○	○	−
Relay-Service コンテナイメージ	○	−	○	○	○	−
Entity-Service(SQL)コンテナイメージ	○	−	○	○	○	−
Entity-Service(TCC)コンテナイメージ	○	−	○	○	○	−
Orchestrator-Service(SQL)コンテナイメージ	○	−	○	○	○	−
Alternate-Service コンテナイメージ	○	−	○	○	○	−
Alternate-Service(SQL)コンテナイメージ	○	−	○	○	○	−
Mediator コンテナイメージ	−	○	○	−	○	−
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	−	○	−	○	○	−

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		トライアル版	
			uCosminexus Application Runtime for Spring Boot 導入あり	uCosminexus Application Runtime for Spring Boot 導入なし		
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。 コンテナイメージの詳細については、取扱説明書「EADS コンテナオーケストレーションツール対応機能」を参照してください。
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。 コンテナイメージの詳細については、取扱説明書「EADS コンテナオーケストレーションツール対応機能」を参照してください。

（凡例）

○：該当します。

−：該当しません。

6.2　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 6-3　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 • 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、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. 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 3. Containerfile の記述例を基に Containerfile を作成する 4. 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 • 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 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 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

6.3　各コンテナベースイメージの作成要件

各コンテナベースイメージの詳細について説明します。

コンテナベースイメージは、「6.4　各コンテナイメージの作成要件」のコンテナイメージを作成する際に使用します。コンテナベースイメージの作成に必要な第三者ソフトウェアの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、リリースノートを参照してください。

6.3.1　uCosminexus Application Runtime 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 for Spring Boot を導入する場合だけ作成します。

次の表に示す uCosminexus Application Runtime 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 6-4　uCosminexus Application Runtime 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	次のどちらかのコンテナイメージを使用します。 • Red Hat Universal Base Image 8※1 • Red Hat Universal Base Image 9※2
前提プログラム	■他社製の Java VM を使用する場合 • Java SE 21 準拠の JDK • uCosminexus Application Runtime for Spring Boot ■uCosminexus Application Runtime for Spring Boot および他社製の Java VM を使用する場合 • curl または curl-minimal • 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	次に示す環境変数を設定してください。そのほかの HMP-PCTO で必要な環境変数は、Kubernetes マニフェストで設定します。 • 環境変数名：LANG

項目	要件
	設定値：UTF-8 • 環境変数名：LC_ALL 設定値：UTF-8
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※1

UBI 8 の任意のベースイメージ（UBI 8 Minimal など）

注※2

UBI 9 の任意のベースイメージ（UBI 9 Minimal など）

6.3.2 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 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 6-5 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 次のどちらかのコンテナイメージを使用します。 • Red Hat Universal Base Image 8※1 • Red Hat Universal Base Image 9※2 ■トライアル版の場合 • Almalinux 9 Minimal 9.5
前提プログラム	• Java SE 21 準拠の JDK • curl または curl-minimal • 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）。

項目	要件
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設定してください。そのほかの HMP-PCTO で必要な環境変数は、Kubernetes マニフェストで設定します。 • 環境変数名：LANG 設定値：UTF-8 • 環境変数名：LC_ALL 設定値：UTF-8
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※1

UBI 8 の任意のベースイメージ（UBI 8 Minimal など）

注※2

UBI 9 の任意のベースイメージ（UBI 9 Minimal など）

6.4　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。

コンテナイメージの一部は、コンテナベースイメージを使用して作成します。コンテナベースイメージの作成要件については、「6.3　各コンテナベースイメージの作成要件」を参照してください。

コンテナイメージの作成に必要な第三者ソフトウェアの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、リリースノートを参照してください。

6.4.1　Orchestrator-Service コンテナイメージ

次の表に示す Orchestrator-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 6-6　Orchestrator-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 for Spring Boot 導入あり）の場合 • uCosminexus Application Runtime for Spring Boot コンテナベースイメージ（「6.3.1　uCosminexus Application Runtime 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 for Spring Boot 導入なし）、またはトライアル版の場合 • JDK コンテナベースイメージ（「6.3.2　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-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。 • 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。
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 for Spring Boot 導入あり）の場合

項目	要件
	• コマンドを「uCosminexus Application Runtime 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-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 for Spring Boot 導入なし）、または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-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	次に示す環境変数を設定してください。そのほかの HMP-PCTO で必要な環境変数は、Kubernetes マニフェストで設定します。 • 環境変数名：LANG 設定値：UTF-8 • 環境変数名：LC_ALL 設定値：UTF-8
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）。

6.4.2 Relay-Service コンテナイメージ

次の表に示す Relay-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 6-7 Relay-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 for Spring Boot 導入あり）の場合 • uCosminexus Application Runtime for Spring Boot コンテナベースイメージ（「6.3.1 uCosminexus Application Runtime 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 for Spring Boot 導入なし）、またはトライアル版の場合 • JDK コンテナベースイメージ（「6.3.2 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。

項目	要件
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 Relay-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。 • 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。
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 for Spring Boot 導入あり）の場合 • コマンドを「uCosminexus Application Runtime 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 Relay-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 for Spring Boot 導入なし）、または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 Relay-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	次に示す環境変数を設定してください。そのほかの HMP-PCTO で必要な環境変数は、Kubernetes マニフェストで設定します。 • 環境変数名：LANG 設定値：UTF-8 • 環境変数名：LC_ALL 設定値：UTF-8
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）。

6.4.3　Entity-Service(SQL)コンテナイメージ（SQL-Participant 限定）

次の表に示す Entity-Service(SQL)コンテナイメージの要件を満たすように作成します。

表 6-8　Entity-Service(SQL)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 for Spring Boot 導入あり）の場合 • uCosminexus Application Runtime for Spring Boot コンテナベースイメージ（「6.3.1　uCosminexus Application Runtime 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 for Spring Boot 導入なし）の場合、またはトライアル版の場合 • JDK コンテナベースイメージ（「6.3.2　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 • ncurses • tar • cpio • gzip • ncompress（コンテナベースイメージが RHEL8 の場合） ■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-Service(SQL)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：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。

項目	要件
	• 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 for Spring Boot 導入あり）の場合 • コマンドを「uCosminexus Application Runtime 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(SQL)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 for Spring Boot 導入なし）、または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(SQL)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設定してください。そのほかの HMP-PCTO で必要な環境変数は、Kubernetes マニフェストで設定します。 • 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 • 環境変数名：LANG 設定値：UTF-8 • 環境変数名：LC_ALL 設定値：UTF-8
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）。

6.4.4 Entity-Service(TCC)コンテナイメージ（TCC-Participant 限定）

次の表に示す Entity-Service(TCC)コンテナイメージの要件を満たすように作成します。

表 6-9　Entity-Service(TCC)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 for Spring Boot 導入あり）の場合 • uCosminexus Application Runtime for Spring Boot コンテナベースイメージ（「6.3.1　uCosminexus Application Runtime 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 for Spring Boot 導入なし）、またはトライアル版の場合 • JDK コンテナベースイメージ（「6.3.2　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 • ncurses • tar • cpio • gzip • ncompress（コンテナベースイメージが RHEL8 の場合） ■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-Service(TCC)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。

項目	要件
コンテナ起動コマンドライン	■通常版（uCosminexus Application Runtime for Spring Boot 導入あり）の場合 • コマンドを「uCosminexus Application Runtime 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(TCC)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 for Spring Boot 導入なし）、または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(TCC) 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設定してください。そのほかの HMP-PCTO で必要な環境変数は、Kubernetes マニフェストで設定します。 • 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 • 環境変数名：LANG 設定値：UTF-8 • 環境変数名：LC_ALL 設定値：UTF-8
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）。

6.4.5 Orchestrator-Service(SQL)コンテナイメージ（SQL-Participant 限定）

次の表に示す Orchestrator-Service(SQL)コンテナイメージの要件を満たすように作成します。

表 6–10 Orchestrator-Service(SQL)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 for Spring Boot 導入あり）の場合 • uCosminexus Application Runtime for Spring Boot コンテナベースイメージ（「6.3.1 uCosminexus Application Runtime 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）

項目	要件
	■通常版（uCosminexus Application Runtime for Spring Boot 導入なし）、またはトライアル版の場合 • JDK コンテナベースイメージ（「6.3.2　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 • ncurses • tar • cpio • gzip • ncompress（コンテナベースイメージが RHEL8 の場合） ■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 Orchestrator-Service(SQL)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。 • 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。

項目	要件
コンテナ起動コマンドライン	■通常版（uCosminexus Application Runtime for Spring Boot 導入あり）の場合 • コマンドを「uCosminexus Application Runtime 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-Service(SQL)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 for Spring Boot 導入なし）、または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-Service(SQL)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設定してください。そのほかの HMP-PCTO で必要な環境変数は、Kubernetes マニフェストで設定します。 • 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 • 環境変数名：LANG 設定値：UTF-8 • 環境変数名：LC_ALL 設定値：UTF-8
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）。

6.4.6 Alternate-Service コンテナイメージ

次の表に示す Alternate-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 6-11 Alternate-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 for Spring Boot 導入あり）の場合 • uCosminexus Application Runtime for Spring Boot コンテナベースイメージ（「6.3.1 uCosminexus Application Runtime 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）

項目	要件
	■通常版（uCosminexus Application Runtime for Spring Boot 導入なし）、またはトライアル版の場合 • JDK コンテナベースイメージ（「6.3.2　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 Alternate-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。 • 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。
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 for Spring Boot 導入あり）の場合 • コマンドを「uCosminexus Application Runtime 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 Alternate-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 for Spring Boot 導入なし）、または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 Alternate-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	次に示す環境変数を設定してください。そのほかの HMP-PCTO で必要な環境変数は、Kubernetes マニフェストで設定します。 • 環境変数名：LANG 設定値：UTF-8 • 環境変数名：LC_ALL 設定値：UTF-8
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。

項目	要件
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）。

6.4.7　Alternate-Service(SQL)コンテナイメージ（SQL-Participant 限定）

次の表に示す Alternate-Service(SQL)コンテナイメージの要件を満たすように作成します。

表 6-12　Alternate-Service(SQL)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 for Spring Boot 導入あり）の場合 • uCosminexus Application Runtime for Spring Boot コンテナベースイメージ（「6.3.1　uCosminexus Application Runtime 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 for Spring Boot 導入なし）、またはトライアル版の場合 • JDK コンテナベースイメージ（「6.3.2　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 • ncurses • tar • cpio • gzip • ncompress（コンテナベースイメージが RHEL8 の場合） ■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 Alternate-Service(SQL)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。 • 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。

項目	要件
	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 for Spring Boot 導入あり）の場合 • コマンドを「uCosminexus Application Runtime 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 Alternate-Service(SQL)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 for Spring Boot 導入なし）、または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 Alternate-Service(SQL)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設定してください。そのほかの HMP-PCTO で必要な環境変数は、Kubernetes マニフェストで設定します。 • 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 • 環境変数名：LANG 設定値：UTF-8 • 環境変数名：LC_ALL 設定値：UTF-8
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）。

6.4.8 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 6-13 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 次のどちらかのコンテナイメージを使用します。 • Red Hat Universal Base Image 8 Minimal • Red Hat Universal Base Image 9 Minimal ■トライアル版の場合 • Almalinux 9 Minimal 9.5
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 17 の uCosminexus Application Runtime with Java for Spring Boot • curl または curl-minimal • glibc • jq ■トライアル版の場合 • java-21-openjdk-headless • curl または curl-minimal • 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」 • 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」

項目	要件
	• 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 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。 • JDK17 ベースの日立 Java VM を使用するため、java コマンドについては uCosminexus Application Runtime with Java for Spring Boot のマニュアルを参照し、利用する日立 Java VM の java コマンド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	次に示す環境変数を設定してください。そのほかの HMP-PCTO で必要な環境変数は、Kubernetes マニフェストで設定します。 • 環境変数名：LANG 設定値：UTF-8 • 環境変数名：LC_ALL 設定値：UTF-8
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。

6.4.9　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 6-14　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	次のどちらかのコンテナイメージを使用します。 • Red Hat Universal Base Image 8 Minimal • Red Hat Universal Base Image 9 Minimal
前提プログラム	• Java SE 17 の uCosminexus Application Runtime with Java for Spring Boot • Hitachi Elastic Application Data Store Client for Java • ncurses • tar • cpio • gzip • ncompress（コンテナベースイメージが RHEL8 の場合）
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 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

項目	要件
	ド」、「環境変数 HMPPCTO_JAVA_OPTIONS の値」、「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）」を指定してください。 • JDK17 ベースの日立 Java VM を使用するため、java コマンドについては uCosminexus Application Runtime with Java for Spring Boot のマニュアルを参照し、利用する日立 Java VM の java コマンド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設定してください。そのほかの HMP-PCTO で必要な環境変数は、Kubernetes マニフェストで設定します。 • 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 • 環境変数名：LANG 設定値：UTF-8 • 環境変数名：LC_ALL 設定値：UTF-8
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。

6.4.10　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 6-15　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 次のどちらかのコンテナイメージを使用します。 • Red Hat Universal Base Image 8 Minimal • Red Hat Universal Base Image 9 Minimal ■トライアル版の場合 • Almalinux 9 Minimal 9.5
前提プログラム	• curl または curl-minimal • jq

項目	要件
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」
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）。

6.4.11　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 6–16　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 9 Minimal 9.5
前提プログラム	java-21-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」

項目	要件
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)。
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。

7

Kubernetes クラスタの作成要件

7.1 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、「6. 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、リリースノート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 7-1 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（「3.8 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	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（「3.7 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

7.2 Kubernetes アプリケーションのグループ定義

HMP-PCTO では、ある業務システムのトランザクションを実現するために相互に通信・連携する Kubernetes アプリケーション群を、同一の「アプリケーショングループ」として定義する必要があります。これは、HMP-PCTO が Kubernetes と連携し Pod 間の通信を制御するためです。

アプリケーショングループの定義方法を次に示します。

1. ある 1 つの業務システムのトランザクションを実現するために構成した一連の Kubernetes アプリケーション群を、1 つのアプリケーショングループとして定義します。
2. Kubernetes クラスタ上に複数の業務システムがある場合、アプリケーショングループを次のとおり定義します。
   - 複数の業務システム間で同一の Kubernetes アプリケーションを 1 つも共有していない場合、各業務システムの Kubernetes アプリケーション群を、それぞれ別のアプリケーショングループとして定義します。
   - 複数の業務システム間で同一の Kubernetes アプリケーションを 1 つ以上共有している場合、それらの業務システムの Kubernetes アプリケーション群を、1 つのアプリケーショングループとしてまとめて定義します。

アプリケーショングループの定義例を次に示します。

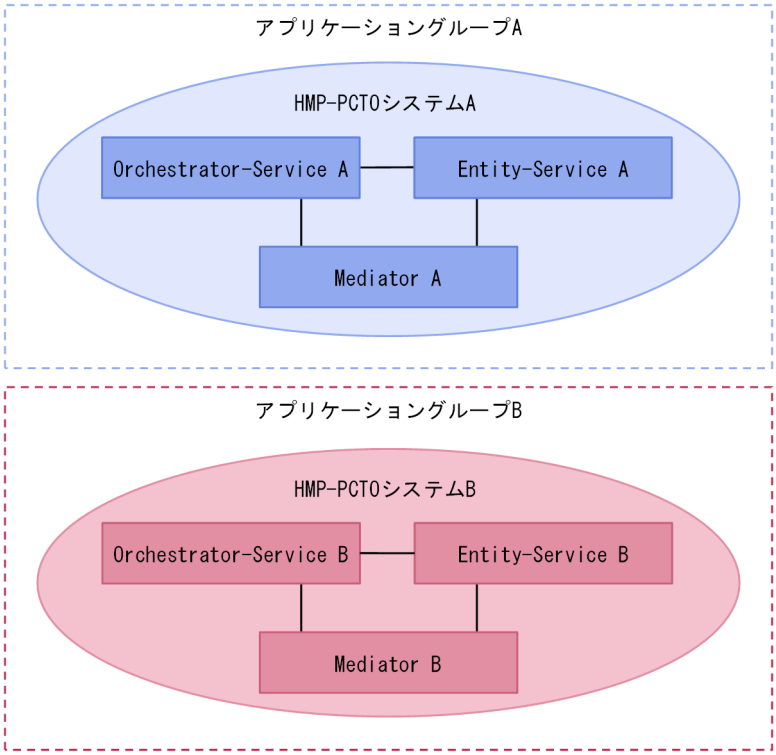
- 複数の業務システム間で同一の Kubernetes アプリケーションを 1 つも共有していない場合のアプリケーショングループ定義例

  次の図の例では、HMP-PCTO を使用する「業務システム A」と「業務システム B」があります。

  「業務システム A」は、「Orchestrator-Service A」、「Entity-Service A」、および「Mediator A」でトランザクションを実現し、「業務システム B」は、「Orchestrator-Service B」、「Entity-Service B」、および「Mediator B」でトランザクションを実現しています。

  これらの業務システム間では、同一の Kubernetes アプリケーションを 1 つも共有していないため、「業務システム A」と「業務システム B」で Kubernetes アプリケーション群を「アプリケーショングループ A」、「アプリケーショングループ B」のように分けて定義します。

図 7-1　複数の業務システム間で同一の Kubernetes アプリケーションを 1 つも共有していない場合のアプリケーショングループ定義例



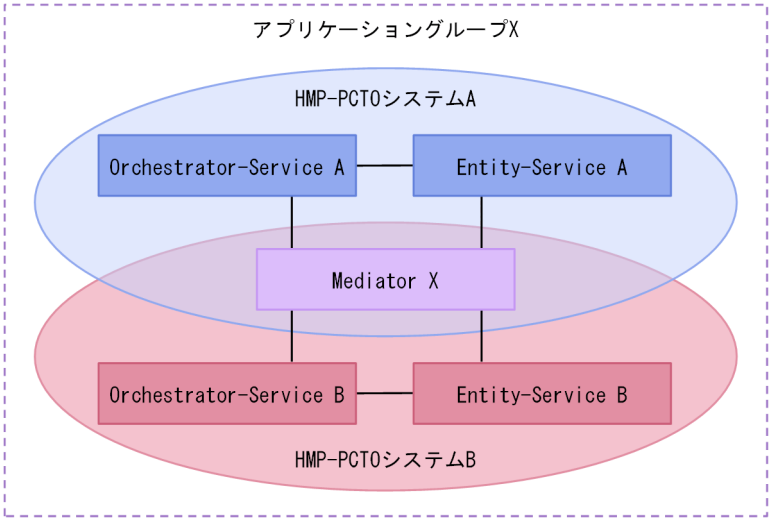
- 複数の業務システム間で同一の Kubernetes アプリケーションを 1 つ以上共有している場合のアプリケーショングループ定義例

  次の図の例では、HMP-PCTO を使用する「業務システム A」と「業務システム B」があります。

  「業務システム A」は、「Orchestrator-Service A」、「Entity-Service A」、および「Mediator X」でトランザクションを実現し、「業務システム B」は、「Orchestrator-Service B」、「Entity-Service B」、および「Mediator X」でトランザクションを実現しています。

  これらの業務システム間では、同一の Kubernetes アプリケーションを 1 つ以上共有しているため（この例では Mediator を共有）、「業務システム A」と「業務システム B」の Kubernetes アプリケーション群をまとめて 1 つの「アプリケーショングループ X」のように定義します。

図 7-2　複数の業務システム間で同一の Kubernetes アプリケーションを 1 つ以上共有している場合のアプリケーショングループ定義例



HMP-PCTO の Kubernetes アプリケーションのグループ定義の要否を次の表に示します。

表 7-2　HMP-PCTO の Kubernetes アプリケーションのグループ定義の要否

Kubernetes アプリケーション	グループ定義の要否
Orchestrator-Service	○
Relay-Service	○
Entity-Service(SQL)	○
Entity-Service(TCC)	○
Orchestrator-Service(SQL)	○
Alternate-Service	○
Alternate-Service(SQL)	○
Mediator	○
TP1-Bridge	○
EADS（通常版の場合）	−
Ext-Cons（トライアル版の場合）	−
Filebeat	−
Jaeger	−
Prometheus	−
Metricbeat	−
Elasticsearch	−
Logstash	−

Kubernetes アプリケーション	グループ定義の要否
Metrics Server	−

（凡例）

○：グループ定義が必要です。

−：グループ定義は不要です。

アプリケーショングループは、各 Kubernetes アプリケーションの Kubernetes マニフェストの「hmppcto-app-group」ラベルおよび Helm チャートの「labels.hmppctoAppGroup」パラメタで定義します。具体的な指定方法は、「7.6　各 Kubernetes アプリケーションの作成要件」内の該当 Kubernetes アプリケーションのセクションおよび「9.　パラメタ」内の該当 Kubernetes アプリケーションのパラメタの説明を参照してください。

7.3 Kubernetes クラスタ内外の通信暗号化

HMP-PCTO ではユーザの通信データの保護のため、コンポーネント間の通信暗号化機能を提供します。プロパティで暗号化通信をするかどうかを指定します。

表 7-3 Kubernetes クラスタ内外の通信暗号化の可否

項番	通信経路	暗号化の可否	暗号化方法
1	ユーザアプリケーションが実行する HMP-PCTO の Pod 間通信（REST 通信）	○	ユーザ責務の次の REST 通信です。 • Spring Boot の RestTemplate • Spring Boot の WebClient Spring Boot や Kubernetes の公式ドキュメントに従って、ユーザアプリケーションの暗号化通信を有効にしてください。 なお、スケーリング対応機能を使用する場合は、HMP-PCTO が提示するとおりにサーバ証明書を作成する必要があります。詳細については、「4.2.7 Kubernetes マニフェストの作成および Helm チャートのカスタマイズ」、および「4.3.9 Kubernetes マニフェストの作成および Helm チャートのカスタマイズ」を参照してください。
2	ユーザアプリケーションが実行する HMP-PCTO の Pod 間通信（gRPC 通信）	×	−
3	HMP-PCTO の Pod 間通信（gRPC 通信）	○	grpc.use-tls パラメタで暗号化通信をするかどうかを指定できます。 詳細は「9.1 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」を参照してください。
4	HMP-PCTO コントロールプレーンの Kubernetes アプリケーションの Pod と HMP-PCTO の Kubernetes アプリケーションで使用するスクリプト間の通信（REST 通信）	×	−
5	ユーザ責務の Kubernetes アプリケーションの Pod と HMP-PCTO の Kubernetes アプリケーションで使用するスクリプト間の通信（REST 通信）	○	トランザクションヘルスチェックスクリプト（health_check.sh）および未決着トランザクション待機スクリプト（wait_transactions.sh）の引数として「--scheme=https」を設定する必要があります。 証明書の検証を無効化する場合は、引数として「--insecure」を設定してください。

項番	通信経路	暗号化の可否	暗号化方法
			詳細については、「7.7 Kubernetes アプリケーションで使用するスクリプト」を参照してください。
6	HMP-PCTO の Pod と DB の間	×	−
7	HMP-PCTO の Pod と EADS の間	×	−
8	ユーザアプリケーションが実行する HMP-PCTO の Pod と TP1-Bridge の Pod 間通信（REST 通信）	×	−
9	TP1-Bridge と TP1-Converter の間	×	−
10	TP1-Bridge と OpenTP1 の間	×	−
11	TP1-Converter と REST API 発行元の間	×	−
12	• Filebeat と Logstash の間 • Logstash と Elasticsearch の間 • Prometheus と Metricbeat の間 • Metricbeat と Elasticsearch の間 • Jaeger と Elasticsearch の間	×	−
13	HMP-PCTO の Pod と Filebeat の間	×	−
14	HMP-PCTO の Pod と Jaeger の間	×	−
15	HMP-PCTO の Pod と Prometheus の間	○	使用するメトリクス収集アプリケーションまたはサービスのスクレイピング設定で暗号化通信を有効にしてください。 詳細は Prometheus の公式ドキュメントを参照してください。 また、HMP-PCTO の Pod の Kubernetes マニフェストファイルで、prometheus.io/scheme アノテーションを指定してください。アノテーションの値で Pod の暗号化通信の状態を表すことができます。 詳細については、「2.6 Kubernetes マニフェスト作成」を参照してください。
16	• Mediator と NFS サーバの間 • EADS と NFS サーバの間 • uCosminexus Application Runtime for Spring Boot と NFS サーバの間	×	−
17	Kubernetes とデプロイ依存関係チェックスクリプトの間	×	−

（凡例）

○：暗号化できます。

×：暗号化できません。

－：該当しません。

7.4 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

7.5 Kubernetes アプリケーションのグレースフルシャットダウン

7.5.1 HMP-PCTO の Kubernetes アプリケーションのグレースフルシャットダウンのサポート有無

HMP-PCTO の Kubernetes アプリケーションのグレースフルシャットダウンのサポート有無の一覧を次の表に示します。

表 7-4 HMP-PCTO の Kubernetes アプリケーションのグレースフルシャットダウンのサポート有無の一覧

Kubernetes アプリケーション	グレースフルシャットダウンのサポート有無
Orchestrator-Service	○
Relay-Service	○
Entity-Service(SQL)	○
Entity-Service(TCC)	−
Orchestrator-Service(SQL)	○
Alternate-Service	○
Alternate-Service(SQL)	○
Mediator	○
TP1-Bridge	−
EADS（通常版の場合）	−
Ext-Cons（トライアル版の場合）	−
Filebeat	−
Jaeger	−
Prometheus	−
Metricbeat	−
Elasticsearch	−
Logstash	−

（凡例）

○：グレースフルシャットダウンをサポートします。

−：該当しません。

7.5.2 HMP-PCTO の Kubernetes アプリケーションのグレースフルシャットダウンが開始できる条件

HMP-PCTO の Kubernetes アプリケーションのグレースフルシャットダウンが開始できる条件を次の表に示します。

表 7-5 HMP-PCTO の Kubernetes アプリケーションのグレースフルシャットダウンが開始できる条件

Kubernetes アプリケーション	グレースフルシャットダウンが開始できる条件（番号順に満たす必要あり）
Orchestrator-Service	1. 新規のトランザクションを受け付けない状態 2. トランザクションルートで処理中の件数が 0 件 3. 処理中リクエスト件数が 0 件
Relay-Service	1. 新規のトランザクションを受け付けない状態 2. 処理中リクエスト件数が 0 件
Entity-Service(SQL)	1. 新規のトランザクションを受け付けない状態 2. 未決着トランザクション件数が 0 件 3. 処理中リクエスト件数が 0 件
Orchestrator-Service(SQL)	1. 新規のトランザクションを受け付けない状態 2. 次の 2 つの条件を満たす（AND 条件） ・トランザクションルートで処理中の件数が 0 件 ・未決着トランザクション件数が 0 件 3. 処理中リクエスト件数が 0 件
Alternate-Service	1. 新規のトランザクションを受け付けない状態 2. トランザクションルートで処理中の件数が 0 件 3. 処理中リクエスト件数が 0 件
Alternate-Service(SQL)	1. 新規のトランザクションを受け付けない状態 2. 次の 2 つの条件を満たす（AND 条件） ・トランザクションルートで処理中の件数が 0 件 ・未決着トランザクション件数が 0 件 3. 処理中リクエスト件数が 0 件
Mediator	1. 新規のトランザクションを受け付けない状態 2. 未決着トランザクション件数が 0 件

各 Kubernetes アプリケーションのグレースフルシャットダウンが開始できる条件について、詳細を説明します。

各 Kubernetes アプリケーションに共通の条件

各 Kubernetes アプリケーションは、新規トランザクションを受け付けない状態にすることを 1 つ目の条件とします。

この理由は、新規トランザクションが流入していると、後続で確認している条件をいつまでも満たせなくなり、グレースフルシャットダウンを開始できなくなるためです。

Orchestrator-Service および Alternate-Service の条件

Orchestrator-Service および Alternate-Service は、トランザクション決着に関わる Pod のため、トランザクションをできる限り決着させるために「トランザクションルートで処理中の件数」が 0 件であることを条件とします。

また、Orchestrator-Service および Alternate-Service は、ユーザのリクエストハンドラでユーザアプリケーション処理を実施しているため、リクエストへの応答をできる限り完了させるために「処理中リクエスト件数」が 0 件であることも条件とします。

Relay-Service の条件

Relay-Service は、ユーザのリクエストハンドラでユーザアプリケーション処理を実施しているため、リクエストへの応答をできる限り完了させるために「処理中リクエスト件数」が 0 件であることを条件とします。

なお、Relay-Service は、トランザクション決着に関わる Pod ではないため、ほかの Kubernetes アプリケーションのような「未決着トランザクション件数」は条件に含みません。

また、グレースフルシャットダウンが開始できる条件が「処理中リクエスト件数」だけなので、グローバルトランザクションが処理中でも、自 Pod はシャットダウンすることがあります。このとき、Relay-Service がスケーリング対応機能を使用している場合は、ほかの Kubernetes アプリケーションから Relay-Service へ再度リクエストが来たとき、並行して稼働している別 Pod でリクエストを受け付けます。

Entity-Service(SQL)の条件

Entity-Service(SQL)は、トランザクション決着に関わる Pod のため、トランザクションをできる限り決着させるために「未決着トランザクション件数」が 0 件であることを条件とします。

Entity-Service(SQL)は、前述の Orchestrator-Service や Relay-Service のユーザアプリケーション処理を完了させるために、できる限り応答を返す必要があります。そのため、「処理中リクエスト件数」が 0 件であることも条件とします。

Orchestrator-Service(SQL)および Alternate-Service(SQL)の条件

Orchestrator-Service(SQL)および Alternate-Service(SQL)は、トランザクション決着に関わる Pod であるため、トランザクションをできる限り決着させるために、「トランザクションルートで処理中の件数」が 0 件であることと、「未決着トランザクション件数」が 0 件であることの両方を満たすことを条件とします。

また、Orchestrator-Service(SQL)および Alternate-Service(SQL)は、ユーザアプリケーション処理を完了させるために、可能な限り応答を返す必要があります。そのため、「処理中リクエスト件数」が 0 件であることも条件とします。

Mediator の条件

Mediator は、トランザクション決着に関わる Pod のため、トランザクションをできる限り決着させるために「未決着トランザクション件数」が 0 件であることを条件とします。

なお、Mediator は、ユーザアプリケーション処理を含まないため、ほかの Kubernetes アプリケーションのような「処理中リクエスト件数」は条件に含みません。

7.5.3　グレースフルシャットダウンの実現方法

グレースフルシャットダウンの実現方法を説明します。

「7.5.2　HMP-PCTO の Kubernetes アプリケーションのグレースフルシャットダウンが開始できる条件」によって、グレースフルシャットダウンの実現方法が異なります。それぞれの実現方法は次のとおりです。

- 新規トランザクションを受け付けない状態

  Kubernetes の preStop フックを使用して、20 秒間待機します。この間、Kubernetes の Service によって転送先リストからリクエストを削除する処理が実施されます。これによって、新規トランザクションを受け付けない状態になります。

- 未決着トランザクション件数およびトランザクションルートで処理中の件数

  「7.7.3　未決着トランザクション待機スクリプト（wait_transactions.sh）」の Kubernetes の preStop フックを使用して、未決着トランザクション件数およびトランザクションルートで処理中の件数が 0 件になるまで待機します。

- 処理中リクエスト件数

  - REST 通信によるリクエスト

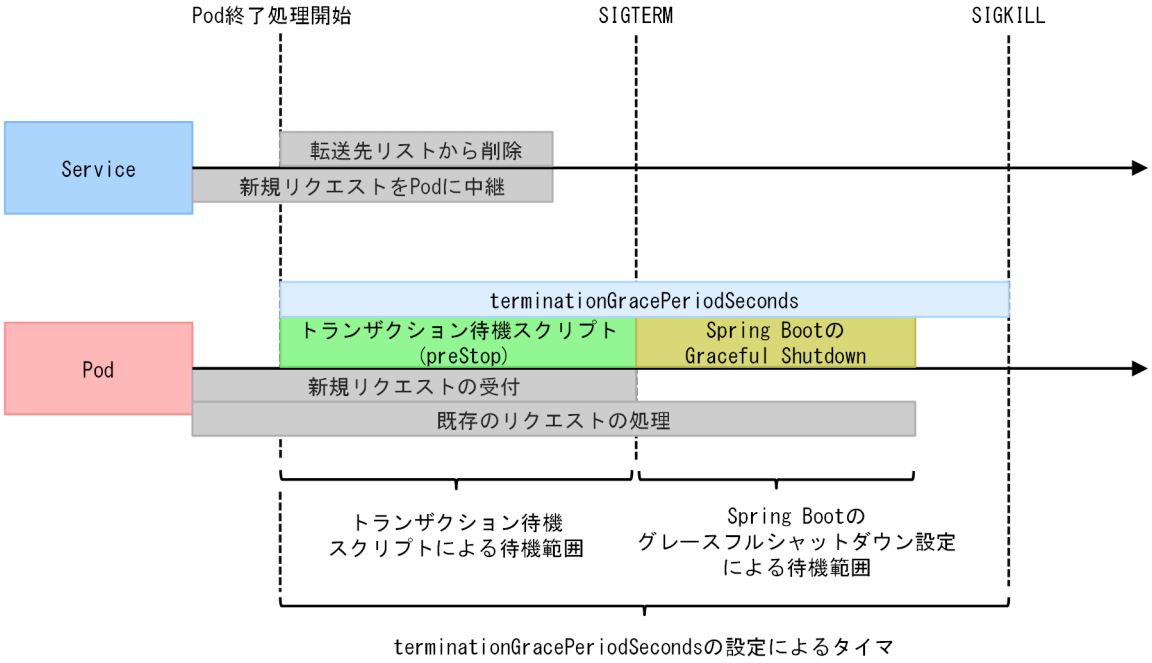
    Spring Boot のグレースフルシャットダウンの設定を使用して、処理中リクエスト件数が 0 件になるまで待機します。

  - gRPC 通信によるリクエスト

    gRPC サーバはグレースフルにシャットダウンするようにアプリケーションを実装してください。実装例については、「2.2.10　通信によるトランザクションブランチの作成（gRPC 通信-クライアント）」を参照してください。

Pod 終了処理時に、未決着トランザクション件数および処理中リクエスト件数が 0 件になるまでの待機範囲を、次の図に示します。

図 7-3　Pod 終了処理時に、未決着トランザクション件数および処理中リクエスト件数が 0 件になるまでの待機範囲



この図に示すとおり、「未決着トランザクション件数およびトランザクションルートで処理中の件数」と「処理中リクエスト件数」の両方を条件に含む Kubernetes アプリケーションの場合は、次の順序で待機を実施します。Pod は、次の 3 つの待機が完了したときにシャットダウンします。

1.「新規トランザクションを受け付けない状態」になるまでの待機
2. トランザクション待機スクリプトによる「未決着トランザクション件数／トランザクションルートで処理中の件数」の待機
3. Spring Boot のグレースフルシャットダウンの設定による「処理中リクエスト件数」の待機

また、上記の待機が完了しないケースを考慮し、Kubernetes の terminationGracePeriodSeconds（安全な停止待機秒数）を指定してください。トランザクション待機スクリプトによる「未決着トランザクション件数」の待機を開始してから指定した秒数が経過した場合に、Pod を強制的にシャットダウンします。

重要

次の状況が発生すると、後述の「7.5.4　HMP-PCTO の Kubernetes アプリケーションが強制シャットダウンしたときの挙動」の対象となります。

1. トランザクションが開始する。
2. トランザクションに参加している Pod がグレースフルシャットダウンを開始する。
3. 2.の Pod が前述の『1.「新規トランザクションを受け付けない状態」になるまでの待機』まで完了する。
4. トランザクション内で時間が掛かるなどの理由によって、他 Pod から 2.の Pod へリクエストするタイミングが 3.のあとになる。

7.5.4 HMP-PCTO の Kubernetes アプリケーションが強制シャットダウンしたときの挙動

グレースフルシャットダウンは、ユーザが事前に指定した猶予時間を過ぎた場合、トランザクション処理やリクエスト処理が処理中でも Pod がシャットダウンします。

このときの Kubernetes アプリケーションのトランザクションの決着結果およびユーザへの影響を次の表に示します。

表 7-6 HMP-PCTO の Kubernetes アプリケーションが強制シャットダウンしたときのトランザクションの決着結果およびユーザへの影響

項番	Kubernetes アプリケーション	トランザクションの決着結果	最終的な Kubernetes アプリケーションの挙動	備考
1	Orchestrator-Service および Orchestrator-Service(SQL)	ロールバックします。	Pod のシャットダウンによって、Orchestrator-Service のクライアントへの応答が返りません。	Participant のトランザクション決着監視タイマによって、仕掛中のトランザクションがロールバックします。
2	Relay-Service	ロールバックします。	KFSG63105-W メッセージを出力します。	Relay-Service の呼び出し元がレスポンスを受け取れないため通信エラーとなり、トランザクションがロールバックに決着します。
3	Entity-Service(SQL)	ロールバックします。	KFSG63105-W メッセージを出力します。	未決着トランザクションの Recovery 機能によってトランザクションがロールバックします。
4	Alternate-Service	• Orchestrator-Service の役割で処理しているトランザクションは項番 1 と同様の動作をします。 • Relay-Service の役割で処理しているトランザクションは項番 2 と同様の動作をします。		
5	Alternate-Service(SQL)	• Orchestrator-Service の役割で処理しているトランザクションは項番 1 と同様の動作をします。 • Relay-Service の役割で処理しているトランザクションは項番 2 と同様の動作をします。 • Entity-Service(SQL)の役割で処理しているトランザクションは項番 3 と同様の動作をします。		
6	Mediator	正常に決着します。	正常に動作します。	残存するほかの Mediator や補助合意機能によってトランザクションが決着します。

7.6　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、Kubernetes 公式リファレンスにある Kubernetes マニフェストの必要なフィールド（作成要件）について説明します。

7.6.1　Orchestrator-Service の要件

Orchestrator-Service は、Client システムからの要求を受け付けて、ほかのサービスを利用して業務を実行する Kubernetes アプリケーションです。トランザクションルートを開始します。

この Kubernetes アプリケーションは、次の表に示す Orchestrator-Service の要件を満たすように作成します。

表 7-7　Orchestrator-Service 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	ワークロードリソース（Deployment など） の.spec.template.metadata.labels に次を指定してください。※1 • キー名：hmppcto-app-group 値：「アプリケーショングループ名」を表す文字列 hmppcto-app-group パラメタの説明については、「付録 B.1　HMP-PCTO 固有の指定が必要なパラメタ」を参照してください。 「アプリケーショングループ」の考え方については、「7.2　Kubernetes アプリケーションのグループ定義」を参照してください。 Service の.metadata.labels に次を指定してください。 • キー名：hmppcto-app-group 値：.spec.template.metadata.labels に指定する同名のキー（hmppcto-app-group）と同じ値を指定してください。
アノテーション	規定しません。 <Prometheus を使用してメトリクス情報を収集する場合> .spec.template.metadata.annotations に次を指定してください。 • キー名：prometheus.io/scrape 値：「true」 値が true の場合にメトリクスの収集対象になります。 • キー名：prometheus.io/path 値：「/hmppcto/metrics※2」 メトリクス収集用のエンドポイントです。 • キー名：prometheus.io/port 値：http ポート番号（Prometheus メトリクス収集用）に指定した値 メトリクス収集用のポートです。

項目	要件
	• キー名：prometheus.io/scheme 値：「http」または「https」 メトリクス収集用のスキーマです。
初期化コンテナ構成	Dependency-Checker コンテナ（「6.4.10　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-Service コンテナ（「6.4.1　Orchestrator-Service コンテナイメージ」を参照してください） • <Elastic Stack 使用時>Filebeat コンテナ Elastic Stack を使用してトラブルシュート情報を収集する場合だけ使用します。
レプリカ数	• スケーリング対応機能を使用する場合：下限「1」、上限「なし」 • スケーリング対応機能を使用しない場合：「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。 ＜ Kubernetes リソース取得用ロール＞ • API グループ 空（コアグループ） • リソース services、endpoints、pods • 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。 • Orchestrator-Service と Orchestrator-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※3 の Namespace がすべて同一の場合：

項目	要件
	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-Service と Orchestrator-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※3 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 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 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「180」 terminationGracePeriodSeconds は、「preStop に設定する未決着トランザクション待機スクリプトの最大待機秒数（120 秒）」と「Spring Boot のグレースフルシャットダウンの設定での最大待機秒数（30 秒）」の合計値以上の値を設定してください。推奨値はそれぞれの最大待機時間のデフォルト値の合計にバッファ（30 秒）を加算した値です。

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

Kubernetes アプリケーションにコンテキストパス（例：/context-path）を設定している場合、prometheus.io/path にはコンテキストパスを含めた正しいパスを指定してください。

コンテキストパスを指定していない場合、メトリクス収集用のエンドポイントは、/hmppcto/metrics になります。

コンテキストパスを指定していない場合の prometheus.io/path の指定：prometheus.io/path: /hmppcto/metrics

コンテキストパスを指定している場合、メトリクス収集用エンドポイントの先頭にコンテキストパスが追加されます。

コンテキストパスを指定している場合の prometheus.io/path の指定：prometheus.io/path: /context-path/hmppcto/metrics

prometheus.io/path の設定が、実際のアプリケーションのエンドポイントと一致しているかを必ず確認してください。設定が誤っていると、Prometheus によるメトリクスの収集ができません。

注※3

Orchestrator-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 7-8　Orchestrator-Service 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 7-8　Orchestrator-Service 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＞ Entity-Service(SQL)、Entity-Service(TCC)、または TP1-Bridge 例を次に示します。 （例） Entity-Service(SQL)を使用する場合：my-namespace/deployments/sql-participant Entity-Service(TCC)を使用する場合：my-namespace/deployments/tcc-participant TP1-Bridge 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 7-9　Orchestrator-Service の Orchestrator-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）。

項目	要件
	• 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：自コンテナの server.port パラメタに設定したポート番号 コンテナポート：自コンテナの server.port パラメタに設定したポート番号
コンテキストパス設定	規定しません。 コンテキストパスを設定した場合は、次の項目で設定したコンテキストパスと同じ値を指定してください。 • 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）の第 6 引数（コンテキストパス） • HMP-PCTO の未決着トランザクション待機スクリプト（wait_transactions.sh）の第 6 引数（コンテキストパス） また、コンテキストパスを設定した場合は、次の項目で設定したコンテキストパスを含んだ値を指定してください。 • メトリクス収集用のエンドポイントを設定するアノテーション prometheus.io/path
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。 • 「9.2.1　共通」のパラメタ • 「9.2.3　Orchestrator」のパラメタ • 「9.2.7　Spring Boot の組み込み HTTP サーバ」のパラメタ
	Spring Boot のグレースフルシャットダウンの設定に関する次のプロパティを、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。 • プロパティ：spring.lifecycle.timeout-per-shutdown-phase 設定する値：任意の秒数 推奨値：「30s」
	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-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 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 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 for Spring Boot の設定ファイル

項目	要件
	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）：自コンテナの server.port パラメタに設定したポート番号 第 6 引数（コンテキストパス）：コンテキストパスに指定した値。コンテキストパスを指定する場合だけ設定してください。指定しない場合は、この引数を省略してください。 オプション（スキーム）：「--scheme={ http ｜ https }」 自コンテナの接続に使用するスキームを指定します。 このオプションは省略できます。省略した場合、「http」が設定された場合の動作となります。 オプション（証明書検証無効化）：「--insecure」 HTTPS 接続時の証明書検証を無効化します。 このオプションは省略できます。省略した場合、証明書検証が行われます。 • 初回 Probe 実行待機秒数（initialDelaySeconds） 推奨値「150」 • 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」

項目	要件
	第 5 引数（HTTP ポート番号）：自コンテナの server.port パラメタに設定したポート番号 第 6 引数（コンテキストパス）：コンテキストパスに指定した値。コンテキストパスを指定する場合だけ設定してください。指定しない場合は、この引数を省略してください。 オプション（スキーム）：「--scheme={ http | https }」 自コンテナの接続に使用するスキームを指定します。 このオプションは省略できます。省略した場合、「http」が設定された場合の動作となります。 オプション（証明書検証無効化）：「--insecure」 HTTPS 接続時の証明書検証を無効化します。 このオプションは省略できます。省略した場合、証明書検証が行われます。 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「10」 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）：自コンテナの server.port パラメタに設定したポート番号 第 6 引数（コンテキストパス）：コンテキストパスに指定した値。コンテキストパスを指定する場合だけ設定してください。指定しない場合は、この引数を省略してください。 オプション（スキーム）：「--scheme={ http | https }」 自コンテナの接続に使用するスキームを指定します。 このオプションは省略できます。省略した場合、「http」が設定された場合の動作となります。 オプション（証明書検証無効化）：「--insecure」 HTTPS 接続時の証明書検証を無効化します。 このオプションは省略できます。省略した場合、証明書検証が行われます。 未決着トランザクション待機スクリプトの最大待機時間は「開始時待機秒数（20 秒）」と「リトライ回数×リトライ間隔秒数」を合計した値です。最大待機時間がトランザクション決着監視タイマの設定値を超えるように引数を調整してください。
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。

Elastic Stack を使用してトラブルシュート情報を収集する場合だけ、次の Filebeat コンテナの要件を満たすように作成します。

表 7-10　Orchestrator-Service 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「3.11.1　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」
CPU/メモリリソース制限	規定しません。

7.6.2　Relay-Service の要件

Relay-Service は、リソースを更新しないでサービス間を中継する Kubernetes アプリケーションです。トランザクションブランチを開始します。

この Kubernetes アプリケーションは、次の表に示す Relay-Service の要件を満たすように作成します。

表 7-11　Relay-Service 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。

項目	要件
ネームスペース	規定しません。
ラベル	ワークロードリソース（Deployment など） の.spec.template.metadata.labels に次を指定してください。※1 • キー名：hmppcto-app-group 値：「アプリケーショングループ名」を表す文字列 hmppcto-app-group パラメタの説明については、「付録 B.1　HMP-PCTO 固有の指定が必要なパラメタ」を参照してください。 「アプリケーショングループ」の考え方については、「7.2　Kubernetes アプリケーションのグループ定義」を参照してください。 Service の.metadata.labels に次を指定してください。 • キー名：hmppcto-app-group 値：.spec.template.metadata.labels に指定する同名のキー（hmppcto-app-group）と同じ値を指定してください。
アノテーション	規定しません。 <Prometheus を使用してメトリクス情報を収集する場合> .spec.template.metadata.annotations に次を指定してください。 • キー名：prometheus.io/scrape 値：「true」 値が true の場合にメトリクスの収集対象になります。 • キー名：prometheus.io/path 値：「/hmppcto/metrics※2」 メトリクス収集用のエンドポイントです。 • キー名：prometheus.io/port 値：http ポート番号（Prometheus メトリクス収集用）に指定した値 メトリクス収集用のポートです。 • キー名：prometheus.io/scheme 値：「http」または「https」 メトリクス収集用のスキーマです。
初期化コンテナ構成	Dependency-Checker コンテナ（「6.4.10　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	• Relay-Service コンテナ（「6.4.2　Relay-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） • <Elastic Stack 使用時>Filebeat コンテナ Elastic Stack を使用してトラブルシュート情報を収集する場合だけ使用します。
レプリカ数	• スケーリング対応機能を使用する場合：下限「1」、上限「なし」 • スケーリング対応機能を使用しない場合：「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。

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。 ＜ Kubernetes リソース取得用ロール＞ • API グループ 空（コアグループ） • リソース services、endpoints、pods • 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。 • Relay-Service と Relay-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※3 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。 • Relay-Service と Relay-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※3 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 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 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「180」

項目	要件
	Orchestrator-Service、Orchestrator-Service(SQL)、Entity-Service(SQL)および Entity-Service(TCC)の terminationGracePeriodSeconds の設定値以上の値を設定してください。 terminationGracePeriodSeconds は、「Spring Boot のグレースフルシャットダウンの設定での最大待機秒数（140 秒）」以上の値を設定してください。推奨値はそれぞれの最大待機時間のデフォルト値の合計にバッファ（40 秒）を加算した値です。

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

Kubernetes アプリケーションにコンテキストパス（例：/context-path）を設定している場合、prometheus.io/path にはコンテキストパスを含めた正しいパスを指定してください。

コンテキストパスを指定していない場合、メトリクス収集用のエンドポイントは、/hmppcto/metrics になります。

コンテキストパスを指定していない場合の prometheus.io/path の指定：prometheus.io/path: /hmppcto/metrics

コンテキストパスを指定している場合、メトリクス収集用エンドポイントの先頭にコンテキストパスが追加されます。

コンテキストパスを指定している場合の prometheus.io/path の指定：prometheus.io/path: /context-path/hmppcto/metrics

prometheus.io/path の設定が、実際のアプリケーションのエンドポイントと一致しているかを必ず確認してください。設定が誤っていると、Prometheus によるメトリクスの収集ができません。

注※3

Relay-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 7-12　Relay-Service 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 7-12　Relay-Service 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、Entity-Service(SQL)、Entity-Service(TCC)、または TP1-Bridge 例を次に示します。

項目	要件
	（例） Entity-Service(SQL)を使用する場合： my-namespace/statefulsets/mediator my-namespace/deployments/sql-participant Entity-Service(TCC)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： 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 7-13　Relay-Service の Relay-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：自コンテナの server.port パラメタに設定したポート番号 コンテナポート：自コンテナの server.port パラメタに設定したポート番号
コンテキストパス設定	規定しません。 コンテキストパスを設定した場合は、次の項目で設定したコンテキストパスと同じ値を指定してください。 • 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）の第 6 引数（コンテキストパス）

項目	要件
	また、コンテキストパスを設定した場合は、次の項目で設定したコンテキストパスを含んだ値を指定してください。 • メトリクス収集用のエンドポイントを設定するアノテーション prometheus.io/path
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。 •「9.2.1　共通」のパラメタ •「9.2.4　Entity-Module」のパラメタ •「9.2.7　Spring Boot の組み込み HTTP サーバ」のパラメタ
	Spring Boot のグレースフルシャットダウンの設定に関する次のプロパティを、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。 • プロパティ：spring.lifecycle.timeout-per-shutdown-phase 設定する値：任意の秒数 推奨値：「140s」
	タイムゾーンの設定は任意です。
	環境変数 HMPPCTO_JAVA_OPTIONS で Relay-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 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 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 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）：自コンテナの server.port パラメタに設定したポート番号

項目	要件
	第 6 引数（コンテキストパス）：コンテキストパスに指定した値。コンテキストパスを指定する場合だけ設定してください。指定しない場合は、この引数を省略してください。 オプション（スキーム）：「--scheme={ http ┃ https }」 自コンテナの接続に使用するスキームを指定します。 このオプションは省略できます。省略した場合、「http」が設定された場合の動作となります。 オプション（証明書検証無効化）：「--insecure」 HTTPS 接続時の証明書検証を無効化します。 このオプションは省略できます。省略した場合、証明書検証が行われます。 • 初回 Probe 実行待機秒数（initialDelaySeconds） 推奨値「150」 • 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）：自コンテナの server.port パラメタに設定したポート番号 第 6 引数（コンテキストパス）：コンテキストパスに指定した値。コンテキストパスを指定する場合だけ設定してください。指定しない場合は、この引数を省略してください。 オプション（スキーム）：「--scheme={ http ┃ https }」 自コンテナの接続に使用するスキームを指定します。 このオプションは省略できます。省略した場合、「http」が設定された場合の動作となります。 オプション（証明書検証無効化）：「--insecure」 HTTPS 接続時の証明書検証を無効化します。 このオプションは省略できます。省略した場合、証明書検証が行われます。 preStop フックを次のとおり設定してください。 • コマンド実行

項目	要件
	グレースフルシャットダウンのために、Pod が宛先に選ばれなくなるまで待機する必要があります。そのため 20 秒間待機するコマンドを設定してください。※
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。

注※

コマンド例を次に示します。

```
- /bin/sh
- -c
- sleep 20
```

Elastic Stack を使用してトラブルシュート情報を収集する場合だけ、次の Filebeat コンテナの要件を満たすように作成します。

表 7-14　Relay-Service 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「3.11.1　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）

項目	要件
	推奨値「5」
CPU/メモリリソース制限	規定しません。

7.6.3 Entity-Service(SQL)の要件（SQL-Participant 限定）

Entity-Service(SQL)は、データベースの参照、更新業務を行う Kubernetes アプリケーションです。トランザクションブランチを開始します。

この Kubernetes アプリケーションは、次の表に示す Entity-Service(SQL)の要件を満たすように作成します。

表 7-15 Entity-Service(SQL)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	Entity-Service(SQL)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 Namespace 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、Namespace 名を検討してください。
ラベル	ワークロードリソース（Deployment など） の.spec.template.metadata.labels に次を指定してください。※1 • キー名：app.kubernetes.io/name 値：app.kubernetes.io/name パラメタの説明については、「付録 B.1 HMP-PCTO 固有の指定が必要なパラメタ」を参照してください。 • キー名：hmppcto-app-group 値：「アプリケーショングループ名」を表す文字列 hmppcto-app-group パラメタの説明については、「付録 B.1 HMP-PCTO 固有の指定が必要なパラメタ」を参照してください。 「アプリケーショングループ」の考え方については、「7.2 Kubernetes アプリケーションのグループ定義」を参照してください。 Service の.metadata.labels に次を指定してください。 • キー名：hmppcto-app-group 値：.spec.template.metadata.labels に指定する同名のキー（hmppcto-app-group）と同じ値を指定してください。
アノテーション	規定しません。 <Prometheus を使用してメトリクス情報を収集する場合> .spec.template.metadata.annotations に次を指定してください。 • キー名：prometheus.io/scrape 値：「true」 値が true の場合にメトリクスの収集対象になります。 • キー名：prometheus.io/path

項目	要件
	値：「/hmppcto/metrics※1」 メトリクス収集用のエンドポイントです。 • キー名：prometheus.io/port 値：http ポート番号（Prometheus メトリクス収集用）に指定した値 メトリクス収集用のポートです。 • キー名：prometheus.io/scheme 値：「http」または「https」 メトリクス収集用のスキーマです。
初期化コンテナ構成	Dependency-Checker コンテナ（「6.4.10　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(SQL)コンテナ（「6.4.3　Entity-Service(SQL)コンテナイメージ（SQL-Participant 限定）」を参照してください） • <Elastic Stack 使用時>Filebeat コンテナ Elastic Stack を使用してトラブルシュート情報を収集する場合だけ使用します。
レプリカ数	• スケーリング対応機能を使用する場合：下限「1」、上限「なし※2」 • スケーリング対応機能を使用しない場合：「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。 ＜ Kubernetes リソース取得用ロール＞ • API グループ 空（コアグループ） • リソース services、endpoints、pods • 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
	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(SQL)と Entity-Service(SQL)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※3 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(SQL)と Entity-Service(SQL)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※3 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 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 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「180」 terminationGracePeriodSeconds は、「preStop に設定する未決着トランザクション待機スクリプトの最大待機秒数（120 秒）」と「Spring Boot のグレースフルシャットダウンの設定での最大待機秒数（30 秒）」の合計値以上の値を設定してください。推奨値はそれぞれの最大待機時間のデフォルト値の合計にバッファ（30 秒）を加算した値です。

注※1

Kubernetes アプリケーションにコンテキストパス（例：/context-path）を設定している場合、prometheus.io/path にはコンテキストパスを含めた正しいパスを指定してください。

コンテキストパスを指定していない場合、メトリクス収集用のエンドポイントは、/hmppcto/metrics になります。

コンテキストパスを指定していない場合の prometheus.io/path の指定：prometheus.io/path: /hmppcto/metrics

コンテキストパスを指定している場合、メトリクス収集用エンドポイントの先頭にコンテキストパスが追加されます。

コンテキストパスを指定している場合の prometheus.io/path の指定：prometheus.io/path: /context-path/hmppcto/metrics

prometheus.io/path の設定が、実際のアプリケーションのエンドポイントと一致しているかを必ず確認してください。設定が誤っていると、Prometheus によるメトリクスの収集ができません。

注※2

スケーリング対応機能を使用する場合、この Pod が接続する次のリソースとの最大接続数を、デプロイするレプリカの最大数に応じて決定してください。各リソースとの最大接続数に関するパラメタは、「9.4　各 RDBMS の使用に必要なパラメタおよび設定（SQL-Participant 限定）」を参照してください。

・EADS サーバへの最大同時接続数（eads.server.maxConnections）

・DB への最大同時接続数

注※3

Entity-Service(SQL)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 7-16　Entity-Service(SQL)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 7-16　Entity-Service(SQL)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）、Entity-Service(SQL)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-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）。
CPU/メモリリソース制限	設定しません（デフォルト）。

表 7-17　Entity-Service(SQL)の Entity-Service(SQL)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	Entity-Service(SQL)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 Namespace 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)。 • ユーザアプリケーションで使用する各種ポート番号 • Entity-Service(SQL)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：自コンテナの server.port パラメタに設定したポート番号 コンテナポート：自コンテナの server.port パラメタに設定したポート番号
コンテキストパス設定	規定しません。 コンテキストパスを設定した場合は、次の項目で設定したコンテキストパスと同じ値を指定してください。 • 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）の第 6 引数（コンテキストパス） • HMP-PCTO の未決着トランザクション待機スクリプト（wait_transactions.sh）の第 6 引数（コンテキストパス） また、コンテキストパスを設定した場合は、次の項目で設定したコンテキストパスを含んだ値を指定してください。 • メトリクス収集用のエンドポイントを設定するアノテーション prometheus.io/path
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。 •「9.2.1　共通」のパラメタ •「9.2.2　JDBC Proxy Driver」のパラメタ •「9.2.4　Entity-Module」のパラメタ •「9.2.5　SQL-Participant」のパラメタ •「9.2.7　Spring Boot の組み込み HTTP サーバ」のパラメタ
	Spring Boot のグレースフルシャットダウンの設定に関する次のプロパティを、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。 • プロパティ：spring.lifecycle.timeout-per-shutdown-phase 設定する値：任意の秒数 推奨値：「30s」
	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-Service(SQL)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。

項目	要件
	uCosminexus Application Runtime 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 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 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）：自コンテナの server.port パラメタに設定したポート番号 第 6 引数（コンテキストパス）：コンテキストパスに指定した値。コンテキストパスを指定する場合だけ設定してください。指定しない場合は、この引数を省略してください。 オプション（スキーム）：「--scheme={ http ｜ https }」 自コンテナの接続に使用するスキームを指定します。 このオプションは省略できます。省略した場合、「http」が設定された場合の動作となります。 オプション（証明書検証無効化）：「--insecure」 HTTPS 接続時の証明書検証を無効化します。 このオプションは省略できます。省略した場合、証明書検証が行われます。 • 初回 Probe 実行待機秒数（initialDelaySeconds） 推奨値「150」 • 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

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）：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）：自コンテナの server.port パラメタに設定したポート番号 第 6 引数（コンテキストパス）：コンテキストパスに指定した値。コンテキストパスを指定する場合だけ設定してください。指定しない場合は、この引数を省略してください。 オプション（スキーム）：「--scheme={ http ｜ https }」 自コンテナの接続に使用するスキームを指定します。 このオプションは省略できます。省略した場合、「http」が設定された場合の動作となります。 オプション（証明書検証無効化）：「--insecure」 HTTPS 接続時の証明書検証を無効化します。 このオプションは省略できます。省略した場合、証明書検証が行われます。 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「10」 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）：自コンテナの server.port パラメタに設定したポート番号 第 6 引数（コンテキストパス）：コンテキストパスに指定した値。コンテキストパスを指定する場合だけ設定してください。指定しない場合は、この引数を省略してください。 オプション（スキーム）：「--scheme={ http ｜ https }」 自コンテナの接続に使用するスキームを指定します。 このオプションは省略できます。省略した場合、「http」が設定された場合の動作となります。 オプション（証明書検証無効化）：「--insecure」 HTTPS 接続時の証明書検証を無効化します。 このオプションは省略できます。省略した場合、証明書検証が行われます。 未決着トランザクション待機スクリプトの最大待機時間は「開始時待機秒数（20 秒）」と「リトライ回数×リトライ間隔秒数」を合計した値です。最大待機時間がトランザクション決着監視タイマの設定値を超えるように引数を調整してください。
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。

Elastic Stack を使用してトラブルシュート情報を収集する場合だけ、次の Filebeat コンテナの要件を満たすように作成します。

表 7-18　Entity-Service(SQL)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「3.11.1　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」
CPU/メモリリソース制限	規定しません。

7.6.4　Entity-Service(TCC)の要件（TCC-Participant 限定）

Entity-Service(TCC)は、外部のサービスを使用してデータベースなどのリソースの参照および更新業務を行う Kubernetes アプリケーションです。トランザクションブランチを開始します。

この Kubernetes アプリケーションは、次の表に示す Entity-Service(TCC)の要件を満たすように作成します。

表 7-19　Entity-Service(TCC)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	Entity-Service(TCC)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 Namespace 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、Namespace 名を検討してください。
ラベル	ワークロードリソース（Deployment など） の.spec.template.metadata.labels に次を指定してください。※1 • キー名：hmppcto-app-group 値：「アプリケーショングループ名」を表す文字列 hmppcto-app-group パラメタの説明については、「付録 B.1　HMP-PCTO 固有の指定が必要なパラメタ」を参照してください。 「アプリケーショングループ」の考え方については、「7.2　Kubernetes アプリケーションのグループ定義」を参照してください。 Service の.metadata.labels に次を指定してください。 • キー名：hmppcto-app-group 値：.spec.template.metadata.labels に指定する同名のキー（hmppcto-app-group）と同じ値を指定してください。
アノテーション	規定しません。 <Prometheus を使用してメトリクス情報を収集する場合> .spec.template.metadata.annotations に次を指定してください。 • キー名：prometheus.io/scrape 値：「true」 値が true の場合にメトリクスの収集対象になります。 • キー名：prometheus.io/path 値：「/hmppcto/metrics※2」 メトリクス収集用のエンドポイントです。 • キー名：prometheus.io/port 値：http ポート番号（Prometheus メトリクス収集用）に指定した値 メトリクス収集用のポートです。 • キー名：prometheus.io/scheme 値：「http」または「https」 メトリクス収集用のスキーマです。
初期化コンテナ構成	Dependency-Checker コンテナ（「6.4.10　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(TCC)コンテナ（「6.4.4　Entity-Service(TCC)コンテナイメージ（TCC-Participant 限定）」を参照してください） • <Elastic Stack 使用時>Filebeat コンテナ Elastic Stack を使用してトラブルシュート情報を収集する場合だけ使用します。
レプリカ数	「1」

項目	要件
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。 ＜ Kubernetes リソース取得用ロール＞ • API グループ 空（コアグループ） • リソース services、endpoints、pods • 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。 • Entity-Service(TCC)と Entity-Service(TCC)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※3 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(TCC)と Entity-Service(TCC)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※3 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 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 for Spring Boot 導入あり）の場合だけ）。

項目	要件
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「150」 terminationGracePeriodSeconds は、「preStop に設定する未決着トランザクション待機スクリプトの最大待機秒数（120 秒）」の値以上の値を設定してください。推奨値はそれぞれの最大待機時間のデフォルト値の合計にバッファ（30 秒）を加算した値です。

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

Kubernetes アプリケーションにコンテキストパス（例：/context-path）を設定している場合、prometheus.io/path にはコンテキストパスを含めた正しいパスを指定してください。

コンテキストパスを指定していない場合、メトリクス収集用のエンドポイントは、/hmppcto/metrics になります。

コンテキストパスを指定していない場合の prometheus.io/path の指定：prometheus.io/path: /hmppcto/metrics

コンテキストパスを指定している場合、メトリクス収集用エンドポイントの先頭にコンテキストパスが追加されます。

コンテキストパスを指定している場合の prometheus.io/path の指定：prometheus.io/path: /context-path/hmppcto/metrics

prometheus.io/path の設定が、実際のアプリケーションのエンドポイントと一致しているかを必ず確認してください。設定が誤っていると、Prometheus によるメトリクスの収集ができません。

注※3

Entity-Service(TCC)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 7-20　Entity-Service(TCC)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 7-20　Entity-Service(TCC)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） 例を次に示します。 （例）

項目	要件
	my-namespace/statefulsets/mediator my-namespace/statefulsets/eads-server-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 7-21　Entity-Service(TCC)の Entity-Service(TCC)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	Entity-Service(TCC)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 Namespace 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：自コンテナの server.port パラメタに設定したポート番号 コンテナポート：自コンテナの server.port パラメタに設定したポート番号
コンテキストパス設定	規定しません。 コンテキストパスを設定した場合は、次の項目で設定したコンテキストパスと同じ値を指定してください。 • 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）の第 6 引数（コンテキストパス） • HMP-PCTO の未決着トランザクション待機スクリプト（wait_transactions.sh）の第 6 引数（コンテキストパス） また、コンテキストパスを設定した場合は、次の項目で設定したコンテキストパスを含んだ値を指定してください。 • メトリクス収集用のエンドポイントを設定するアノテーション prometheus.io/path
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。

項目	要件
	• 「9.2.1　共通」のパラメタ • 「9.2.4　Entity-Module」のパラメタ • 「9.2.6　TCC-Participant」のパラメタ • 「9.2.7　Spring Boot の組み込み HTTP サーバ」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-Service(TCC)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 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 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 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）：自コンテナの server.port パラメタに設定したポート番号 第 6 引数（コンテキストパス）：コンテキストパスに指定した値。コンテキストパスを指定する場合だけ設定してください。指定しない場合は、この引数を省略してください。 オプション（スキーム）：「--scheme={ http ｜ https }」 自コンテナの接続に使用するスキームを指定します。 このオプションは省略できます。省略した場合、「http」が設定された場合の動作となります。 オプション（証明書検証無効化）：「--insecure」 HTTPS 接続時の証明書検証を無効化します。 このオプションは省略できます。省略した場合、証明書検証が行われます。

項目	要件
	• 初回 Probe 実行待機秒数（initialDelaySeconds） 推奨値「150」 • 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）：自コンテナの server.port パラメタに設定したポート番号 第 6 引数（コンテキストパス）：コンテキストパスに指定した値。コンテキストパスを指定する場合だけ設定してください。指定しない場合は、この引数を省略してください。 オプション（スキーム）：「--scheme={ http | https }」 自コンテナの接続に使用するスキームを指定します。 このオプションは省略できます。省略した場合、「http」が設定された場合の動作となります。 オプション（証明書検証無効化）：「--insecure」 HTTPS 接続時の証明書検証を無効化します。 このオプションは省略できます。省略した場合、証明書検証が行われます。 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「10」 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）：自コンテナの server.port パラメタに設定したポート番号 第 6 引数（コンテキストパス）：コンテキストパスに指定した値。コンテキストパスを指定する場合だけ設定してください。指定しない場合は、この引数を省略してください。 オプション（スキーム）：「--scheme={ http | https }」 自コンテナの接続に使用するスキームを指定します。

項目	要件
	このオプションは省略できます。省略した場合、「http」が設定された場合の動作となります。 オプション（証明書検証無効化）：「--insecure」 HTTPS 接続時の証明書検証を無効化します。 このオプションは省略できます。省略した場合、証明書検証が行われます。 未決着トランザクション待機スクリプトの最大待機時間は「開始時待機秒数（20 秒）」と「リトライ回数×リトライ間隔秒数」を合計した値です。最大待機時間がトランザクション決着監視タイマの設定値を超えるように引数を調整してください。
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。

Elastic Stack を使用してトラブルシュート情報を収集する場合だけ、次の Filebeat コンテナの要件を満たすように作成します。

表 7-22　Entity-Service(TCC)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「3.11.1　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」
CPU/メモリリソース制限	規定しません。

7.6.5　Orchestrator-Service(SQL)の要件（SQL-Participant 限定）

Orchestrator-Service(SQL)は、Orchestrator の役割に加えて、データベースの参照および更新業務を行う Kubernetes アプリケーションです。トランザクションルートとトランザクションブランチを開始します。

この Kubernetes アプリケーションは、次の表に示す Orchestrator-Service(SQL)の要件を満たすように作成します。

表 7-23　Orchestrator-Service(SQL)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	Orchestrator-Service(SQL)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 Namespace 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、Namespace 名を検討してください。
ラベル	ワークロードリソース（Deployment など） の.spec.template.metadata.labels に次を指定してください。※1 • キー名：app.kubernetes.io/name 値：app.kubernetes.io/name パラメタの説明については、「付録 B.1　HMP-PCTO 固有の指定が必要なパラメタ」を参照してください。 • キー名：hmppcto-app-group 値：「アプリケーショングループ名」を表す文字列 hmppcto-app-group パラメタの説明については、「付録 B.1　HMP-PCTO 固有の指定が必要なパラメタ」を参照してください。 「アプリケーショングループ」の考え方については、「7.2　Kubernetes アプリケーションのグループ定義」を参照してください。 Service の.metadata.labels に次を指定してください。 • キー名：hmppcto-app-group 値：.spec.template.metadata.labels に指定する同名のキー（hmppcto-app-group）と同じ値を指定してください。
アノテーション	規定しません。 <Prometheus を使用してメトリクス情報を収集する場合> .spec.template.metadata.annotations に次を指定してください。 • キー名：prometheus.io/scrape 値：true 値が true の場合にメトリクスの収集対象になります。 • キー名：prometheus.io/path 値：/hmppcto/metrics※1 メトリクス収集用のエンドポイントです。 • キー名：prometheus.io/port 値：http ポート番号（Prometheus メトリクス収集用）に指定した値 メトリクス収集用のポートです。 • キー名：prometheus.io/scheme

項目	要件
	値：「http」または「https」 メトリクス収集用のスキーマです。
初期化コンテナ構成	Dependency-Checker コンテナ（「6.4.10　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-Service(SQL)コンテナ（「6.4.5　Orchestrator-Service(SQL)コンテナイメージ（SQL-Participant 限定）」を参照してください） • <Elastic Stack 使用時>Filebeat コンテナ Elastic Stack を使用してトラブルシュート情報を収集する場合だけ使用します。
レプリカ数	• スケーリング対応機能を使用する場合：下限「1」、上限「なし※2」 • スケーリング対応機能を使用しない場合：「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。 ＜ Kubernetes リソース取得用ロール＞ • API グループ 空（コアグループ） • リソース services、endpoints、pods • 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。 • Orchestrator-Service(SQL)と Orchestrator-Service(SQL)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※3 の Namespace がすべて同一の場合：

項目	要件
	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-Service(SQL)と Orchestrator-Service(SQL)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※3 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 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 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「180」 terminationGracePeriodSeconds は、「preStop に設定する未決着トランザクション待機スクリプトの最大待機秒数（120 秒）」と「Spring Boot のグレースフルシャットダウンの設定での最大待機秒数（30 秒）」の合計値以上の値を設定してください。推奨値はそれぞれの最大待機時間のデフォルト値の合計にバッファ（30 秒）を加算した値です。

注※1

Kubernetes アプリケーションにコンテキストパス（例：/context-path）を設定している場合、prometheus.io/path にはコンテキストパスを含めた正しいパスを指定してください。

コンテキストパスを指定していない場合、メトリクス収集用のエンドポイントは、/hmppcto/metrics になります。

コンテキストパスを指定していない場合の prometheus.io/path の指定：prometheus.io/path: /hmppcto/metrics

コンテキストパスを指定している場合、メトリクス収集用エンドポイントの先頭にコンテキストパスが追加されます。

コンテキストパスを指定している場合の prometheus.io/path の指定：prometheus.io/path: /context-path/hmppcto/metrics

prometheus.io/path の設定が、実際のアプリケーションのエンドポイントと一致しているかを必ず確認してください。設定が誤っていると、Prometheus によるメトリクスの収集ができません。

注※2

スケーリング対応機能を使用する場合、この Pod が接続する次のリソースとの最大接続数を、デプロイするレプリカの最大数に応じて決定してください。各リソースとの最大接続数に関するパラメタは、「9.4　各 RDBMS の使用に必要なパラメタおよび設定（SQL-Participant 限定）」を参照してください。

・EADS サーバへの最大同時接続数（eads.server.maxConnections）

・DB への最大同時接続数

注※3

Orchestrator-Service(SQL)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 7-24　Orchestrator-Service(SQL)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 7-24　Orchestrator-Service(SQL)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、Orchestrator-Service(SQL)が使用する DB/外部サービス 例を次に示します。 （例） Mediator を使用する場合：my-namespace/statefulsets/mediator EADS を使用する場合：my-namespace/statefulsets/eads-server-eads-server Ext-Cons を使用する場合：my-namespace/deployments/ext-cons Orchestrator-Service(SQL)を使用する場合：my-namespace/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）。
CPU/メモリリソース制限	設定しません（デフォルト）。

表 7-25　Orchestrator-Service(SQL)の Orchestrator-Service(SQL)のコンテナの要件

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	Orchestrator-Service(SQL)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 Namespace 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

項目	要件
	• Orchestrator-Service(SQL)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：自コンテナの server.port パラメタに設定したポート番号 コンテナポート：自コンテナの server.port パラメタに設定したポート番号
コンテキストパス設定	規定しません。 コンテキストパスを設定した場合は、次の項目で設定したコンテキストパスと同じ値を指定してください。 • 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）の第 6 引数（コンテキストパス） • HMP-PCTO の未決着トランザクション待機スクリプト（wait_transactions.sh）の第 6 引数（コンテキストパス） また、コンテキストパスを設定した場合は、次の項目で設定したコンテキストパスを含んだ値を指定してください。 • メトリクス収集用のエンドポイントを設定するアノテーション prometheus.io/path
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。 • 「9.2.1　共通」のパラメタ • 「9.2.2　JDBC Proxy Driver」のパラメタ • 「9.2.3　Orchestrator」のパラメタ • 「9.2.5　SQL-Participant」のパラメタ • 「9.2.7　Spring Boot の組み込み HTTP サーバ」のパラメタ
	Spring Boot のグレースフルシャットダウンの設定に関する次のプロパティを、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。 • プロパティ：spring.lifecycle.timeout-per-shutdown-phase 設定する値：任意の秒数 推奨値：「30s」
	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-Service(SQL)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 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 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。

項目	要件
	• ファイル：uCosminexus Application Runtime 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）：自コンテナの server.port パラメタに設定したポート番号 第 6 引数（コンテキストパス）：コンテキストパスに指定した値。コンテキストパスを指定する場合だけ設定してください。指定しない場合は、この引数を省略してください。 オプション（スキーム）：「--scheme={ http ｜ https }」 自コンテナの接続に使用するスキームを指定します。 このオプションは省略できます。省略した場合、「http」が設定された場合の動作となります。 オプション（証明書検証無効化）：「--insecure」 HTTPS 接続時の証明書検証を無効化します。 このオプションは省略できます。省略した場合、証明書検証が行われます。 • 初回 Probe 実行待機秒数（initialDelaySeconds） 推奨値「150」 • 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）：自コンテナの server.port パラメタに設定したポート番号 第６引数（コンテキストパス：コンテキストパスに指定した値。コンテキストパスを指定する場合だけ設定してください。指定しない場合は、この引数を省略してください。 オプション（スキーム）：「--scheme={ http ┃ https }」 自コンテナの接続に使用するスキームを指定します。 このオプションは省略できます。省略した場合、「http」が設定された場合の動作となります。 オプション（証明書検証無効化）：「--insecure」 HTTPS 接続時の証明書検証を無効化します。 このオプションは省略できます。省略した場合、証明書検証が行われます。 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「10」 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）：自コンテナの server.port パラメタに設定したポート番号 第６引数（コンテキストパス）：コンテキストパスに指定した値。コンテキストパスを指定する場合だけ設定してください。指定しない場合は、この引数を省略してください。 オプション（スキーム）：「--scheme={ http ┃ https }」 自コンテナの接続に使用するスキームを指定します。 このオプションは省略できます。省略した場合、「http」が設定された場合の動作となります。 オプション（証明書検証無効化）：「--insecure」 HTTPS 接続時の証明書検証を無効化します。 このオプションは省略できます。省略した場合、証明書検証が行われます。 未決着トランザクション待機スクリプトの最大待機時間は「開始時待機秒数（20 秒）」と「リトライ回数×リトライ間隔秒数」を合計した値です。最大待機時間がトランザクション決着監視タイマの設定値を超えるように引数を調整してください。
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。

Elastic Stack を使用してトラブルシュート情報を収集する場合だけ、次の Filebeat コンテナの要件を満たすように作成します。

表 7-26　Orchestrator-Service(SQL)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「3.11.1　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」
CPU/メモリリソース制限	規定しません。

7.6.6　Alternate-Service の要件

Alternate-Service は、役割がトランザクションルートまたはトランザクションブランチに変わる Kubernetes アプリケーションです。

この Kubernetes アプリケーションは、次の表に示す Alternate-Service の要件を満たすように作成します。

表 7-27　Alternate-Service 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。

項目	要件
ネームスペース	規定しません。
ラベル	ワークロードリソース（Deployment など） の.spec.template.metadata.labels に次を指定してください。※1 • キー名：hmppcto-app-group 値：「アプリケーショングループ名」を表す文字列 hmppcto-app-group パラメタの説明については、「付録 B.1　HMP-PCTO 固有の指定が必要なパラメタ」を参照してください。 「アプリケーショングループ」の考え方については、「7.2　Kubernetes アプリケーションのグループ定義」を参照してください。 Service の.metadata.labels に次を指定してください。 • キー名：hmppcto-app-group 値：.spec.template.metadata.labels に指定する同名のキー（hmppcto-app-group）と同じ値を指定してください。
アノテーション	規定しません。 <Prometheus を使用してメトリクス情報を収集する場合> .spec.template.metadata.annotations に次を指定してください。 • キー名：prometheus.io/scrape 値：「true」 値が true の場合にメトリクスの収集対象になります。 • キー名：prometheus.io/path 値：「/hmppcto/metrics※2」 メトリクス収集用のエンドポイントです。 • キー名：prometheus.io/port 値：http ポート番号（Prometheus メトリクス収集用）に指定した値 メトリクス収集用のポートです。 • キー名：prometheus.io/scheme 値：「http」または「https」 メトリクス収集用のスキーマです。
初期化コンテナ構成	Dependency-Checker コンテナ（「6.4.10　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	• Alternate-Service コンテナ（「6.4.6　Alternate-Service コンテナイメージ」を参照してください） • <Elastic Stack 使用時>Filebeat コンテナ Elastic Stack を使用してトラブルシュート情報を収集する場合だけ使用します。
レプリカ数	• スケーリング対応機能を使用する場合：下限「1」、上限「なし」 • スケーリング対応機能を使用しない場合：「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。

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。 ＜ Kubernetes リソース取得用ロール＞ • API グループ 空（コアグループ） • リソース services、endpoints、pods • 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。 • Alternate-Service と Alternate-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※3 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。 • Alternate-Service と Alternate-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※3 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 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 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「180」

項目	要件
	terminationGracePeriodSeconds は、「preStop に設定する未決着トランザクション待機スクリプトの最大待機秒数（120 秒）」と「Spring Boot のグレースフルシャットダウンの設定での最大待機秒数（30 秒）」の合計値以上の値を設定してください。推奨値はそれぞれの最大待機時間のデフォルト値の合計にバッファ（30 秒）を加算した値です。

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

Kubernetes アプリケーションにコンテキストパス（例：/context-path）を設定している場合、prometheus.io/path にはコンテキストパスを含めた正しいパスを指定してください。

コンテキストパスを指定していない場合、メトリクス収集用のエンドポイントは、/hmppcto/metrics になります。

コンテキストパスを指定していない場合の prometheus.io/path の指定：prometheus.io/path: /hmppcto/metrics

コンテキストパスを指定している場合、メトリクス収集用エンドポイントの先頭にコンテキストパスが追加されます。

コンテキストパスを指定している場合の prometheus.io/path の指定：prometheus.io/path: /context-path/hmppcto/metrics

prometheus.io/path の設定が、実際のアプリケーションのエンドポイントと一致しているかを必ず確認してください。設定が誤っていると、Prometheus によるメトリクスの収集ができません。

注※3

Alternate-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 7-28　Alternate-Service 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 7-28　Alternate-Service 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、Entity-Service(SQL)、Entity-Service(TCC)、または TP1-Bridge 例を次に示します。 （例） Mediator を使用する場合：my-namespace/statefulsets/mediator

項目	要件
	Entity-Service(SQL)を使用する場合：my-namespace/deployments/sql-participant Entity-Service(TCC)を使用する場合：my-namespace/deployments/tcc-participant TP1-Bridge 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 7-29　Alternate-Service の Alternate-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：自コンテナの server.port パラメタに設定したポート番号 コンテナポート：自コンテナの server.port パラメタに設定したポート番号
コンテキストパス設定	規定しません。 コンテキストパスを設定した場合は、次の項目で設定したコンテキストパスと同じ値を指定してください。 • 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）の第 6 引数（コンテキストパス） • HMP-PCTO の未決着トランザクション待機スクリプト（wait_transactions.sh）の第 6 引数（コンテキストパス） また、コンテキストパスを設定した場合は、次の項目で設定したコンテキストパスを含んだ値を指定してください。 • メトリクス収集用のエンドポイントを設定するアノテーション prometheus.io/path

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。 • 「9.2.1　共通」のパラメタ • 「9.2.3　Orchestrator」のパラメタ • 「9.2.4　Entity-Module」のパラメタ • 「9.2.7　Spring Boot の組み込み HTTP サーバ」のパラメタ
	Spring Boot のグレースフルシャットダウンの設定に関する次のプロパティを、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。 • プロパティ：spring.lifecycle.timeout-per-shutdown-phase 設定する値：任意の秒数 推奨値：「30s」
	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 Alternate-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 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 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 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）：自コンテナの server.port パラメタに設定したポート番号

項目	要件
	第 6 引数（コンテキストパス）：コンテキストパスに指定した値。コンテキストパスを指定する場合だけ設定してください。指定しない場合は、この引数を省略してください。 オプション（スキーム）：「--scheme={ http ｜ https }」 自コンテナの接続に使用するスキームを指定します。 このオプションは省略できます。省略した場合、「http」が設定された場合の動作となります。 オプション（証明書検証無効化）：「--insecure」 HTTPS 接続時の証明書検証を無効化します。 このオプションは省略できます。省略した場合、証明書検証が行われます。 • 初回 Probe 実行待機秒数（initialDelaySeconds） 推奨値「150」 • 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）：自コンテナの server.port パラメタに設定したポート番号 第 6 引数（コンテキストパス）：コンテキストパスに指定した値。コンテキストパスを指定する場合だけ設定してください。指定しない場合は、この引数を省略してください。 オプション（スキーム）：「--scheme={ http ｜ https }」 自コンテナの接続に使用するスキームを指定します。 このオプションは省略できます。省略した場合、「http」が設定された場合の動作となります。 オプション（証明書検証無効化）：「--insecure」 HTTPS 接続時の証明書検証を無効化します。 このオプションは省略できます。省略した場合、証明書検証が行われます。 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）

項目	要件
	第 1 引数（リトライ回数）：推奨値「10」 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）：自コンテナの server.port パラメタに設定したポート番号 第 6 引数（コンテキストパス）：コンテキストパスに指定した値。コンテキストパスを指定する場合だけ設定してください。指定しない場合は、この引数を省略してください。 オプション（スキーム）：「--scheme={ http ｜ https }」 自コンテナの接続に使用するスキームを指定します。 このオプションは省略できます。省略した場合、「http」が設定された場合の動作となります。 オプション（証明書検証無効化）：「--insecure」 HTTPS 接続時の証明書検証を無効化します。 このオプションは省略できます。省略した場合、証明書検証が行われます。 未決着トランザクション待機スクリプトの最大待機時間は「開始時待機秒数（20 秒）」と「リトライ回数×リトライ間隔秒数」を合計した値です。最大待機時間がトランザクション決着監視タイマの設定値を超えるように引数を調整してください。
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。

Elastic Stack を使用してトラブルシュート情報を収集する場合だけ、次の Filebeat コンテナの要件を満たすように作成します。

表 7-30　Alternate-Service 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「3.11.1　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」
CPU/メモリリソース制限	規定しません。

7.6.7　Alternate-Service(SQL)の要件（SQL-Participant 限定）

Alternate-Service(SQL)は、Alternate-Service と同様に、役割がトランザクションルートまたはトランザクションブランチに変わる Kubernetes アプリケーションです。Alternate-Service の役割に加えて、データベースの参照、更新業務も行います。

この Kubernetes アプリケーションは、次の表に示す Alternate-Service(SQL)の要件を満たすように作成します。

表 7-31　Alternate-Service(SQL)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	Alternate-Service(SQL)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 Namespace 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、Namespace 名を検討してください。
ラベル	ワークロードリソース（Deployment など） の.spec.template.metadata.labels に次を指定してください。※1 • キー名：app.kubernetes.io/name 値：app.kubernetes.io/name パラメタの説明については、「付録 B.1　HMP-PCTO 固有の指定が必要なパラメタ」を参照してください。 • キー名：hmppcto-app-group 値：「アプリケーショングループ名」を表す文字列 hmppcto-app-group パラメタの説明については、「付録 B.1　HMP-PCTO 固有の指定が必要なパラメタ」を参照してください。

項目	要件
	「アプリケーショングループ」の考え方については、「7.2　Kubernetes アプリケーションのグループ定義」を参照してください。 Service の.metadata.labels に次を指定してください。 • キー名：hmppcto-app-group 値：.spec.template.metadata.labels に指定する同名のキー（hmppcto-app-group）と同じ値を指定してください。
アノテーション	規定しません。 <Prometheus を使用してメトリクス情報を収集する場合> .spec.template.metadata.annotations に次を指定してください。 • キー名：prometheus.io/scrape 値：「true」 値が true の場合にメトリクスの収集対象になります。 • キー名：prometheus.io/path 値：「/hmppcto/metrics※1」 メトリクス収集用のエンドポイントです。 • キー名：prometheus.io/port 値：http ポート番号（Prometheus メトリクス収集用）に指定した値 メトリクス収集用のポートです。 • キー名：prometheus.io/scheme 値：「http」または「https」 メトリクス収集用のスキーマです。
初期化コンテナ構成	Dependency-Checker コンテナ（「6.4.10　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	• Alternate-Service(SQL)コンテナ（「6.4.7　Alternate-Service(SQL)コンテナイメージ（SQL-Participant 限定）」を参照してください） • <Elastic Stack 使用時>Filebeat コンテナ Elastic Stack を使用してトラブルシュート情報を収集する場合だけ使用します。
レプリカ数	• スケーリング対応機能を使用する場合：下限「1」、上限「なし※2」 • スケーリング対応機能を使用しない場合：「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。 ＜ Kubernetes リソース取得用ロール＞ • API グループ 空（コアグループ） • リソース

項目	要件
	services、endpoints、pods • 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。 • Alternate-Service(SQL)と Alternate-Service(SQL)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※3 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。 • Alternate-Service(SQL)と Alternate-Service(SQL)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※3 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 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 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「180」 terminationGracePeriodSeconds は、「preStop に設定する未決着トランザクション待機スクリプトの最大待機秒数（120 秒）」と「Spring Boot のグレースフルシャットダウンの設定での最大待機秒数（30 秒）」の合計値以上の値を設定してください。推奨値はそれぞれの最大待機時間のデフォルト値の合計にバッファ（30 秒）を加算した値です。

注※1

Kubernetes アプリケーションにコンテキストパス（例：/context-path）を設定している場合、prometheus.io/path にはコンテキストパスを含めた正しいパスを指定してください。

コンテキストパスを指定していない場合、メトリクス収集用のエンドポイントは、/hmppcto/metrics になります。

コンテキストパスを指定していない場合の prometheus.io/path の指定：prometheus.io/path: /hmppcto/metrics

コンテキストパスを指定している場合、メトリクス収集用エンドポイントの先頭にコンテキストパスが追加されます。

コンテキストパスを指定している場合の prometheus.io/path の指定：prometheus.io/path: /context-path/hmppcto/metrics

prometheus.io/path の設定が、実際のアプリケーションのエンドポイントと一致しているかを必ず確認してください。設定が誤っていると、Prometheus によるメトリクスの収集ができません。

注※2

スケーリング対応機能を使用する場合、この Pod が接続する次のリソースとの最大接続数を、デプロイするレプリカの最大数に応じて決定してください。各リソースとの最大接続数に関するパラメタは、「9.4　各 RDBMS の使用に必要なパラメタおよび設定（SQL-Participant 限定）」を参照してください。

・EADS サーバへの最大同時接続数（eads.server.maxConnections）

・DB への最大同時接続数

注※3

Alternate-Service(SQL)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 7-32　Alternate-Service(SQL)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 7-32　Alternate-Service(SQL)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、Alternate-Service(SQL)が使用する DB/外部サービス 例を次に示します。 （例） Mediator を使用する場合：my-namespace/statefulsets/mediator EADS を使用する場合：my-namespace/statefulsets/eads-server-eads-server Ext-Cons を使用する場合：my-namespace/deployments/ext-cons Alternate-Service(SQL)を使用する場合：my-namespace/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）

項目	要件
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 7-33　Alternate-Service(SQL)の Alternate-Service(SQL)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	Alternate-Service(SQL)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 Namespace 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 • Alternate-Service(SQL)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：自コンテナの server.port パラメタに設定したポート番号 コンテナポート：自コンテナの server.port パラメタに設定したポート番号
コンテキストパス設定	規定しません。 コンテキストパスを設定した場合は、次の項目で設定したコンテキストパスと同じ値を指定してください。 • 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）の第 6 引数（コンテキストパス） • HMP-PCTO の未決着トランザクション待機スクリプト（wait_transactions.sh）の第 6 引数（コンテキストパス） また、コンテキストパスを設定した場合は、次の項目で設定したコンテキストパスを含んだ値を指定してください。 • メトリクス収集用のエンドポイントを設定するアノテーション prometheus.io/path
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。 •「9.2.1　共通」のパラメタ •「9.2.2　JDBC Proxy Driver」のパラメタ •「9.2.3　Orchestrator」のパラメタ •「9.2.4　Entity-Module」のパラメタ •「9.2.5　SQL-Participant」のパラメタ •「9.2.7　Spring Boot の組み込み HTTP サーバ」のパラメタ

項目	要件
	Spring Boot のグレースフルシャットダウンの設定に関する次のプロパティを、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。 • プロパティ：spring.lifecycle.timeout-per-shutdown-phase 設定する値：任意の秒数 推奨値：「30s」
	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 Alternate-Service(SQL)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 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 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 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）：自コンテナの server.port パラメタに設定したポート番号 第 6 引数（コンテキストパス）：コンテキストパスに指定した値。コンテキストパスを指定する場合だけ設定してください。指定しない場合は、この引数を省略してください。 オプション（スキーム）：「--scheme={ http ┃ https }」 自コンテナの接続に使用するスキームを指定します。 このオプションは省略できます。省略した場合、「http」が設定された場合の動作となります。 オプション（証明書検証無効化）：「--insecure」

項目	要件
	HTTPS 接続時の証明書検証を無効化します。 このオプションは省略できます。省略した場合、証明書検証が行われます。 • 初回 Probe 実行待機秒数（initialDelaySeconds） 推奨値「150」 • 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）：自コンテナの server.port パラメタに設定したポート番号 第 6 引数（コンテキストパス）：コンテキストパスに指定した値。コンテキストパスを指定する場合だけ設定してください。指定しない場合は、この引数を省略してください。 オプション（スキーム）：「--scheme={ http ｜ https }」 自コンテナの接続に使用するスキームを指定します。 このオプションは省略できます。省略した場合、「http」が設定された場合の動作となります。 オプション（証明書検証無効化）：「--insecure」 HTTPS 接続時の証明書検証を無効化します。 このオプションは省略できます。省略した場合、証明書検証が行われます。 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「10」 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）：自コンテナの server.port パラメタに設定したポート番号

項目	要件
	第 6 引数（コンテキストパス）：コンテキストパスに指定した値。コンテキストパスを指定する場合だけ設定してください。指定しない場合は、この引数を省略してください。 オプション（スキーム）：「--scheme={ http ｜ https }」 自コンテナの接続に使用するスキームを指定します。 このオプションは省略できます。省略した場合、「http」が設定された場合の動作となります。 オプション（証明書検証無効化）：「--insecure」 HTTPS 接続時の証明書検証を無効化します。 このオプションは省略できます。省略した場合、証明書検証が行われます。 未決着トランザクション待機スクリプトの最大待機時間は「開始時待機秒数（20 秒）」と「リトライ回数×リトライ間隔秒数」を合計した値です。最大待機時間がトランザクション決着監視タイマの設定値を超えるように引数を調整してください。
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。

Elastic Stack を使用してトラブルシュート情報を収集する場合だけ、次の Filebeat コンテナの要件を満たすように作成します。

表 7-34　Alternate-Service(SQL)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「3.11.1　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」

項目	要件
	• 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。

7.6.8 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。レプリカ数の要件については、取扱説明書「HMP-PCTO テクニカルガイド」の Mediator 数の設定要件に関する説明を参照して決定します。

表 7-35 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	各 Kubernetes リソースの 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 値：「*V.R.S*※1」 備考：バージョン • キー名：helm.sh/chart 値：「mediator-*V.R.S*※1」 備考：<チャート名>-<バージョン> StatefulSet の.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

項目	要件
	• キー名：hmppcto-app-group 値：「アプリケーショングループ名」を表す文字列 hmppcto-app-group パラメタの説明については、「付録 B.1　HMP-PCTO 固有の指定が必要なパラメタ」を参照してください。 「アプリケーショングループ」の考え方については、「7.2　Kubernetes アプリケーションのグループ定義」を参照してください。 Service の.metadata.labels に次を指定してください。 • キー名：hmppcto-app-group 値：.spec.template.metadata.labels に指定する同名のキー（hmppcto-app-group）と同じ値を指定してください。
アノテーション	Prometheus を使用してメトリクス情報を収集するために.spec.template.metadata.annotations に次を指定します。 • キー名：prometheus.io/scrape 値：「true」 値が true の場合にメトリクスの収集対象になります。 • キー名：prometheus.io/path 値：「/hmppcto/metrics」 メトリクス収集用のエンドポイントです。 • キー名：prometheus.io/port 値：http ポート番号（Prometheus メトリクス収集用）に指定した値 メトリクス収集用のポートです。 • キー名：prometheus.io/scheme 値：「http」 メトリクス収集用のスキーマです。
初期化コンテナ構成	Dependency-Checker コンテナ（「6.4.10　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（「6.4.8　Mediator コンテナイメージ」を参照してください）
レプリカ数	• ThreeQuarters 合意方式の場合：5～64 • PrivilegedValue 合意方式の場合：2～64
Service 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＞

項目	要件
	• 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。 • 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※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。 • 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※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	• 可用性の観点から、Pod の数だけワーカーノードを用意し、各ワーカーノードに Pod を 1 つずつ配置することを推奨します。 なお、インフラストラクチャのコスト削減の観点などで、用意するワーカーノードの数が Pod の数を下回る場合でも、Mediator は問題なく動作しますが、ワーカーノード数を削減する代わりに、ノード障害時のトランザクション決着（補助合意）に時間が掛かるおそれがあります。 また、可用性の観点から、各ワーカーノードに配置する Pod の数は均等にすることを推奨します。 • Mediator の ConsensusLog の永続ボリュームにローカル永続ボリューム（local）を使用する場合、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＞

項目	要件
	• 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。 • Linux のカーネルは、NFS に対するダイレクト I/O に対応しているため、NFS ストレージとして使用するダイレクト I/O にも対応している必要があります。NFS ストレージとして使用するストレージまたはストレージサービスは、問いません。
	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「150」 terminationGracePeriodSeconds は、「preStop に設定する未決着トランザクション待機スクリプトの最大待機秒数（120 秒）」以上の値を設定してください。推奨値は最大待機時間のデフォルト値の合計にバッファ（30 秒）を加算した値です。

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

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 7-36 Mediator の Dependency-Checker コンテナの要件」を参照してください。

表 7-36　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＞ Elastic Stack、Jaeger および Prometheus を使用してトラブルシュート情報を収集する場合、dependencyCheck.enabled パラメタに true を設定し、dependencyCheck.targets パラメタに対象のトラブルシュート情報を収集する際に必要なアプリケーションを指定してください。 この項目は、クラウド環境（Amazon Web Services 環境で Jaeger を使用する場合、および Microsoft Azure 環境で Jaeger と Elasticsearch を使用する場合）にも適用されます。

項目	要件
	Google Cloud Platform 環境でトラブルシュート情報を収集する場合は、Elastic Stack、Jaeger および Prometheus の Pod のデプロイを待機する必要がありません。その場合は dependencyCheck.enabled パラメタに false を設定してください。 記載例については、「3.8.2　Mediator の Helm チャートのカスタマイズ」を参照してください。
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 7-37　Mediator の 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 および 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：自コンテナの server.port パラメタに設定したポート番号 コンテナポート：自コンテナの server.port パラメタに設定したポート番号
コンテキストパス設定	設定しません。 Mediator のエンドポイントについて、ユーザがコンテキストパスを意識する必要はないため、設定する必要はありません。
コンフィグ設定	次の HMP-PCTO のパラメタを、Helm チャートの values.yaml に設定してください。 • 「9.2.1　共通」のパラメタ • 「9.3.1　Mediator」のパラメタ
	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。

項目	要件
	• 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 に true を設定する場合）は、次のとおりファイルを配置します。 • ファイル：接続元クライアント（Orchestrator および 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

項目	要件
	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）：自コンテナの server.port パラメタに設定したポート番号 第 6 引数（コンテキストパス）：設定しません。 オプション（スキーム）：設定しません。 オプション（証明書検証無効化）：設定しません。 • 初回 Probe 実行待機秒数（initialDelaySeconds） 推奨値「150」 • 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）：自コンテナの server.port パラメタに設定したポート番号 第 6 引数（コンテキストパス）：設定しません。 オプション（スキーム）：設定しません。 オプション（証明書検証無効化）：設定しません。 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「10」

項目	要件
	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）：自コンテナの server.port パラメタに設定したポート番号 第 6 引数（コンテキストパス）：設定しません。 オプション（スキーム）：設定しません。 オプション（証明書検証無効化）：設定しません。 未決着トランザクション待機スクリプトの最大待機時間は「開始時待機秒数（20 秒）」と「リトライ回数×リトライ間隔秒数」を合計した値になります。最大待機時間がトランザクション決着監視タイマの設定値を超えるように引数を調整してください。
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 7-38　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
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

7.6.9　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 7-39　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	各 Kubernetes リソースの 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：「tp1-bridge」 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：「tp1-bridge-*V.R.S*※1」 備考：<チャート名>-<バージョン> StatefulSet の.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：「tp1-bridge」 備考：チャート名 • キー名：hmppcto-app-group 値：「アプリケーショングループ名」を表す文字列 hmppcto-app-group パラメタの説明については、「付録 B.1　HMP-PCTO 固有の指定が必要なパラメタ」を参照してください。 「アプリケーショングループ」の考え方については、「7.2　Kubernetes アプリケーションのグループ定義」を参照してください。 Service の.metadata.labels に次を指定してください。 • キー名：hmppcto-app-group 値：.spec.template.metadata.labels に指定する同名のキー（hmppcto-app-group）と同じ値を指定してください。
アノテーション	Prometheus を使用してメトリクス情報を収集するために.spec.template.metadata.annotations に次を指定します。

項目	要件
	• キー名：prometheus.io/scrape 値：「true」 値が true の場合にメトリクスの収集対象になります。 • キー名：prometheus.io/path 値：「/hmppcto/metrics」 メトリクス収集用のエンドポイントです。 • キー名：prometheus.io/port 値：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、アプリケーションからの OpenTP1 のサービス呼び出し要求の受信用）に指定した値 メトリクス収集用のポートです。 • キー名：prometheus.io/scheme 値：「http」 メトリクス収集用のスキーマです。
初期化コンテナ構成	Dependency-Checker コンテナ（「6.4.10　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（「6.4.9　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） • <Elastic Stack 使用時>Filebeat コンテナ Elastic Stack を使用してトラブルシュート情報を収集する場合だけ使用します。
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。 ＜ Kubernetes リソース取得用ロール＞ • API グループ 空（コアグループ） • リソース services、endpoints、pods • 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

項目	要件
	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。 • 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「150」 terminationGracePeriodSeconds は、「preStop に設定する未決着トランザクション待機スクリプトの最大待機秒数（120 秒)」以上の値を設定してください。推奨値はそれぞれの最大待機時間のデフォルト値の合計にバッファ（30 秒）を加算した値です。

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 7–40 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 7-40　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） 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（通常版の場合だけ）、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-eads-server（通常版の場合だけ） 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 7-41　TP1-Bridge の 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 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：自コンテナの http.port パラメタに設定したポート番号 コンテナポート：自コンテナの http.port パラメタに設定したポート番号
コンテキストパス設定	設定しません。

項目	要件
	TP1-Bridge の処理を呼び出すためのエンドポイントは固定のため、コンテキストパスを設定できません。 エンドポイントの詳細は「2.2.15　TP1-Bridge の使用（通常版かつ TP1-Bridge 限定）」の「(1)　コーディング例」を参照してください。
コンフィグ設定	次の HMP-PCTO のパラメタを、Helm チャートの values.yaml に設定してください。 • 「9.2.1　共通」のパラメタ • 「9.3.2　TP1-Bridge（通常版かつ TP1-Bridge 限定）」のパラメタ
	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 「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。
	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。

項目	要件
	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）：自コンテナの http.port パラメタに設定したポート番号 第 6 引数（コンテキストパス）：設定しません。 オプション（スキーム）：設定しません。 オプション（証明書検証無効化）：設定しません。 • 初回 Probe 実行待機秒数（initialDelaySeconds） 推奨値「150」 • 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）：自コンテナの http.port パラメタに設定したポート番号

項目	要件
	第 6 引数（コンテキストパス）：設定しません。 オプション（スキーム）：設定しません。 オプション（証明書検証無効化）：設定しません。 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「10」 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）：自コンテナの http.port パラメタに設定したポート番号 第 6 引数（コンテキストパス）：設定しません。 オプション（スキーム）：設定しません。 オプション（証明書検証無効化）：設定しません。 未決着トランザクション待機スクリプトの最大待機時間は「開始時待機秒数（20 秒）」と「リトライ回数×リトライ間隔秒数」を合計した値になります。最大待機時間がトランザクション決着監視タイマの設定値を超えるように引数を調整してください。
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。

Elastic Stack を使用してトラブルシュート情報を収集する場合だけ、次の Filebeat コンテナの要件を満たすように作成します。

表 7-42　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「3.11.1　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
	• 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） 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 7-43　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.tpl	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

ディレクトリ/ファイル			説明
		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
	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

7.6.10　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 7–44　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

項目	要件
	• 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（「6.4.10　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	Ext-Cons コンテナ（「6.4.11　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 7-45　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） ＜対象 Kubernetes アプリケーション＞ トラブルシュート情報を収集する場合に必要なアプリケーションを指定してください。 Google Cloud Platform 環境でトラブルシュート情報を収集する場合は、Elastic Stack、Jaeger および Prometheus の Pod のデプロイを待機する必要がありません。その場合は dependencyCheck.enabled パラメタに false を設定してください。 記載例については、「3.8.2　Mediator の Helm チャートのカスタマイズ」を参照してください。
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 7-46　Ext-Cons の 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 自コンテナの server.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 自コンテナの 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。 •「9.2.1　共通」のパラメタ •「9.3.4　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 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 7-47　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

ディレクトリ/ファイル	説明
values.schema.json	values.yaml の設定値のバリデーション用ファイル

7.6.11　Filebeat の要件

Elastic Stack を使用してトラブルシュート情報を収集する場合だけ使用します。次の表に示す要件を満たすように作成します。

表 7-48　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	規定しません。
ラベル	規定しません。
アノテーション	規定しません。
初期化コンテナ構成	規定しません。
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。 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。Red Hat OpenShift Container Platform 環境で使用する場合は、Red Hat OpenShift Container Platform の SCC の設定に応じて Pod の ServiceAccount に必要な権限を付与してください。
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。 ＜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。

項目	要件
	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」

コンテナごとの要件を次に示します。

表 7-49　Filebeat 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	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	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。

7.6.12　Jaeger の要件

Jaeger を使用してトラブルシュート情報を収集する場合だけ使用します。次の表に示す要件を満たすように作成します。この要件は、クラウド環境（例：Amazon Web Services 環境）で Jaeger を使用する場合にも適用されます。

表 7-50　Jaeger の要件

項目	要件
ワークロードリソース	規定しません。
ネームスペース	規定しません。
ラベル	規定しません。
アノテーション	規定しません。
初期化コンテナ構成	規定しません。
コンテナ構成	Jaeger 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。 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。Red Hat OpenShift Container Platform 環境で使用する場合は、Red Hat OpenShift Container Platform の SCC の設定に応じて Pod の ServiceAccount に必要な権限を付与してください。
アクセス制御	規定しません。
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」

コンテナごとの要件を次に示します。

表 7-51　Jaeger の Jaeger コンテナの要件

項目	要件
コマンド	規定しません。
ネットワーク設定	デプロイしたノードの IP で、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、または Mediator 向け） • プロトコル TCP • コンテナポート 4318

項目	要件
	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	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。

7.6.13 Prometheus の要件

Prometheus を使用してトラブルシュート情報を収集する場合だけ使用します。次の表に示す要件を満たすように作成します。

表 7-52 Prometheus の要件

項目	要件
ワークロードリソース	規定しません。
ネームスペース	規定しません。
ラベル	規定しません。
アノテーション	規定しません。
初期化コンテナ構成	規定しません。
コンテナ構成	Prometheus 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。 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。Red Hat OpenShift Container Platform 環境で使用する場合は、Red Hat OpenShift Container Platform の SCC の設定に応じて Pod の ServiceAccount に必要な権限を付与してください。

項目	要件
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。 ＜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」

コンテナごとの要件を次に示します。

表 7-53　Prometheus の 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。
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。

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	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。

7.6.14 Metricbeat の要件

Elastic Stack を使用してトラブルシュート情報を収集する場合だけ使用します。次の表に示す要件を満たすように作成します。

表 7-54 Metricbeat の要件

項目	要件
ワークロードリソース	規定しません。
ネームスペース	規定しません。
ラベル	規定しません。
アノテーション	規定しません。
初期化コンテナ構成	規定しません。
コンテナ構成	Metricbeat 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。 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。Red Hat OpenShift Container Platform 環境で使用する場合は、Red Hat OpenShift Container Platform の SCC の設定に応じて Pod の ServiceAccount に必要な権限を付与してください。
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。 <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

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 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」

コンテナごとの要件を次に示します。

表 7-55　Metricbeat の 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）。 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	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。

7.6.15　Elasticsearch の要件

Elastic Stack を使用してトラブルシュート情報を収集する場合だけ使用します。次の表に示す要件を満たすように作成します。この要件は、クラウド環境（例：Microsoft Azure 環境）で Elasticsearch を使用する場合にも適用されます。

表 7-56　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	規定しません。
ラベル	規定しません。
アノテーション	規定しません。
初期化コンテナ構成	規定しません。
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。
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。 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。Red Hat OpenShift Container Platform 環境で使用する場合は、Red Hat OpenShift Container Platform の SCC の設定に応じて Pod の ServiceAccount に必要な権限を付与してください。
アクセス制御	規定しません。
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	• 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。

項目	要件
	• 安全な停止待機秒数（terminationGracePeriodSeconds） 推奨値「120」

コンテナごとの要件を次に示します。

表 7-57　Elasticsearch の 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	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。

7.6.16　Logstash の要件

Elastic Stack を使用してトラブルシュート情報を収集する場合だけ使用します。次の表に示す要件を満たすように作成します。

表 7-58　Logstash の要件

項目	要件
ワークロードリソース	規定しません。
ネームスペース	規定しません。
ラベル	規定しません。
アノテーション	規定しません。
初期化コンテナ構成	規定しません。
コンテナ構成	Logstash コンテナ

項目	要件
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。 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。Red Hat OpenShift Container Platform 環境で使用する場合は、Red Hat OpenShift Container Platform の SCC の設定に応じて Pod の ServiceAccount に必要な権限を付与してください。
アクセス制御	規定しません。
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「120」

コンテナごとの要件を次に示します。

表 7-59　Logstash の 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	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。

7.6.17　Metrics Server の要件

スケーリング対応機能で利用するメトリクス情報を Metrics Server で収集する場合、公式ドキュメントを参照して Metrics Server を作成してください。利用できる Metrics Server のバージョンについては、リリースノートを参照してください。

このとき、「9.3.10　Metrics Server」を参照し記載のある設定項目に適切な値を設定して、Metrics Server を作成してください。

7.7 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。

7.7.1 デプロイ依存関係チェックスクリプト（dependency_check.sh）

(1) 機能

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。

＜コマンド処理内容＞

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。

(2) 形式

コマンドは、次のどちらかの形式で指定します。

<形式 1>

```
dependency_check.sh <retry-count>
                    <retry-interval-seconds>
                    <connect-timeout-seconds>
                    <read-timeout-seconds>
                    {{<namespace>/<resource>/{<name>|<label-selector>}} [...]}
```

<形式 2>

```
dependency_check.sh --version
```

(3) オプション

コマンドのオプションを次の表に示します。

表 7-60　デプロイ依存関係チェックスクリプトのコマンドのオプション一覧

オプション	説明	必須／省略可	指定範囲	デフォルト値
<retry-count>	第 1 引数としてリクエストの最大リトライ回数を指定します。	必須	0～1,080	－
<retry-interval-seconds>	第 2 引数としてリクエストのリトライ間隔秒数を指定します。	必須	0～360	－
<connect-timeout-seconds>	第 3 引数としてリクエストの接続タイムアウト秒数を指定します。	必須	1～360	－
<read-timeout-seconds>	第 4 引数としてリクエストの応答タイムアウト秒数を指定します。	必須	1～360	－
{{<namespace>/<resource>/{<name>|<label-selector>}} [...]}	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。 • <namespace> ターゲット Kubernetes オブジェクトの Namespace を指定します。 • <resource> ターゲット Kubernetes オブジェクトの Kubernetes リソース種別を次の中から指定します。 ・pods ・replicasets ・deployments ・statefulsets ・daemonsets • <name> 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。 • <label-selector> 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。 • 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	必須	－	－

オプション	説明	必須／省略可	指定範囲	デフォルト値
	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。			
--version	この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。ほかのオプションは無視されます。	省略可	－	－

＜オプション指定例＞

- 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-eads-server ¥
  my-namespace/statefulsets/postgresql
```

- バージョン表示オプションを指定する例：

```
dependency_check.sh --version
```

＜オプションのチェック内容＞

次の条件をどれか 1 つでも満たしていない場合は、KFSG62109-E 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。

- 必須オプションが未指定の場合
- オプションの値が指定範囲外の場合
- 存在しないオプションを指定した場合

(4) 出力例

コマンドの出力例を示します。

- コマンド正常終了時の例：

```
KFSG82104-I The dependency check processing will now start. (retry count = 30, retry inte
rval seconds = 10, connect timeout seconds = 10, read timeout seconds = 10, targets = my-
namespace/deployments/sql-participant)
KFSG62111-W The target Kubernetes object is not ready yet. (target = my-namespace/deploym
ents/sql-participant)
```

```
KFSG62111-W The target Kubernetes object is not ready yet. (target = my-namespace/deploym
ents/sql-participant)
KFSG62111-W The target Kubernetes object is not ready yet. (target = my-namespace/deploym
ents/sql-participant)
KFSG82105-I The dependency check processing finished.
```

- バージョン表示オプション指定時の例：

```
dependency_check.sh 1.0.0
```

- オプション指定誤り時の例：

```
KFSG62109-E One or more command arguments are invalid. (command line = dependency_check.s
h )
Usage: dependency_check.sh <retry-count> <retry-interval-seconds> <connect-timeout-second
s> <read-timeout-seconds> {<namespace>/<resource>/[<name>|<label-selector>]} [...]
       dependency_check.sh --version
```

(5) リターンコード

コマンドのリターンコードを次の表に示します。

表 7-61 デプロイ依存関係チェックスクリプトのコマンドのリターンコード

リターンコード	説明
0	コマンドの実行に成功しました。
1	コマンドの実行に失敗しました。

7.7.2 トランザクションヘルスチェックスクリプト（health_check.sh）

(1) 機能

Orchestrator-Service、Relay-Service、Entity-Service(SQL)、Entity-Service(TCC)、Orchestrator-Service(SQL)、Alternate-Service、Alternate-Service(SQL)、Mediator、および TP1-Bridge がトランザクションを受け付けられる状態になるまで postStart フックで待機し、Liveness Probe で Pod の起動中にトランザクションを受け付けられるかを定期的に確認するスクリプトです。

＜コマンド処理内容＞

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「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。

(2) 形式

コマンドは、次のどちらかの形式で指定します。

<形式 1>

```
health_check.sh <retry-count>
                <retry-interval-seconds>
                <connect-timeout-seconds>
                <read-timeout-seconds>
                <http-port>
                [<context-path>]
                [--scheme={http|https}]
                [--insecure]
```

<形式 2>

```
health_check.sh --version
```

(3) オプション

コマンドのオプションを次の表に示します。

表 7-62 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のオプション一覧

オプション	説明	必須／省略可	指定範囲	デフォルト値
<retry-count>	第 1 引数としてリクエストの最大リトライ回数を指定します。	必須	0～1,080	－
<retry-interval-seconds>	第 2 引数としてリクエストのリトライ間隔秒数を指定します。	必須	0～360	－
<connect-timeout-seconds>	第 3 引数としてリクエストの接続タイムアウト秒数を指定します。	必須	1～360	－
<read-timeout-seconds>	第 4 引数としてリクエストの応答タイムアウト秒数を指定します。	必須	1～360	－
<http-port>	第 5 引数として Spring Boot アプリケーションの HTTP ポート番号を指定します。	必須	1,024～65,535	－
<context-path>	第 6 引数として Spring Boot アプリケーションのコンテキストパスを指定します。 必ず"/"から始まるパスで指定してください。	省略可	－	/
--scheme={http|https}	自コンテナの接続に使用するスキームを指定します。	省略可	http|https	http

オプション	説明	必須／省略可	指定範囲	デフォルト値
--insecure	HTTPS 接続時の証明書検証を無効化する場合に指定します。	省略可	－	－
--version	この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。ほかのオプションは無視されます。	－	－	－

＜オプション指定例＞

- 必須オプションだけ指定する例：

```
health_check.sh 30 10 10 10 20555
```

- コンテキストパスを指定する例：

```
health_check.sh 30 10 10 10 20555 /path
```

- スキーム指定および証明書検証を無効化する例：

```
health_check.sh 30 10 10 10 20555 --scheme=https --insecure
```

- バージョン表示オプションを指定する例：

```
health_check.sh --version
```

＜オプションのチェック内容＞

次の条件をどれか 1 つでも満たしていない場合は、KFSG62100-E 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。

- 必須オプションが未指定の場合
- オプションの値が指定範囲外の場合
- 存在しないオプションを指定した場合
- context-path の指定値が"/"から始まるパスではない場合

(4) 出力例

コマンドの出力例を示します。

- コマンド正常終了時の例：

```
KFSG82100-I The transaction health check processing will now start. (retry count = 30, re
try interval seconds = 10, connect timeout seconds = 10, read timeout seconds = 10, http
port = 20555, context pass = , scheme = http, insecure option = false)
KFSG62105-W The service is not ready yet.
KFSG62105-W The service is not ready yet.
KFSG62105-W The service is not ready yet.
KFSG82101-I The transaction health check processing finished.
```

- バージョン表示オプション指定時の例：

```
health_check.sh 1.0.0
```

- オプション指定誤り時の例：

```
KFSG62100-E One or more command arguments are invalid. (command line = health_check.sh )
Usage: health_check.sh <retry-count> <retry-interval-seconds> <connect-timeout-seconds> <
read-timeout-seconds> <http-port> [context-path] [--scheme=http|https] [--insecure]
.......health_check.sh --version
```

(5) リターンコード

コマンドのリターンコードを次の表に示します。

表 7-63 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のリターンコード

リターンコード	説明
0	コマンドの実行に成功しました。
1	コマンドの実行に失敗しました。

7.7.3 未決着トランザクション待機スクリプト（wait_transactions.sh）

(1) 機能

Orchestrator-Service、Entity-Service(SQL)、Entity-Service(TCC)、Orchestrator-Service(SQL)、Alternate-Service、Alternate-Service(SQL)、Mediator、および TP1-Bridge がスケールインなどによってシャットダウンする際に、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。各 Kubernetes アプリケーションの preStop フックで使用します。

＜コマンド処理内容＞

このコマンド開始直後に 20 秒間待機します。これは、Kubernetes が Pod 終了開始直後に実施する「Service のリクエスト転送先リストからの Pod の宛先の削除」が完了するのを待つためです。これによって、このコマンドが後続で未決着トランザクションの待機処理を開始するまでに、新規トランザクション処理のリクエストの流入がない状態にできます。

そのあと、未決着トランザクションの有無を判定するために、localhost に対してコマンド引数で指定された HTTP ポート番号の「グレースフルシャットダウン確認 API（/hmppcto/graceful)」へ GET リクエストを実行し、JSON レスポンスに含まれる「グレースフルシャットダウンできるかを表す値("status"キー)」の値が「true」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「false」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。

(2) 形式

コマンドは、次のどちらかの形式で指定します。

<形式 1>

```
wait_transactions.sh <retry-count>
                     <retry-interval-seconds>
                     <connect-timeout-seconds>
                     <read-timeout-seconds>
                     <http-port>
                     [<context-path>]
                     [--scheme={http|https}]
                     [--insecure]
```

<形式 2>

```
wait_transactions.sh --version
```

(3) オプション

コマンドのオプションを次の表に示します。

表 7-64 未決着トランザクション待機スクリプトのコマンドのオプション一覧

オプション	説明	必須／省略可	指定範囲	デフォルト値
<retry-count>	第 1 引数としてリクエストの最大リトライ回数を指定します。	必須	0～1,080	－
<retry-interval-seconds>	第 2 引数としてリクエストのリトライ間隔秒数を指定します。	必須	0～360	－
<connect-timeout-seconds>	第 3 引数としてリクエストの接続タイムアウト秒数を指定します。	必須	1～360	－
<read-timeout-seconds>	第 4 引数としてリクエストの応答タイムアウト秒数を指定します。	必須	1～360	－
<http-port>	第 5 引数として Spring Boot アプリケーションの HTTP ポート番号を指定します。	必須	1,024～65,535	－
<context-path>	第 6 引数として Spring Boot アプリケーションのコンテキストパスを指定します。 必ず"/"から始まるパスで指定してください。	省略可	－	/
--scheme={http|https}	自コンテナの接続に使用するスキームを指定します。	省略可	http|https	http
--insecure	HTTPS 接続時の証明書検証を無効化する場合に指定します。	省略可	－	－

オプション	説明	必須／省略可	指定範囲	デフォルト値
--version	この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。ほかのオプションは無視されます。	－	－	－

＜オプション指定例＞

- 必須オプションだけ指定する例：

```
wait_transactions.sh 30 10 10 10 20555
```

- コンテキストパスを指定する例：

```
wait_transactions.sh 30 10 10 10 20555 /path
```

- スキーム指定および証明書検証を無効化する例：

```
wait_transactions.sh 30 10 10 10 20555 --scheme=https --insecure
```

- バージョン表示オプションを指定する例：

```
wait_transactions.sh --version
```

＜オプションのチェック内容＞

次の条件をどれか 1 つでも満たしていない場合は、KFSG62101-E 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。

- 必須オプションが未指定の場合
- オプションの値が指定範囲外の場合
- 存在しないオプションを指定した場合
- context-path の指定値が"/"から始まるパスではない場合

(4) 出力例

コマンドの出力例を示します。

- コマンド正常終了時の例：

```
KFSG82102-I The processing that waits for active transactions will now start. (retry coun
t = 30, retry interval seconds = 10, connect timeout seconds = 10, read timeout seconds =
 10, http port = 20555, context pass = , scheme = http, insecure option = false)
KFSG62106-W The service still has active transactions.
KFSG62106-W The service still has active transactions.
KFSG62106-W The service still has active transactions.
KFSG82103-I The processing that waits for active transactions finished.
```

- バージョン表示オプション指定時の例：

```
wait_transactions.sh 1.0.0
```

- オプション指定誤り時の例：

```
KFSG62101-E One or more command arguments are invalid. (command line = wait_transactions.
sh )
Usage: wait_transactions.sh <retry-count> <retry-interval-seconds> <connect-timeout-secon
ds> <read-timeout-seconds> <http-port> [context-path] [--scheme=http|https] [--insecure]
.......wait_transactions.sh --version
```

(5) リターンコード

コマンドのリターンコードを次の表に示します。

表 7-65 未決着トランザクション待機スクリプトのコマンドのリターンコード

リターンコード	説明
0	コマンドの実行に成功しました。
1	コマンドの実行に失敗しました。

8 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。

8.1 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。

8.2　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。

8.2.1　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 8-1　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	TxPropagationRestTemplateInterceptor クラス	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。
3	jp.co.Hitachi.soft.hmppcto.filter.propagation	TxPropagationGrpcClientInterceptor クラス	gRPC 通信時に、gRPC の ClientInterceptor を使用してトランザクションを伝搬するインタセプタクラスです。
4	jp.co.Hitachi.soft.hmppcto.filter.propagation	TxPropagationWebClientFactory クラス	Spring Framework の WebClient を使用したトランザクションを伝搬するファクトリクラスです。
5	jp.co.Hitachi.soft.hmppcto.filter.propagation	TxPropagationOptionalSettings クラス	トランザクションを伝搬するためのオプションを設定するクラスです。
6	jp.co.Hitachi.soft.hmppcto.filter	TxPropagationGrpcServerInterceptor クラス	gRPC の ServerInterceptor を使用して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	jp.co.Hitachi.soft.hmppcto.filter	GrpcTraceInterceptorFactory クラス	HMP-PCTO が取得する分散トレースに、アプリケーションの gRPC 通信の送受信イベントを出力するインタセプタを提供するためのファクトリクラスです。

項番	パッケージ名	クラス・インタフェース名	説明
8	jp.co.Hitachi.soft.hmppcto.grpc	TxPropagationGrpcNameResolverProvider クラス	接続先の名前解決をするクラスを生成するクラスです。スケーリング対応機能を使用する場合に使用します。
9	jp.co.Hitachi.soft.hmppcto.grpc	TxPropagationGrpcLoadBalancerProvider クラス	通信の接続先を決定するクラスを生成するクラスです。スケーリング対応機能を使用する場合に使用します。

8.2.2 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	REST 通信による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.branch.api	@PctoGrpcBranchRollbackFor アノテーション	gRPC 通信による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。
3	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。
4	jp.co.Hitachi.soft.hmppcto.transaction.api	@PctoTransactionBranch アノテーション	トランザクションブランチの作成（トランザクションブランチの開始/終了）と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。

8.2.3 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 8-2　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場合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.lb.exception	PodNotFoundException	AP に指定したサービスに紐づく Pod が存在しない場合に送出する例外です。リクエストの実行の前処理で送出します。	否
4	jp.co.Hitachi.soft.hmppcto.lb.exception	ServiceNotFoundException	AP に指定したサービスが存在しない場合に送出する例外です。リクエストの実行の前処理で送出します。	否
5	jp.co.Hitachi.soft.hmppcto.transaction	TransactionBranchException	トランザクションではない呼び出し元で @PctoTransactionBranch を指定したメソッドを呼び出した場合に送出する例外です。	否
6	jp.co.Hitachi.soft.hmppcto.transaction	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

8.2.4　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 8-3　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。
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
```

8.2.5　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 8-4　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、レスポンスを返すメソッドです。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。

jp.co.Hitachi.soft.hmppcto.lb.exception.PodNotFoundException：

AP に指定したサービスに紐づく Pod が存在しない場合に送出する例外です。リクエストの実行の前処理で送出します。

jp.co.Hitachi.soft.hmppcto.lb.exception.ServiceNotFoundException：

AP に指定したサービスが存在しない場合に送出する例外です。リクエストの実行の前処理で送出します。

8.2.6 TxPropagationGrpcClientInterceptor クラス

説明

gRPC 通信時に、gRPC の ClientInterceptor を使用してトランザクションを伝搬するインタセプタクラスです。

メソッド

TxPropagationGrpcClientInterceptor クラスのメソッドの一覧を次の表に示します。

表 8–5 TxPropagationGrpcClientInterceptor クラスのメソッドの一覧

メソッド	説明
interceptCall io.grpc.ClientCall<ReqT,RespT> interceptCall(io.grpc.MethodDescriptor<ReqT,RespT> method,io.grpc.CallOptions callOptions, io.grpc.Channel next)	io.grpc.ClientCall による gRPC の呼び出しをインタセプトし、トランザクションを伝搬するメソッドです。

(1) interceptCall io.grpc.ClientCall<ReqT,RespT> interceptCall(io.grpc.MethodDescriptor<ReqT,RespT> method,io.grpc.CallOptions callOptions, io.grpc.Channel next)

説明

io.grpc.ClientCall による gRPC の呼び出しをインタセプトし、トランザクションを伝搬するメソッドです。

形式

```
public <ReqT,RespT> io.grpc.ClientCall<ReqT,RespT> interceptCall(io.grpc.MethodDescriptor
<ReqT,RespT> method, io.grpc.CallOptions callOptions, io.grpc.Channel next)
```

引数

io.grpc.MethodDescriptor<ReqT,RespT> method：

呼び出されるリモートメソッドを指定します。

io.grpc.CallOptions callOptions：

この呼び出しに使用するオプションを指定します。

io.grpc.Channel next：

インタセプトされているチャネルを指定します。

戻り値

<ReqT,RespT> io.grpc.ClientCall<ReqT,RespT>：

リモート呼び出しのオブジェクトを返します。

8.2.7 TxPropagationWebClientFactory クラス

説明

トランザクションの伝搬を REST で実現する際に、Spring Framework の WebClient に指定するファクトリクラスです。TxPropagationWebClientFactory クラスの webClientFilter()メソッドまたはこのメソッドの戻り値を org.springframework.web.reactive.function.client.WebClient.Builder クラスの filter()メソッドの引数として指定して、WebClient のインスタンスを生成してください。

メソッド

TxPropagationWebClientFactory クラスのメソッドの一覧を次の表に示します。

表 8-6 TxPropagationWebClientFactory クラスのメソッドの一覧

メソッド	説明
webClientFilter()	WebClient によるリクエストをインタセプトするメソッドです。

パッケージ名称

```
jp.co.Hitachi.soft.hmppcto.filter.propagation
```

(1) webClientFilter()

説明

HTTP リクエストに、HMP-PCTO がトランザクション制御に使用する HTTP ヘッダを追加してトランザクションを伝搬します。HTTP レスポンスから、HTTP ヘッダのトランザクション制御情報を読み取ります。

形式

```
org.springframework.web.reactive.function.client.ExchangeFunction webClientFilter()
```

戻り値

org.springframework.web.reactive.function.client.ExchangeFunction：

WebClient に登録できるフィルタ関数を返します。

例外

リクエストの実行の前処理で HMP-PCTO が送出する例外を次に示します。

これ以外の例外は、WebClient の各メソッドが送出する例外をそのまま送出します。

jp.co.Hitachi.soft.hmppcto.lb.exception.PodNotFoundException：

AP に指定したサービスに紐づく Pod が存在しない場合に送出する例外です。

jp.co.Hitachi.soft.hmppcto.lb.exception.ServiceNotFoundException：

AP に指定したサービスが存在しない場合に送出する例外です。

8.2.8 TxPropagationOptionalSettings クラス

説明

トランザクションを伝搬するためのオプションを設定するクラスです。

メソッド

TxPropagationOptionalSettings クラスのメソッドの一覧を次の表に示します。

表 8–7 TxPropagationOptionalSettings クラスのメソッドの一覧

メソッド	説明
isPrePrepared()	このチャネルでサービスを呼び出した場合に、先行プリペア機能を使用するかどうかを返すメソッドです。
setPrePrepared(boolean prePreparedState)	このチャネルでサービスを呼び出した場合に、先行プリペア機能を使用するかどうかを切り替えるメソッドです。

(1) isPrePrepared()

説明

このチャネルでサービスを呼び出した場合に、先行プリペア機能を使用するかどうかを返すメソッドです。スレッド単位で値を記憶するため、現在のスレッドの値を返します。

形式

```
public static boolean isPrePrepared()
```

戻り値

先行プリペア機能を使用する場合、true を返します。

先行プリペア機能を使用しない場合、false を返します。

(2) setPrePrepared(boolean prePreparedState)

説明

このチャネルでサービスを呼び出した場合に、先行プリペア機能を使用するかどうかを切り替えるメソッドです。一度設定した値は再設定されるまでスレッド単位で記憶します。先行プリペア機能を使用するかどうかを切り替えたいタイミングで実行してください。現在のスレッドの値は isPrePrepared() メソッドで確認できます。チャネル作成時の初期値は false です。

形式

```
public static void setPrePrepared(boolean prePreparedState)
```

引数

boolean prePreparedState：

先行プリペア機能を使用するかどうかを指定します。

true を指定した場合、先行プリペア機能を使用します。

false を指定した場合、先行プリペア機能を使用しません。

8.2.9 TxPropagationGrpcServerInterceptor クラス

説明

gRPC の ServerInterceptor を使用してトランザクションを伝搬するインタセプタクラスです。

メソッド

TxPropagationGrpcServerInterceptor クラスのメソッドの一覧を次の表に示します。

表 8–8 TxPropagationGrpcServerInterceptor クラスのメソッドの一覧

メソッド	説明
interceptCall(io.grpc.ServerCall<ReqT,RespT> call, io.grpc.Metadata headers, io.grpc.ServerCallHandler<ReqT,RespT> next)	io.grpc.ServerCall をインタセプトし、トランザクションを伝搬するメソッドです。

(1) interceptCall(io.grpc.ServerCall<ReqT,RespT> call, io.grpc.Metadata headers, io.grpc.ServerCallHandler<ReqT,RespT> next)

説明

io.grpc.ServerCall をインタセプトし、トランザクションを伝搬するメソッドです。

形式

```
public <ReqT,RespT>io.grpc.ServerCall.Listener<ReqT> interceptCall(io.grpc.ServerCall<ReqT,RespT> call, io.grpc.Metadata headers, io.grpc.ServerCallHandler<ReqT,RespT> next)
```

引数

io.grpc.ServerCall<ReqT,RespT> call：

サーバを呼び出すオブジェクトを指定します。

io.grpc.Metadata headers：

クライアントから送信された io.grpc.Metadata を指定します。

io.grpc.ServerCallHandler<ReqT,RespT> next：

次のインタセプタをディスパッチするためのオブジェクトを指定します。

戻り値

<ReqT,RespT>io.grpc.ServerCall.Listener<ReqT>：

リクエストを処理するためのリスナーを返します。

8.2.10 GrpcTraceInterceptorFactory クラス

説明

HMP-PCTO が取得する分散トレースに、アプリケーションの gRPC 通信の送受信イベントを出力するインタセプタを提供するためのファクトリクラスです。

このクラスのメソッドを使用して取得したインタセプタを gRPC クライアントおよび gRPC サーバに追加することで、HMP-PCTO が取得する分散トレースにアプリケーションの gRPC 通信の送受信イベントが出力されます。

メソッド

GrpcTraceInterceptorFactory クラスのメソッドの一覧を次の表に示します。

表 8-9 GrpcTraceInterceptorFactory クラスのメソッドの一覧

メソッド	説明
getClientInterceptor()	gRPC クライアントに追加する分散トレース出力用のインタセプタを取得するメソッドです。

メソッド	説明
getServerInterceptor()	gRPC サーバに追加する分散トレース出力用のインタセプタを取得するメソッドです。

(1) getClientInterceptor()

説明

gRPC クライアントに追加する分散トレース出力用のインタセプタを取得するメソッドです。

形式

```
io.grpc.ServerInterceptor getServerInterceptor()
```

戻り値

io.grpc.ClientInterceptor：

gRPC クライアントに追加する分散トレース出力用のインタセプタを返します。

(2) getServerInterceptor()

説明

gRPC サーバに追加する分散トレース出力用のインタセプタを取得するメソッドです。

形式

```
io.grpc.ServerInterceptor getServerInterceptor()
```

戻り値

io.grpc.ServerInterceptor：

gRPC サーバに追加する分散トレース出力用のインタセプタを返します。

8.2.11 TxPropagationGrpcNameResolverProvider クラス

説明

接続先の名前解決をするクラスを生成するクラスです。スケーリング対応機能を使用する場合に使用します。

定数

TxPropagationGrpcNameResolverProvider クラスの定数の一覧を次の表に示します。

表 8-10 TxPropagationGrpcNameResolverProvider クラスの定数の一覧

定数	備考
public static final String SCHEME_SCALING	スケーリング対応機能を使用する場合に、ManagedChannelBuilder またはそのサブクラスに設定するスキーム名の定数です。

8.2.12　TxPropagationGrpcLoadBalancerProvider クラス

説明

通信の接続先を決定するクラスを生成するクラスです。スケーリング対応機能を使用する場合に使用します。

定数

TxPropagationGrpcLoadBalancerProvider クラスの定数の一覧を次の表に示します。

表 8-11　TxPropagationGrpcLoadBalancerProvider クラスの定数の一覧

定数	備考
public static final String POLICY_SCALING	スケーリング対応機能を使用する場合に、ManagedChannelBuilder またはそのサブクラスに設定するロードバランシングポリシー名の定数です。

8.2.13　@PctoBranchRollbackFor アノテーション

説明

REST 通信による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。HTTP のサービス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呼び出し元のサービスに連絡します。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 8-12　@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	jakarta.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	jakarta.ws.rs.core.Response.Status.Family[]	{jakarta.ws.rs.core.Response.Status.Family.CLIENT_ERROR, jakarta.ws.rs.core.Response.Status.Family.SERVER_ERROR}

パッケージ名称

```
jp.co.Hitachi.soft.hmppcto.interceptor
```

8.2.14　@PctoGrpcBranchRollbackFor アノテーション

説明

gRPC 通信による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。gRPC のサービス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、メソッドレベルで指定することを推奨します。

dontRollbackOn 属性で、このトランザクションをロールバックにマークしない gRPC ステータスコードを設定します。デフォルトの動作は、io.grpc.Status.Code.OK 以外をロールバックにマークします。

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 のインタセプタが、gRPC 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呼び出し元のサービスに連絡します。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。

適用可能要素

クラス、メソッド

属性

@PctoGrpcBranchRollbackFor アノテーションの属性の一覧を次の表に示します。

表 8-13　@PctoGrpcBranchRollbackFor 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	トランザクションをロールバックにマークしない gRPC のステータスコードを設定します。	任意	io.grpc.Status.Code[]	なし

パッケージ名称

```
jp.co.Hitachi.soft.hmppcto.interceptor
```

8.2.15 @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 8-14　@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
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
```

8.2.16　@PctoTransactionBranch アノテーション

説明

トランザクションブランチを作成するアノテーションです。このアノテーションを指定することで、トランザクションルートの範囲内で JDBC Proxy Driver を使用できます。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ブランチを開始し、メソッドの終了後にトランザクションブランチを終了します。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、同じトランザクションブランチの範囲内です。トランザクションブランチ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、元のトランザクションブランチの範囲内になるため、ネストしている指定を無視します。@PctoTransactional アノテーションを指定したメソッドにも、このアノテーションを指定できます。

prePreparedOn 属性を指定して、先行プリペア機能を使用できます。prePreparedOn 属性に true を指定すると、先行プリペア機能を使用します。prePreparedOn 属性を省略または false を指定すると、先行プリペア機能を使用しません。

適用可能要素

クラス、メソッド

属性

@PctoTransactionBranch アノテーションの属性の一覧を次の表に示します。

表 8-15　@PctoTransactionBranch 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
prePreparedOn	先行プリペア機能を使用するかどうかを指定します。 true：先行プリペア機能を使用します。 false：先行プリペア機能を使用しません。	任意	boolean	false

パッケージ名称

```
jp.co.Hitachi.soft.hmppcto.transaction.api
```

8.2.17　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
```

8.2.18　TerminationFailureException 例外

説明

Mediator との通信障害でトランザクションの決着結果が不明な場合、または Mediator がトランザクションの決着結果不明を返却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、分散トレース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
```

8.2.19 PodNotFoundException 例外

説明

AP に指定したサービスに紐づく Pod が存在しない場合に送出する例外です。
TxPropagationRestTemplateInterceptor クラスの送信リクエストの実行の前処理、または TxPropagationGrpcNameResolverProvider クラスと TxPropagationGrpcLoadBalancerProvider クラスで登録したクラスで送出します。

メソッド

なし。

パッケージ名称

```
jp.co.Hitachi.soft.hmppcto.lb.exception
```

継承関係

```
java.lang.Object
  java.lang.Throwable
    java.lang.Exception
      java.io.IOException
        jp.co.Hitachi.soft.hmppcto.lb.exception.PodNotFoundException
```

8.2.20 ServiceNotFoundException 例外

説明

AP に指定したサービスが存在しない場合に送出する例外です。
TxPropagationRestTemplateInterceptor クラスの送信リクエストの実行の前処理、または TxPropagationGrpcNameResolverProvider クラスで登録したクラスで送出します。

メソッド

なし。

パッケージ名称

```
jp.co.Hitachi.soft.hmppcto.lb.exception
```

継承関係

```
java.lang.Object
  java.lang.Throwable
```

```
    java.lang.Exception
      java.io.IOException
        jp.co.Hitachi.soft.hmppcto.lb.exception.ServiceNotFoundException
```

8.2.21 TransactionBranchException 例外

説明

トランザクションではない呼び出し元で@PctoTransactionBranch を指定したメソッドを呼び出した場合に送出する例外です。

具体的には、@PctoTransactionBranch を指定したメソッドを実行しないで呼び出し元に TransactionBranchException を送出します。

メソッド

なし。

パッケージ名称

```
jp.co.Hitachi.soft.hmppcto.transaction
```

継承関係

```
java.lang.Object
  java.lang.Throwable
    java.lang.Exception
      jp.co.Hitachi.soft.hmppcto.transaction.TransactionBranchException
```

8.2.22 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.transaction
```

継承関係

```
java.lang.Object
  java.lang.Throwable
```

```
java.lang.Exception
  jp.co.Hitachi.soft.hmppcto.transaction.TransactionSystemException
```

8.3　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。
- メソッドの戻り値が java.sql のインタフェースを実装したクラスの場合、SQL-Participant で使用している JDBC Driver のクラスではなく、「表 8-16　クラス・インタフェースの一覧」に示した HMP-PCTO の JDBC Proxy Driver がサポートするクラスにラップして返却します。そのため JDBC の仕様にないベンダ独自のメソッドについては、サポートしません。メソッドの戻り値が java.sql のインタフェースを実装したクラス以外の場合は、戻り値をそのまま返却します。

8.3.1　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 8-16　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。

項番	パッケージ名	クラス・インタフェース名	説明
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。
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	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。

8.3.2 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 8-17 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
3	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

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
			次の場合に送出する例外です。 • Connection インスタンスが失われている場合 • Connection インスタンスの取得時に DB で障害が発生した場合 • SQL-Participant が閉塞している場合 • トランザクション決着監視タイマによる rollback 処理後に SQL 実行を要求した場合	
4	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
5	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
6	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

8.3.3 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 8-18　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

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。

PctoInternalContradictionException：

次の場合に例外を送出します。

- Connection インスタンスが失われている場合
- Connection インスタンスの取得時に DB で障害が発生した場合
- SQL-Participant が閉塞している場合

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。

8.3.4 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 8-19 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。

メソッド	説明
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。

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

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。

PctoInternalContradictionException：

次の場合に例外を送出します。

- Connection インスタンスが失われている場合
- SQL-Participant が閉塞している場合
- 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。

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

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。

PctoInternalContradictionException：

次の場合に例外を送出します。

- Connection インスタンスが失われている場合
- SQL-Participant が閉塞している場合
- 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。

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

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

PctoInternalContradictionException：

次の場合に例外を送出します。

- Connection インスタンスが失われている場合
- SQL-Participant が閉塞している場合
- 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。

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

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

PctoInternalContradictionException：

次の場合に例外を送出します。

- Connection インスタンスが失われている場合
- SQL-Participant が閉塞している場合

- 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。

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

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

PctoInternalContradictionException：

次の場合に例外を送出します。

- Connection インスタンスが失われている場合
- SQL-Participant が閉塞している場合
- 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。

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

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

PctoInternalContradictionException：

以下の場合に例外を送出します。

- Connection インスタンスが失われている場合
- SQL-Participant が閉塞している場合
- 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。

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

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

PctoInternalContradictionException：

次の場合に例外を送出します。

- Connection インスタンスが失われている場合
- SQL-Participant が閉塞している場合
- 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。

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

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

PctoInternalContradictionException：

次の場合に例外を送出します。

- Connection インスタンスが失われている場合
- SQL-Participant が閉塞している場合
- 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。

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

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。

PctoInternalContradictionException：

次の場合に例外を送出します。

- Connection インスタンスが失われている場合
- SQL-Participant が閉塞している場合
- 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。

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

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

PctoInternalContradictionException：

次の場合に例外を送出します。

- Connection インスタンスが失われている場合
- SQL-Participant が閉塞している場合
- 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。

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

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。

PctoInternalContradictionException：

次の場合に例外を送出します。

- Connection インスタンスが失われている場合
- SQL-Participant が閉塞している場合
- 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。

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

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

PctoInternalContradictionException：

次の場合に例外を送出します。

- Connection インスタンスが失われている場合
- SQL-Participant が閉塞している場合
- 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。

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。

8.3.5 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 8–20　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。
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。

メソッド	説明
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

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

PctoInternalContradictionException：

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

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。

PctoInternalContradictionException：

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

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。

PctoInternalContradictionException：

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

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。

PctoInternalContradictionException：

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.SQLEx
ception
```

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

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。

PctoInternalContradictionException：

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 を送出し、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

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。

PctoInternalContradictionException：

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

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。

PctoInternalContradictionException：

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

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。

PctoInternalContradictionException：

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

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。

PctoInternalContradictionException：

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

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

PctoInternalContradictionException：

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 が送出されます。

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

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

PctoInternalContradictionException：

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

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

PctoInternalContradictionException：

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

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

PctoInternalContradictionException：

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 が送出されます。

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

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

PctoInternalContradictionException：

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

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

PctoInternalContradictionException：

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

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

PctoInternalContradictionException：

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 が送出されます。

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

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

PctoInternalContradictionException：

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。

8.3.6 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 8-21　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。

メソッド	説明
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。
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。

メソッド	説明
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 8-22　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

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

PctoInternalContradictionException：

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

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。

PctoInternalContradictionException：

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

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。

PctoInternalContradictionException：

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

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。

PctoInternalContradictionException：

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。

戻り値

なし。

例外

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。

PctoInternalContradictionException：

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.SQL
Exception
```

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

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

PctoInternalContradictionException：

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

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

PctoInternalContradictionException：

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

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。

PctoInternalContradictionException：

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

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

PctoInternalContradictionException：

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

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

PctoInternalContradictionException：

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

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

PctoInternalContradictionException：

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

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

PctoInternalContradictionException：

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

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。

PctoInternalContradictionException：

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

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

PctoInternalContradictionException：

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

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

PctoInternalContradictionException：

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

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

PctoInternalContradictionException：

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

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

PctoInternalContradictionException：

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

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

PctoInternalContradictionException：

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

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。

PctoInternalContradictionException：

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

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。

PctoInternalContradictionException：

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送出します。

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。

戻り値

なし。

例外

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

PctoInternalContradictionException：

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 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.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。

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

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

PctoInternalContradictionException：

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

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

PctoInternalContradictionException：

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

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

PctoInternalContradictionException：

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

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

PctoInternalContradictionException：

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

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

PctoInternalContradictionException：

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。

8.3.7 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 8-23　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。
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。

メソッド	説明
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 8-24　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 8-25　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

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。

PctoInternalContradictionException：

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

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。

PctoInternalContradictionException：

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

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。

PctoInternalContradictionException：

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

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

PctoInternalContradictionException：

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

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。

PctoInternalContradictionException：

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

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

PctoInternalContradictionException：

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

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

PctoInternalContradictionException：

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

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

PctoInternalContradictionException：

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

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

PctoInternalContradictionException：

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

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

PctoInternalContradictionException：

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

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。

PctoInternalContradictionException：

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。

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

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

PctoInternalContradictionException：

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 が送出されます。

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

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。

PctoInternalContradictionException：

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

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

PctoInternalContradictionException：

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

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。

PctoInternalContradictionException：

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

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

PctoInternalContradictionException：

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

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

PctoInternalContradictionException：

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

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

PctoInternalContradictionException：

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。

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

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。

PctoInternalContradictionException：

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

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。

PctoInternalContradictionException：

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

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。

PctoInternalContradictionException：

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

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

PctoInternalContradictionException：

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。

8.3.8 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 8-26　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。
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。

メソッド	説明
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。
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。

メソッド	説明
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。
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

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。

PctoInternalContradictionException：

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

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。

PctoInternalContradictionException：

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

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。

PctoInternalContradictionException：

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

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。

PctoInternalContradictionException：

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

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

PctoInternalContradictionException：

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.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

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

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

PctoInternalContradictionException：

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

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。

PctoInternalContradictionException：

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

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。

PctoInternalContradictionException：

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

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

PctoInternalContradictionException：

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

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

PctoInternalContradictionException：

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

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

PctoInternalContradictionException：

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

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

PctoInternalContradictionException：

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

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

PctoInternalContradictionException：

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

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

PctoInternalContradictionException：

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

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。

PctoInternalContradictionException：

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

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。

PctoInternalContradictionException：

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

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

PctoInternalContradictionException：

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

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

PctoInternalContradictionException：

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

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

PctoInternalContradictionException：

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

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

PctoInternalContradictionException：

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

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

PctoInternalContradictionException：

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

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

PctoInternalContradictionException：

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

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

PctoInternalContradictionException：

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

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

PctoInternalContradictionException：

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

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

PctoInternalContradictionException：

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

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

PctoInternalContradictionException：

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

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

PctoInternalContradictionException：

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

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。

PctoInternalContradictionException：

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.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

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。

PctoInternalContradictionException：

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

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

PctoInternalContradictionException：

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

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

PctoInternalContradictionException：

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 が送出されます。

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

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。

PctoInternalContradictionException：

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 が送出されます。

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

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。

PctoInternalContradictionException：

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

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

PctoInternalContradictionException：

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

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

PctoInternalContradictionException：

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

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。

PctoInternalContradictionException：

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

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。

PctoInternalContradictionException：

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

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

PctoInternalContradictionException：

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

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

PctoInternalContradictionException：

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

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

PctoInternalContradictionException：

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

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

PctoInternalContradictionException：

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

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

PctoInternalContradictionException：

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

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

PctoInternalContradictionException：

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

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

PctoInternalContradictionException：

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

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。

PctoInternalContradictionException：

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

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

PctoInternalContradictionException：

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

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

PctoInternalContradictionException：

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。

8.3.9 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 8-27　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

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。

PctoInternalContradictionException：

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

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。

PctoInternalContradictionException：

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。

8.3.10　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 8-28　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。
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

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。

PctoInternalContradictionException：

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

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。

PctoInternalContradictionException：

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

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。

PctoInternalContradictionException：

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

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。

PctoInternalContradictionException：

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

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。

PctoInternalContradictionException：

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。

8.3.11　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 8-29　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 が送出されます。

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

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。

PctoInternalContradictionException：

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

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。

PctoInternalContradictionException：

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

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。

PctoInternalContradictionException：

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。

8.3.12 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 8-30 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

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。

PctoInternalContradictionException：

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

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。

PctoInternalContradictionException：

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

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。

PctoInternalContradictionException：

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。

8.3.13　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 8-31　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

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。

PctoInternalContradictionException：

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

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。

PctoInternalContradictionException：

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。

8.3.14　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 8–32　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 が送出されます。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

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。

PctoInternalContradictionException：

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

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。

PctoInternalContradictionException：

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。

8.3.15 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
```

8.3.16 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
```

8.3.17 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します。

次の場合に送出する例外です。

- Connection インスタンスが失われている場合
- Connection インスタンスの取得時に DB で障害が発生した場合
- SQL-Participant が閉塞している場合
- トランザクション決着監視タイマによる rollback 処理後に SQL 実行を要求した場合

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
```

8.3.18 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
```

8.3.19 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
```

8.3.20　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
```

8.3.21　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 8-33　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	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。
4	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。
5	A0002	1010	PctoInternalContradictionException 例外 または PctoTransactionStartException 例外	SQL-Participant が閉塞しています。
6	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。

項番	SQLState	ErrorCode	例外	意味
7	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。
8	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。
9	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。

8.4　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。

8.4.1　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 8-34　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。

8.4.2　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 8-35　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。

8.4.3　例外の一覧（TCC-Participant 限定）

なし。

8.4.4 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 8-36　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。

8.4.5　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 8-37　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「8.2.13 @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 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 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「(5)　executeRecover メソッドの実装」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 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 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 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。

8.4.6 @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
```

9 パラメタ

9.1 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 9-1 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	9.2.1 共通
		JDBC Proxy Driver	9.2.2 JDBC Proxy Driver
		Orchestrator	9.2.3 Orchestrator
		Entity-Module	9.2.4 Entity-Module
		SQL-Participant	9.2.5 SQL-Participant
		TCC-Participant	9.2.6 TCC-Participant
		Spring Boot の組み込み HTTP サーバ	9.2.7 Spring Boot の組み込み HTTP サーバ
HMP-PCTO のコンポーネント	Helm チャートの values.yaml に設定	Mediator	9.3.1 Mediator
		TP1-Bridge	9.3.2 TP1-Bridge（通常版かつ TP1-Bridge 限定）
		EADS（通常版限定）	9.3.3 EADS（通常版限定）
		Ext-Cons（トライアル版の場合）	9.3.4 Ext-Cons（トライアル版限定）
HMP-PCTO と RAS のコンポーネント	Kubernetes マニフェストに設定	Filebeat	9.3.5 Filebeat
		Jaeger	9.3.6 Jaeger
		Prometheus	9.3.7 Prometheus
		Metricbeat	9.3.8 Metricbeat
		Logstash	9.3.9 Logstash
		Metrics Server	9.3.10 Metrics Server

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 9-2 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	9.4.1 PostgreSQL（SQL-Participant 限定）

連携先製品	パラメタ設定方法	参照先
SQL Server（SQL-Participant 限定）	SQL Server のサーバ側のオプションに設定	9.4.2　SQL Server（SQL-Participant 限定）
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	9.4.3　HiRDB（SQL-Participant 限定）
Oracle Database（SQL-Participant 限定）	Oracle Database のサーバ側のパラメタに設定	9.4.4　Oracle Database（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	9.5　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	9.6　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。

このマニュアルに記載している OSS のパラメタ、およびこの章で説明しているパラメタ以外の他社製品のパラメタの一覧については、「付録 B　HMP-PCTO に関連する OSS および他社製品のパラメタ一覧」を参照してください。各パラメタの詳細は、関連の OSS および他社製品の公式ドキュメントを参照してください。

9.2　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。

9.2.1　共通

Orchestrator-Service、Relay-Service、Entity-Service(SQL)、Entity-Service(TCC)、Orchestrator-Service(SQL)、Alternate-Service および Alternate-Service(SQL)で共通のパラメタを説明します。

(1)　server のパラメタ

各パラメタの接頭辞は、server です。

表 9-3　共通の server のパラメタ

パラメタ	説明	型	設定範囲	デフォルト値
shutdown	Spring Boot が終了する際に、処理中のリクエストをどのように処理するかを設定します。 スケーリング対応機能を使用する場合は、 グレースフルシャットダウンをサポートしている HMP-PCTO の Kubernetes アプリケーションで graceful を設定し、Spring Boot のグレースフルシャットダウンを設定してください。 詳細については、「7.5　Kubernetes アプリケーションのグレースフルシャットダウン」を参照してください。	String	Spring Boot の仕様に依存します。	Spring Boot の仕様に依存します。
port	HMP-PCTO の Kubernetes アプリケーションで使用するポート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。 この設定値は次で使用します。 • REST 通信のリクエスト受信 • 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 • メトリクス収集のポート番号	Integer	Spring Boot の仕様に依存します。	Spring Boot の仕様に依存します。

(2)　spring のパラメタ

各パラメタの接頭辞は、spring です。

表 9-4　共通の spring のパラメタ

パラメタ	説明	型	設定範囲	デフォルト値
lifecycle.timeout-per-shutdown-phase	server.shutdown パラメタに graceful を設定し、Spring Boot のグレースフルシャットダウンの設定をした場合に、このパラメタで設定した秒数の間、終了を待機します。 スケーリング対応機能を使用する場合は、 グレースフルシャットダウンをサポートしている HMP-PCTO の Kubernetes アプリケーションに graceful を設定し、Spring Boot のグレースフルシャットダウンの設定およびこのパラメタの設定をしてください。 詳細については、「7.5　Kubernetes アプリケーションのグレースフルシャットダウン」を参照してください。	Duration	Spring Boot の仕様に依存します。	Spring Boot の仕様に依存します。

(3)　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 9-5　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	HMP-PCTO の各コンテナイメージを識別するための識別子を設定してください。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。 システムで一意になるようにコンテナイメージの役割に応じた値を設定してください。 異なる役割のコンテナイメージに同一の値を設定した場合、どのコンテナイメージが出力した分散トレース情報かを判別できません。 Kubernetes マニフェストファイルの spec.replicas パラメタに 2 以上を設定する場合やスケーリング対応機能を使用する場合は、このパラメタを必ず指定してください。 複数の Pod が起動した場合、同じサービス名となり、分散トレース情報の検索が容易になります。 このパラメタを設定した場合は、次のサービス名となります。 • Orchestrator-Service/Relay-Service/Entity-Service(SQL)/Entity-	String	32 文字	なし

パラメタ	説明	型	設定範囲	デフォルト値
	Service(TCC)/Orchestrator-Service(SQL)/Alternate-Service/Alternate-Service(SQL)/TP1-Bridge： <このパラメタの設定値> • Mediator： <このパラメタの設定値>@<Mediator のホスト名> また、このパラメタの設定を省略した場合は、次のサービス名となります。 • Orchestrator-Service： Orchestrator@<Orchestrator-Service のホスト名> • Relay-Service： Entity-Service@<Relay-Service のホスト名> • Entity-Service(SQL)： SQL-Participant@<Entity-Service(SQL)のホスト名> • Entity-Service(TCC)： TCC-Participant@<Entity-Service(TCC)のホスト名> • Orchestrator-Service(SQL)： SQL-Participant@<Orchestrator-Service(SQL)のホスト名> • Alternate-Service： Orchestrator@<Alternate-Service のホスト名> • Alternate-Service(SQL)： SQL-Participant@<Alternate-Service(SQL)のホスト名> • TP1-Bridge： TP1-Bridge@<TP1-Bridge のホスト名> • Mediator： Mediator@<Mediator のホスト名>			
tracing.url	HMP-PCTO が分散トレースの送信先に使用する分散トレース収集アプリケーションまたはサービスの接続先（URL）を指定します。	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.use-trace の指定値が true、かつ分散トレース送信先が Google Cloud Platform 以外の場合は省略できません。 OpenTelemetry Collector の場合 次の形式で指定してください。 • OpenTelemetry Collector が同一 namespace の場合 http://<OpenTelemetry Collector の.metadata.name フィールドに指定した値>:<ポート番号>/v1/traces • OpenTelemetry Collector が異なる namespace の場合 http://<OpenTelemetry Collector の.metadata.name フィールドに指定した値>.<Namespace 名>.svc.cluster.local:<ポート番号>/v1/traces 指定例 • OpenTelemetry Collector が同一 namespace の場合 http://otel-collector:4318/v1/traces • OpenTelemetry Collector が異なる namespace の場合 http://otel-collector.my-namespace.svc.cluster.local:4318/v1/traces Google Cloud Platform の場合 指定は不要です。 Pod のサービスアカウントに紐づいている、Google Cloud Platform のプロジェクトの Cloud Trace に、自動的に送信します。			
tracing.use-trace	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。	Boolean	true | false	true

パラメタ	説明	型	設定範囲	デフォルト値
	• 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。			
tracing.debug	デバッグ情報を含むスパン取得有無を指定します。 • true デバッグ情報を含むスパンを取得します。 • false デバッグ情報を含むスパンを取得しません。 HMP-PCTO ではデバッグ情報を含むスパンを取得することで、詳細な処理の内容を確認できます。 このパラメタに true を指定するとデバッグ情報を含むスパンを取得します。デバッグ情報を含むスパンを取得すると、分散トレースのディスク使用量が増加します。	Boolean	true | false	false
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.use-trace 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収集アプリケーションまたはサービスに送信されます。 • 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-in-millis パラメタで設定した値以上の時間が経過した場合	Integer	10,000-100,000	20,000
tracing.delay-time-in-millis	分散トレースを分散トレース収集アプリケーションまたはサービスに送信する間隔をミリ秒単位で指定します。tracing.use-trace パラ	Integer	1,000-5,000	1,000

パラメタ	説明	型	設定範囲	デフォルト値
	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収集アプリケーションまたはサービスに送信されます。			
tracing.batch-size	分散トレースを分散トレース収集アプリケーションまたはサービスに送信する際に、一度に送信する分散トレース数を指定します。 tracing.use-trace 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収集アプリケーションまたはサービスに送信されます。	Integer	1,000-10,000	2,000
tracing.grpc.debug	gRPC 通信の送受信メッセージのスパンイベントの追加情報の取得有無を設定します。 • true gRPC 通信の送受信メッセージのスパンイベントの追加情報を取得します。 • false gRPC 通信の送受信メッセージのスパンイベントの追加情報を取得しません。 対象のスパンイベント • sent message(hmppcto) • received message(hmppcto) このパラメタに true を設定すると HMP-PCTO の Orchestrator、Participant および Mediator が実施している gRPC 通信の送受信メッセージの追加情報がスパンイベントとして取得され、gRPC 通信の内容を確認できます。 ただし、gRPC 通信のメッセージ長によって、分散トレースの取得処理が増え、処理時間や分散トレースのディスク使用量が増加します。 このパラメタに false を設定することで gRPC 通信の送受信メッセー	Boolean	true | false	false

パラメタ	説明	型	設定範囲	デフォルト値
	ジをスパンイベントの追加情報として取得しなくなるため、処理時間が短縮され、分散トレースのディスク使用量を抑えることができます。			
tracing.jdbc.debug	SQL-Participant での JDBC Driver メソッド開始および終了のスパンイベントの取得有無を設定します。 • true JDBC Driver メソッド開始および終了のスパンイベントを取得します。 • false JDBC Driver メソッド開始および終了のスパンイベントを取得しません。 対象のスパンイベント • start:jdbc <JDBC Driver のクラス名>.<JDBC Driver のメソッド名> • end:jdbc このパラメタに true を設定すると、HMP-PCTO の SQL-Participant で実行された JDBC Driver メソッド開始および終了のスパンイベントが取得され、JDBC Driver メソッドの発行状況および処理時間を確認できます。 ただし、JDBC Driver メソッドの発行回数によって、分散トレースの取得処理が増え、処理時間や分散トレースのディスク使用量が増加します。 このパラメタに false を設定することで JDBC Driver メソッド開始および終了のスパンイベントを取得しなくなるため、処理時間が短縮され、分散トレースのディスク使用量を抑えることができます。	Boolean	true | false	false
metrics.collection.tomcat	Tomcat のメトリクスを設定します。 • true Spring Boot の組み込み HTTP サーバの Tomcat のメトリクスを取得します。	Boolean	true | false	true

パラメタ	説明	型	設定範囲	デフォルト値
	• false Spring Boot の組み込み HTTP サーバの Tomcat のメトリクスは取得しません。 このパラメタに true を設定すると、Spring Boot の組み込み HTTP サーバの Tomcat のメトリクス（接続数、スレッド数、レスポンスタイムなど）を取得できます。このパラメタを使用するには、アプリケーションで Spring Boot の組み込み HTTP サーバに Tomcat を使用し、HMP-PCTO の hmppcto-server-tomcat 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。 このパラメタの設定を true にすると、メトリクスの取得対象が増えるため、ディスク使用量が増加します。 Tomcat を使用した HTTP リクエストの処理を行わない場合、Tomcat のメトリクスは不要なため、このパラメタには false を設定してください。不要なデータの蓄積を防ぎ、ディスク使用量を抑えることができます。			

（凡例）－：該当しません。

9.2.2 JDBC Proxy Driver

JDBC Proxy Driver を使用する場合に設定するパラメタを説明します。JDBC Proxy Driver を使用しない場合は設定不要です。

JDBC Proxy Driver のパラメタは、Entity-Service(SQL)、Orchestrator-Service(SQL)、Alternate-Service(SQL)に設定します。

(1) spring.datasource のパラメタ

各パラメタの接頭辞は、spring.datasource 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 9–6　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。 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。 実際に DB に接続するための URL は SQL-Participant で設定します。	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。 実際に 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。 実際に 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。

(2)　spring.sql のパラメタ

各パラメタの接頭辞は、spring.sql です。

表 9–7　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。 never	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。

9.2.3 Orchestrator

Orchestrator のパラメタを説明します。

Orchestrator のパラメタは、Orchestrator-Service、Orchestrator-Service(SQL)、Alternate-Service、および Alternate-Service(SQL)に設定します。

(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 9-8 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。アルゴリズムの詳細は、取扱説明書「HMP-PCTO テクニカルガイド」の分散合意習得方式の説明を参照してください。	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>.<Namespace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 Service 名以降の文字を指定してください。	String	−	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
	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>.<Namespace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。			
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	1,024-65,535	20,551
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 以上かつ奇数の値を設定してください。 設定する値については、取扱説明書「HMP-PCTO テクニカルガイド」の Mediator 数の設定要件に関する説明を参照して決定してください。	Integer	2-16	なし 必ず設定してください。
mediator.initial-size	Orchestrator 起動時にチェックを行う Mediator 数を設定してください。 mediator.size で設定した数以上の値を設定してください。mediator.size で設定した数より小さい値を設定した場合は、Orchestrator の初期化時に異常終了します。 設定する値については、取扱説明書「HMP-PCTO テクニカルガイド」の Mediator 数の設定要件に関する説明を参照して決定してください。	Integer	2-64	mediator.size 設定値
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。	Integer	2-16	mediator.size 設定値

パラメタ	説明	型	設定範囲	デフォルト値
	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 以上かつ奇数の値を設定してください。 設定する値については、取扱説明書「HMP-PCTO テクニカルガイド」の Mediator 数の設定要件に関する説明を参照して決定してください。			
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、「3.12.1　スレッドプール」を参照してください。	Integer	1-1,000	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	1-60,000	10,000

branch.qualifier.list	AP がサービスを呼び出す際に、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。
	REST 通信で呼び出す場合は、次の形式で指定します。
	`<サービスのホスト名>:<サービスのポート番号><サービスのエンドポイント>@識別子`
	gRPC 通信で呼び出す場合は、次の形式で指定します。
	`<gRPCサーバのホスト名>:<gRPCサーバのポート番号><protobufで定義したサービス`
		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。

パラメタ	説明	型	設定範囲	デフォルト値
	の完全修飾名>/<protobufで定義したメソッド名>@識別子 識別子は、32 文字以内で設定してください。 複数のサービスを設定する場合は「,」で区切って設定してください。 REST 通信時の設定例： host1:80/service1@qualifier1,host2:80/service2@qualifier2 gRPC 通信時の設定例： host1-grpc:80com.example.grpc.service.GrpcService/method@qualifier1			
orchestrator.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
orchestrator.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します。	String	−	なし 暗号化通信をする場合は、必ず指定してください。
orchestrator.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。
orchestrator.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。
orchestrator.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	0-2,147,483,647	0
orchestrator.grpc.tls.session.timeout-in-seconds	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	0-65,535	0

パラメタ	説明	型	設定範囲	デフォルト値
orchestrator.grpc.keepalive.interval-in-seconds	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	0-100,000	0
orchestrator.grpc.keepalive.timeout-in-seconds	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	1-65,535	60
orchestrator.grpc.watchtime-in-seconds	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	0-65,535	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、Pod の起動処理を異常終了します。	Integer	1-10,800	60
use-scaling	スケーリング対応機能を使用するかどうかを指定します。 • true：スケーリング対応機能を使用します。 • false：スケーリング対応機能を使用しません。 Orchestrator、Entity-Module、SQL-Participant、TCC-Participant、TP1-Bridge および Mediator それぞれに設定するこのパラメタには、同じ値を指定してください。	Boolean	true | false	true
set-routing	ルーティング方法を指定します。 • inner：HMP-PCTO が提供するルーティング機能を使用し、ルーティングします。	String	inner | user	inner

パラメタ	説明	型	設定範囲	デフォルト値
	• user：ユーザ側でロードバランサを構成し、ロードバランサでルーティングします。 このパラメタは use-scaling パラメタに true を指定している場合だけ有効になります。 Orchestrator、Entity-Module、SQL-Participant それぞれのこのパラメタの設定値はすべて統一してください。 gRPC 通信を使用する場合は、このパラメタの指定を省略するか inner を指定してください。			

（凡例）－：該当しません。

9.2.4 Entity-Module

Entity-Module のパラメタについて説明します。

Entity-Module のパラメタは、Relay-Service、Entity-Service(SQL)、Entity-Service(TCC)、Alternate-Service、Alternate-Service(SQL)に設定します。

(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 9-9 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 を使用します。 • TCC：TCC-Participant を使用します。 TP1-Bridge を使用する場合、指定は不要です。	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。			
lb.list-timeout-in-seconds	Orchestrator の lb.list-timeout-in-seconds 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。			
use-scaling	Orchestrator の use-scaling と同じです。			

パラメタ	説明	型	設定範囲	デフォルト値
	「(1)　jp.co.Hitachi.soft.hmppcto のパラメタ」を参照してください。			
set-routing	Orchestrator の set-routing 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。			

9.2.5　SQL-Participant

SQL-Participant のパラメタについて説明します。

SQL-Participant のパラメタは、Entity-Service(SQL)、Orchestrator-Service(SQL)、Alternate-Service(SQL)に設定します。

(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 9–10　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 1,000 ミリ秒以上を指定した場合は 1,000 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 1,000 ミリ秒以上の場合の誤差は最大 1,000 ミリ秒です。	Integer	100-10,000	1,000
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

パラメタ	説明	型	設定範囲	デフォルト値
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.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	0-65,535,000	60,000
sql.pod.service-name	SQL-Participant を構成する Pod を定義した Kubernetes の Service の Service 名と Namespace 名を「.」でつないだ形式で設定します。 Service 名と Namespace 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 Namespace 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.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、「2.3　コンテナイメージ作成」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.jdbc.driver.additional-library-paths	Oracle Database で XMLType 型を利用する場合に、必要な JDBC Driver の追加ライブラリの絶対パスを指定します。「2.3　コンテナイメージ作成」でコンテナ内に配置したパスを設定してください。 Oracle Database で XMLType 型を利用する場合は、次を指定してください。 • xdb • xmlparserv2_sans_jaxp_services 詳細は、Oracle Database のマニュアルを参照してください。	String	－	なし

パラメタ	説明	型	設定範囲	デフォルト値
	なお、複数指定する場合は「;」区切りで指定してください。 例：/deployments/xdb-23.6.0.24.10.jar;/deployments/xmlparserv2_sans_jaxp_services-23.6.0.24.10.jar			
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 • Oracle Database の場合 oracle.jdbc.xa.client.OracleXADataSource	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	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

	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。
	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、「9.4.3　HiRDB（SQL-Participant 限定）」を参照してください。
	＜ Oracle Database の場合＞
	次の 2 つの方法があります。
	詳細は、Oracle Database の公式ドキュメントの JDBC 接続プロパティを参照してください。
	(1)非データベース URL 形式
	`serverName=oracle-db-sample;portNumber=1521;user=appuser1;password=appuser1;driverType=thin;serviceName=FREEPDB1;nativeXA=true`
	次のプロパティを指定できます。
	• serverName
	• portNumber
	• user			

パラメタ	説明	型	設定範囲	デフォルト値
	• password • driverType 「Thin」を必ず指定してください。 • serviceName • nativeXA 「true」を指定することを推奨します。 (2)データベース URL 形式 `URL=jdbc:oracle:thin:appuser1/appuser1@(DESCRIPTION =(ADDRESS = (PROTOCOL = TCP)(HOST = oracle-db-sample)(PORT = 1521))(CONNECT_DATA =(SERVER = DEDICATED)(SERVICE_NAME = FREEPDB1)));nativeXA=true` 次のプロパティを指定できます。 • URL • nativeXA 「true」を指定することを推奨します。			
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。 スケーリング対応機能を使用する場合は、定期実行の未決着トランザクションの Recovery 処理でコネクションを使用するため、上記の指定値に 2 以上を加えた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 • HiRDB の場合 pd_max_users • Oracle Database の場合 SESSIONS 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。	Integer	1-100,000	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
	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.connection-retry-limit	SQL-Participant のコネクションプールの初期化で、DB とのコネクションの取得に失敗した場合に行うリトライの最大リトライ回数を設定します。	Integer	1-30	20
sql.pool.connection-retry-interval-in-seconds	SQL-Participant のコネクションプールの初期化で、DB とのコネクションの取得に失敗した場合に行うリトライのリトライ間隔を秒単位で設定します。	Integer	1-10	3
sql.pool.transaction-timeout-in-seconds	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、次の動作になります。0 を設定してトランザクションタイムアウト時間を無制限または無限に近い値とすることを推奨します。 • DB 側でトランザクションタイムアウト時間を設定できる場合（postgreSQL および HiRDB)、DB 側の設定が有効になります。 DB 側で無制限または無限に近い値を設定することを推奨します。 • DB 側にトランザクションタイムアウト時間を設定できない場合（Oracle Database および SQL Server)、無制限となります。 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します。 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確認してください。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の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の動作となります。	Integer	0-65,535 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 の仕様を確認してください。	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。	Long	100-6,000,000	60,000

パラメタ	説明	型	設定範囲	デフォルト値
	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。 業務サービスはトランザクションルートとなるサービスから呼び出され、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 1,000 ミリ秒以上の場合の誤差は最大 1,000 ミリ秒です。また、この設定値に 1,000 ミリ秒以上を指定した場合は、1,000 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 1,000 ミリ秒以上の場合の誤差は最大 1,000 ミリ秒です。また、この設定値に 1,000 ミリ秒以上を指定した場合は、1,000 の単位に切り上げたタイマ値とします。	Long	100-6,000,000	1,000
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（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）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	Integer	1-1,000	12

パラメタ	説明	型	設定範囲	デフォルト値
	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。			
sql.participant-processing.queue-size	SQL-Participant がトランザクション処理（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）で実行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。 理論上の必要な設定値は次のとおりです。 （m ＋ 7×m）×p×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	1-2,147,483,647	4,096,000
recover.interval-in-millis	recover 処理の実施の間隔をミリ秒単位で指定します。 このパラメタは、スケーリング対応機能を使用している場合だけ有効になります。	Integer	100-6,000,000	60,000
mediator-process-management.thread-pool-size	Mediator へ送信を行う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	1-32,767	30
mediator-communication.thread-pool-size	Participant が Mediator からの受信に使用する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	1-1,000	30
mediator-communication.queue-size	Participant が Mediator 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。 理論上必要な設定値は次のとおりです。 （m ＋ 7×m）×p×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	Integer	1-2,147,483,647	4,096,000

パラメタ	説明	型	設定範囲	デフォルト値
	t：同時実行するトランザクション数			
sql.db-type	SQL-Participant が接続する RDBMS を指定します。	String	PostgreSQL | SQLServer | HiRDB | OracleDatabase	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。 • 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します。	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.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.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	0-2,147,483,647	0
grpc.tls.session.timeout-in-seconds	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	0-65,535	0
grpc.keepalive.interval-in-seconds	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。	Integer	0-100,000	0

パラメタ	説明	型	設定範囲	デフォルト値
	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。			
grpc.keepalive.timeout-in-seconds	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	1-65,535	60
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。			
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。			
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。			
use-scaling	Orchestrator の use-scaling 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。			
set-routing	Orchestrator の set-routing 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。			

（凡例）－：該当しません。

9.2.6　TCC-Participant

TCC-Participant のパラメタについて説明します。

TCC-Participant のパラメタは、Entity-Service(TCC)に設定します。

(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 9-11　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 Namespace 名を「.」でつないだ形式で設定します。	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。

パラメタ	説明	型	設定範囲	デフォルト値
	Service 名と Namespace 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 Namespace 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`			
tcc.recover.enabled	TCC-Participant が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。 このパラメタは、実運用では必ず true を設定してください。運用テスト中に executeRecover がない状態で動作確認したい場合に false を指定します。	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。 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。	Integer	-2,147,483,648-2,147,483,647	2,147,483,647
tcc.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。 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。	Long	100-6,000,000	60,000

パラメタ	説明	型	設定範囲	デフォルト値
	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1,000ミリ秒以上の場合の誤差は最大1,000ミリ秒です。また、この設定値に1,000ミリ秒以上を指定した場合は1,000の単位に切り上げたタイマ値とします。			
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。 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1,000ミリ秒以上の場合の誤差は最大1,000ミリ秒です。また、この設定値に1,000ミリ秒以上を指定した場合は1,000の単位に切り上げたタイマ値とします。	Long	100-6,000,000	60,000
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	1-1,000	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。	Integer	1-2,147,483,647	4,096,000

パラメタ	説明	型	設定範囲	デフォルト値
	理論上必要な設定値は次のとおりです。 （m ＋ 7×m）×p×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	SQL-Participant の mediator-process-management.thread-pool-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。	Integer	1-32,767	30
mediator-communication.thread-pool-size	SQL-Participant の mediator-communication.thread-pool-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。	Integer	1-1,000	30
mediator-communication.queue-size	SQL-Participant の mediator-communication.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。	Integer	1-2,147,483,647	4,096,000
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 1,000 ミリ秒以上を指定した場合は 1,000 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 1,000 ミリ秒以上の場合の誤差は最大 1,000 ミリ秒です。	Integer	100-10,000	1,000
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

パラメタ	説明	型	設定範囲	デフォルト値
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。			20,551
grpc.use-tls	暗号化通信を行う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.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します。	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.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.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	0-2,147,483,647	0

パラメタ	説明	型	設定範囲	デフォルト値
grpc.tls.session.timeout-in-seconds	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	0-65,535	0
grpc.keepalive.interval-in-seconds	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	0-100,000	0
grpc.keepalive.timeout-in-seconds	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	1-65,535	60
use-scaling	スケーリング対応機能を使用するかどうかを指定します。 true：スケーリング対応機能を使用します。 false：スケーリング対応機能を使用しません。 TCC-Participant はスケーリング対応機能に対応していないため、必ず false を指定してください。 このパラメタの設定値はすべて統一する必要があるため、Orchestrator、Entity-Module、SQL-Participant、TCC-Participant、TP1-Bridge、Mediator それぞれの、このパラメタにも必ず false を指定してください。	Boolean	true | false	false

（凡例）－：該当しません。

9.2.7 Spring Boot の組み込み HTTP サーバ

HMP-PCTO で Spring Boot の組み込み HTTP サーバを使用する場合に、指定する必要があるパラメタについて説明します。

Spring Boot の組み込み HTTP サーバのパラメタは、Orchestrator-Service、Relay-Service、Entity-Service(SQL)、Entity-Service(TCC)、Orchestrator-Service(SQL)、Alternate-Service、Alternate-Service(SQL)に設定します。

(1) server.tomcat のパラメタ

各パラメタの接頭辞は、server.tomcat です。

Spring Boot の組み込み HTTP サーバに Tomcat を使用する場合に設定します。

次に示したもの以外のプロパティは任意に設定できます。

表 9-12　Spring Boot の組み込み HTTP サーバの server.tomcat のパラメタ

パラメタ	説明	型	設定範囲	デフォルト値
accesslog.enabled	Tomcat のアクセスログの出力有無を設定します。 トラブルシュートで必要な情報を取得するため、true を設定してください。	Boolean	Spring Boot の仕様に依存します。	Spring Boot の仕様に依存します。

accesslog.pattern	Tomcat のアクセスログのパターンを設定します。
	トラブルシュートで必要な情報を取得するため、次の推奨フォーマットを含んだ値を設定してください。
	• uCosminexus Application Runtime for Spring Boot を導入する場合
	`%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" "%{traceparent}i" "%{tracestate}i"`
	• uCosminexus Application Runtime for Spring Boot を導入しない場合
	`%h %{X-Forwarded-For}i %u %{[dd/MMM/yyyy:HH:mm:ss.SSS Z]}t "%r" %s %b %D "%{Referer}i" "%{User-Agent}i" "%{traceparent}i" "%{tracestate}i"`
		String	Spring Boot の仕様に依存します。※1	Spring Boot の仕様に依存します。

accesslog.locale	Tomcat のアクセスログのロケールを設定します。 デフォルトでは、使用する Java VM のデフォルトロケールが設定されます。 ほかのトラブルシュート情報に合わせて、英語ロケール（en_US）を設定してください。	String	Spring Boot の仕様に依存します。	Spring Boot の仕様に依存します。
basedir※2	Tomcat のベースディレクトリを設定します。任意のディレクトリを指定してください。 Tomcat のベースディレクトリはアプリケーションを起動したディレク	String	Spring Boot の仕様に依存します。	Spring Boot 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
	トリの配下に作成され、Tomcat のアクセスログなどが配置されます。 server.tomcat.accesslog.directory、server.tomcat.accesslog.prefix、server.tomcat.accesslog.suffix、および server.tomcat.accesslog.file-date-format をデフォルト値のまま使用した場合は、次のパスに Tomcat のアクセスログが出力されます。 `<アプリケーション起動ディレクトリ>/<server.tomcat.basedirの指定値>/logs/access_log.<yyyy-MM-dd>.log`			
accesslog.max-days	Tomcat のアクセスログを何日間保持するかを設定します。 デフォルト値の「-1」は無制限を意味するため、ログファイルが単調増加してしまいます。運用要件に合わせて有限の日数を指定してください。	Integer	Spring Boot の仕様に依存します。	Spring Boot の仕様に依存します。

注※1

任意の区切り文字で設定できます。推奨値では半角スペースです。また、日時は任意の書式で設定できます。

注※2

アクセスログファイルを永続化するために Kubernetes の PV としてマウントしたディレクトリを出力先にすると、複数のレプリカ Pod で同じ出力先を使用してしまいます。そのため、Pod 名称を使用してディレクトリを作成するなど、出力ディレクトリを Pod ごとに一意にしてください。

出力ディレクトリがレプリカ Pod ごとに一意になると、Pod が停止および再起動した場合、停止前に使用されていたアクセスログの上書きや削除がされません。そのため、定期的に外部ツールなどで移動や削除などの運用をしてください。

(2) server.undertow のパラメタ

各パラメタの接頭辞は、server.undertow です。

Spring Boot の組み込み HTTP サーバに Undertow を使用する場合に設定します。

次に示したもの以外のプロパティは任意に設定できます。

表 9-13 Spring Boot の組み込み HTTP サーバの server.undertow のパラメタ

パラメタ	説明	型	設定範囲	デフォルト値
accesslog.enabled	Undertow のアクセスログの出力有無を設定します。	Boolean	Spring Boot の仕様に依存します。	Spring Boot 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
	トラブルシュートで必要な情報を取得するため、true を設定してください。			
accesslog.pattern	Undertow のアクセスログのパターンを設定します。 トラブルシュートで必要な情報を取得するため、次の推奨フォーマットを含んだ値を設定してください。 `%h %{i,X-Forwarded-For} %u [%{time,dd/MMM/yyyy:HH:mm:ss.SSS Z}] "%r" %s %b %D "%{i,Referer}" "%{i,User-Agent}" "%{i,traceparent}" "%{i,tracestate}"`	String	Spring Boot の仕様に依存します。※1	Spring Boot の仕様に依存します。
accesslog.dir※2	Undertow のアクセスログを出力する場合に、Undertow のアクセスログのディレクトリを設定します。任意のディレクトリを指定してください。 server.undertow.accesslog.prefix および server.undertow.accesslog.suffix にデフォルト値を設定し、server.undertow.accesslog.rotate に false を設定した場合、次のパスに Undertow のアクセスログが出力されます。 `<server.undertow.accesslog.dir の指定値>/access_log.log`	String	Spring Boot の仕様に依存します。	Spring Boot の仕様に依存します。
options.server.record-request-start-time	Undertow のリクエスト開始時刻を内部に記録し、リクエスト時間を記録できるようにします。 アクセスログに「%D（ミリ秒の処理時間)」を出力するため、true を設定してください。 未設定または false を設定した場合、アクセスログには"-"が表示されます。	Boolean	Spring Boot の仕様に依存します。	Spring Boot の仕様に依存します。

注※1

任意の区切り文字で設定できます。推奨値では半角スペースです。また、日時は任意の書式で設定できます。

「%D（ミリ秒の処理時間)」を出力するためには、server.undertow.options.server.record-request-start-time に true を設定してください。未設定または false を設定した場合、アクセスログには"-"が表示されます。

注※2

Undertow には Tomcat のようにアクセスログの保持日数の設定（server.tomcat.accesslog.max-days）ができないため、ログファイルが単調増加してしまいます。外部ツールなどで移動や削除などの運用をしてください。

アクセスログファイルを永続化するために Kubernetes の PV としてマウントしたディレクトリを出力先にすると、複数のレプリカ Pod で同じ出力先を使用してしまいます。そのため、Pod 名称を使用してディレクトリを作成するなど、出力ディレクトリを Pod ごとに一意にしてください。

出力ディレクトリがレプリカ Pod ごとに一意になると、Pod が停止および再起動した場合、停止前に使用されていたアクセスログの上書きや削除がされません。そのため、定期的に外部ツールなどで移動や削除などの運用をしてください。

9.3 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。

9.3.1 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 9-14 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。 オートスケールを使用する場合は、設定しないでください。	Integer	2-64	なし オートスケールを使用しない場合は、 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

パラメタ	説明	型	設定範囲	デフォルト値

resources	Mediator 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.enabled	デプロイ依存関係チェック（初期化コンテナ）を有効にするかどうかを指定してください。 • true：有効にします。 • false：無効にします。 Elastic Stack、Jaeger および Prometheus を使用してトラブルシュート情報を収集する場合、dependencyCheck.enabled パラメタに true を設定し、dependencyCheck.targets パラメタに対象のトラブルシュート情報を収集する際に必要なアプリケーションを指定してください。 この項目は、クラウド環境（Amazon Web Services 環境で Jaeger を使用する場合、および Microsoft Azure 環境で Jaeger と Elasticsearch を使用する場合）にも適用されます。 Google Cloud Platform 環境でトラブルシュート情報を収集する場合は、Elastic Stack、Jaeger および Prometheus の Pod のデプロイを待機する必要がないため、デプロイ依存関係チェックが不要になります。その場合はこのパラメタに false を設定してください。 Helm チャートの記載例については、「3.8.2　Mediator の Helm チャートのカスタマイズ」を参照してください。	Boolean	true | false	false
dependencyCheck.image	「3.4.5　Dependency-Checker コンテナイメージの作成（通常版限	String	−	なし

パラメタ	説明	型	設定範囲	デフォルト値
	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。			dependencyCheck.enabled に true を設定した場合には、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	0-1,080	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「7.7.1　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>	－	なし dependencyCheck.enabled に true を設定した場合には、必ず設定してください。
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.retryCount	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リトライ回数を設定してください。	Integer	0-1,080	0

パラメタ	説明	型	設定範囲	デフォルト値
livenessProbe.healthCheck.retryIntervalSeconds	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リトライ間隔秒数を設定してください。	Integer	0-360	0
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	0-1,080	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。 未決着トランザクション待機スクリプトの最大待機時間がトランザクション決着監視タイマの設定値を超えるように調整してください。 未決着トランザクション待機スクリプトの最大待機時間は「開始時待機秒数（20 秒）」と「リトライ回数×リトライ間隔秒数」を合計した値です。	Integer	0-1,080	10
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

パラメタ	説明	型	設定範囲	デフォルト値
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。 preStop による未決着トランザクション待機スクリプトの最大待機秒数以上の値を設定してください。 デフォルト値である 160 秒は、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（10 回）×リトライ間隔秒数（10 秒）+開始時待機時間（20 秒）+バッファ（40 秒）で算出した値です。	Integer	0-2,147,483,647	160

affinity	Pod の割り当てノードの制御を設定してください。
	デフォルト値は、1 ノードにつき 1 つの Pod だけを配置する指定です。
	可用性の観点から、Pod の数だけワーカーノードを用意し、各ワーカーノードに Pod を 1 つずつ配置することを推奨します。
	例：ワーカーノード数「5」に対してレプリカ数「5」の場合、各ノードに Pod を 1 つ起動させる。
	なお、インフラストラクチャのコスト削減の観点などで、用意するワーカーノードの数が Pod の数を下回る場合でも、Mediator は問題なく動作しますが、ワーカーノード数を削減する代わりに、ノード障害時のトランザクション決着（補助合意）に時間が掛かるおそれがあります。
	また、可用性の観点から、各ワーカーノードに配置する Pod の数は均等にすることを推奨します。
		Object	－	`podAntiAffinity:`
				`  requiredDuringSchedulingIgnoredDuringExecution:`
				`  - labelSelector:`
				`  matchExpressions:`
				`  - key: app.kubernetes.io/name`
				`  operator: In`
				`  values:`
				`  - mediator`
				`  topologyKey: kubernetes.io/hostname`

securityContext	securityContext を指定します。 Google Cloud Platform 環境の場合、fsGroup: 0 を指定してください。	Object	－	{} 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
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。 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	1-1,000	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 ＋ 7×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	1-2,147,483,647	5,120,000
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	1-1,000	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。	Integer	1-2,147,483,647	256,000

パラメタ	説明	型	設定範囲	デフォルト値
	（m ＋ 7×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			
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	1-1,000	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 ＋ 7×m）×p ＋ 1} ×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	1-2,147,483,647	258,000
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 1,024（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	Long	256-100,000	256

パラメタ	説明	型	設定範囲	デフォルト値
	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。			
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	1-1,000	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。通常はデフォルト値を使用してください。	Integer	1-2,147,483,647	100
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	0-3,600	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。 1,024 の倍数を設定してください。	Integer	1,024-16,384	4,096

パラメタ	説明	型	設定範囲	デフォルト値
	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、1,024 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。			
consensusLog.storageClassName	「3.7.2　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プトのコマンド引数の http ポート番号にも使用します。	Integer	1,024-65,535	20,554
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	1,024-65,535	20,551
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行うかどうかを指定します。	Boolean	true | false	false

パラメタ	説明	型	設定範囲	デフォルト値
	• 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。			
grpc.keepalive.intervalInSeconds	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	0-100,000	0
grpc.keepalive.timeoutInSeconds	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	1-65,535	60
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	Integer	1-17,000	512

パラメタ	説明	型	設定範囲	デフォルト値
	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。 1,400 +(800×トランザクションに参加した Participant 数)			
tracing.instanceId	共通の tracing.instance-id と同じです。 「(3)　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。 「(3)　jp.co.Hitachi.soft.hmppcto のパラメタ」を参照してください。			
tracing.useTrace	共通の tracing.use-trace と同じです。 「(3)　jp.co.Hitachi.soft.hmppcto のパラメタ」を参照してください。			
tracing.debug	共通の tracing.debug と同じです。 「(3)　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。 「(3)　jp.co.Hitachi.soft.hmppcto のパラメタ」を参照してください。			
tracing.delayTimeInMillis	共通の tracing.delay-time-in-millis と同じです。 「(3)　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。 「(3)　jp.co.Hitachi.soft.hmppcto のパラメタ」を参照してください。			
tracing.grpc.debug	共通の tracing.grpc.debug と同じです。 「(3)　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	String	−	-XX:+UseG1GC

パラメタ	説明	型	設定範囲	デフォルト値
	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。			
ucars.enabled	Mediator コンテナで uCosminexus Application Runtime for Spring Boot を有効にするかどうかを設定します。 • 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）	Boolean	true | false	true
ucars.snapshots.claimName	「3.7.10　uCosminexus Application Runtime 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 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。
labels.hmppctoAppGroup	ラベル「hmppcto-app-group」の値に指定する「アプリケーショングループ名」を設定します。 hmppcto-app-group パラメタの説明については、「付録 B.1　HMP-PCTO 固有の指定が必要なパラメタ」を参照してください。 「アプリケーショングループ」については、「7.2　Kubernetes アプリケーションのグループ定義」を参照してください。	String	63 バイト※以内の文字列	なし 必ず設定してください。
use-scaling	Orchestrator の use-scaling 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。			

（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。

9.3.2　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。

表 9–15　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
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
labels.appKubernetesIoName	ラベル「app.kubernetes.io/name」に指定する値を設定します。 ほかの Kubernetes アプリケーションの「app.kubernetes.io/name」に設定した値と異なる値を設定してください。	String	16 バイト以内の文字列	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
	Blue/Green デプロイメントを実施する場合は、Blue と Green の「labels.appKubernetesIoName」に同じ値を設定してください。			
labels.hmppctoAppGroup	Mediator の labels.hmppctoAppGroup と同じです。「(1)　Helm チャートのパラメタ」を参照してください。			

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	0-1,080	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	List<String>	−	• - my-namespace/statefulsets/mediator

パラメタ	説明	型	設定範囲	デフォルト値
	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、「7.7.1　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。			• - my-namespace/statefulsets/eads-server-eads-server
filebeat.enabled	Filebeat のサイドカーコンテナを有効にするかどうかを指定してください。 • true：有効にします。 • false：無効にします。	Boolean	true | false	true
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
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

パラメタ	説明	型	設定範囲	デフォルト値
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	0-1,080	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	0-1,080	30
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

パラメタ	説明	型	設定範囲	デフォルト値
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	0-2,147,483,647	330
securityContext	securityContext を設定します。 Google Cloud Platform 環境の場合、fsGroup: 0 を指定してください。	Object	−	{} 必ず設定してください。
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	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	1-65,535	8,008

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.connectTimeInSeconds	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.watchTimeInSeconds	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	0-65,535	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	5,001-65,535	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	0-65,535	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	0-65,535	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	4,096-1,048,576	4,096
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	4,096-1,048,576	4,096
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	16-1,048,576	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	4,096-1,048,576	4,096
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	4,096-1,048,576	4,096
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、Java VM の送信バッファサイズが適用されます。	Integer	8,192-2,147,483,647	省略した場合は、Java VM の送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
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、Java VM の受信バッファサイズが適用されます。	Integer	8,192-2,147,483,647	省略した場合は、Java VM の受信バッファサイズを適用します。
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 1,000 ミリ秒以上を設定した場合は 1,000 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 1,000 ミリ秒以上の場合の誤差は、最大 1,000 ミリ秒です。	Integer	100-10,000	1,000
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.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。	Long	0-65,535,000	60,000

パラメタ	説明	型	設定範囲	デフォルト値

tp1.pod.serviceName	TP1-Bridge を構成する Pod を定義した Kubernetes の Service の Service 名と Namespace 名を「.」でつないだ形式で設定します。
	Service 名と Namespace 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 Namespace 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.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。	Integer	1-1,024	なし 必ず設定してください。
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	Long	100-6,000,000	60,000

パラメタ	説明	型	設定範囲	デフォルト値
	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 1,000 ミリ秒以上の場合の誤差は、最大 1,000 ミリ秒です。また、この設定値に 1,000 ミリ秒以上を設定した場合は、1,000 の単位に切り上げたタイマ値とします。			
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 • 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	Long	100-6,000,000	1,000

パラメタ	説明	型	設定範囲	デフォルト値
	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 1,000 ミリ秒以上の場合の誤差は、最大 1,000 ミリ秒です。また、この設定値に 1,000 ミリ秒以上を設定した場合は 1,000 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	1-1,000	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 ＋ 7×m）×p×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 および TP1-Bridge の数 t：同時実行するトランザクション数	Integer	1-2,147,483,647	4,096,000
mediatorProcessManagement.threadPoolSize	SQL-Participant の mediator-process-management.thread-pool-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。	Integer	1-32,767	30

パラメタ	説明	型	設定範囲	デフォルト値
mediatorCommunication.threadPoolSize	SQL-Participant の mediator-communication.thread-pool-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。	Integer	1-1,000	30
mediatorCommunication.queueSize	SQL-Participant の mediator-communication.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。	Integer	1-2,147,483,647	4,096,000
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	1,024-65,535	20,554
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の Kubernetes オブジェクト名を設定してください。	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	0-2,147,483,647	0

パラメタ	説明	型	設定範囲	デフォルト値
grpc.tls.session.timeoutInSeconds	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	0-65,535	0
grpc.keepalive.intervalInSeconds	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	0-100,000	0
grpc.keepalive.timeoutInSeconds	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	1-65,535	60
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。			20,551
tracing.instanceId	共通の tracing.instance-id と同じです。 「(3)　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。 「(3)　jp.co.Hitachi.soft.hmppcto のパラメタ」を参照してください。			
tracing.useTrace	共通の tracing.use-trace と同じです。 「(3)　jp.co.Hitachi.soft.hmppcto のパラメタ」を参照してください。			
tracing.debug	共通の tracing.debug と同じです。 「(3)　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。 「(3)　jp.co.Hitachi.soft.hmppcto のパラメタ」を参照してください。			
tracing.delayTimeInMillis	共通の tracing.delay-time-in-millis と同じです。 「(3)　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。			

パラメタ	説明	型	設定範囲	デフォルト値
	「(3)　jp.co.Hitachi.soft.hmppcto のパラメタ」を参照してください。			
tracing.grpc.debug	共通の tracing.grpc.debug と同じです。 「(3)　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。「-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。	String	－	-XX:+UseG1GC
ucars.snapshots.claimName	「3.7.10　uCosminexus Application Runtime 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。※2	String	－	• eads.client.connect.servers=sv1,sv2,sv3

パラメタ	説明	型	設定範囲	デフォルト値
	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。			• 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
use-scaling	スケーリング対応機能を使用するかどうかを指定します。 true：スケーリング対応機能を使用する false：スケーリング対応機能を使用しない TP1-Bridge はスケーリング対応機能に対応していないため、必ず false を指定してください。 このパラメタの設定値はすべて統一する必要があるため、Orchestrator、Entity-Module、SQL-Participant、TCC-Participant、TP1-Bridge、Mediator それぞれの、このパラメタにも必ず false を指定してください。	Boolean	true | false	false

（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

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
```

9.3.3 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 の機能を使用する Kubernetes アプリケーションの 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 9-16　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「(g)　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。

パラメタ	値	説明
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。 トランザクションの同時実行数×10
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※÷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）× 3,600（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「(23)　EADS のセットアップ（通常版限定）」で、eztool compaction コマンドを定期的に実行する際の 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 3,600（秒/時）で割った値を設定してください。

注※

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、eztool コマンド実行用および生存監視ツール実行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 ファイルのサンプルは、取扱説明書「EADS コンテナオーケストレーションツール対応機能」を参照してください。

values.yaml ファイルでの設定は、EADS サーバの Helm チャートのパラメタ値およびプロパティファイルに展開されます。

(b) Helm チャートのパラメタ

Helm チャートのパラメタを説明します。

パラメタの仕様や指定方法および、生存監視ツールのパラメタについては、取扱説明書「EADS コンテナオーケストレーションツール対応機能」を参照してください。

表 9-17 EADS の Helm チャートのパラメタ

パラメタ	説明
release.server	EADS サーバのリリース名を指定します。 このマニュアルでは、eads-server を指定した前提で各種例を記載しています。それ以外の値を指定する場合は、eads-server を読み替えてください。
release.service	EADS サーバ用サービスのリリース名を指定します。 このマニュアルでは、eads-service を指定した前提で各種例を記載しています。それ以外の値を指定する場合は、eads-service を読み替えてください。
namespace	デプロイ先の Namespace 名を指定します。 このマニュアルでは、my-namespace を指定した前提で各種例を記載しています。それ以外の値を指定する場合は、my-namespace を読み替えてください。
caches.name	キャッシュ名を指定します。 このマニュアルでは、hmppctoCache を指定した前提で各種例を記載しています。それ以外の値を指定する場合は、hmppctoCache を読み替えてください。
caches.pvc.accessModes	キャッシュに使用する永続ボリュームのアクセスモードとして、StatefulSet 定義内の volumeClaimTemplates 配列内の accessModes フィールドで使用されます。 永続ボリュームをローカル永続ボリューム（local）とする場合、ReadWriteOnce を指定してください。 永続ボリュームを NFS ストレージとする場合、ReadWriteMany を指定してください。
caches.pvc.storageClassName	キャッシュに使用する永続ボリュームの種類が NFS ストレージの場合の StorageClass の名称を指定します。

パラメタ	説明
	このマニュアルでは、eads-storage を指定した前提で各種例を記載しています。それ以外の値を指定する場合は、eads-storage を読み替えてください。
logs.accessModes	ログに使用する永続ボリュームのアクセスモードとして、StatefulSet 定義内の volumeClaimTemplates 配列内の accessModes フィールドで使用されます。 永続ボリュームをローカル永続ボリューム（local）とする場合、ReadWriteOnce を指定してください。 永続ボリュームを NFS ストレージとする場合、ReadWriteMany を指定してください。
logs.storageClassName	ログに使用する永続ボリュームの種類が NFS ストレージの場合の StorageClass の名称を指定します。 このマニュアルでは、eads-storage を指定した前提で各種例を記載しています。それ以外の値を指定する場合は、eads-storage を読み替えてください。
logs.dirType	ログの出力先ディレクトリの種別を指定します。 • local：ワーカーノード上の領域を使用するディレクトリ • nfs：NFS 上の領域を使用するディレクトリ デフォルトは local です。 永続ボリュームを NFS ストレージとする場合、nfs を指定してください。
store.storageClassName	ストアデータファイルに使用する永続ボリュームの種類が NFS ストレージの場合の StorageClass の名称を指定します。 このマニュアルでは、eads-storage を指定した前提で各種例を記載しています。それ以外の値を指定する場合は、eads-storage を読み替えてください。

(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_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 9-18 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。

パラメタ	説明
	推奨値は、トランザクションに参加する Participant 数×Participant の Pod のレプリカの最大数※1×eads.client.connectionPool.poolsize×2（安全係数※2）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	8,192 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。 10,000 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。 2,500 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。 2,500 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.logger.multipleWritePrevention.enable	ログファイルの多重書き込み防止機能を有効にするかを指定します。true の場合有効となり、それ以外では無効となります。デフォルトは false（無効）です。 永続ボリュームを NFS ストレージとする場合、true を指定してください。 多重書き込み防止機能およびこのパラメタの仕様や指定方法については、取扱説明書「EADS コンテナオーケストレーションツール対応機能」を参照してください。

注※1

スケーリング対応機能を使用しない場合は 1、スケーリング対応機能を使用する場合はオートスケール用 HPA の Kubernetes マニフェストに指定した「スケールアウトできる最大の Pod 数」です。

注※2

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。

(e) 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 9-19 eads_shared.properties のパラメタ

パラメタ	説明
eads.replication.factor （values.yaml の「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 9-16 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 9-16 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 9-16 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 9-16 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 9-16 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。

(f) 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 9-20　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。

(g)　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 9-21　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 9-16　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 9-16　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 9-16　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.multipleWritePrevention.enable	キャッシュファイルの多重書き込み防止機能を有効にするかを指定します。true の場合有効となり、それ以外では無効となります。デフォルトは false（無効）です。 永続ボリュームを NFS ストレージとする場合、true を指定してください。 多重書き込み防止機能およびこのパラメタの仕様や指定方法については、取扱説明書「EADS コンテナオーケストレーションツール対応機能」を参照してください。

(h)　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。

(i)　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 の機能を使用する Kubernetes アプリケーションの 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　Entity-Service(SQL)の Kubernetes マニフェストの作成（SQL-Participant 限定）」、または「2.6.5　Entity-Service(TCC)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 9-22　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。 HMP-PCTO は、EADS のクライアント名として、Pod 名の UUID を設定します。UUID の「-」は「_」に変換して設定します。EADS クライアントログの出力先ディレクトリは、eads-log/<Pod 名の UUID>となります。
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
```

9.3.4　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 9-23　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 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
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	0-1,080	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「7.7.1　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>	−	なし 必ず設定してください。
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	1,024-65,535	20,556
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。

9.3.5　Filebeat

Filebeat のパラメタを説明します。

(1)　Filebeat のパラメタ

ここで示す Filebeat の各パラメタは、公式ドキュメントに定義されたデフォルト値を前提とし、HMP-PCTO での設計方針に基づいて、推奨設定値を定めています。

Filebeat の各パラメタの詳細な仕様および制約については、Elastic の公式ドキュメントを参照してください。

表 9-24　Filebeat のパラメタ

パラメタ	説明	型	推奨または必須の設定値	OSS デフォルト値※
queue.disk.max_size	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。	String	10GB	10GB
queue.disk.read_ahead	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。	String	512	512

注※

OSS デフォルト値は、Filebeat 8.13.0 でのデフォルト値です。

9.3.6　Jaeger

Jaeger は公式ドキュメントを参照し、インストールします。この時、HMP-PCTO として制約のあるパラメタがあります。制約について、次の表に示します。

表 9-25　Jaeger で HMP-PCTO として制約のあるパラメタ

パラメタ	説明	制約
image	起動するコンテナイメージのバージョンやビルド元を明示的に管理するために使用します。	Jaeger のコンテナイメージを設定してください。 使用できる Jaeger のバージョンについては、リリースノートを参照してください。

Jaeger の各パラメタは公式ドキュメントに従って設定してください。

(1) Jaeger のパラメタ

ここで示す Jaeger の各パラメタは、公式ドキュメントに定義されたデフォルト値を前提とし、HMP-PCTO での設計方針に基づいて、推奨設定値を定めています。

Jaeger の各パラメタの詳細な仕様および制約については、Jaeger の公式ドキュメントを参照してください。

表 9-26　Jaeger のパラメタ

パラメタ	説明	型	設定範囲	推奨または必須の設定値	OSS デフォルト値※1
exporters.jaeger_storage_exporter.queue.queue_size	Exporter の内部キューのサイズを設定してください。	Integer	－※2	2,000	1,000
exporters.jaeger_storage_exporter.queue.num_consumers	Exporter の内部で動作するワーカの数を設定してください。	Integer	－※2	10	10
processors.batch.timeout	分散トレースのバッチを Exporter に渡すまでの最大待機時間を設定してください。	Duration	－※2	200ms	200ms
processors.batch.send_batch_size	分散トレースがその件数に達した場合に、Exporter へバッチを即時送信するためのトリガ件数を設定してください。	Integer	－※2	8,192	8,192
processors.batch.send_batch_max_size	分散トレースのバッチを Exporter へ送信する場合の、1 回当たりの最大件数を設定してください。 0 を指定した場合は、上限なしとなります。	Integer	－※2	0	0

（凡例）

－：該当しません。

注※1

OSS デフォルト値は、Jaeger 2.15.1 でのデフォルト値です。

注※2

Jaeger に対する設定値のため、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 の公式ドキュメントを参照してください。

9.3.7 Prometheus

Prometheus は公式ドキュメントを参照し、インストールします。この時、HMP-PCTO として制約のあるパラメタがあります。制約について、次の表に示します。

表 9-27　Prometheus で HMP-PCTO として制約のあるパラメタ

パラメタ	説明	制約
image	起動するコンテナイメージのバージョンやビルド元を明示的に管理するために使用します。	Prometheus のコンテナイメージを設定してください。 使用できる Prometheus のバージョンについては、リリースノートを参照してください。
service.port	外部アクセスや他コンポーネントとの連携に使用します。	Prometheus 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。

(1)　Prometheus のパラメタ

ここで示す Prometheus の各パラメタは、公式ドキュメントに定義されたデフォルト値を前提とし、HMP-PCTO での設計方針に基づき、推奨設定値を定めています。

Prometheus の各パラメタの詳細な仕様および制約については、Prometheus の公式ドキュメントを参照してください。

表 9-28　Prometheus で HMP-PCTO として制約のあるパラメタ

パラメタ	説明	型	設定範囲	推奨または必須の設定値	OSS デフォルト値※1
global.scrape_interval	Prometheus がメトリクスの取得対象から情報を収集する間隔を設定してください。	String	―※2	15s	1m
global.scrape_configs	メトリクスの収集対象を設定してください。 このパラメタの設定を変更することでメトリクスの収集対象の追加や、ラベルの設定ができるようになります。設定できる値については、Prometheus の公式ドキュメントを参照してください。 HMP-PCTO として必要なメトリクスを収集するための収集対象については、「3.7.13 Prometheus の Kubernetes マニフェスト作成」を参照してください。	String	―	―	―
--storage.tsdb.retention.time	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。	String	―※2	7d	15d
remote_write.queue_config.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	String	―※2	10,000	10,000

パラメタ	説明	型	設定範囲	推奨または必須の設定値	OSS デフォルト値※1
	キューのサイズを設定してください。				
remote_write.queue_config.max_shards	Metricbeat に送信するデータの最大サイズを設定してください。 この設定値は、送信が遅延した場合などに有効になる最大サイズを指定するものです。	String	－※2	50	50
remote_write.queue_config.max_samples_per_send	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。	String	－※2	2,000	2,000
http_config.enable_http2	Prometheus が Remote Write で外部ストレージにデータを送信する場合、HTTP/2 を使用するかどうかを設定してください。	Boolean	－※2	true	true

（凡例）

－：該当しません。

注※1

OSS デフォルト値は、Prometheus 3.5.0 でのデフォルト値です。

注※2

Prometheus への設定値のため、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 の公式ドキュメントを参照してください。

9.3.8 Metricbeat

Metricbeat のパラメタを説明します。

(1) Metricbeat のパラメタ

ここで示す Metricbeat の各パラメタは、公式ドキュメントに定義されたデフォルト値を前提とし、HMP-PCTO での設計方針に基づいて、推奨設定値を定めています。

Metricbeat の各パラメタの詳細な仕様および制約については、Elastic の公式ドキュメントを参照してください。

表 9-29 Metricbeat のパラメタ

パラメタ	説明	型	推奨または必須の設定値	OSS デフォルト値※
queue.disk.max_size	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。	String	10GB	10GB

パラメタ	説明	型	推奨または必須の設定値	OSS デフォルト値※
queue.disk.read_ahead	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。	String	512	512

注※

OSS デフォルト値は、Metricbeat 8.13.0 でのデフォルト値です。

9.3.9 Logstash

Logstash のパラメタを説明します。

(1) Logstash のパラメタ

ここで示す Logstash の各パラメタは、公式ドキュメントに定義されたデフォルト値を前提とし、HMP-PCTO での設計方針に基づいて、推奨設定値を定めています。

Logstash の各パラメタの詳細な仕様および制約については、Elastic の公式ドキュメントを参照してください。

表 9-30 Logstash のパラメタ

パラメタ	説明	型	推奨または必須の設定値	OSS デフォルト値※
pipeline.batch.size	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。	String	125	125
queue.max_events	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。queue.max_bytes パラメタを有効になるように設定している場合、この設定が有効となります。	String	0（無制限）	0（無制限）
queue.max_bytes	Logstash の内部キューの最大容量を設定してください。queue.max_events パラメタを有効になるように設定している場合、この設定は無効となります。	String	なし	なし

注※

OSS デフォルト値は、Logstash 8.13.0 でのデフォルト値です。

9.3.10 Metrics Server

Metrics Server は公式ドキュメントを参照し、インストールします。この時、HMP-PCTO として制約のあるパラメタがあります。制約について、次の表に示します。

表 9-31　Metrics Server で HMP-PCTO として制約のあるパラメタ

パラメタ	説明	制約
image	起動するコンテナイメージのバージョンやビルド元を明示的に管理するために使用します。	Metrics Server のコンテナイメージを設定してください。 使用できる Metrics Server のバージョンについては、リリースノートを参照してください。
service.port	外部アクセスや他コンポーネントとの連携に使用します。	Metrics Server 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。

9.4　各 RDBMS の使用に必要なパラメタおよび設定（SQL-Participant 限定）

次に示す RDBMS を使用する際に必要なパラメタおよび設定について説明します。

- PostgreSQL
- SQL Server
- HiRDB
- Oracle Database

9.4.1　PostgreSQL（SQL-Participant 限定）

(1)　パラメタ

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 9-32　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×Participant の Pod のレプリカの最大数※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、「(1)　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。

注※

スケーリング対応機能を使用しない場合は 1、スケーリング対応機能を使用する場合はオートスケール用 HPA の Kubernetes マニフェストに指定した「スケールアウトできる最大の Pod 数」です。

(2)　接続するユーザに付与する権限

HMP-PCTO が RDBMS に接続する際のユーザに、次に示す権限を付与してください。詳細については、PostgreSQL のマニュアルを参照してください。

- CONNECT 権限
- 他ユーザの未決着トランザクションを決着させる場合は、スーパーユーザ権限

9.4.2 SQL Server（SQL-Participant 限定）

(1) パラメタ

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 9-33 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×Participant の Pod のレプリカの最大数※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、「(1) jp.co.Hitachi.soft.hmppcto のパラメタ」を参照してください。

注※

スケーリング対応機能を使用しない場合は 1、スケーリング対応機能を使用する場合はオートスケール用 HPA の Kubernetes マニフェストに指定した「スケールアウトできる最大の Pod 数」です。

(2) 接続するユーザに付与する権限

HMP-PCTO が RDBMS に接続する際のユーザに、次に示す権限を付与してください。詳細については、SQL Server のマニュアルを参照してください。

- SqlJDBCXAUser ロール

9.4.3 HiRDB（SQL-Participant 限定）

(1) パラメタ

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 9-34　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×Participant の Pod のレプリカの最大数※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、「(1)　jp.co.Hitachi.soft.hmppcto のパラメタ」を参照してください。

注※

スケーリング対応機能を使用しない場合は 1、スケーリング対応機能を使用する場合はオートスケール用 HPA の Kubernetes マニフェストに指定した「スケールアウトできる最大の Pod 数」です。

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 9-35　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。

(2)　接続するユーザに付与する権限

HMP-PCTO が RDBMS に接続する際のユーザに、次に示す権限を付与してください。詳細については、HiRDB のマニュアルを参照してください。

- CONNECT 権限

9.4.4　Oracle Database（SQL-Participant 限定）

(1)　パラメタ

Oracle Database 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、Oracle Database 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含めた詳細については、Oracle Database のマニュアルを参照してください。

表 9-36　Oracle Database のパラメタ

パラメタ	説明
SESSIONS	Oracle Database に同時接続する最大数を設定します。sql.pool.max-pool-size に設定した値×Participant の Pod のレプリカの最大数※に指定した値より大きい値を指定してください。 詳細については、「(1)　jp.co.Hitachi.soft.hmppcto のパラメタ」を参照してください。

注※

スケーリング対応機能を使用しない場合は 1、スケーリング対応機能を使用する場合はオートスケール用 HPA の Kubernetes マニフェストに指定した「スケールアウトできる最大の Pod 数」です。

(2)　接続するユーザに付与する権限

HMP-PCTO が RDBMS に接続する際のユーザに、次に示す権限を付与してください。詳細については、Oracle Database のマニュアルを参照してください。

- SYS.DBA_PENDING_TRANSACTIONS の SELECT 権限
- FORCE ANY TRANSACTION 権限

9.5 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。

9.5.1 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 9-37 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。

9.5.2 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 9-38　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。

9.5.3　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 9-39　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。

9.5.4　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 9-40　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。

9.6 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。

9.6.1 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 9-41 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.directory	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。

（凡例）−：該当しません。

(2) webserver.connector のパラメタ

表 9-42 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	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
	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。			
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。			

(3) ejbserver.application のパラメタ

表 9-43 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。			
ejbserver.application.userlog.Logger.jp.co	ロガーが出力するメッセージのレベルを設定します。			

パラメタ	説明	型	設定範囲	デフォルト値
.Hitachi.soft.hmppcto.converter.level	「INFO」を設定してください。			

9.6.2　usrconf.cfg（通常版かつ TP1-Bridge 限定）

(1)　add のパラメタ

表 9–44　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。		

10 メッセージ

10.1　メッセージの概要

10.1.1　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 10-1　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 10–2　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

10.1.2　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 10–3　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

10.1.3　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。

10.2 メッセージ一覧

10.2.1 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、スケーリング対応機能が無効なときは、Participant の起動を中止します。スケーリング対応機能が有効なときは、処理を継続します。

処理の種別が 004 以外の場合、Participant の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］

処理の種別が 004 の場合、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 を起動または再起動してください。

それ以外の場合、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.grpc.port

(S)gRPC サーバの起動処理を終了します。

(O)なし。

〔対策〕

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。

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 または 005 の場合は Participant の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 Participant の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 メッセージまたは 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。
  jp.co.Hitachi.soft.hmppcto.sql.db-type パラメタに PostgreSQL を指定している場合は PostgreSQL の max_prepared_transactions パラメタ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 001 または 005 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)なし。

〔対策〕

jp.co.Hitachi.soft.hmppcto.sql.datasource.class-name パラメタ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。
jp.co.Hitachi.soft.hmppcto.sql.datasource.property.values パラメタに指定したプロパティ名を見直してください。そのあと、Pod を再起動してください。

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（jp.co.Hitachi.soft.hmppcto.sql.pool.transaction-timeout-in-seconds パラメタ）

(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、Pod 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)なし。

〔対策〕

直後に出力するスタックトレースを参照し、jp.co.Hitachi.soft.hmppcto.sql.datasource.property.values パラメタに指定した値を見直してください。そのあと、Pod を再起動してください。

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 jp.co.Hitachi.soft.hmppcto.sql.pool.max-pool-size パラメタを超えている場合、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：ラベル「app.kubernetes.io/name」に指定した値
- TP1-Bridge を使用する場合：labels.appKubernetesIoName パラメタに指定した値

(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 メッセージを参照して原因を取り除き、Pod 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。

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)

Participant の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)

Participant の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。
　スケーリング対応機能が有効な場合は、未決着トランザクションの Recovery を終了し処理を続行します。そのあと、jp.co.Hitachi.soft.hmppcto.recover.interval-in-millis パラメタに設定した間隔で、再度未決着トランザクションの Recovery を実行します。

(O)なし。

〔対策〕

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

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：ラベル「app.kubernetes.io/name」に指定した値
- TP1-Bridge を使用する場合：labels.appKubernetesIoName パラメタに指定した値

(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 Participant 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〕

このメッセージに先行して出力される KFSG60000-E メッセージ、KFSG60034-E メッセージまたは KFSG60035-E メッセージに従い対処してください。

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。「3.12.2　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)

Participant の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)なし。

〔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)

Participant の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 メッセージが出力された場合は、Participant を起動しません。KFSG60024-E メッセージ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)

Participant の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)なし。

［対策］

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。

KFSG60024-E (Y)M

The transaction branch could not be settled. (rootxid = aa....aa)

Participant の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)なし。

[対策]

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。

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. (service name = aa....aa, pod hostname = bb....bb, pod IP address = cc....cc)

Participant への通信処理でエラーが発生しました。

aa....aa：Participant のドメイン名（Participant 情報取得前の場合は*）

bb....bb：Participant の Pod のホスト名（Participant 情報取得前の場合は*）

cc....cc：Participant の Pod の IP アドレス（Participant 情報取得前の場合は*）

(S)Participant への通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)なし。

〔対策〕

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。

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 Participant の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］

KFSG62014-W 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、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。

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 メッセージまたは 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。必要に応じてデータソースの接続情報を確認してください。

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)なし。

〔対策〕

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

KFSG60034-E (Y)M

An internal contradiction was detected. (branchxid = aa....aa, information 1 = bb....bb, information 2 = cc....cc)

内部矛盾を検知しました。

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

(S)処理の種別が 004 の場合、スケーリング対応機能が無効なときは、Participant の起動を中止します。スケーリング対応機能が有効なときは、処理を継続します。

処理の種別が 004 の以外の場合、Participant の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］

保守員に連絡してください。

KFSG60035-E (Y)M

Invalid data was received from the Resource Manager. (branchxid = aa....aa, information 1 = bb....bb, information 2 = cc....cc)

リソースマネージャから不正なデータを受信しました。

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

(S)処理の種別が 004 の場合、スケーリング対応機能が無効なときは、Participant の起動を中止します。スケーリング対応機能が有効なときは、処理を継続します。

処理の種別が 004 の以外の場合、Participant の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］

保守員に連絡してください。

KFSG60036-E (Y)M

An error occurred in the processing to set the transaction timeout for the Resource Manager. Enable the transaction timeout setting on the Resource Manager side. (transaction timeout = aa....aa, information 1 = bb....bb)

リソースマネージャへ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。リソースマネージャ側のトランザクションタイムアウト設定が有効になります。

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-seconds）

bb....bb：リソースマネージャが返却したエラーコード

(S)処理を続行します。

(O)なし。

〔対策〕

なし。

KFSG60037-E (Y)M

The library file does not exist. (file path = aa....aa)

ライブラリファイルが存在しません。

aa....aa：ライブラリファイルのパス（jp.co.Hitachi.soft.hmppcto.sql.jdbc.driver.additional-library-paths パラメタに指定したパス）

(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〕

ライブラリファイルが存在するかどうか、または jp.co.Hitachi.soft.hmppcto.sql.jdbc.driver.additional-library-paths パラメタの値が正しいかどうかを確認してください。

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)システム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)

次のどれかの場合に、このメッセージを出力します。

- 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。
- Mediator への gRPC 通信の送信を行う前に、Mediator の Pod が存在しないことを検知しました。
  メッセージの次行から HMP-PCTO の Exception 情報を出力します。
- Mediator への gRPC 通信の送信を行う前に、Mediator の Pod が再起動し、該当する Pod に IP アドレスがまだ割り当てられていないことを検知しました。
  メッセージの次行から HMP-PCTO 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)なし。

〔対策〕
Exception 情報を確認し、次のとおり対処してください。

- IllegalStateException が出力されている場合：
  HMP-PCTO の Exception 情報です。Mediator の Pod が存在しないことを検知したことで出力します。Mediator の Pod が存在しない要因を、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。
- UnknownHostException が出力されている場合：
  HMP-PCTO の Exception 情報です。Mediator の Pod が再起動し、該当する Pod に IP アドレスがまだ割り当てられていないことを検知したことで出力します。Mediator の Pod が再起動した要因を、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。
- 上記以外の Exception 情報が出力されている場合：
  gRPC 通信の Exception 情報です。Exception 情報を確認して対策を行うか、または Mediator などに問題が発生していないかの確認結果や、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。

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)システム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)システム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、必要に応じて 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 または 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]

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)なし。

[対策]

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。

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)なし。

［対策］

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。

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)なし。

［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。

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)なし。

［対策］

このメッセージに先行して出力される KFSG61105-E メッセージ、KFSG61106-E メッセージ、KFSG61107-E メッセージ、または KFSG61108-E メッセージに従い対処してください。

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、Mediator および該当する Mediator と接続して、HMP-PCTO の機能を使用するアプリケーションの Pod を停止してから、ConsensusLog ファイルを削除してください。そのあと、各 Pod 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)なし。

〔対策〕
reason の内容に従って対処し、Mediator を再起動してください。

- 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 メッセージ、または 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、または 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：consensusLog.size パラメタの値

(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］

consensusLog.size パラメタの値を見直し、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（jp.co.Hitachi .soft.hmppcto.consensus-log.io.watch-in-seconds パラメタ）

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：consensusLog.size パラメタの値

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〕

consensusLog.size パラメタの値を見直し、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)なし。

[対策]

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。

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、該当する Mediator と接続して、HMP-PCTO の機能を使用するアプリケーションの Pod を停止してから、ConsensusLog ファイルを削除してください。そのあと、各 Pod 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.transaction-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。「3.12.2　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、Mediator および該当する Mediator と接続して、HMP-PCTO の機能を使用するアプリケーションの Pod を停止してから、ConsensusLog ファイルを削除してください。そのあと、各 Pod 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)なし。

〔対策〕

このメッセージに先行して出力される KFSG61105-E メッセージ、KFSG61107-E メッセージに従い対処してください。

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

TccService#executeTry() method was executed outside the scope of a transaction.

トランザクションの範囲外で TccService#executeTry()メソッドが実行されました。

(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、また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.transaction-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。「3.12.2　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リプレイインターバルタイマまたは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)なし。

〔対策〕

- メッセージが繰り返し出力される場合は、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
- 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。「3.12.2　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> [context-path] [--scheme=http|https] [--insecure]

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> [context-path] [--scheme=http|https] [--insecure]
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 メッセージ、または KFSG82102-I メッセージに含まれているコマンドライン引数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。

・「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。

  ・HTTP ステータスコードが「000」の場合、コマンド引数に指定したポート番号が正しいかどうかを確認してください。誤りがある場合はコマンド引数のポート番号を修正したあと、Pod を再起動してください。

  ・HTTP ステータスコードが「404」の場合、コンテキストパスの設定とコマンド引数の第 6 引数に指定した値が一致しているか確認してください。コンテキストパスが未指定の場合は第 6 引数を省略してください。その後、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。

  ・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 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 メッセージ、または KFSG82102-I メッセージに含まれているコマンドライン引数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。

  ・「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 メッセージ、または KFSG82103-I 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 メッセージ、または KFSG62108-E 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 メッセージ、または KFSG62108-E メッセージ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.

サービスにまだ未決着トランザクションが存在します。

(S)処理を続行します。

(O)なし。

〔対策〕

- 直後に KFSG82101-I メッセージ、または KFSG82103-I 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 メッセージ、または KFSG62108-E 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 メッセージ、または KFSG62108-E メッセージ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]

- Mediator および Participant のメッセージログファイルを確認してください。このメッセージが出力される直前の時間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たあと、Pod を再起動してください。
- Mediator および Participant のメッセージログファイルで、メッセージが出力される直前の時間に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、また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、また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。
- コマンド引数に指定したポート番号が正しいかどうかを確認してください。誤りがある場合はコマンド引数のポート番号を修正したあと、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。

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]

- Mediator および Participant のメッセージログファイルを確認してください。このメッセージが出力される直前の時間に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。
- Mediator および Participant のメッセージログファイルで、メッセージが出力される直前の時間に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、また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、または

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 を再起動してください。

- コマンド引数に指定したポート番号が正しいかどうかを確認してください。誤りがある場合はコマンド引数のポート番号を修正したあと、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。

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 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 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 メッセージ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 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 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 メッセージ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 メッセージ、KFSG62103-W メッセージ、または KFSG62104-W 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 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、また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、また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「7.6　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。
- 同一のアプリケーショングループ内で、hmppcto-app-group ラベルの値に指定しているアプリケーショングループ名が統一され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。「アプリケーショングループ」については、「7.2　Kubernetes アプリケーションのグループ定義」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。

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。

KFSG62300-E (Y)J+M

A request was received that cannot be processed because of a duplicate IP address. (branchxid = aa....aa, IP address = bb....bb)

IP アドレスが重複したため、処理できないリクエストを受信しました。

トランザクションの処理中に終了した Pod と起動した Pod で IP アドレスが重複したおそれがあります。

aa....aa：ブランチ XID

bb....bb：IP アドレス、またはサービス名

(S)リクエスト受信後、対応したアプリケーションを呼び出さないで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〕

- gRPC 通信でスケーリング対応機能を使用する場合に、スキーム名およびロードバランシングポリシー名が正しく指定されているかを確認してください。正しく指定されていない場合、詳細情報の IP アドレスにサービス名が出力されることがあります。
- 繰り返し表示される場合は、保守員に連絡してください。

KFSG62301-E(Y)J+M

The specified in AP Service does not exist. (service = aa....aa, branchxid = bb....bb)

アプリケーションで指定した Service が存在しません。

aa....aa：アプリケーションで指定した Service

bb....bb：ブランチ XID。gRPC 通信時に接続先の名前解決で発生した場合は*。

(S)リクエストを送信しないで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〕

- アプリケーションで指定した Service を示すドメイン名が「2.2.8　通信によるトランザクションブランチの作成（REST 通信-クライアント）」の「Service を示すドメイン名の指定形式」になっていることを確認してください。
- Kubernetes マニフェストの該当の Service の定義が、アプリケーションで指定した Service のドメイン名・ポート番号と一致していることを確認してください。
- Kubernetes 上で該当の Service が起動した状態になっていることを確認してください。
- 同一のアプリケーショングループ内で、hmppcto-app-group ラベルの値に指定しているアプリケーショングループ名が統一され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。「アプリケーショングループ」については、「7.2　Kubernetes アプリケーションのグループ定義」を参照してください。

KFSG62302-E(Y)J+M

There is no Pod associated with the Service specified in the AP (service = aa....aa, branchxid = bb....bb)

アプリケーションで指定した Service に紐づく Pod が存在しません。

aa....aa：アプリケーションで指定した Service。トランザクション外で送信した場合は*。

bb....bb：ブランチ XID。トランザクション外で送信した場合、または gRPC 通信時に接続先の名前解決で発生した場合は*。

(S)リクエストを送信しないで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〕

- 指定した Service に紐づく Pod が起動した状態になっていることを確認してください。
- 同一のアプリケーショングループ内で、hmppcto-app-group ラベルの値に指定しているアプリケーショングループ名が統一され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。「アプリケーショングループ」については、「7.2　Kubernetes アプリケーションのグループ定義」を参照してください。

KFSG62303-E (Y)J+M

An error occurred in the request to the Kubernetes API server. (branchxid = aa....aa)

Kubernetes API サーバへのリクエストでエラーが発生しました。

aa....aa：ブランチ XID

トランザクション処理中でない場合は*

(S)Pod 起動時の場合は、Pod の起動を停止します。

トランザクション処理中の場合は、処理を中断しトランザクションをロールバックします。

未決着トランザクションの Recovery 機能やトランザクション完了確認機能の処理で出力した場合は、処理をスキップします。

(O)システム管理者に連絡してください。

［対策］

次の対策を実施してください。

- 該当のコンテナで「7.6　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。
- 直後に出力されるスタックトレースを参照し、Kubernetes API サーバへのリクエスト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。

KFSG62304-E (Y)J+M

The label value is invalid or not specified. (label = aa....aa, value = bb....bb, reason = cc....cc)

ラベルの指定値が不正または指定されていません。

aa....aa：ラベル名

bb....bb：ラベルの指定値

指定されていない場合は、「null」が表示されます。

cc....cc：理由コード

VALUE：ラベルの指定値が不正です。

NULL：必須ラベルが指定されていません。

(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〕

Participant の機能を使用する Kubernetes アプリケーションの Kubernetes マニフェストのラベルの指定値を見直してください。そのあと、Pod を再起動してください。

ラベル（app.kubernetes.io/name）の設定例については、「2.6　Kubernetes マニフェスト作成」を参照してください。

KFSG62305-E (Y)J+M

Transaction rolled back due to communication with different Pods of the same Service within a single transaction. (branchxid = aa....aa, expected pod uid = bb....bb, actual pod uid = cc....cc)

同一トランザクション内で同一のサービスに対して、異なる Pod へ通信が行われたため、トランザクションをロールバックしました。

aa....aa：ブランチ XID

bb....bb：トランザクションの最初に通信した Pod の uid

cc....cc：エラー発生時に通信した Pod の uid

(S)リクエスト受信後、対応したアプリケーションを呼び出さないで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〕

アプリケーションの実装および配置したロードバランサの設定を見直し、同一トランザクション内で複数回同一のサービスに対して通信する場合に、同一の Pod に対して通信するように修正してください。

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。

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)なし。

〔対策〕

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。

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.initial-size 設定値)

bb....bb：現在の Mediator 起動数

(S)Orchestrator を異常終了します。

(O)なし。

〔対策〕

Mediator の Helm チャートに設定したレプリカ数が mediator.initial-size の設定値以上かどうか確認してください。また、Mediator の状態を確認して、Mediator が mediator.initial-size の設定値以上起動しているの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。

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。

次のどちらかの場合に、このメッセージを出力します。

- 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。
- Mediator への gRPC 通信の送信を行う前に、Mediator の Pod が存在しないことを検知しました。
  メッセージの次行から HMP-PCTO の Exception 情報を出力します。

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)なし。

〔対策〕

Exception 情報を確認し、次のとおり対処してください。

- IllegalStateException が出力されている場合：
  HMP-PCTO の Exception 情報です。Mediator の Pod が存在しないことを検知したことで出力します。Mediator の Pod が存在しない要因を、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。
- 上記以外の Exception 情報が出力されている場合：
  gRPC 通信の Exception 情報です。Exception 情報を確認して対策を行うか、または Mediator などに問題が発生していないかの確認結果や、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。

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)なし。

〔対策〕

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。

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)なし。

〔対策〕

- 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。
- 同一のアプリケーショングループ内で、hmppcto-app-group ラベルの値に指定しているアプリケーショングループ名が統一され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。「アプリケーショングループ」については、「7.2　Kubernetes アプリケーションのグループ定義」を参照してください。

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)なし。

〔対策〕

このメッセージの前に出力されている KFSG63101-E メッセージ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オペレーション名「j.c.H.s.h.p.s.GrpcNotifyTerminationResultRequestSender」のスパンタグ consensus-value で確認できます。consensus-value に COMMIT、ROLLBACK のどちらかを出力します。

"root-xid"="< KFSG63104-E メッセージの aa....aa 出力値>"のスパンタグを検索し、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「j.c.H.s.h.p.s.GrpcNotifyTerminationResultRequestSender」のオペレーションを検索し、スパンタグを確認してください。

オペレーション名「j.c.H.s.h.p.s.GrpcNotifyTerminationResultRequestSender」の span が出力されていない場合は、まだトランザクションが決着していません。Mediator、Participant に出力されているメッセージを参照してトランザクションが決着していない原因を調査してください。

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 Service：送信先のサービスとの通信障害、または送信先のサービスからの応答にトランザクション制御用のヘッダがない

Rollback request from Service：送信先のサービス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〕

次の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 Service	アプリケーションが送信先のサービス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。また、Spring Boot や通信経路上のルータなどでエラーが発生していないか調査してください。
Rollback request from Service	送信先のサービスで出力している KFSG63107-W メッセージを参照して、アプリケーション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:」以降がアプリケーションが送出した例外の情報です。この情報やアプリケーションが出力している情報から、アプリケーション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

bb....bb の出力値	原因の調査方法
	• KFSG60022-E • 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)

Entity-Module の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-Module：Entity-Module との通信障害、または送信先のサービスからの応答にトランザクション制御用のヘッダがない

Participant failure：Participant の障害

The HTTP status of the rollback target：Rollback 対象の HTTP ステータス

The gRPC status of the rollback target：Rollback 対象の gRPC ステータス

cc....cc：HTTP ステータスコードまたは gRPC ステータスコード（ロールバックにマークした理由が Rollback 対象の HTTP ステータスまたは Rollback 対象の gRPC ステータス以外の場合は「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〕

次の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-Module	アプリケーションが Entity-Module 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。また、Spring Boot や通信経路上のルータなどでエラーが発生していないか調査してください。
Participant failure	Participant、または Entity-Module 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。 • KFSG60010-W • KFSG60011-E • KFSG60012-E • KFSG60013-W • KFSG60016-E • KFSG63002-E • KFSG63005-E • KFSG63200-E • KFSG63201-E • KFSG63208-W
The HTTP status of the rollback target	アプリケーション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。
The gRPC status of the rollback target	アプリケーションがロールバック対象の gRPC 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)なし。

〔対策〕

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。

KFSG63109-E (Y)M

An error occurred in the initialization of the entity module. Initialization of the entity module failed.

Entity-Module の初期化時にエラーが発生しました。Entity-Module の初期化に失敗しました。

(S)Entity-Module を異常終了します。

(O)なし。

〔対策〕

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。

KFSG63110-E (Y)J+M

The transaction branch could not be started.

トランザクションブランチを開始できませんでした。

(S)要求を無視します。

(O)なし。

〔対策〕

トランザクションルートまたはトランザクションブランチの範囲内で@PctoTransactionBranch アノテーションを使用するように、プログラムを修正してください。

KFSG63111-E (Y)J+M

Mediator has returned an unknown transaction settlement result, so the transaction settlement result is unknown. If the transaction has already been settled on the participant side due to communication delay between the participant and the mediator, the settlement result of the transaction will be unknown. (rootxid = aa....aa)

Mediator からトランザクションの決着結果不明が返却されたため、トランザクションの決着結果は不明です。Participant - Mediator 間の通信遅延などによって、Participant 側でトランザクションがすでに決着している場合、トランザクションの決着結果が不明になります。

aa....aa：ルート XID

(S)処理を続行します。

(O)なし。

〔対策〕

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オペレーション名「j.c.H.s.h.p.s.GrpcNotifyTerminationResultRequestSender」のスパンタグ consensus-value で確認できます。consensus-value に COMMIT、ROLLBACK のどちらかを出力します。

"root-xid"="< aa....aa 出力値>"のスパンタグを検索し、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「j.c.H.s.h.p.s.GrpcNotifyTerminationResultRequestSender」のオペレーションを検索しスパンタグを確認してください。

KFSG63112-E (Y)M

gRPC NameResolver is configured without the scaling function.

スケーリング対応機能を使用しないで、gRPC 通信用の NameResolver を指定しています。

(S)ManagedChannel のビルド処理を中断します。

(O)システム管理者に連絡してください。

[対策]

スケーリング対応機能を使用する場合

jp.co.Hitachi.soft.hmppcto.use-scaling パラメタに true を指定し、スケーリング対応機能を使用する設定をしてください。また、ルーティング方法を設定するパラメタの jp.co.Hitachi.soft.hmppcto.set-routing パラメタを省略するか、inner を指定してください。

スケーリング対応機能を使用しない場合

ユーザアプリケーションの gRPC チャネル作成処理で HMP-PCTO が提供する TxPropagationGrpcNameResolverProvider を使用しないよう修正してください。

KFSG63113-E (Y)M

gRPC LoadBalancer is configured without the scaling function.

スケーリング対応機能を使用しないで、gRPC 通信用の LoadBalancer を指定しています。

(S)gRPC 通信処理を中断します。

(O)システム管理者に連絡してください。

[対策]

スケーリング対応機能を使用する場合

jp.co.Hitachi.soft.hmppcto.use-scaling パラメタに true を指定し、スケーリング対応機能を使用する設定をしてください。また、ルーティング方法を設定するパラメタの jp.co.Hitachi.soft.hmppcto.set-routing パラメタを省略するか、inner を指定してください。

スケーリング対応機能を使用しない場合

ユーザアプリケーションの gRPC チャネル作成処理で HMP-PCTO が提供する TxPropagationGrpcNameResolverProvider を使用しないよう修正してください。

KFSG63114-E (Y)M

The destination host name is invalid.

アプリケーションが設定した宛先の文字列が不正です。

(S)ManagedChannel のビルド処理を中断します。

(O)システム管理者に連絡してください。

〔対策〕

スケーリング対応機能を使用する場合、「2.2　Java アプリケーションの開発」に記載しているコーディング例を参照し、宛先の指定方法を見直してください。

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 を呼び出したアプリケーションにエラー応答を返します。

(O)なし。

〔対策〕

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。

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 を呼び出したアプリケーション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 を呼び出したアプリケーション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 を呼び出したアプリケーションを、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)なし。

〔対策〕

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。

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 を呼び出したアプリケーション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 を呼び出したアプリケーションにエラー応答を返します。

(O)なし。

〔対策〕

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。

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)なし。

〔対策〕

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。

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)なし。

〔対策〕

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。

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)なし。

〔対策〕

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。

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)なし。

〔対策〕

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。

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)なし。

〔対策〕

tp1.data-conversion.definition-file.directory 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。

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)なし。

〔対策〕

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。

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そろっていません。

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)なし。

〔対策〕

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。

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)なし。

〔対策〕

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。

KFSG63308-E (Y)C

Failed to create message. (message ID = aa....aa, details = bb....bb, exception = cc....cc)

メッセー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)なし。

〔対策〕

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。

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)なし。

〔対策〕

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。

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)なし。

〔対策〕

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

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)なし。

〔対策〕

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。

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)なし。

〔対策〕

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。

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ます。
- 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>.<Namespace 名>

bb....bb：Participant のポート番号　*を出力します。

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)なし。

〔対策〕
　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分散トレースを確認してください。分散トレースを確認する際は、オペレーション名「j.c.H.s.h.t.o.t.PeriodicallyRecoveryJob:execute」に記録されるスパンイベント「hostname」を参照し、トランザクションで利用している Participant を特定してください。
　また、次の Participant のパラメタの設定値を見直してください。
  - SQL-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

KFSG64603-E (Y)M

Failed to process the running dates file. (path = aa....aa, reason = bb....bb)

稼働日ファイルの処理に失敗しました。

aa....aa：稼働日ファイル用ディレクトリのパスまたは稼働日ファイルのパス

bb....bb：理由コード

Directory creation：稼働日ファイル用ディレクトリの作成に失敗しました。

File deletion：稼働日ファイルの削除に失敗しました。

File creation：稼働日ファイルの作成に失敗しました。

File read：稼働日ファイルの読み込みに失敗しました。

File write：稼働日ファイルの書き込みに失敗しました。

(S) Mediator の起動時は、Mediator を停止します。それ以外は処理を続行します。

(O)システム管理者に連絡してください。

〔対策〕

稼働日ファイルの格納ディレクトリのアクセス権や容量を見直してください。また、ディスクに障害が起きていないか確認してください。

KFSG64604-E (Y)M

Failed to initialize the running dates file output timer.

稼働日ファイル出力タイマの初期化に失敗しました。

(S)Pod の起動処理を終了します。

(O)システム管理者に連絡してください。

〔対策〕

直後に出力するスタックトレースを参照し、稼働日ファイル出力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。

10.2.2 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)なし。

〔対策〕

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。

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)なし。

〔対策〕

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。

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)なし。

〔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)なし。

[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)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、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)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、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)

メッセー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)システム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。

KFSG74105-I (Y)M

HMP-PCTO Version. (Version = aa....aa)

起動した HMP-PCTO のバージョンです。

aa....aa：HMP-PCTO バージョン

(S)処理を続行します。

(O)なし。

〔対策〕
　なし。

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)なし。

〔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)なし。

〔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)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、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)なし。

〔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)なし。

〔対策〕

なし。

KFSG74205-W (Y)M

The trace function is disabled. (reason = aa....aa)

分散トレース機能が無効です。

aa....aa：理由コード

PARAMETER：jp.co.Hitachi.soft.hmppcto.tracing.use-trace に false が設定されています。

LIBRARY：jp.co.Hitachi.soft.hmppcto.tracing.use-trace に true が設定されていますが、分散トレース機能を使用するためのライブラリがアプリケーションに組み込まれていないか、読み込むことができません。

(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〕

- 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。
- jp.co.Hitachi.soft.hmppcto.tracing.use-trace パラメタに true を設定してください。そのあと、Pod を再起動してください。
- 分散トレース機能を使用するためのライブラリがアプリケーションに組み込まれているか確認してください。組み込まれていない場合、「2.2.17　分散トレース機能の設定」を参照して組み込んでください。

解決しない場合は保守員に連絡してください。

KFSG74206-I (Y)M

The trace will be sent to Tracing service. (service = aa....aa, instance-id = bb....bb, URL = cc....cc)

分散トレースを分散トレース収集アプリケーションサービスに送信します。

aa....aa：送信先の分散トレース収集アプリケーションまたはサービス

分散トレース収集サービスとして Cloud Trace を使用する場合：GCP

分散トレース収集サービスとして OpenTelemetry Collector を使用する場合：OTLP（http/protobuf）

bb....bb：jp.co.Hitachi.soft.hmppcto.tracing.instance-id パラメタの設定値。未設定の場合は自動生成された ID

cc....cc：jp.co.Hitachi.soft.hmppcto.tracing.url パラメタの設定値

(S)処理を続行します。

(O)なし。

[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)なし。

[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)なし。

[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)なし。

[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)なし。

〔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)なし。

〔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)なし。

〔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)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、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)なし。

〔対策〕
なし。

10.2.3 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)なし。

〔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)なし。

〔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)なし。

[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)なし。

[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)なし。

[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)なし。

〔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)なし。

〔対策〕

なし。

KFSG80007-I (Y)M

The commit processing retry has completed. (branchxid = aa....aa)

Commit 処理のリトライが完了しました。

aa....aa：ブランチ XID

(S)Commit 処理のリトライを終了します。

(O)なし。

〔対策〕

なし。

KFSG80008-I (Y)M

The rollback processing retry has completed. (branchxid = aa....aa)

Rollback 処理のリトライが完了しました。

aa....aa：ブランチ XID

(S)Rollback 処理のリトライを終了します。

(O)なし。

〔対策〕

なし。

KFSG81001-I (Y)M

The SQL-Participant initialization processing is waiting for the obtaining connection to database. (retry count = aa....aa, retry interval in seconds = bb....bb)

SQL-Participant の初期化処理でデータベースとのコネクション取得を待ち合わせます。

aa....aa：データベースとのコネクション取得をリトライする回数
（jp.co.Hitachi.soft.hmppcto.sql.pool.connection-retry-limit 設定値）

bb....bb：データベースとのコネクション取得をリトライする間隔（単位：秒）
（jp.co.Hitachi.soft.hmppcto.sql.pool.connection-retry-interval-in-seconds 設定値）

(S)データベースとのコネクション取得を 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。

KFSG81002-I (Y)M

The obtaining connection to database finished.

データベースとのコネクション取得が完了しました。

(S)処理を続行します。

(O)なし。

［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)なし。

［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)なし。

〔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)なし。

〔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)なし。

〔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)なし。

[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)なし。

[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が使用するスレッドプールサイズ

bb....bb：Participant が Mediator からの受信に使用するスレッドが使用するスレッドプールサイズ

cc....cc：Participant が Mediator からの受信に使用する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, context path = ff....ff, scheme = gg....gg, insecure option = hh....hh)

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

ff....ff：コンテキストパス。指定されていない場合は空文字を表示します。

gg....gg：スキーマ設定。--scheme オプションが指定されていない場合は http を表示します。

hh....hh：証明書検証オプション。--insecure オプションが指定されている場合は true を、--insecure オプションが指定されていない場合は false を表示します。

(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, context path = ff....ff, scheme = gg....gg, insecure option = hh....hh)

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

ff....ff：コンテキストパス。指定されていない場合は空文字を表示します。

gg....gg：スキーマ設定。--scheme オプションが指定されていない場合は http を表示します。

hh....hh：証明書検証オプション。--insecure オプションが指定されている場合は true を、--insecure オプションが指定されていない場合は false を表示します。

(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。

KFSG82200-I (Y)M

The scaling function is enabled.

スケーリング対応機能が有効になっています。

(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 module finished.

Entity-Module 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。

11 システム運用に必要となる機能

11.1　概要

11.1.1　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 11-1　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 11-1　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 11-2　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 11-3　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 11-1　分散トレースのイメージ（SQL-Participant を使用する場合）

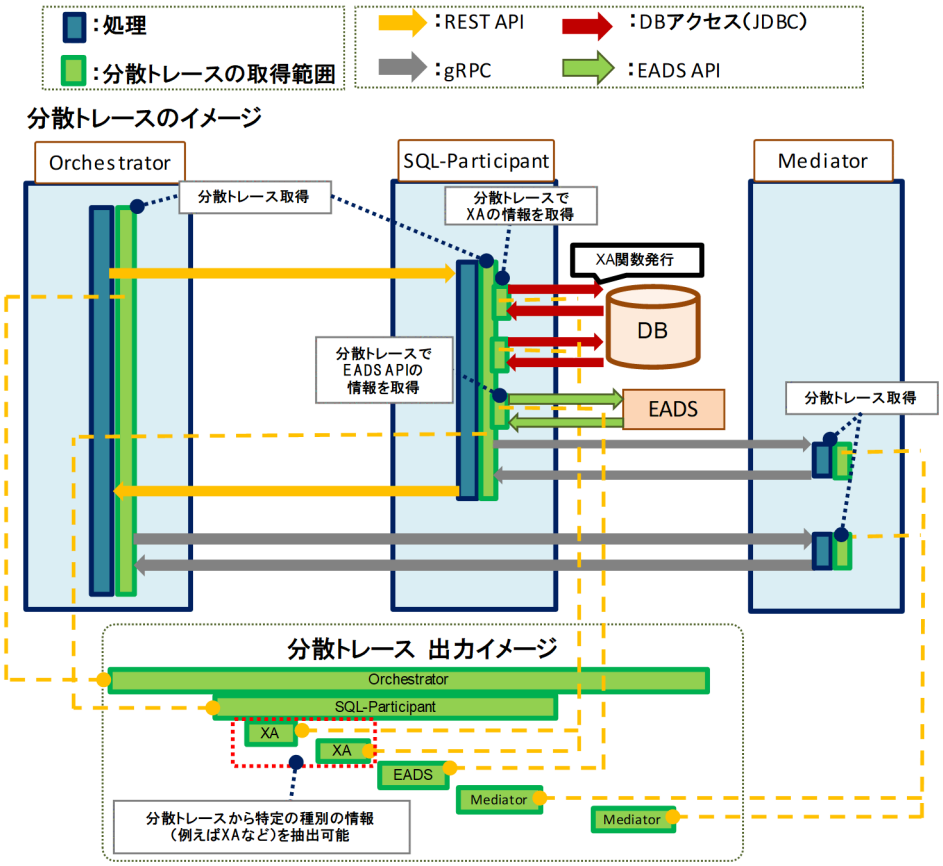


図 11-2　分散トレースのイメージ（TCC-Participant を使用する場合）

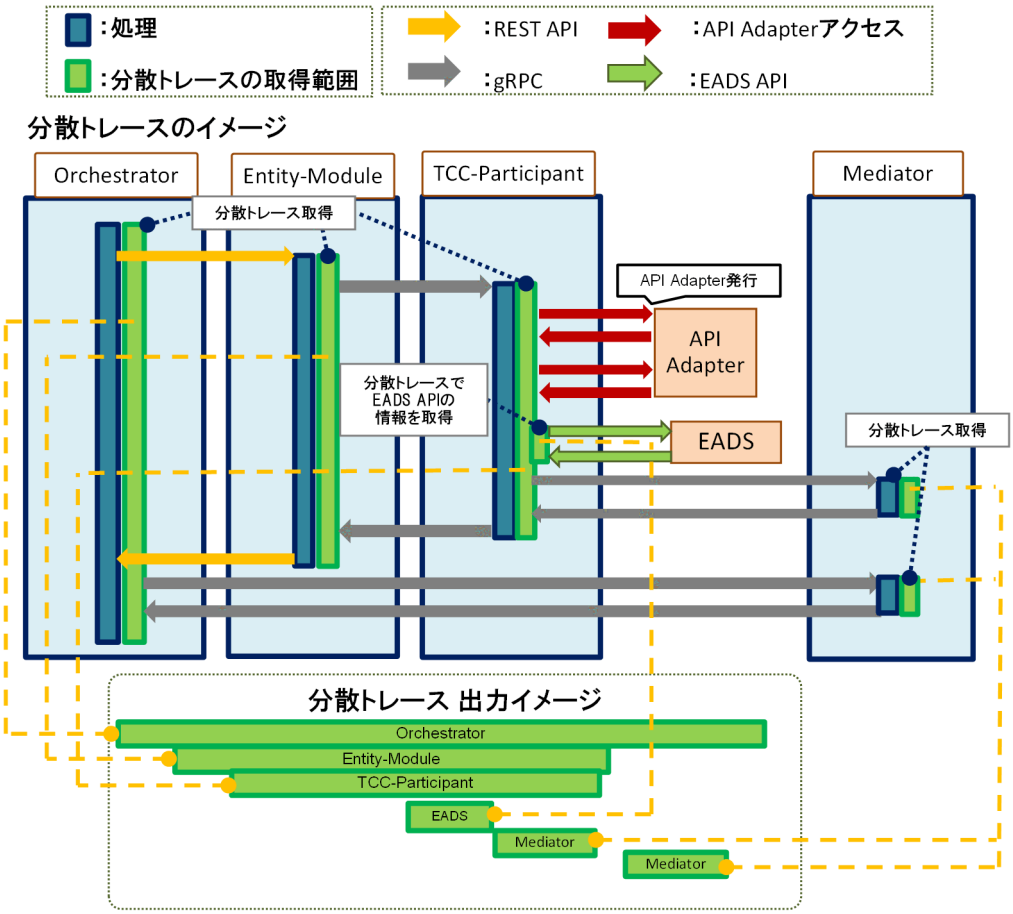
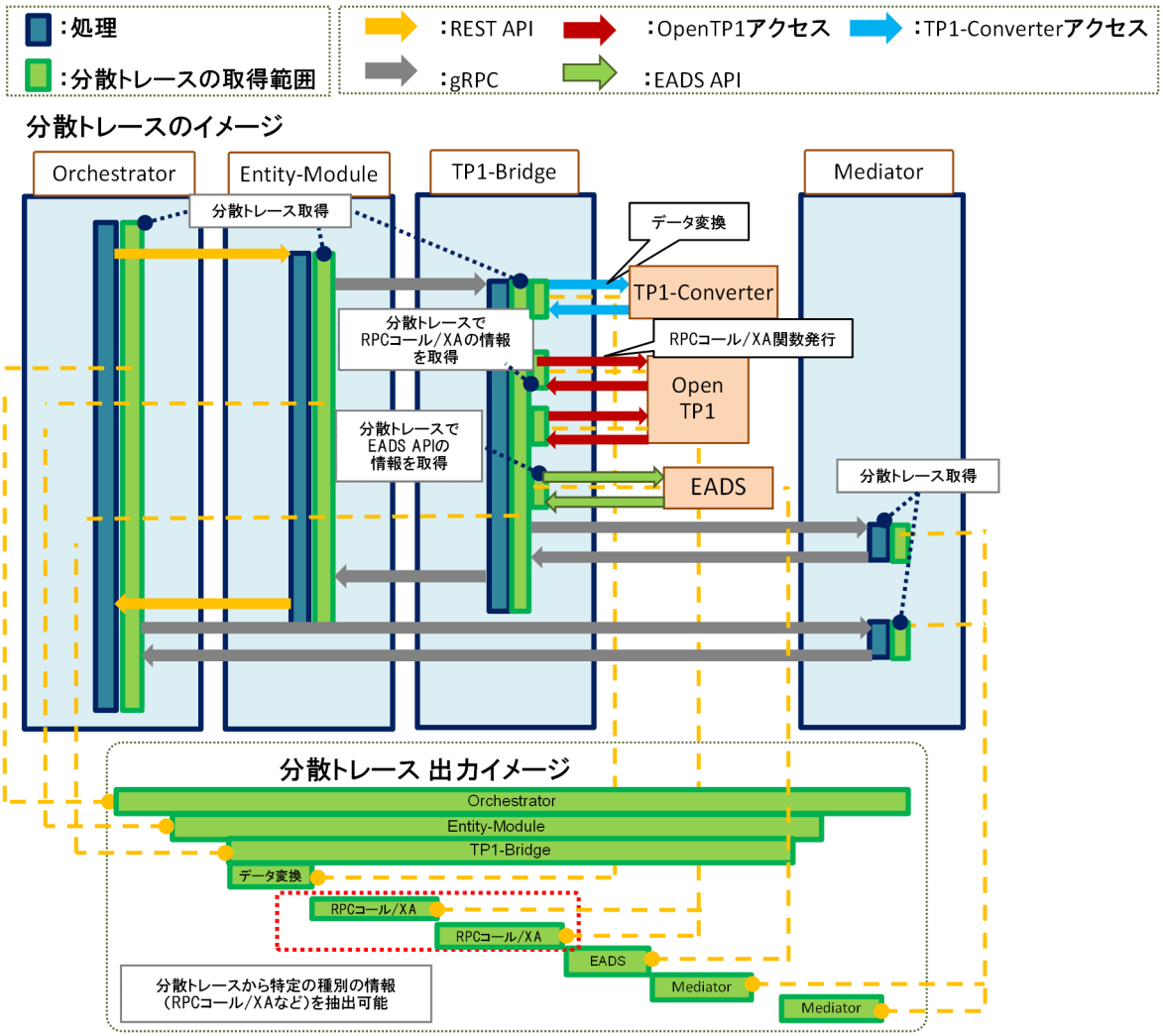


図 11-3　分散トレースのイメージ（TP1-Bridge を使用する場合）

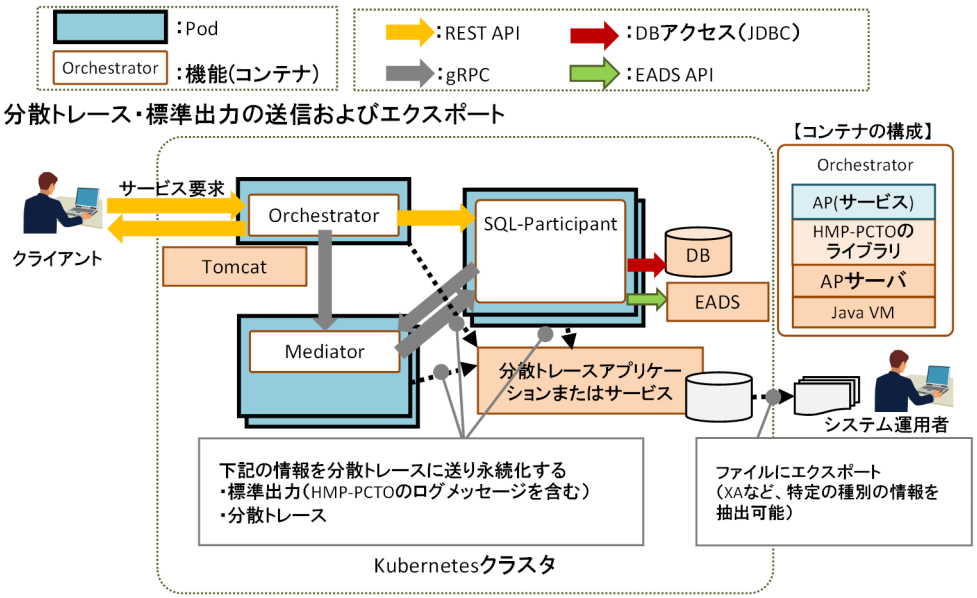


要件「情報を永続化できること」を満たすため、HMP-PCTO はログ、分散トレース、およびメトリクスをそれぞれの管理基盤に送り永続化します。メトリクス情報の永続化の方法については、「11.4.2　サポートするアプリケーションおよびサービス」を参照してください。

また、管理基盤に Elastic Stack を使用する場合は、障害が発生するケースに備えて、Elasticsearch をクラスタ構成にできます。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。

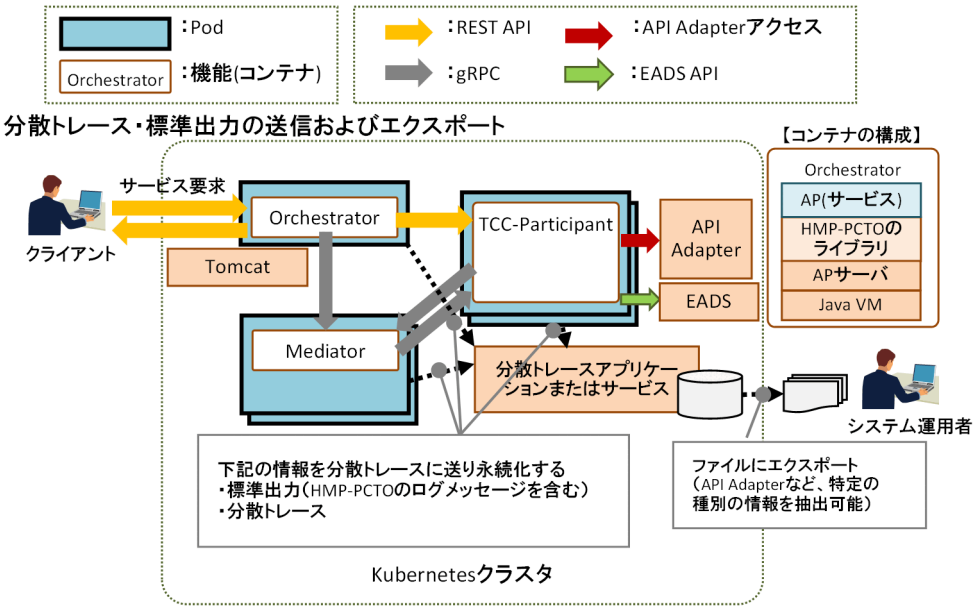
分散トレースの情報は、ファイルにエクスポートする際に特定の種別の情報を抽出できます。例えば、SQL-Participant 使用時の場合、XA 関数発行時のトレースに絞って出力することで、1 トランザクション当たりの XA 関数の処理時間の合計を抽出できます。SQL-Participant 使用時の場合の、HMP-PCTO 分散トレース・標準出力の送信およびエクスポートのイメージを次の図に示します。

図 11-4　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）



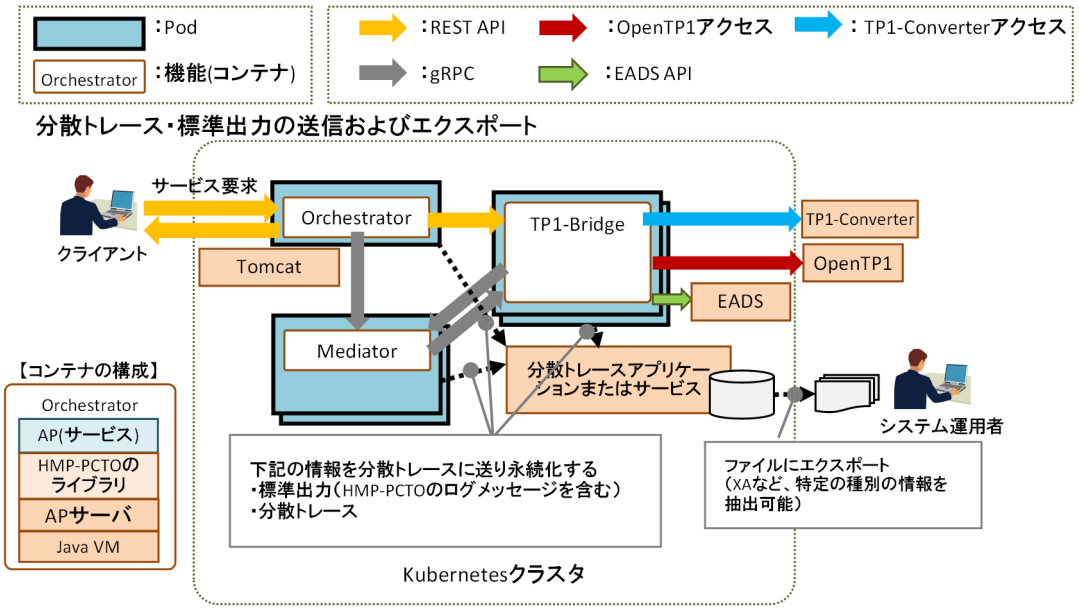
TCC-Participant 使用時の場合、API Adapter 発行時のトレースに絞って出力することで、1 トランザクション当たりの処理時間の合計を抽出できます。TCC-Participant を使用する場合の、HMP-PCTO 分散トレース・標準出力の送信およびエクスポートイメージを次の図に示します。

図 11-5　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 11-6　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 11-1　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 11-2　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	Spring Boot の組み込み HTTP サーバのアクセスログ	AP の REST API 通信についてアクセスログに記録されたヘッダ情報をトラブルシュートに利用します。
6	uCosminexus Application Runtime 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。
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。

項番	名称	説明
10	Oracle Database のトレース・ログ情報	Oracle Database のトレース・ログ情報を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 11-3　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	Spring Boot の組み込み HTTP サーバのアクセスログ	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）

項番	情報取得先 名称	取得する情報
9	uCosminexus Application Runtime for Spring Boot のログ	uCosminexus Application Runtime for Spring Boot のトレース情報（HTTP リクエストのトレース） uCosminexus Application Runtime 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、Spring Boot の組み込み HTTP サーバ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 • Java VM スタックトレース情報 • Java VM の GC ログ • Java VM ログ（Java VM 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。
13	Oracle Database のトレース・ログ情報	• SQL トレース • ログ情報

(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、そのあと、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 11-4　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）	Spring Boot の組み込み HTTP サーバアクセスログ		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。

項番	情報			説明
12		Oracle Database のトレース・ログ情報	SQL 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。
13			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。
14	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。
15	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。
16			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。
17			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。
18			Java VM 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。
19			Java VM 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。
20			Java VM ログ（Java VM 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。
21		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。
22		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。
23			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。
24			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。
25			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。

項番	情報			説明
26			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 11-4　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 11-7　HMP-PCTO 各情報の紐づけのイメージ（SQL-Participant を使用する場合）

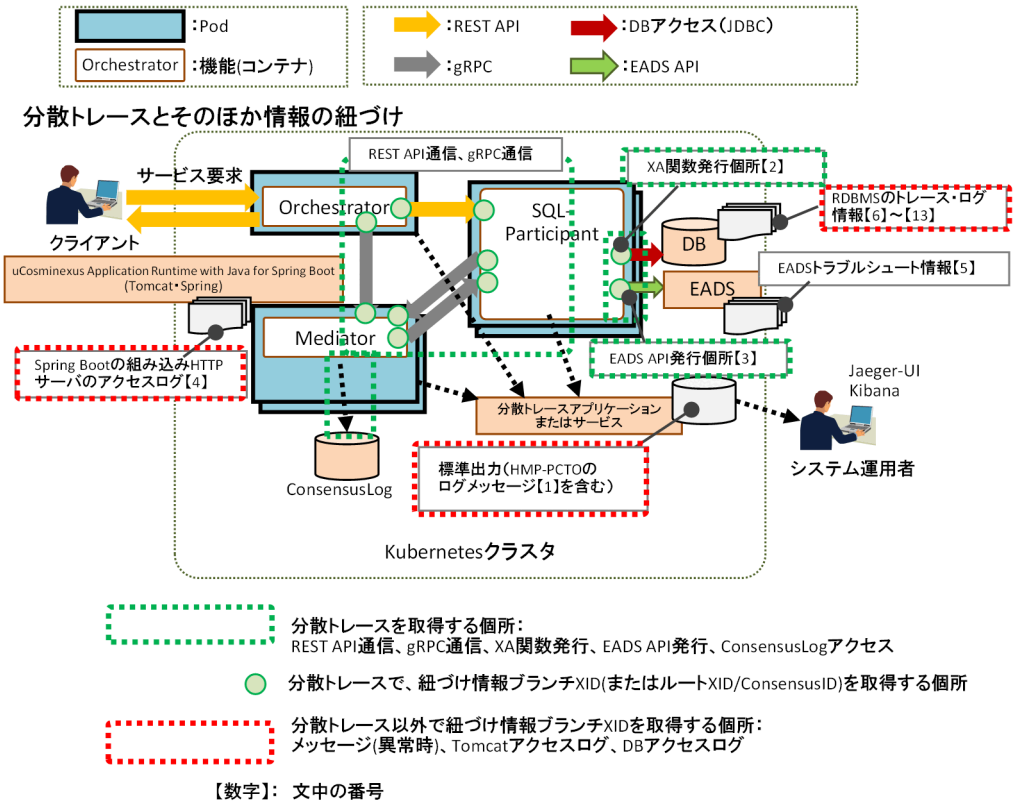


図 11–8　HMP-PCTO 各情報の紐づけのイメージ（TCC-Participant を使用する場合）

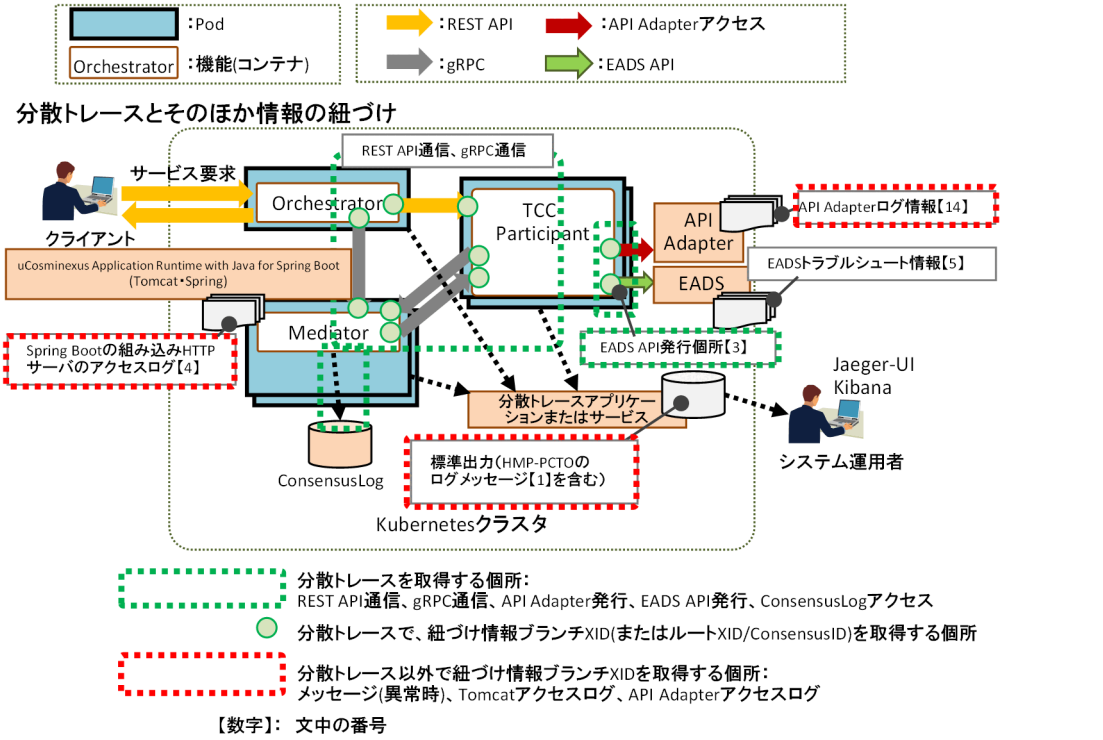
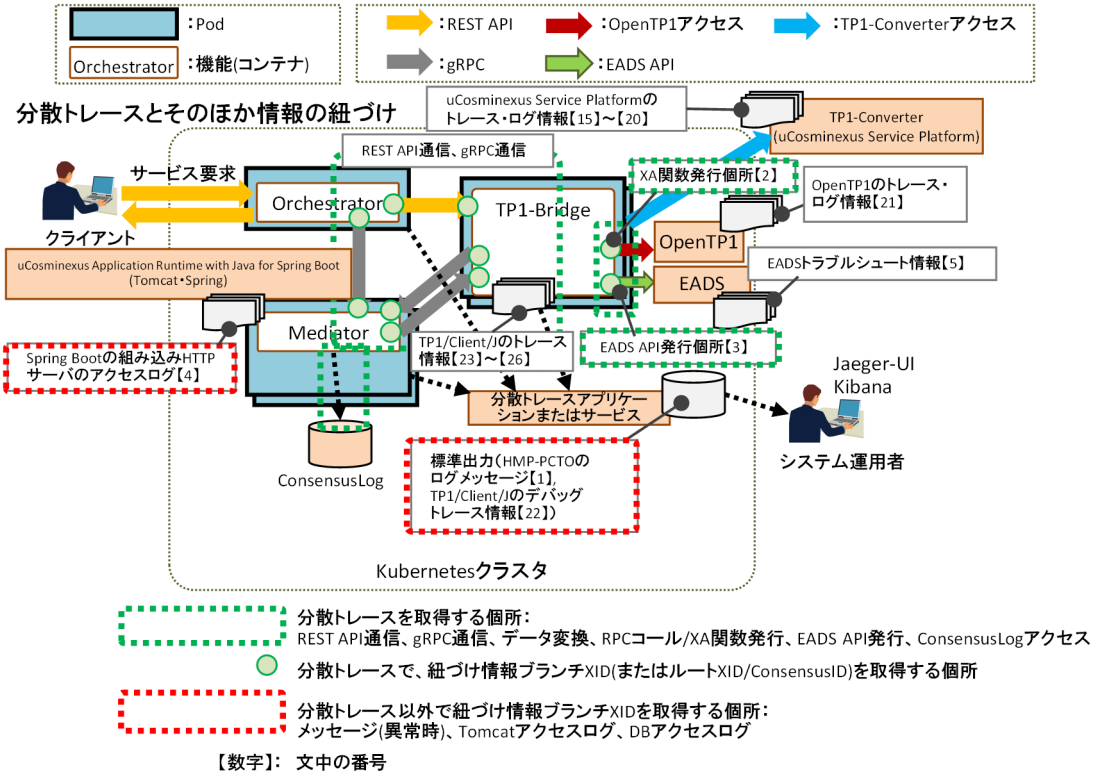


図 11–9　HMP-PCTO 各情報の紐づけのイメージ（TP1-Bridge を使用する場合）



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 11–7　HMP-PCTO 各情報の紐づけのイメージ（SQL-Participant を使用する場合）」、「図 11–8　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 11–9　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、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 11-5 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 ＜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：Proposal Log、ConsensusLog 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 の組み込み HTTP サーバの 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 11-6 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。

11.2　メッセージログ

11.2.1　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。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してください。

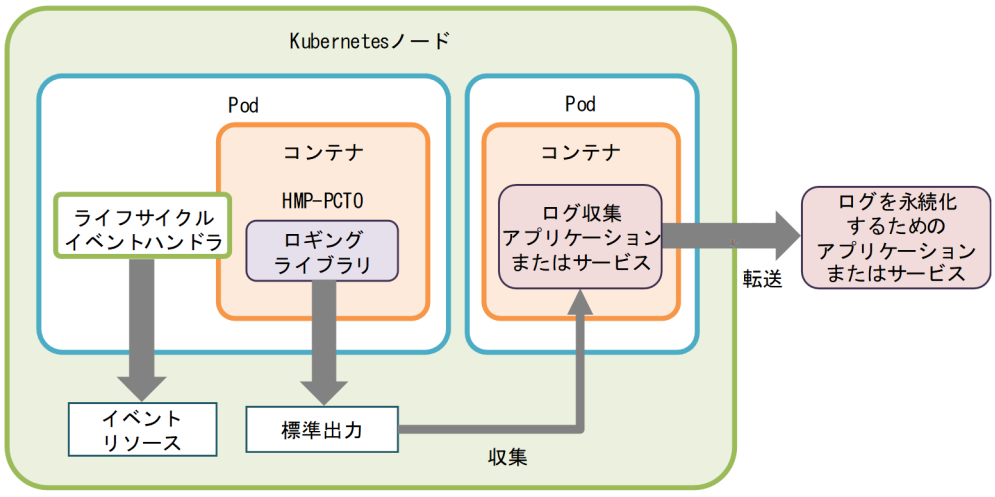
標準出力の内容は、ノード上でログ収集アプリケーションまたはサービスを使用して収集します。収集したログは、ログを永続化するためのアプリケーションまたはサービスに転送され、永続化および可視化されます。障害発生時は、必要に応じて保存されたログをエクスポートし、障害調査者に送付してください。詳細については、「11.6　HMP-PCTO の障害調査」を参照してください。

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。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、適切に対応してください。

なお、ログ収集アプリケーションおよびサービスは、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、「11.6　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 11-10　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。

- ユーザアプリケーションのログ出力形式

  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。

11.2.2　サポートするアプリケーションおよびサービス

HMP-PCTO では、次に示すアプリケーションおよびサービスをサポートしています。

表 11-7　サポートするアプリケーションおよびサービス

項番	環境	ログ収集アプリケーションおよびサービス	ログを永続化するためのアプリケーションおよびサービス	ログ閲覧アプリケーションおよびサービス
1	Elastic Stack 使用環境	Filebeat	• Logstash • Elasticsearch	• Elasticsearch • Kibana
2	Microsoft Azure 環境	Azure Monitor	Azure Monitor Log Analytics	Azure Monitor
3	Google Cloud Platform 環境	Cloud Logging	Cloud Logging	Cloud Logging
4	Amazon Web Services 環境	Fluent Bit	Amazon CloudWatch	Amazon CloudWatch
5	New Relic 使用時	New Relic	New Relic	New Relic UI（Web UI）
6	Red Hat OpenShift Container Platform 環境	Red Hat OpenShift Logging Operator	Loki Operator	Red Hat OpenShift Container Platform（Web UI）

各アプリケーションおよびサービスの設定・運用方法については、次の説明を参照してください。

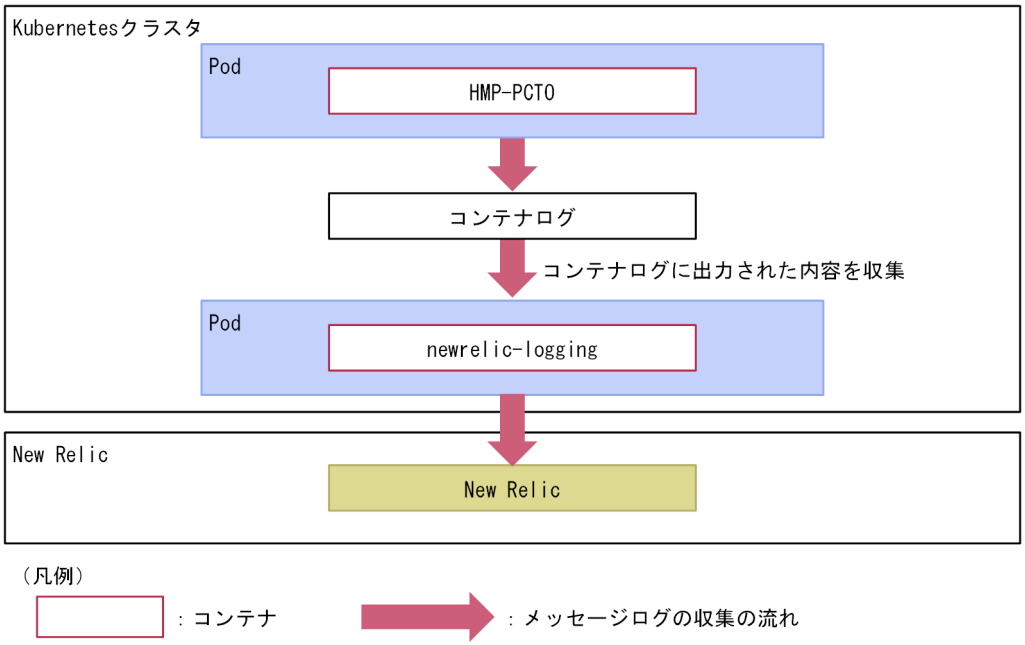
- 「3.6　共有ファイルシステムでの作業」
- 「3.7　Kubernetes マニフェスト作成」
- 「3.8　Helm チャートのカスタマイズ」
- 「3.9　HMP-PCTO のコントロールプレーンおよび RAS の Kubernetes アプリケーションの運用」
- 「3.11　ログの取得および監視の設定」

(1)　New Relic を使用した場合の構成

New Relic へメッセージログを出力する際の構成図を示します。

- HMP-PCTO のメッセージログを収集する場合、newrelic-logging コンポーネントをデプロイまたは有効化する必要があります。

図 11-11　New Relic 使用時の構成図



11.2.3　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、「10.　メッセージ」を参照してください。

11.2.4　パラメタ

メッセージログに関するパラメタはありません。

ログ収集アプリケーションまたはサービスを使用したサイドカーコンテナに関するパラメタについては、「3.11.1　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。

11.3　分散トレース

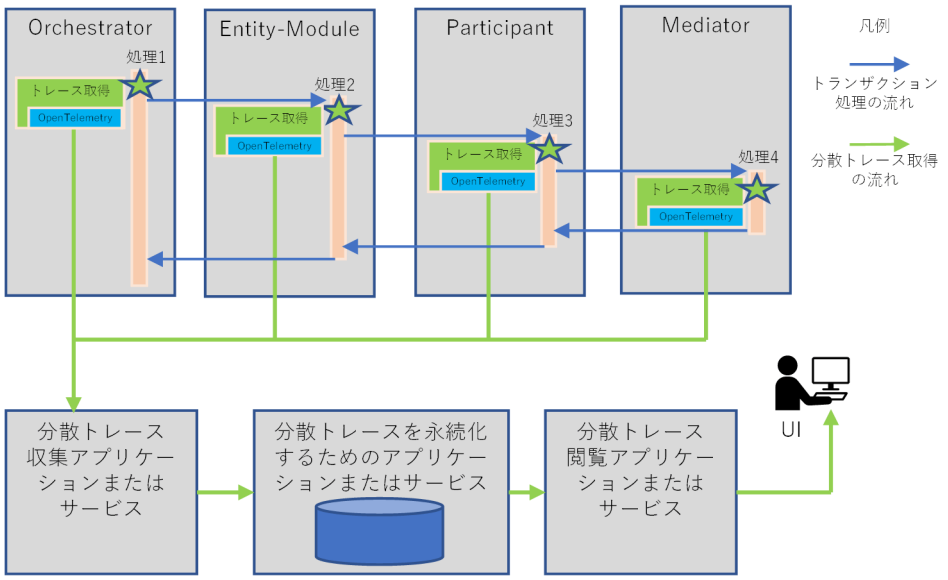
11.3.1　概要

HMP-PCTO は 1 つのトランザクション内で、Orchestrator、Mediator、Entity-Module、および 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、分散トレース収集アプリケーションまたはサービスに送信します。収集された分散トレースについては、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ともに分散トレースを使用した調査ができません。そのため、分散トレースの構成は次のようになります。

- 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
- 分散トレース収集アプリケーションまたはサービスにトレース情報を送信
- 分散トレースを永続化するためのアプリケーションまたはサービスにデータを送信しデータを永続化
- 分散トレース閲覧アプリケーションまたはサービス上でトレース情報を参照

図 11-12　分散トレースの概要



11.3.2　サポートするアプリケーションおよびサービス

HMP-PCTO では、次に示すアプリケーションおよびサービスをサポートしています。

表 11-8　サポートするアプリケーションおよびサービス

項番	環境	分散トレース収集アプリケーションおよびサービス	分散トレースを永続化するためのアプリケーションおよびサービス	分散トレース閲覧アプリケーションおよびサービス
1	Elastic Stack 使用環境	• Jaeger • OpenTelemetry Collector※	Elasticsearch	Jaeger
2	Microsoft Azure 環境	• Jaeger • OpenTelemetry Collector※	Elasticsearch	Jaeger
3	Google Cloud Platform 環境	• Cloud Trace • OpenTelemetry Collector※	Cloud Trace	Cloud Trace
4	Amazon Web Services 環境	• Jaeger • OpenTelemetry Collector※	Amazon OpenSearch Service	Jaeger
5	New Relic 使用時	• OpenTelemetry Collector※ • New Relic	New Relic	New Relic UI（Web UI）
6	Red Hat OpenShift Container Platform 環境	• Tempo Operator • OpenTelemetry Collector※	TempoStack	Red Hat OpenShift Container Platform（Web UI）

注※　確率的サンプリングおよびテイルベースサンプリングなどを実施する場合は、分散トレース収集アプリケーションやサービスにデータを送信するコンポーネントで実施する必要があります。そのため、分散トレース収集アプリケーションやサービスにデータを送信するコンポーネントとして、OpenTelemetry Collector を経由して分散トレースを収集するように構成してください。

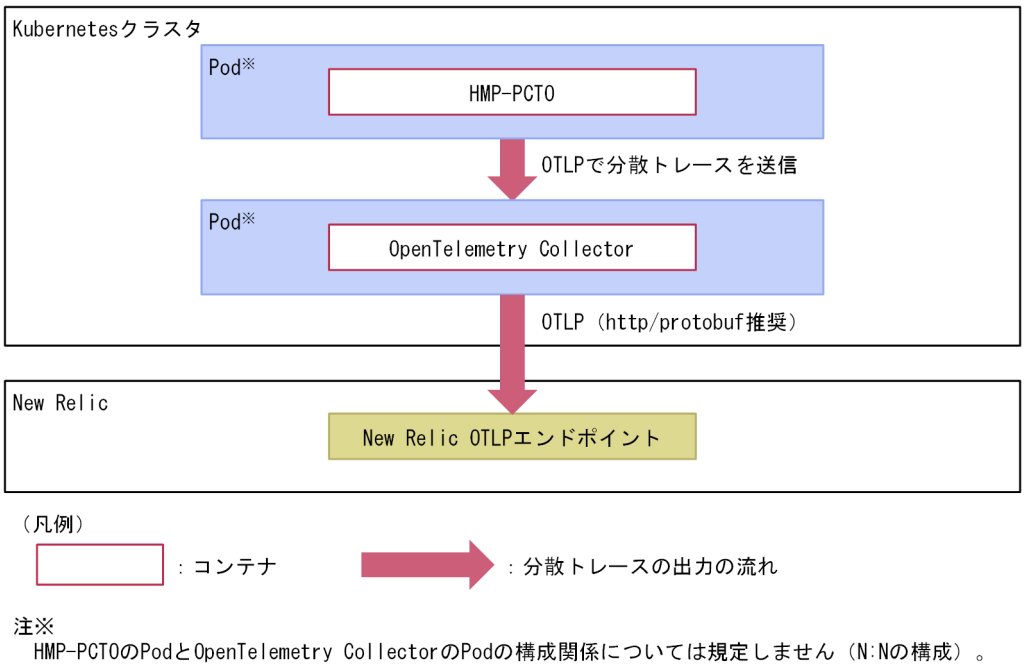
(1)　New Relic を使用した場合の構成

New Relic へ分散トレースを出力する際の構成図を示します。

- OpenTelemetry Collector への送信形式には、http/protobuf、http/json、および gRPC の 3 種類があります。

  HMP-PCTO の Pod から OpenTelemetry Collector への送信形式は、デフォルトの http/protobuf だけをサポートします。
- New Relic OTLP エンドポイントは http/protobuf だけに対応するため、OpenTelemetry Collector から New Relic OTLP エンドポイントへの OTLP は http/protobuf を選択する必要があります。

図 11-13　New Relic 使用時の構成図



注※
HMP-PCTOのPodとOpenTelemetry CollectorのPodの構成関係については規定しません（N:Nの構成）。

11.3.3　分散トレース機能の使用

(1)　分散トレース機能の設定

分散トレース機能を使用する場合、使用する分散トレース収集サービスによって異なるライブラリを組み込む必要があります。

ライブラリの組み込み方法については、「2.2.17　分散トレース機能の設定」を参照してください。

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。

(2) 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収集アプリケーションまたはサービスに送信しますが、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、分散トレース収集アプリケーションまたはサービス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、分散トレース収集アプリケーションまたはサービス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、「11.6 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

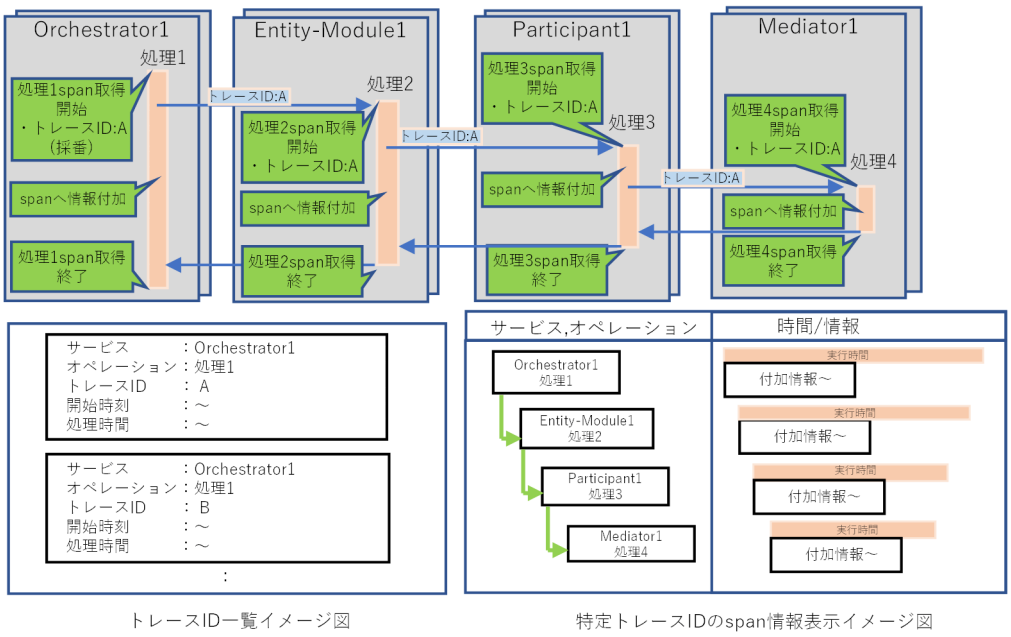
「(a) 分散トレース送信失敗時の検知、および分散トレース機能の抑止について」にあるとおり、ユーザが運用で対処して分散トレース送信失敗を回復する必要があるため、HMP-PCTO ではトレース情報の出力先を別の出力先に切り替えません。

そのため、ユーザが運用で対処するまでは、分散トレースを永続化するための収集アプリケーションまたはサービスにトレース情報が送信されず、トレース情報がロストするため注意してください。

11.3.4 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-Module、Participant、および 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。

図 11-14　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。

分散トレース閲覧アプリケーションまたはサービスにアクセスし、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。

11.3.5　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)　スパンタグ

次の key（情報名）、value（情報値）の形式で情報を取得します。分散トレース閲覧アプリケーションまたはサービスで、スパンタグの情報名、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、スパンタグに設定します。

例：受信した要求電文に含まれていたブランチ XID など

図 11-15　スパンタグ（Jaeger の場合）

図 11-16　スパンタグ（Google Cloud Platform の Cloud Trace の場合）

図 11-17　スパンタグ（New Relic の場合）

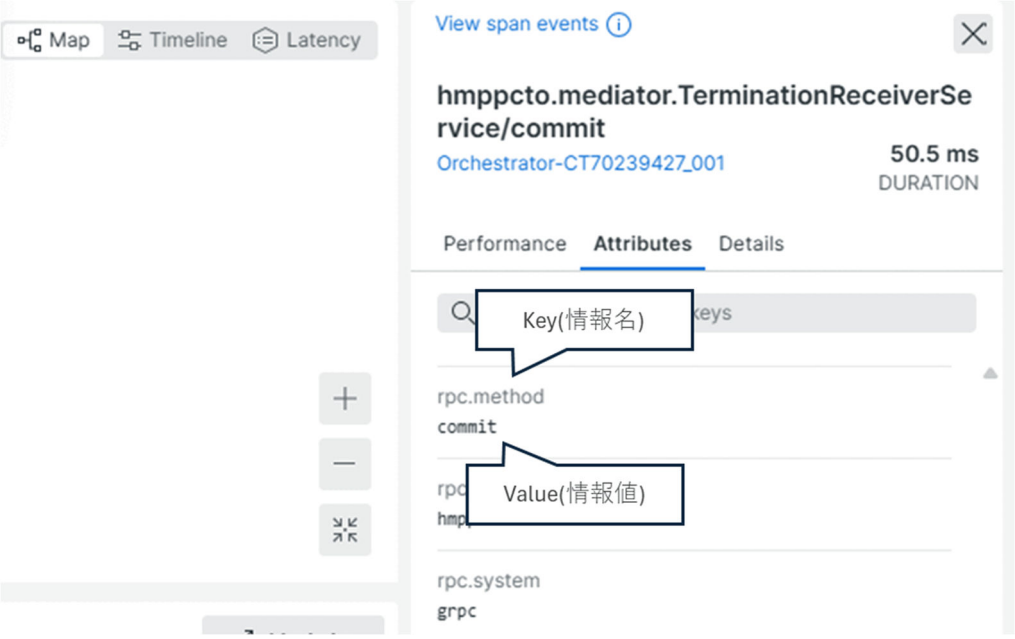
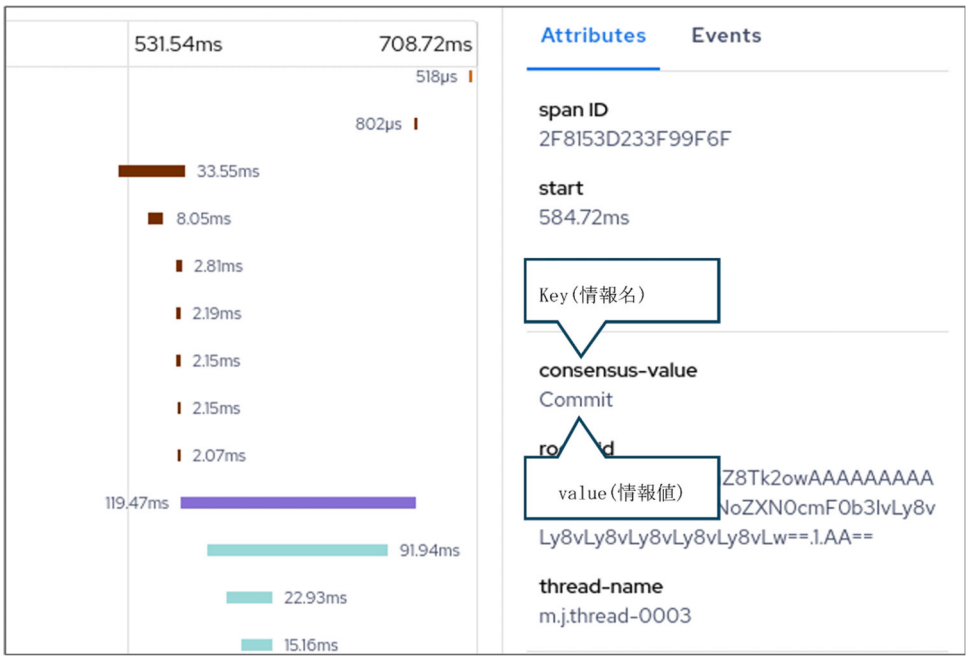


図 11-18　スパンタグ（Red Hat OpenShift Container Platform 環境の場合）



(2)　スパンイベント

Jaeger の場合、「図 11-19　スパンイベント形式 1（Jaeger の場合）」のスパンイベント形式 1、「図 11-20　スパンイベント形式 2（Jaeger の場合）」のスパンイベント形式 2 のどちらかの形式で情報を取得します。Google Cloud Platform の Cloud Trace の場合、「図 11-21　スパンイベント（Google Cloud Platform の Cloud Trace の場合）」の key（情報名）、value（情報値）、情報説明の形式で情報を取得します。New Relic の場合、「図 11-22　スパンイベント（New Relic の場合）」の message、情報説明の形式で情報を取得します。

スパンタグは、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スパンイベントに設定します（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）。

Jaeger の場合で、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ときは、スパンイベント形式 2 で情報を取得します。追加情報が不要であれば、スパンイベント形式 1 で取得します。

スパンタグとスパンイベント形式 2 の使い分け

対象イベントだけに関わる情報の場合はスパンイベント形式 2 へ、対象 Span 全体に関わる情報の場合はスパンタグに設定します。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スパンタグに、ファイル I/O の結果はスパンイベント形式 2 に設定します。

図 11-19　スパンイベント形式 1（Jaeger の場合）

図 11-20　スパンイベント形式 2（Jaeger の場合）

図 11-21　スパンイベント（Google Cloud Platform の Cloud Trace の場合）

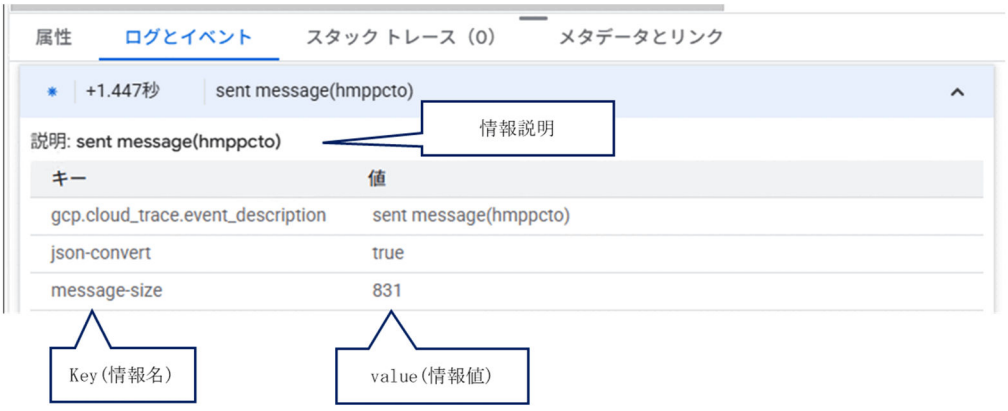
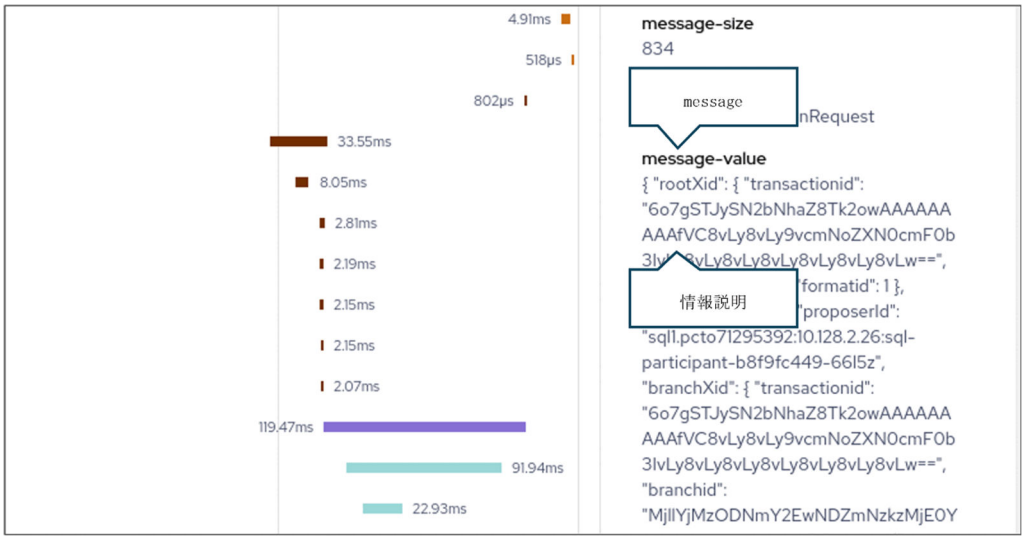


図 11-22　スパンイベント（New Relic の場合）

図 11-23　スパンイベント（Red Hat OpenShift Container Platform 環境の場合）



(3)　スパンコンテキスト

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。分散トレース閲覧アプリケーションまたはサービスでスパンコンテキストの名称、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 11-24　スパンコンテキスト（Jaeger の場合）

図 11-25　スパンコンテキスト（Google Cloud Platform の Cloud Trace の場合）

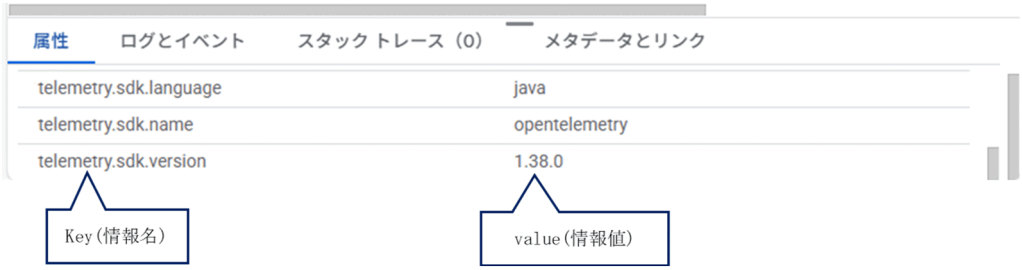


図 11-26　スパンコンテキスト（New Relic の場合）

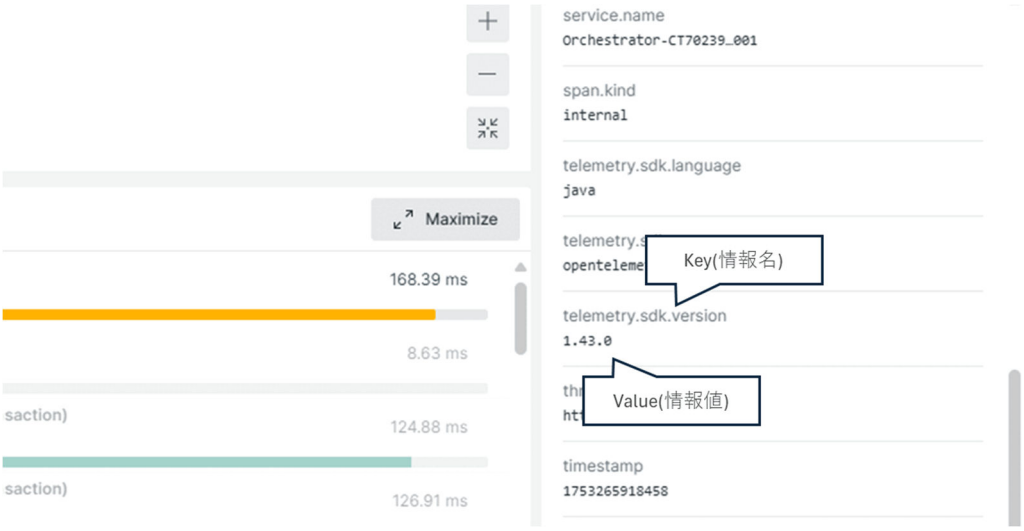
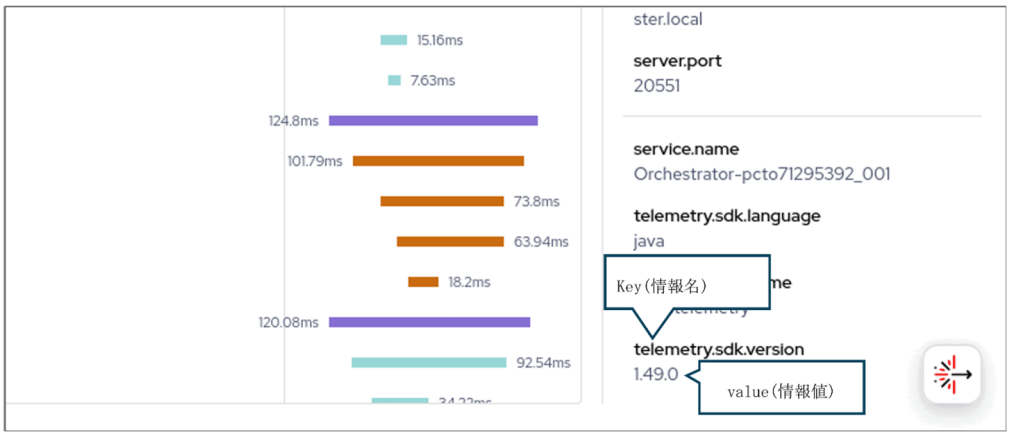


図 11-27　スパンコンテキスト（Red Hat OpenShift Container Platform 環境の場合）



11.3.6　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.use-trace
- jp.co.Hitachi.soft.hmppcto.tracing.queue-size
- jp.co.Hitachi.soft.hmppcto.tracing.delay-time-in-millis
- jp.co.Hitachi.soft.hmppcto.tracing.batch-size
- jp.co.Hitachi.soft.hmppcto.tracing.debug
- jp.co.Hitachi.soft.hmppcto.tracing.grpc.debug
- jp.co.Hitachi.soft.hmppcto.tracing.jdbc.debug

パラメタの詳細については、「9.　パラメタ」を参照してください。

11.4　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。

11.4.1　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 11–9　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：Proposal Log、ConsensusLog 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 • Jaeger の例：受信した Span 数、Exporter 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 11-9　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 11-10　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-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	Orchestrator Entity-Module SQL-Participant TCC-Participant	Tomcat で管理されるスレッド数（Spring Boot の組み込み HTTP サーバに Tomcat を使用する場合）
5	TP1-Bridge	Tomcat で管理されるスレッド数
6	Jaeger	• Exporter の内部キュー内の Span 数 • Exporter 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 などを用いたメトリクス収集アプリケーションまたはサービスの使用を推奨します。メトリクス収集アプリケーションまたはサービスでメトリクスの情報を収集することで、HMP-PCTO の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 11-11　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。HMP-PCTO が対応しているメトリクス収集アプリケーションまたはサービスは Prometheus だけです。このため、HMP-PCTO は Prometheus 向けのデータ形式でメトリクス情報を出力します。

Prometheus では、HMP-PCTO が Micrometer で取得したメトリクスデータを収集します。このため、HMP-PCTO では、Prometheus にメトリクスデータを送信できるように、Prometheus に対応した Exporter を Micrometer で実装しています。

11.4.2 サポートするアプリケーションおよびサービス

HMP-PCTO では、次に示すアプリケーションおよびサービスをサポートしています。

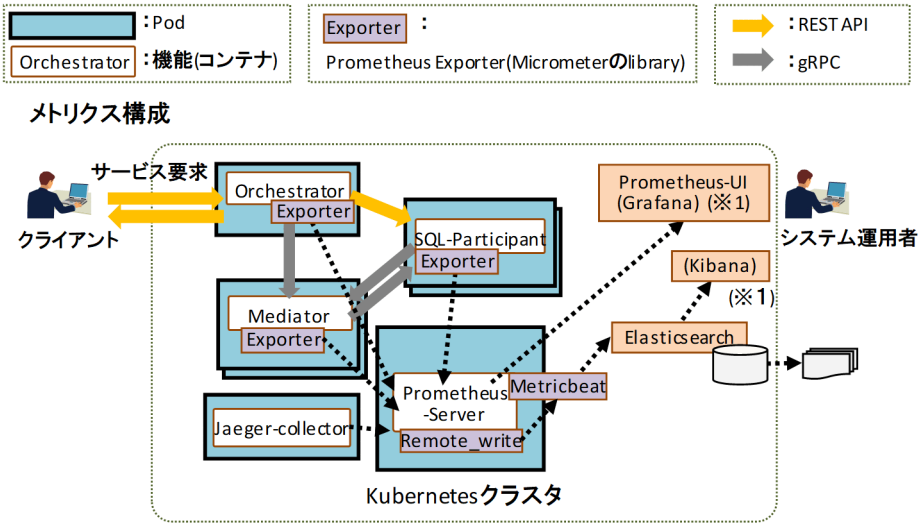
表 11-12 サポートするアプリケーションおよびサービス

項番	環境	メトリクス収集アプリケーションおよびサービス	メトリクスを永続化するためのアプリケーションおよびサービス	メトリクス閲覧アプリケーションおよびサービス
1	Elastic Stack 使用環境	• Prometheus • Metricbeat	Elasticsearch	Prometheus
2	Microsoft Azure 環境	Prometheus 用のマネージドサービス	Azure Monitor	Grafana 用のマネージドサービス
3	Google Cloud Platform 環境	Google Cloud Managed Service for Prometheus	Cloud Monitoring	Cloud Monitoring
4	Amazon Web Services 環境	• AWS Distro for OpenTelemetry Collector • Amazon Managed Service for Prometheus	Amazon CloudWatch	Amazon CloudWatch
5	New Relic 使用時	New Relic	New Relic	New Relic UI（Web UI）
6	Red Hat OpenShift Container Platform 環境	Red Hat OpenShift Container Platform モニタリングの Prometheus	Red Hat OpenShift Container Platform モニタリングの Prometheus	Red Hat OpenShift Container Platform（Web UI）

(1) Elastic Stack および Prometheus を使用した場合の構成

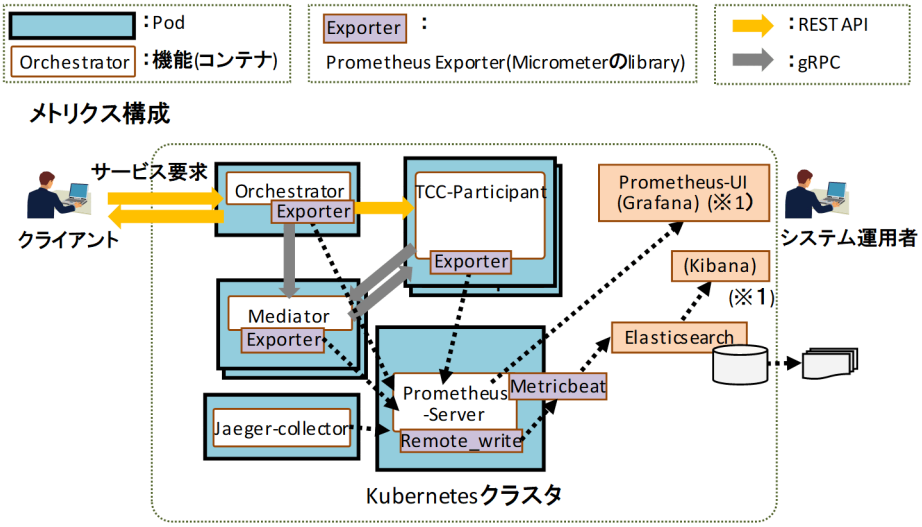
Elastic Stack および Prometheus を使用してメトリクス収集を行う際の構成図を次に示します。

図 11–28　HMP-PCTO メトリクスの構成（SQL-Participant の場合）



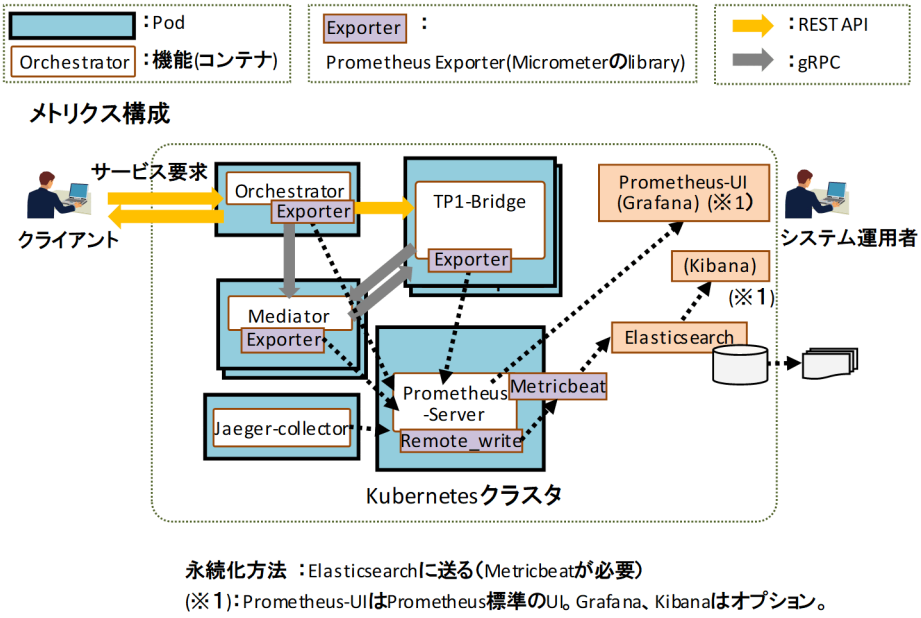
永続化方法 ：Elasticsearchに送る(Metricbeatが必要)
(※1)：Prometheus-UIはPrometheus標準のUI。Grafana、Kibanaはオプション。

図 11–29　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 11-30　HMP-PCTO メトリクスの構成（TP1-Bridge の場合）



「表 11-9　メトリクスの評価対象」の評価対象ごとの取得手段を次の表に示します。

表 11-13　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 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 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 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 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 に送信するメトリクスデータを生成します。 • Jaeger は、Jaeger 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	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 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、メトリクス閲覧アプリケーションまたはサービス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。

重要

Prometheus を利用してメトリクス収集を行う場合

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。

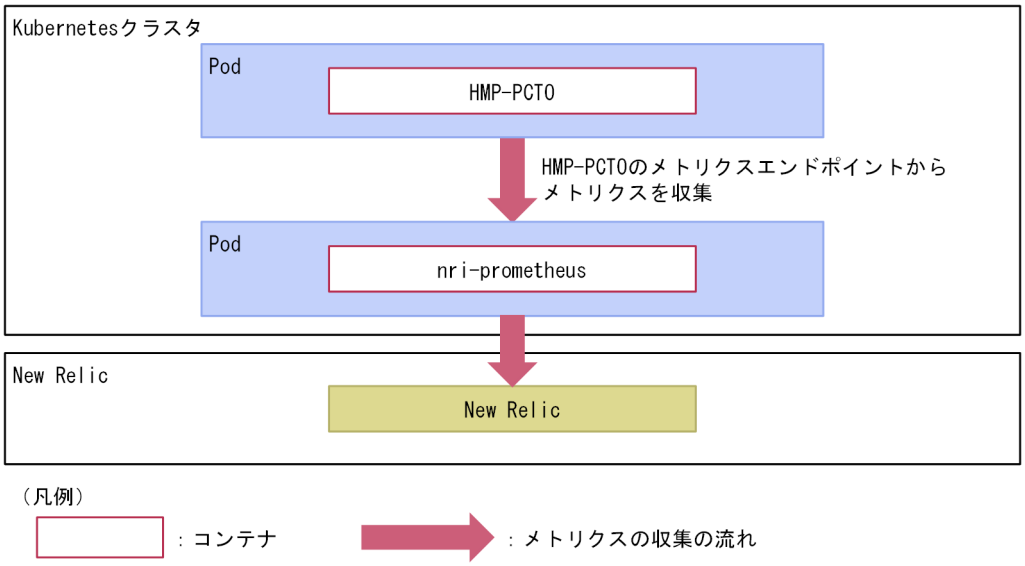
Prometheus-Server が収集したメトリクスデータを閲覧するには、Prometheus の Web UI を使用します。詳細については、「11.5.3　メトリクス」を参照してください。

(2)　New Relic を使用した場合の構成

New Relic へメトリクスを出力する際の構成図を示します。

- HMP-PCTO のメトリクスを収集する場合、nri-prometheus コンポーネントをデプロイまたは有効化する必要があります。

図 11-31　New Relic 使用時の構成図



11.4.3　パラメタ

次のパラメタを使用します。

- jp.co.Hitachi.soft.hmppcto.metrics.collection.tomcat

パラメタの詳細については、「9.　パラメタ」を参照してください。

11.5　データおよびファイル

11.5.1　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 11-14　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。

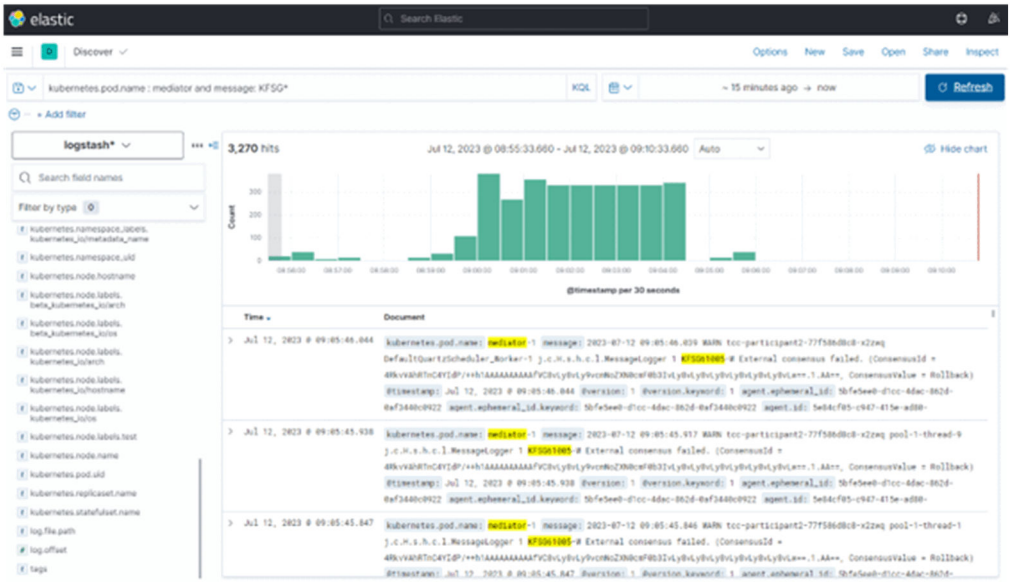
  Kibana の Discover 機能で HMP-PCTO のメッセージを抽出するためのフィールドを次の表に示します。このフィールドは Filebeat によって設定されています。

  表 11-15　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 Namespace 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、「3.7　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、「3.7　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、「3.7　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 11–32　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 11–16　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 HMP-PCTO で作成される非同期スレッドは、「表 11–17　スレッド名」に記載しているスレッド名で出力します。

出力情報	説明
classname	メッセージ出力元クラス名 短縮名で表示します。 ＜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
KFSG$N_1N_2N_3N_4N_5$-X	メッセージ ID
message	メッセージテキスト

表 11–17　スレッド名

コンポーネント	スレッドの役割	スレッド名
Orchestrator	Orchestrator から Mediator への送信に使用するスレッド	m.t-thread-*xxxx*
SQL-Participant	SQL-Participant がトランザクション処理で実行するスレッド	p-p.thread-*xxxx*
	Mediator に送信を行うスレッドを管理するスレッド	m-p-m.thread-*xxxx*
	Participant が Mediator との送受信に使用するスレッド	m-c.thread-*xxxx*
TCC-Participant	TCC-Participant がトランザクション処理で実行するスレッド	p-p.thread-*xxxx*
	Mediator に送信を行うスレッドを管理するスレッド	m-p-m.thread-*xxxx*
	Participant が Mediator との送受信に使用するスレッド	m-c.thread-*xxxx*
TP1-Bridge	TP1-Bridge がトランザクション処理で実行するスレッド	p-p.thread-*xxxx*
	Mediator に送信を行うスレッドを管理するスレッド	m-p-m.thread-*xxxx*
	Participant が Mediator との送受信に使用するスレッド	m-c.thread-*xxxx*
Mediator	Mediator から Participant への送信に使用するスレッド	p.s.thread-*xxxx*
	Mediator の分散合意処理に使用するスレッド	m.j.thread-*xxxx*
	Mediator が受信に使用するスレッド	g.s.thread-*xxxx*
	ConsensusLog ファイルへの書き込みに使用するスレッド	c.i.thread-*xxxx*

（凡例）

xxxx：スレッドのシーケンス番号

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) 文字コード

(a) HMP-PCTO が出力するメッセージログ

HMP-PCTO は、メッセージログの文字コードに UTF-8 を指定して、出力します。

(4) 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 Logback を使用し、その設定をリソースディレクトリの直下の 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="HMPPCTO_STDOUT" level="INFO" additivity="false">
        <appender-ref ref="HMPPCTO_STDOUT" />
</logger>
<logger name="HMPPCTO_STDERR" level="ERROR" additivity="false">
        <appender-ref ref="HMPPCTO_STDERR" />
</logger>
```

(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、リソースディレクトリの直下に次の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

11.5.2 分散トレース

(1) 出力先

使用するアプリケーションまたはサービスによって、分散トレースの出力先が異なります。

(a) Jaeger を使用する場合

Jaeger と連携した Elasticsearch に分散トレースが出力されます。

Amazon Web Services 環境で分散トレースを収集する際は Jaeger と連携した Amazon OpenSearch Service に分散トレースが出力されます。詳細については、Amazon Web Services の公式ドキュメントを参照してください。

(b) Google Cloud Platform の Cloud Trace を使用する場合

Google Cloud Platform 環境で分散トレースを収集する際は、Cloud Trace に分散トレースが出力されます。詳細については、Google Cloud Platform の公式ドキュメントを参照してください。

(c) New Relic を使用する場合

New Relic 使用時に分散トレースを収集する際は、OpenTelemetry Collector を経由して分散トレースが収集され、New Relic に出力されます。詳細については、New Relic の公式ドキュメントを参照してください。

(d) Red Hat OpenShift Container Platform 環境の場合

Red Hat OpenShift Container Platform 環境で分散トレースを収集する際は、OpenTelemetry Collector を経由して収集され、さらに TempoStack を経由し、Red Hat OpenShift Container Platform モニタリングに出力されます。詳細については、Red Hat OpenShift Container Platform の公式ドキュメントを参照してください。

(2) 出力形式

(a) Jaeger の Web UI による参照

収集された分散トレースは、Jaeger の Web UI の次の URL に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（「3.7.12 Jaeger の Kubernetes マニフェスト作成」の ConfigMap で指定している Jaeger の extensions.jaeger_query.http.endpoint に指定した値）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。

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 11–33　Search Page

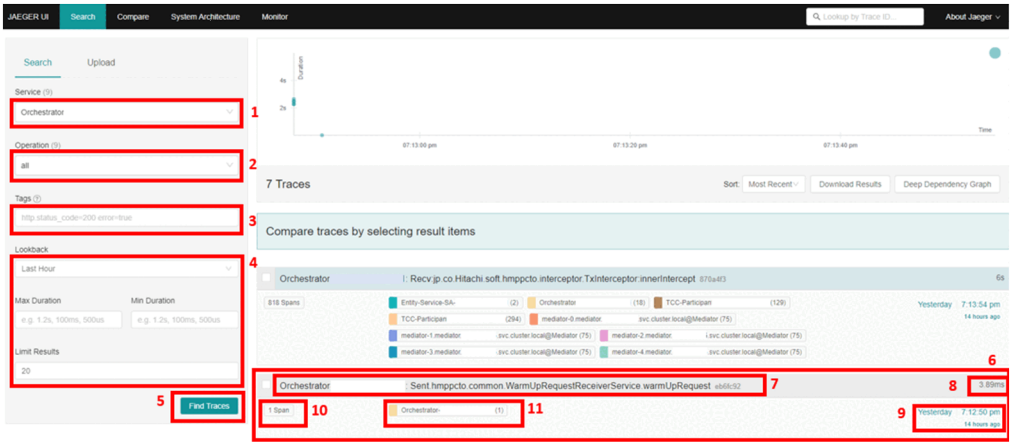


表 11–18　Search Page 項目

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、「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、「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。
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、「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。

（凡例）−：該当しません。

Trace Page

「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 11-34　Trace Page

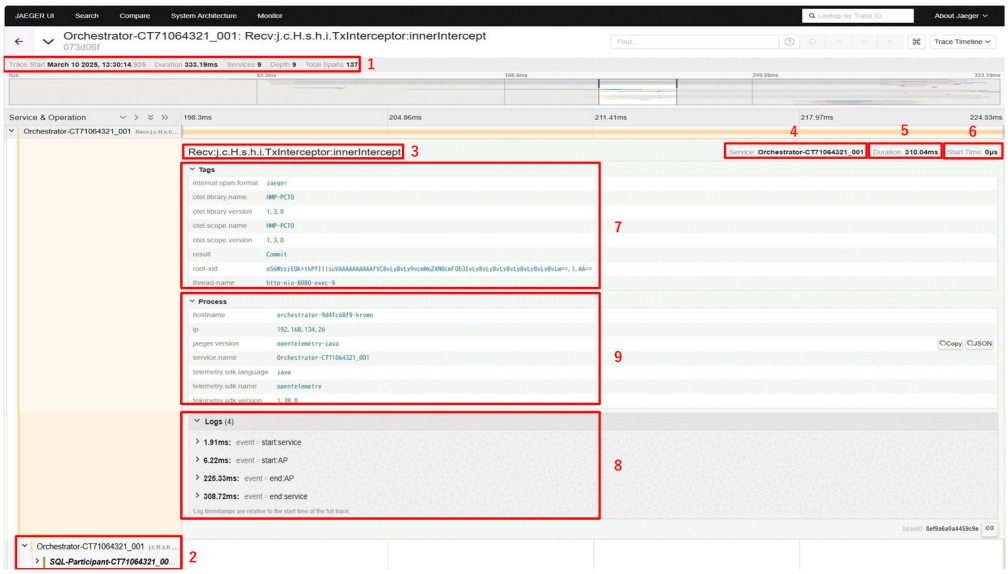


表 11-19　Trace Page 項目

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 • 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「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`
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`

図中番号	項目名	説明
		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

（凡例）－：該当しません。

(b)　Google Cloud Platform の Cloud Trace による参照

Google Cloud Platform 環境で分散トレースを閲覧する場合は、Cloud Trace を使用して分散トレースを参照します。詳細については Google Cloud Platform の公式ドキュメントを参照してください。

(c)　New Relic 使用時の参照方法

New Relic 使用時に分散トレースを閲覧する場合は、New Relic UI（Web UI）を使用して分散トレースを参照します。詳細については New Relic の公式ドキュメントを参照してください。

(d)　Red Hat OpenShift Container Platform の OpenShift（Web UI）による参照

Red Hat OpenShift Container Platform 環境で分散トレースを閲覧する場合は、Red Hat OpenShift Container Platform（Web UI）を使用して分散トレースを参照します。詳細については、Red Hat OpenShift Container Platform の公式ドキュメントを参照してください。

11.5.3　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。

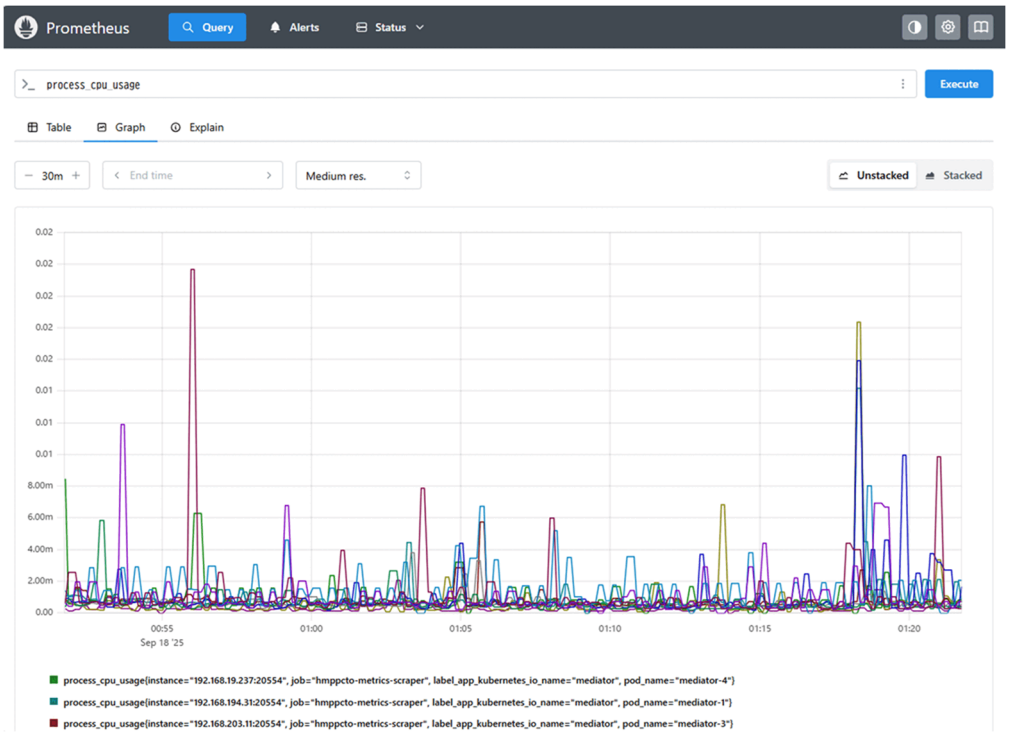
また、Jaeger から出力されるメトリクスについても Prometheus サーバで収集します。

(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 11-35　Prometheus サーバの標準の Web UI



画面上部の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)　Grafana による参照

Prometheus で収集されたメトリクスは、メトリクス閲覧アプリケーションである Grafana で確認できます。Microsoft Azure 環境でメトリクスを収集した場合は、マネージド Grafana を使用してメトリクスを確認してください。Grafana の詳細については、公式ドキュメントを参照してください。

(c)　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
```

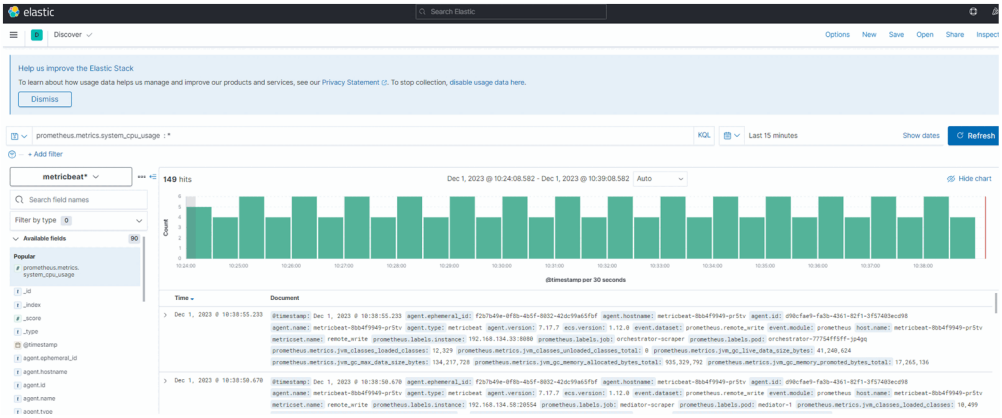
注※1

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 11-36　Kibana によるメトリクス参照例



(d)　Google Cloud Platform 環境の場合の参照方法

Google Cloud Platform 環境でメトリクスを確認する場合は、Cloud Monitoring を使用してメトリクスを参照します。詳細については、Google Cloud Platform の公式ドキュメントを参照してください。

(e)　Amazon Web Services 環境の場合の参照方法

Amazon Web Services 環境でメトリクスを確認する場合は、Amazon CloudWatch を使用してメトリクスを参照します。詳細については、Amazon Web Services の公式ドキュメントを参照してください。

(f)　New Relic 使用時の参照方法

New Relic 使用時にメトリクスを確認する場合は、New Relic UI（Web UI）を使用してメトリクスを参照します。詳細については、New Relic の公式ドキュメントを参照してください。

(g)　Red Hat OpenShift Container Platform モニタリングの Prometheus 使用時の参照方法

Red Hat OpenShift Container Platform モニタリングの Prometheus 使用時にメトリクスを確認する場合は、Red Hat OpenShift Container Platform（Web UI）を使用してメトリクスを参照します。詳細については、Red Hat OpenShift Container Platform の公式ドキュメントを参照してください。

(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については、「付録 D　HMP-PCTO のメトリクス取得項目」を参照してください。

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。

(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 に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-Service コンテナイメージ

Orchestrator-Service コンテナイメージのエンドポイントを次に示します。

表 11-20 Orchestrator-Service コンテナイメージのエンドポイント

項番	エンドポイント名	役割
1	/hmppcto/metrics	Prometheus にメトリクスを公開するためのエンドポイント

Relay-Service コンテナイメージ

Relay-Service コンテナイメージのエンドポイントを次に示します。

表 11-21 Relay-Service コンテナイメージのエンドポイント

項番	エンドポイント名	役割
1	/hmppcto/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(SQL)コンテナイメージ

Entity-Service(SQL)コンテナイメージのエンドポイントを次に示します。

表 11-22　Entity-Service(SQL)コンテナイメージのエンドポイント

項番	エンドポイント名	役割
1	/hmppcto/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(TCC)コンテナイメージ

Entity-Service(TCC)コンテナイメージのエンドポイントを次に示します。

表 11-23　Entity-Service(TCC)コンテナイメージのエンドポイント

項番	エンドポイント名	役割
1	/hmppcto/metrics	Prometheus にメトリクスを公開するためのエンドポイント

Orchestrator-Service(SQL)コンテナイメージ

Orchestrator-Service(SQL)コンテナイメージのエンドポイントを次に示します。

表 11-24　Orchestrator-Service(SQL)コンテナイメージのエンドポイント

項番	エンドポイント名	役割
1	/hmppcto/metrics	Prometheus にメトリクスを公開するためのエンドポイント

Alternate-Service コンテナイメージ

Alternate-Service コンテナイメージのエンドポイントを次に示します。

表 11-25　Alternate-Service コンテナイメージのエンドポイント

項番	エンドポイント名	役割
1	/hmppcto/metrics	Prometheus にメトリクスを公開するためのエンドポイント

Alternate-Service(SQL)コンテナイメージ

Alternate-Service(SQL)コンテナイメージのエンドポイントを次に示します。

表 11-26　Alternate-Service(SQL)コンテナイメージのエンドポイント

項番	エンドポイント名	役割
1	/hmppcto/metrics	Prometheus にメトリクスを公開するためのエンドポイント

Mediator コンテナイメージ

Mediator コンテナイメージのエンドポイントを次に示します。

表 11-27　Mediator コンテナイメージのエンドポイント

項番	エンドポイント名	役割
1	/hmppcto/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 コンテナイメージ

TP1-Bridge コンテナイメージのエンドポイントを次に示します。

表 11-28　TP1-Bridge コンテナイメージのエンドポイント

項番	エンドポイント名	役割
1	/hmppcto/metrics	Prometheus にメトリクスを公開するためのエンドポイント

11.5.4　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 11-29　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 （形式識別子/フォーマット ID）	XID の形式を示す数値。 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 11-30　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。

名称	オフセット（単位：バイト）	長さ（単位：バイト）	説明
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-Module を呼び出す Orchestrator、または 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。

11.6 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。

11.6.1 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 11-31 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)	ログを永続化するためのアプリケーションまたはサービス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 の標準出力をログ収集アプリケーションまたはサービスに送信します。ログ収集アプリケーションまたはサービスは、ログを永続化するためのアプリケーションまたはサービスに情報を送信します。 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。ログ閲覧アプリケーションまたはサービスで、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 ・Spring Boot ・Java VM ・Java ランタイム ・TP1/Client/J
2	分散トレースの情報	分散トレースを永続化するためのアプリケーションまたはサービス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分散トレース収集アプリケーションまたはサービスに送信します。分散トレース収集アプリケーションまたはサービスは、分散トレースを永続化するためのアプリケーションまたはサービスに情報を送信します。
3	メトリクスの情報	メトリクス情報を永続化するためのアプリケーションまたはサービス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メトリクス収集アプリケーションまたはサービスに送信します。メトリクス収集アプリケーションまたはサービスはメトリクス情報を永続化するためのアプリケーションまたはサービスに情報を送信します。
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。

項番	情報の種類	取得する情報
5	連携製品が出力したトラブルシュート情報	各連携製品コンテナのボリュームに出力される情報です。 • PostgreSQL のログ情報 • EADS サーバのトラブルシュート情報
6		HMP-PCTO コンテナのボリュームに出力される情報です。 • EADS クライアントのトラブルシュート情報 • Spring Boot の組み込み HTTP サーバのアクセスログ • TP1/Client/J のトレース情報
7		連携製品が稼働するマシン上に出力される情報です。 • HiRDB のトレース・ログ情報 • SQL Server のトレース・ログ情報 • Oracle Database 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 11-32　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)	ログを永続化するためのアプリケーションまたはサービスに送られた標準出力の情報を、ログ閲覧アプリケーションまたはサービスで確認します。
2	分散トレースの情報	分散トレースを永続化するためのアプリケーションまたはサービスに送られた分散トレースの情報を、分散トレース閲覧アプリケーションまたはサービスで確認します。
3	メトリクスの情報	メトリクスを永続化するためのアプリケーションまたはサービスに送られたメトリクスの情報を、メトリクス閲覧アプリケーションまたはサービスで確認します。
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	各連携製品のコンテナのボリュームに出力される情報を次のどちらかの方法で収集し、情報を確認します。 • 各連携製品のマニュアルに記載されている出力場所や、出力方法を参照して収集します。 • Elastic Stack を使用している場合、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 コンテナに組み込むことで Elasticsearch に情報が格納され、リアルタイムで確認できます。
6		HMP-PCTO コンテナのボリュームに出力される情報を次のどちらかの方法で収集し情報を確認します。 • uCosminexus Application Runtime for Spring Boot のスナップショットログ収集機能を使用することで、指定したボリュームの内容をスナップショットログとして出力できます。

項番	情報の種類	調査方法
		• Elastic Stack を使用している場合、各コンテナ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 コンテナに組み込むことで Elasticsearch に情報が格納され、リアルタイムで確認できます。
7		連携製品が稼働するマシン上に出力される情報を次の方法で収集して確認します。 • 各連携製品のマニュアルに記載されている出力場所や、出力方法を参照して収集します。

注　Microsoft Azure 環境、Google Cloud Platform 環境、Amazon Web Services 環境、New Relic、または Red Hat OpenShift Container Platform 環境を使用している場合、各コンテナのボリュームに出力される情報について、サイドカーコンテナの組み込みによる情報の収集ができません。そのため、uCosminexus Application Runtime for Spring Boot のスナップショットログ機能を使用して、HMP-PCTO コンテナのボリュームに出力される情報を収集してください。

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 11–33　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)	ログを永続化するためのアプリケーションまたはサービスに格納された標準出力の情報	• Elastic Stack 使用環境の場合 Elasticsearch のスナップショット機能で収集します。 • Microsoft Azure 環境の場合 Azure Monitor のエクスポート機能で収集します。 • Google Cloud Platform 環境の場合 Cloud Logging のログエクスプローラで収集します。 • Amazon Web Services 環境の場合 Amazon CloudWatch を使用して収集します。 • New Relic 使用時の場合 New Relic の NRQL を使用して収集します。 • Red Hat OpenShift Container Platform 環境の場合 Red Hat OpenShift Container Platform が提供する Red Hat OpenShift Logging Operator を使用して収集します。Red Hat OpenShift Logging Operator 単体では永続化できないため、Red Hat OpenShift Container Platform が提供する Loki Operator を使用して永続化します。

項番	情報の種類	収集する情報	収集方法
2	分散トレースの情報	分散トレースを永続化するためのアプリケーションまたはサービスに格納された分散トレース情報	• Elastic Stack 使用環境の場合 Elasticsearch のスナップショット機能で収集します。 • Microsoft Azure 環境の場合 Elasticsearch のスナップショット機能で収集します。 • Google Cloud Platform 環境の場合 Cloud Trace の Cloud Console で収集します。 • Amazon Web Services 環境の場合 Amazon OpenSearch Service で収集します。 • New Relic 使用時の場合 New Relic の NRQL を使用して収集します。 • Red Hat OpenShift Container Platform 環境の場合 Red Hat OpenShift Container Platform が提供する Tempo Operator を使用して収集します。 Tempo Operator 単体では永続化できないため、Red Hat OpenShift Container Platform が提供する TempoStack を使用して永続化します。
3	メトリクスの情報	メトリクスを永続化するためのアプリケーションまたはサービスに格納されたメトリクス情報	• Elastic Stack 使用環境の場合 Elasticsearch のスナップショット機能で収集します。 • Microsoft Azure 環境の場合 Azure Monitor のエクスポート機能で収集します。 • Google Cloud Platform 環境の場合 Cloud Monitoring のメトリクスエクスプローラで収集します。 • Amazon Web Services 環境の場合 Amazon CloudWatch を使用して収集します。 • New Relic 使用時の場合 New Relic の NRQL を使用して収集します。 • Red Hat OpenShift Container Platform 環境の場合 Thanos Querier API を使用して収集します。

項番	情報の種類	収集する情報	収集方法
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	各連携製品コンテナのボリュームに出力される情報 • PostgreSQL のログ情報 • EADS サーバのトラブルシュート情報	• 連携製品が稼働するコンテナ上でトラブルシュート情報の収集・出力を実施します。 • uCosminexus Application Runtime 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。 • Elastic Stack 使用環境の場合 Elasticsearch のスナップショット機能で収集します。
6		HMP-PCTO コンテナのボリュームに出力される情報 • EADS クライアントのトラブルシュート情報 • Spring Boot の組み込み HTTP サーバのアクセスログ • TP1/Client/J のトレース情報	• uCosminexus Application Runtime 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。 • Elastic Stack 使用環境の場合 Elasticsearch のスナップショット機能で収集します。
7		連携製品が稼働するマシン上に出力される情報 • HiRDB のトレース・ログ情報 • SQL Server のトレース・ログ情報 • Oracle Database 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。

11.6.2　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 11-34　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	「logstash」

項番	情報の種類	Elasticsearch の index 名のプレフィックス
	（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	
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) Microsoft Azure に格納された情報の収集方法

Azure Monitor のエクスポート機能を使用して情報を収集します。収集先として、Azure Blob Storage などの永続ストレージに出力してください。

＜ Azure Monitor のエクスポート機能で収集する手順＞

1. Azure ポータルにログイン
2. 対象の Azure Monitor Log Analytics ワークスペースを選択

3.「エクスポート」オプションを選択

4. エクスポート先として Azure Blob Storage を指定

サポート部署へ送付するなどの目的で、Azure Blob Storage にエクスポートされた情報を 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。

< Azure Monitor Log Analytics の情報を date.tgz というファイルにまとめる例>

```
az storage blob download-batch -d . --source <コンテナ名> --account-name <ストレージアカウント名>
tar cvfz log_analytics_data.tgz *
```

(3) Google Cloud Platform に格納された情報の収集方法

Cloud Logging、Cloud Trace および Cloud Monitoring の情報を取得してください。

- Cloud Logging のログ エクスプローラで収集する場合
  ログ エクスプローラを開き、「操作」メニューをクリックしたあと、「ダウンロード」をクリックしてください。
  詳細については、Google Cloud Platform の公式ドキュメントを参照してください。
- Cloud Trace から Cloud Console で収集する場合
  Cloud Console から Cloud Trace API を使用して、対象の分散トレースを取得してください。
  詳細については、Google Cloud Platform の公式ドキュメントを参照してください。
- Cloud Monitoring のメトリクスエクスプローラで収集する場合
  「CSV 形式でダウンロード」ボタンをクリックしてください。
  詳細については、Google Cloud Platform の公式ドキュメントを参照してください。

(4) Amazon Web Services に格納された情報の収集方法

Amazon CloudWatch、および Amazon OpenSearch Service を使用して情報を収集します。

<Amazon CloudWatch で HMP-PCTO の標準出力の情報を収集する場合>

1. Amazon CloudWatch を開き「Log Management」を選択
2. 収集対象のロググループを選択
3. 収集対象のログイベントを含むログストリームを選択
4.「アクション」を選択してから「検索結果をダウンロード」を選択

<Amazon CloudWatch でメトリクスの情報を収集する場合>

1. Amazon CloudWatch を開き「すべてのメトリクス」を選択
2. 収集対象のメトリクスを選択
3.「アクション」を選択してから「.csv 形式でダウンロード」を選択

<Amazon OpenSearch Service で分散トレース情報を収集する場合>

Jaeger-UI の「Download Results」から、分散トレースを取得する例を次に示します。

1. Jaeger-UI を開き対象の分散トレースを検索する
2. 「Download Results」を選択

(5) New Relic に格納された情報の収集方法

NRQL を使用して New Relic に保存されたメッセージログ、分散トレースおよびメトリクスの情報を取得してください。メッセージログは、New Relic で一度に出力できる最大行数までを New Relic UI を使用して取得できます。

詳細については、New Relic の公式ドキュメントを参照してください。

(6) Red Hat OpenShift Container Platform に格納された情報の収集方法

Red Hat OpenShift Container Platform の Web UI、または各 API を使用して保存されたメッセージログ、分散トレースおよびメトリクスの情報を取得してください。

詳細については、Red Hat OpenShift Container Platform の公式ドキュメントを参照してください。

<メッセージログ情報を収集する場合>

Red Hat OpenShift Container Platform の Web UI 上で、または Loki Operator に対して LogQL を使用して、保存されたメッセージログ情報を取得してください。

<分散トレース情報を収集する場合>

Tempo Query API を使用して、保存された分散トレース情報を取得してください。

<メトリクス情報を収集する場合>

Thanos Querier API を使用して、保存されたメトリクス情報を取得してください。

(7) 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 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 for Spring Boot の機能で収集したい場合は、uCosminexus Application Runtime 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。

(8) 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。

11.6.3 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

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、障害発生を検知できます。

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 11–37　HMP-PCTO 障害発生時の流れ（SQL-Participant を使用する場合）

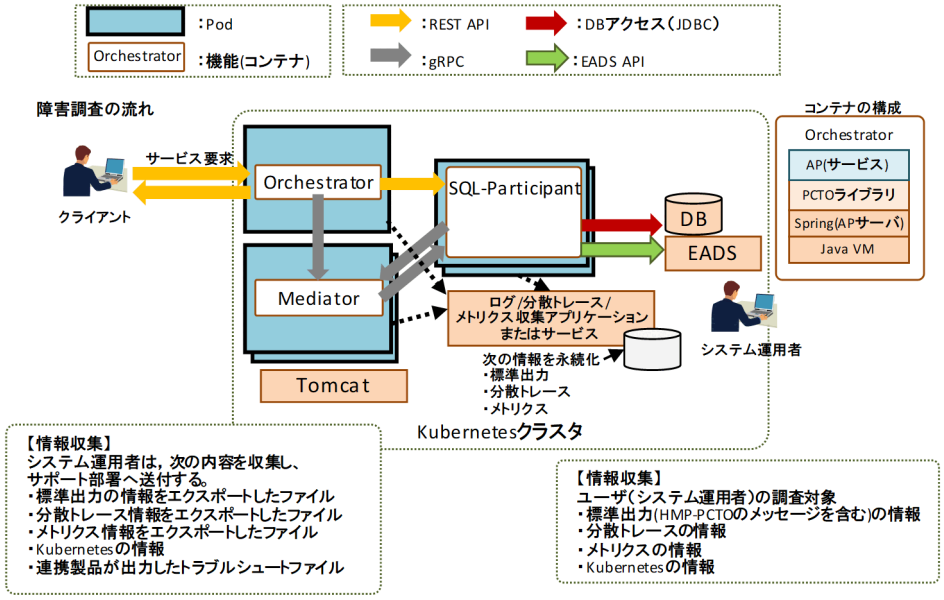


図 11–38　HMP-PCTO 障害発生時の流れ（TCC-Participant を使用する場合）

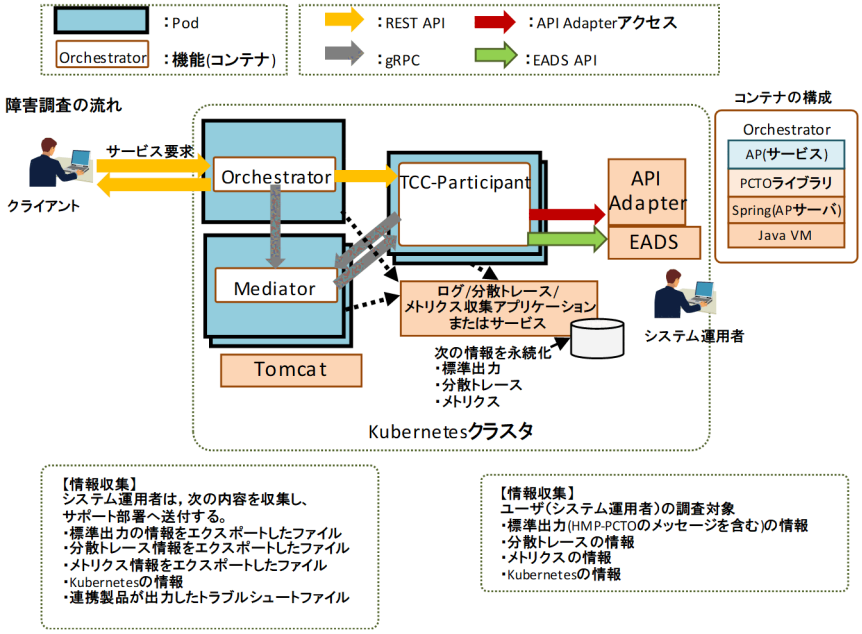
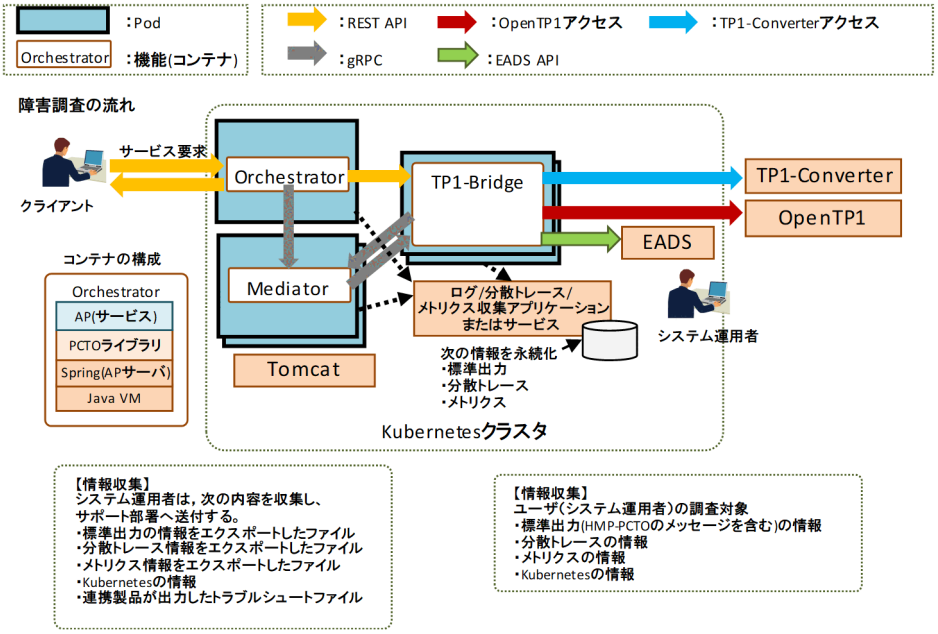


図 11–39　HMP-PCTO 障害発生時の流れ（TP1-Bridge を使用する場合）



(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、「11.6.2　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 11–35　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 11-36　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 メッセージ（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

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

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、「10.　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「hmppcto_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、「10. 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。

  - Spring Boot
  - 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。

- EADS（トライアル版では不要です）
- uCosminexus Application Runtime 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 Boot
- 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。

- Spring Boot の組み込み HTTP サーバのアクセスログ
  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。

11.7　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。

11.7.1　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

(a)　共通

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 11–37　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	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。
		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）	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実行されている場合、KFSG82102-I メッセージがイベントリソースに出力される）	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。
2	プロセスダウン	KFSG74103-E メッセージが出力された（「ロガーがメッセージプロパティファイルを取	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。

項番	障害の種類	現象	主な原因	対策
		得できませんでした。」)		

(b) Elastic Stack 使用環境の場合

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 11-38 メッセージログに関する障害の種類と対策（Elastic Stack 使用環境の場合）

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	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	• 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。
			可視化ツールにアクセスする 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いったん停止し、Elasticsearch の容量を拡大してから、再度、各コンポーネントを開始してください。

(c) Microsoft Azure 環境の場合

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 11-39 メッセージログに関する障害の種類と対策（Microsoft Azure 環境の場合）

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	Microsoft Azure ポータル上で HMP-PCTO のメッセージを確認できない	リソースグループ内に Log Analytics ワークスペースが作成されていない	Microsoft Azure ポータルからリソースグループ内に Log Analytics ワークスペースを作成してください。
			AKS クラスタが Log Analytics ワークスペースに接続されていない	Azure ポータルまたは Azure CLI を使用して、AKS クラスタのモニタリングを有効化し、Log Analytics ワークスペースに接続してください。
			Azure Monitor Agent がデプロイされていない	AKS クラスタ内で Azure Monitor Agent がデプロイされていることを確認してください。
			Log Analytics ワークスペースへのアクセス権がない	利用ユーザに適切なアクセス権限を設定してください。

項番	障害の種類	現象	主な原因	対策
			メッセージログの検索クエリに誤りがある	Kusto の形式に沿った検索クエリが使用されていることを確認してください。

(d) Google Cloud Platform 環境の場合

特になし。

(e) Amazon Web Services 環境の場合

特になし。

(f) New Relic 使用時の場合

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 11-40 メッセージログに関する障害の種類と対策（New Relic 使用時の場合）

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	New Relic UI 上で HMP-PCTO のメッセージを確認できない	New Relic の Kubernetes integration のコンポーネントがデプロイされていない	Kubernetes クラスタ内に Kubernetes integration コンポーネントがデプロイされていることを確認してください。

(g) Red Hat OpenShift Container Platform 環境の場合

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 11-41 メッセージログに関する障害の種類と対策（Red Hat OpenShift Container Platform 環境の場合）

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	Red Hat OpenShift Container Platform の Web UI 上で HMP-PCTO のメッセージを確認できない	• Red Hat OpenShift Logging Operator が動作していない • Loki Operator が動作していない	OpenShift クラスタ内、openshift-logging プロジェクトの Pod がデプロイされていることを確認してください。

(2) 分散トレース

(a) 共通

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 11-42　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	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。

(b)　Elastic Stack および Jaeger 使用環境の場合

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 11-43　分散トレースに関する障害の種類と対策（Elastic Stack および Jaeger 使用環境の場合）

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 Pod から Jaeger 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（「(3) jp.co.Hitachi.soft.hmppcto のパラメタ」の tracing.url）で指定する URL が、Jaeger のホスト名・ポートと合っているか確認してください。
			Jaeger から Elasticsearch への送信が失敗している	Jaeger および Elasticsearch の状態を確認し、再起動してください。
			Jaeger 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 を一度アンデプロイします。そのあと、Exporter の内部キューの容量の設定（「(1)　Jaeger のパラメタ」）で内部キューのサイズを変更し、再度 Jaeger をデプロイしてください。
			Jaeger の Pod が動作していない	Jaeger の状態を確認し、再起動してください。
			Jaeger の Web UI にアクセスする URL が誤っている	URL が正しいかどうか確認してください。

(c)　Microsoft Azure 環境の場合

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 11-44　分散トレースに関する障害の種類と対策（Microsoft Azure 環境の場合）

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	HMP-PCTO の Pod から Jaeger 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（「(3) jp.co.Hitachi.soft.hmppcto のパラメタ」の tracing.url）で指定する URL

項番	障害の種類	現象	主な原因	対策
		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		が、Jaeger のホスト名およびポートと合っているか確認してください。
			Jaeger から Elasticsearch への送信が失敗している	Jaeger および Elasticsearch の状態を確認し、再度起動してください。
			Jaeger 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 を一度アンデプロイします。そのあと、Exporter の内部キューの容量の設定（「9.3.6　Jaeger」）で内部キューのサイズを変更し、再度 Jaeger をデプロイしてください。
			Jaeger の Pod が動作していない	Jaeger の状態を確認し、再度起動してください。
			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。

(d)　Google Cloud Platform 環境の場合

特になし。

(e)　Amazon Web Services 環境の場合

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 11-45　分散トレースに関する障害の種類と対策（Microsoft Azure 環境の場合）

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 Pod からの Jaeger 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（「(3) jp.co.Hitachi.soft.hmppcto のパラメタ」の tracing.url）で指定する URL が、Jaeger のホスト名・ポートと合っているか確認してください。
			Jaeger から Amazon OpenSearch Service への送信が失敗している	Jaeger および Amazon OpenSearch Service の状態を確認し、再度起動してください。
			Jaeger 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 を一度アンデプロイします。そのあと、Jaeger の内部キューの容量の設定（「(1)　Jaeger のパラメタ」）で内部キューのサイズを変更し、再度 Jaeger をデプロイしてください。
			Jaeger の Pod が動作していない（標準出力に「Error 400 (Bad Request): Failed to parse mapping～」が出力されている）	Jaeger を一度アンデプロイします。そのあと、Jaeger の環境変数に「ES_VERSION」を追加し、再度 Jaeger をデプロイしてください。

項番	障害の種類	現象	主な原因	対策
			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。

(f) New Relic 使用時の場合

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 11-46 分散トレースに関する障害の種類と対策（New Relic 使用時の場合）

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	• New Relic UI 上で分散トレースを確認できない • 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 Pod からの分散トレース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（「(3) jp.co.Hitachi.soft.hmppcto のパラメタ」の tracing.url）で指定する URL が、OpenTelemetry Collector のホスト名およびポートと合っているかを確認してください。
			OpenTelemetry Collector から New Relic へ送信する設定が誤っている	New Relic OTLP エンドポイントおよび New Relic のライセンスキーが正しいかどうか確認してください。

(g) Red Hat OpenShift Container Platform 環境の場合

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 11-47 分散トレースに関する障害の種類と対策（Red Hat OpenShift Container Platform 環境の場合）

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	• Red Hat OpenShift Container Platform の Web UI 上で分散トレースを確認できない • 標準出力に、「WARNING ～ Failed to export traces:～」というメッセージが出力されるされる	HMP-PCTO の Pod からの分散トレース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（「(3) jp.co.Hitachi.soft.hmppcto のパラメタ」の tracing.url）で指定する URL が、OpenTelemetry Collector のホスト名およびポートと合っているかを確認してください。
			OpenTelemetry Collector から Tempo Operator へ送信する設定が誤っている	Tempo Operator エンドポイントが正しいかどうか確認してください。

(3) メトリクス

(a) 共通

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 11-48 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	HMP-PCTO の Pod が動作していることを確認してください。

(b) Elastic Stack および Prometheus 使用環境の場合

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 11-49 メトリクスに関する障害の種類と対策（Elastic Stack および Prometheus 使用環境の場合）

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	Prometheus-Server が動作していない	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（「9.3.7 Prometheus」）を見直してください。
			Prometheus の Web UI にアクセスする 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。
			可視化ツールにアクセスする URL が誤っている	URL が正しいかどうか確認してください。

(c) Microsoft Azure 環境の場合

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 11-50　メトリクスに関する障害の種類と対策（Microsoft Azure 環境の場合）

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	Microsoft Azure ポータル上で HMP-PCTO のメトリクスの情報を確認できない	Azure Monitor Agent がデプロイされていない	AKS クラスタ内で Azure Monitor Agent がデプロイされていることを確認してください。
			• リソースグループ内に Azure Monitor ワークスペースが作成されていない • リソースグループ内に Grafana ワークスペースが作成されていない	Microsoft Azure ポータルからリソースグループ内に対応するワークスペースを作成してください。
			Application Insights インスタンスへのアクセス権がない	利用ユーザに適切なアクセス権限を設定してください。

(d)　Google Cloud Platform 環境の場合

特になし。

(e)　Amazon Web Services 環境の場合

特になし。

(f)　New Relic 使用時の場合

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 11-51　メトリクスに関する障害の種類と対策（New Relic 使用時の場合）

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	New Relic UI 上で HMP-PCTO のメトリクスの情報を確認できない	nri-prometheus コンポーネントがデプロイされていない	Kubernetes クラスタ内に nri-prometheus コンポーネントがデプロイされていることを確認してください。
			HMP-PCTO の Pod がメトリクスの収集対象になっていない	HMP-PCTO の Kubernetes マニフェストに設定されたアノテーション prometheus.io/*の値を確認してください。 アノテーションの詳細については、「3.7.17　New Relic によるメトリクス収集の Kubernetes マニフェスト作成」を参照してください。

(g)　Red Hat OpenShift Container Platform 環境の場合

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 11-52　メトリクスに関する障害の種類と対策（Red Hat OpenShift Container Platform 環境の場合）

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	Red Hat OpenShift Container Platform の Web UI 上で HMP-PCTO のメトリクスの情報を確認できない	ServiceMonitor がデプロイされていない	Kubernetes クラスタ内にメトリクス収集用の ServiceMonitor がデプロイされていることを確認してください。詳細については、「3.7.18　Red Hat OpenShift Container Platform モニタリングの Prometheus によるメトリクス収集の Kubernetes マニフェスト作成」を参照してください。
			HMP-PCTO の Pod がメトリクスの収集対象になっていない	Red Hat OpenShift Container Platform の Web UI から ServiceMonitor の動作状況を確認してください。

11.8　稼働日ファイルに関する注意事項

HMP-PCTO では、Mediator が稼働した日を記録したファイルを稼働日ファイルとして保存します。HMP-PCTO が稼働した日を特定する場合に、稼働日ファイルを使用します。

稼働日ファイルは、次の規則で保存されます。

- 保存場所

  稼働日ファイル用ディレクトリ（running-dates）に保存されます。

  稼働日ファイル用ディレクトリは、Mediator の初回起動時に、Mediator の ConsensusLog の永続ボリュームマウント用ディレクトリ下に作成されます。

- ファイル名

  Mediator ごとに、Mediator を稼働した月単位で次の命名規則で作成されます。

  <ConsensusLog Path※1>/running-dates/<Mediator DNS のホスト名>_<利用月※2>

  注※1

  Mediator の ConsensusLog の永続ボリュームマウント用ディレクトリの絶対パス（jp.co.Hitachi.soft.hmppcto.mediator.consensus-log.path の指定値）です。

  注※2

  YYYY-MM 形式です。

- 保存期間

  稼働日ファイルは、作成されたあと 3 か月間保存され、その翌月以降の稼働日に削除されます。

稼働日ファイルの書き込み容量不足の対策として、Mediator1 台につき、1.5MB のディスク容量を確保してください。

稼働日ファイルを使用する場合、Mediator の ConsensusLog の永続ボリュームマウント用ディレクトリおよび Mediator の Namespace を、ほかの HMP-PCTO の Mediator と異なる名前に指定してください。

重要

当月の稼働日ファイルに対して、読み込み、書き込み、および削除をしないでください。これらを実施した場合、HMP-PCTO の稼働日を特定できなくなるおそれがあります。

付録

付録 A　提供ファイル

HMP-PCTO およびオプション製品の提供ファイルについて説明します。

HMP-PCTO およびオプション製品は、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の Kubernetes アプリケーションおよび RAS 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。

(1)　Hitachi Microservices Platform - Paxos Commit Transaction Orchestrator

表 A-1　提供ファイル一覧（Hitachi Microservices Platform - Paxos Commit Transaction Orchestrator）

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-server-http-filter¥V.R.S	Entity-Service 用ライブラリ配置ディレクトリ
15	hmppcto-server-http-filter-V.R.S.jar	Server 側 http filter 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
16	hmppcto-server-http-filter-V.R.S.module	Server 側 http filter 用 module ファイル
17	hmppcto-server-http-filter-V.R.S.pom	Server 側 http filter 用 pom ファイル
18	hmppcto-server-grpc-interceptor¥V.R.S	Server 側 gRPC interceptor 用ライブラリ配置ディレクトリ
19	hmppcto-server-grpc-interceptor-V.R.S.jar	Server 側 gRPC interceptor 用ライブラリファイル
20	hmppcto-server-grpc-interceptor-V.R.S.module	Server 側 gRPC interceptor 用 module ファイル
21	hmppcto-server-grpc-interceptor-V.R.S.pom	Server 側 gRPC interceptor 用 pom ファイル
22	hmppcto-jdbc-proxy-driver¥V.R.S	JDBC Proxy Driver 用ライブラリ配置ディレクトリ
23	hmppcto-jdbc-proxy-driver-V.R.S.jar	JDBC Proxy Driver 用ライブラリファイル
24	hmppcto-jdbc-proxy-driver-V.R.S.module	JDBC Proxy Driver 用 module ファイル
25	hmppcto-jdbc-proxy-driver-V.R.S.pom	JDBC Proxy Driver 用 pom ファイル
26	hmppcto-propagation-rest-template¥V.R.S	RestTemplate 用ライブラリ配置ディレクトリ
27	hmppcto-propagation-rest-template-V.R.S.jar	RestTemplate 用ライブラリファイル
28	hmppcto-propagation-rest-template-V.R.S.module	RestTemplate 用 module ファイル
29	hmppcto-propagation-rest-template-V.R.S.pom	RestTemplate 用 pom ファイル
30	hmppcto-propagation-grpc¥V.R.S	gRPC 用ライブラリ配置ディレクトリ
31	hmppcto-propagation-grpc-V.R.S.jar	gRPC 用ライブラリファイル
32	hmppcto-propagation-grpc-V.R.S.module	gRPC 用 module ファイル
33	hmppcto-propagation-grpc-V.R.S.pom	gRPC 用 pom ファイル
34	hmppcto-propagation-webclient¥V.R.S	WebClient 用ライブラリ配置ディレクトリ
35	hmppcto-propagation-webclient-V.R.S.jar	WebClient 用ライブラリファイル
36	hmppcto-propagation-webclient-V.R.S.module	WebClient 用 module 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
37	hmppcto-propagation-webclient-V.R.S.pom	WebClient 用 pom ファイル
38	hmppcto-mediator¥V.R.S	Mediator 用ライブラリ配置ディレクトリ
39	hmppcto-mediator-V.R.S.jar	Mediator 用ライブラリファイル
40	hmppcto-mediator-V.R.S.module	Mediator 用 module ファイル
41	hmppcto-mediator-V.R.S.pom	Mediator 用 pom ファイル
42	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
43	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
44	hmppcto-participant-common-V.R.S.jar	SQL-Participant/TCC-Participant 共通用ライブラリファイル
45	hmppcto-participant-common-V.R.S.module	SQL-Participant/TCC-Participant 共通用 module ファイル
46	hmppcto-participant-common-V.R.S.pom	SQL-Participant/TCC-Participant 共通用 pom ファイル
47	hmppcto-participant-sql¥V.R.S	SQL-Participant 用ライブラリ配置ディレクトリ
48	hmppcto-participant-sql-V.R.S.jar	SQL-Participant 用ライブラリファイル
49	hmppcto-participant-sql-V.R.S.module	SQL-Participant 用 module ファイル
50	hmppcto-participant-sql-V.R.S.pom	SQL-Participant 用 pom ファイル
51	hmppcto-participant-tcc¥V.R.S	TCC-Participant 用ライブラリ配置ディレクトリ
52	hmppcto-participant-tcc-V.R.S.jar	TCC-Participant 用ライブラリファイル
53	hmppcto-participant-tcc-V.R.S.module	TCC-Participant 用 module ファイル
54	hmppcto-participant-tcc-V.R.S.pom	TCC-Participant 用 pom ファイル
55	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
56	hmppcto-lb-V.R.S.jar	ロードバランス用ライブラリ
57	hmppcto-lb-V.R.S.module	ロードバランス用 module ファイル
58	hmppcto-lb-V.R.S.pom	ロードバランス用 pom ファイル
59	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
60	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
61	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

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
62	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
63	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
64	hmppcto-participant-adapter-V.R.S.jar	Participant アダプタ用ライブラリ
65	hmppcto-participant-adapter-V.R.S.module	Participant アダプタ用 module ファイル
66	hmppcto-participant-adapter-V.R.S.pom	Participant アダプタ用 pom ファイル
67	hmppcto-observability-modules-otlp-http¥V.R.S	トラブルシュート機能（OTLP（http/protobuf））用ライブラリ配置ディレクトリ
68	hmppcto-observability-modules-otlp-http-V.R.S.jar	トラブルシュート機能（OTLP（http/protobuf））用ライブラリ
69	hmppcto-observability-modules-otlp-http-V.R.S.module	トラブルシュート機能（OTLP（http/protobuf））用 module ファイル
70	hmppcto-observability-modules-otlp-http-V.R.S.pom	トラブルシュート機能（OTLP（http/protobuf））用 pom ファイル
71	hmppcto-observability-modules-gcp¥V.R.S	トラブルシュート機能（Google Cloud Platform）用ライブラリ配置ディレクトリ
72	hmppcto-observability-modules-gcp-V.R.S.jar	トラブルシュート機能（Google Cloud Platform）用ライブラリ
73	hmppcto-observability-modules-gcp-V.R.S.module	トラブルシュート機能（Google Cloud Platform）用 module ファイル
74	hmppcto-observability-modules-gcp-V.R.S.pom	トラブルシュート機能（Google Cloud Platform）pom ファイル
75	hmppcto-server-tomcat	Tomcat 用ライブラリ配置ディレクトリ
76	hmppcto-server-tomcat-V.R.S.jar	Tomcat 用ライブラリファイル
77	hmppcto-server-tomcat-V.R.S.module	Tomcat 用 module ファイル
78	hmppcto-server-tomcat-V.R.S.pom	Tomcat 用 pom ファイル
79	java-projects	HMP-PCTO Java プロジェクト配置用ディレクトリ
80	mediator	Mediator 用 Java プロジェクトディレクトリ
81	src¥main¥java¥jp¥co¥Hitachi¥soft¥hmppcto¥Application.java	Mediator 用 Java ソースファイル
82	build.gradle	Mediator 用 build.gradle ファイル
83	settings.gradle	Mediator 用 settings.gradle 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
84	gradle.properties	Mediator 用 gradle.properties ファイル
85	scripts	HMP-PCTO スクリプトファイル配置用ディレクトリ
86	dependency_check.sh	デプロイ依存関係チェックスクリプト
87	health_check.sh	トランザクションヘルスチェックスクリプト
88	wait_transactions.sh	未決着トランザクション待機スクリプト
89	templates	HMP-PCTO テンプレートファイル配置用ディレクトリ
90	container-files	Containerfile 配置ディレクトリ
91	RHEL8	RHEL8 用の Containerfile 配置ディレクトリ
92	mediator	Mediator 用 Containerfile 配置ディレクトリ
93	normal	Mediator 用 Containerfile 配置ディレクトリ （uCosminexus Application Runtime for Spring Boot の修正パッチがない場合）
94	Containerfile	Mediator 用 Containerfile（uCosminexus Application Runtime for Spring Boot の修正パッチがない場合）
95	patch	Mediator 用 Containerfile 配置ディレクトリ （uCosminexus Application Runtime for Spring Boot の修正パッチがある場合）
96	Containerfile	Mediator 用 Containerfile（uCosminexus Application Runtime for Spring Boot の修正パッチがある場合）
97	dependency-checker	Dependency-Checker 用 Containerfile 配置ディレクトリ
98	Containerfile	Dependency-Checker 用 Containerfile
99	RHEL9	RHEL9 用の Containerfile 配置ディレクトリ
100	mediator	Mediator 用 Containerfile 配置ディレクトリ
101	normal	Mediator 用 Containerfile 配置ディレクトリ （uCosminexus Application Runtime for Spring Boot の修正パッチがない場合）
102	Containerfile	Mediator 用 Containerfile（uCosminexus Application Runtime for Spring Boot の修正パッチがない場合）
103	patch	Mediator 用 Containerfile 配置ディレクトリ （uCosminexus Application Runtime for Spring Boot の修正パッチがある場合）
104	Containerfile	Mediator 用 Containerfile（uCosminexus Application Runtime for Spring Boot の修正パッチがある場合）
105	dependency-checker	Dependency-Checker 用 Containerfile 配置ディレクトリ
106	Containerfile	Dependency-Checker 用 Containerfile

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
107	helm-charts	HMP-PCTO の Helm チャート配置ディレクトリ
108	mediator-V.R.S.tgz	Mediator 用 Helm チャートパッケージファイル
109	bundled-softwares	同梱ソフトウェア配置ディレクトリ
110	EADS-client	Hitachi Elastic Application Data Store Client for Java インストーラ配置ディレクトリ
111	X64LIN	Hitachi Elastic Application Data Store Client for Java インストーラ
112	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。
113	technical-auxiliary-tools	技術補助資材配置ディレクトリ
114	eads	EADS 用技術補助資材配置ディレクトリ
115	eads_operation.sh	EADS 運用コマンド
116	readme.txt	EADS 運用コマンド使用方法テキスト

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

(2) Hitachi Microservices Platform - Bridge Option for OpenTP1

表 A-2 提供ファイル一覧（Hitachi Microservices Platform - Bridge Option for OpenTP1）

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-tp1-bridge¥V.R.S	TP1-Bridge 用ライブラリ配置ディレクトリ
3	hmppcto-tp1-bridge-V.R.S.jar	TP1-Bridge 用ライブラリファイル
4	hmppcto-tp1-bridge-V.R.S.module	TP1-Bridge 用 module ファイル
5	hmppcto-tp1-bridge-V.R.S.pom	TP1-Bridge 用 pom ファイル
6	java-projects	HMP-PCTO Java プロジェクト配置用ディレクトリ
7	tp1-bridge	TP1-Bridge 用 Java プロジェクトディレクトリ
8	src¥main¥java¥jp¥co¥Hitachi¥soft¥hmppcto¥Application.java	TP1-Bridge 用 Java ソースファイル

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
9	build.gradle	TP1-Bridge 用 build.gradle ファイル
10	settings.gradle	TP1-Bridge 用 settings.gradle ファイル
11	gradle.properties	TP1-Bridge 用 gradle.properties ファイル
12	libs	TP1-Bridge 用外部ライブラリ格納フォルダ
13	templates	HMP-PCTO テンプレートファイル配置用ディレクトリ
14	container-files	Containerfile 配置ディレクトリ
15	RHEL8	RHEL8 用の Containerfile 配置ディレクトリ
16	tp1-bridge	TP1-Bridge 用 Containerfile 配置ディレクトリ
17	normal	TP1-Bridge 用 Containerfile 配置ディレクトリ (uCosminexus Application Runtime for Spring Boot の修正パッチがない場合)
18	Containerfile	TP1-Bridge 用 Containerfile 配置（uCosminexus Application Runtime for Spring Boot の修正パッチがない場合)
19	patch	TP1-Bridge 用 Containerfile 配置ディレクトリ (uCosminexus Application Runtime for Spring Boot の修正パッチがある場合)
20	Containerfile	TP1-Bridge 用 Containerfile 配置（uCosminexus Application Runtime for Spring Boot の修正パッチがある場合)
21	RHEL9	RHEL9 用の Containerfile 配置ディレクトリ
22	tp1-bridge	TP1-Bridge 用 Containerfile 配置ディレクトリ
23	normal	TP1-Bridge 用 Containerfile 配置ディレクトリ (uCosminexus Application Runtime for Spring Boot の修正パッチがない場合)
24	Containerfile	TP1-Bridge 用 Containerfile 配置（uCosminexus Application Runtime for Spring Boot の修正パッチがない場合)
25	patch	TP1-Bridge 用 Containerfile 配置ディレクトリ (uCosminexus Application Runtime for Spring Boot の修正パッチがある場合)
26	Containerfile	TP1-Bridge 用 Containerfile 配置（uCosminexus Application Runtime for Spring Boot の修正パッチがある場合)
27	helm-charts	HMP-PCTO の Helm チャート配置ディレクトリ
28	tp1-bridge-V.R.S.tgz	TP1-Bridge 用 Helm チャートパッケージファイル

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
29	TP1-Converter	TP1-Converter 用ファイル配置ディレクトリ
30	lib	TP1-Converter 用ライブラリ配置ディレクトリ
31	tp1converter.war	TP1-Converter 用アプリケーションライブラリ
32	usrconf	TP1-Converter 用 J2EE サーバ定義ファイル配置ディレクトリ
33	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
34	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 に関連する OSS および他社製品のパラメタ一覧

このマニュアルに記載している OSS のパラメタ、および 9 章で説明しているパラメタ以外の他社製品のパラメタの一覧を次の表に示します。各パラメタの詳細は、関連の OSS および他社製品の公式ドキュメントを参照してください。

付録 B.1　HMP-PCTO 固有の指定が必要なパラメタ

(1)　Kubernetes のパラメタ

Kubernetes の各パラメタの詳細は Kubernetes の公式ドキュメントを参照してください。

表 B-1　Kubernetes のパラメタ（HMP-PCTO 固有の指定が必要なパラメタ）

項番	パラメタ名	説明
1	spec.template.metadata.labels.app.kubernetes.io/instance	－
2	spec.template.metadata.labels.app.kubernetes.io/name	「app.kubernetes.io/name」には、ほかの Kubernetes アプリケーションに設定した値と異なる値を 16 バイト以内で指定してください。 Blue/Green デプロイメントを実施する場合、Blue と Green の「app.kubernetes.io/name」に同一の値を設定してください。
3	spec.template.metadata.labels.hmppcto-app-group	アプリケーショングループ名を表す任意の文字列を 63 バイト以内で指定してください。 例：「x-system」 1 つの Kubernetes アプリケーションで指定できるアプリケーショングループ名は 1 つだけです。 「アプリケーショングループ」については、「7.2　Kubernetes アプリケーションのグループ定義」を参照してください。

（凡例）－：該当しません。

付録 B.2　HMP-PCTO 固有の指定が不要なパラメタ

(1)　Kubernetes のパラメタ

Kubernetes の各パラメタの詳細は Kubernetes の公式ドキュメントを参照してください。

表 B–2　Kubernetes のパラメタ（HMP-PCTO 固有の指定が不要なパラメタ）

項番	パラメタ名
1	apiVersion
2	automountServiceAccountToken
3	kind
4	metadata.labels
5	metadata.labels.app.kubernetes.io/instance
6	metadata.labels.app.kubernetes.io/managed-by
7	metadata.labels.app.kubernetes.io/name
8	metadata.labels.app.kubernetes.io/version
9	metadata.name
10	metadata.namespace
11	provisioner
12	roleRef.apiGroup
13	roleRef.kind
14	roleRef.name
15	rules
16	spec.accessModes
17	spec.advanced.horizontalPodAutoscalerConfig.behavior.scaleDown.policies
18	spec.behavior.scaleDown.policies
19	spec.behavior.scaleDown.policies.periodSeconds
20	spec.behavior.scaleDown.policies.type
21	spec.behavior.scaleDown.policies.value
22	spec.behavior.scaleUp.policies
23	spec.behavior.scaleUp.policies.periodSeconds
24	spec.behavior.scaleUp.policies.type
25	spec.behavior.scaleUp.policies.value

項番	パラメタ名
26	spec.capacity.storage
27	spec.claimRef
28	spec.csi.volumeHandle
29	spec.local
30	spec.local.path
31	spec.metrics
32	spec.metrics.resource.name
33	spec.metrics.resource.target.averageUtilization
34	spec.metrics.resource.target.type
35	spec.nfs
36	spec.nodeAffinity
37	spec.nodeAffinity.required.nodeSelectorTerms.matchExpressions
38	spec.nodeAffinity.required.nodeSelectorTerms.matchExpressions.operator
39	spec.nodeAffinity.required.nodeSelectorTerms.matchExpressions.values
40	spec.persistentVolumeReclaimPolicy
41	spec.replicas
42	spec.resources.requests.storage
43	spec.selector
44	spec.selector.matchLabels
45	spec.selector.matchLabels.app.kubernetes.io/instance
46	spec.selector.matchLabels.app.kubernetes.io/name
47	spec.storageClassName
48	spec.template.metadata.annotations
49	spec.template.spec.containers.env
50	spec.template.spec.containers.lifecycle.postStart
51	spec.template.spec.containers.lifecycle.preStop
52	spec.template.spec.containers.livenessProbe
53	spec.template.spec.containers.livenessProbe.exec.command
54	spec.template.spec.containers.livenessProbe.failureThreshold
55	spec.template.spec.containers.livenessProbe.httpGet.path
56	spec.template.spec.containers.livenessProbe.httpGet.port

項番	パラメタ名
57	spec.template.spec.containers.livenessProbe.initialDelaySeconds
58	spec.template.spec.containers.livenessProbe.periodSeconds
59	spec.template.spec.containers.livenessProbe.timeoutSeconds
60	spec.template.spec.containers.readinessProbe
61	spec.template.spec.containers.readinessProbe.exec.command
62	spec.template.spec.containers.readinessProbe.failureThreshold
63	spec.template.spec.containers.readinessProbe.httpGet.path
64	spec.template.spec.containers.readinessProbe.httpGet.port
65	spec.template.spec.containers.readinessProbe.initialDelaySeconds
66	spec.template.spec.containers.readinessProbe.periodSeconds
67	spec.template.spec.containers.readinessProbe.timeoutSeconds
68	spec.template.spec.containers.volumeMounts
69	spec.template.spec.imagePullSecrets
70	spec.template.spec.securityContext
71	spec.template.spec.terminationGracePeriodSeconds
72	spec.template.spec.volumes
73	spec.template.spec.volumes[].persistentVolumeClaim.claimName
74	spec.volumeName
75	subjects
76	subjects[].kind
77	subjects[].name
78	subjects[].namespace

付録 B.3　Azure Kubernetes Service（AKS）関連パラメタ

Azure Kubernetes Service（AKS）関連の各パラメタの詳細は Microsoft の公式ドキュメントを参照してください。

表 B-3　Azure Kubernetes Service（AKS）関連パラメタ

項番	パラメタ名
1	spec.csi.volumeAttributes.containerName
2	spec.csi.volumeAttributes.folderName

項番	パラメタ名
3	spec.csi.volumeAttributes.protocol
4	spec.csi.volumeAttributes.shareName
5	spec.csi.volumeAttributes.storageAccount

付録 B.4　Helm 関連パラメタ

Helm 関連の各パラメタの詳細は Helm の公式ドキュメントを参照してください。

表 B-4　Helm 関連パラメタ

項番	パラメタ名
1	metadata.labels.helm.sh/chart

付録 B.5　KEDA 関連パラメタ

KEDA 関連の各パラメタの詳細は KEDA の公式ドキュメントを参照してください。

表 B-5　KEDA 関連パラメタ

項番	パラメタ名
1	spec.triggers
2	spec.triggers.metadata.query
3	spec.triggers.metadata.serverAddress
4	spec.triggers.metadata.threshold
5	spec.triggers.type

付録 B.6　Prometheus 関連パラメタ

Prometheus 関連の各パラメタの詳細は Prometheus の公式ドキュメントを参照してください。

表 B-6　Prometheus 関連パラメタ

項番	パラメタ名
1	spec.template.metadata.annotations.prometheus.io/path
2	spec.template.metadata.annotations.prometheus.io/port
3	spec.template.metadata.annotations.prometheus.io/scrape

項番	パラメタ名
4	spec.template.metadata.annotations.prometheus.io/scheme

付録 B.7　Spring Boot 関連パラメタ

Spring Boot 関連の各パラメタの詳細は Spring Boot の公式ドキュメントを参照してください。

表 B–7　Spring Boot 関連パラメタ

項番	パラメタ名
1	data.application.properties.spring.lifecycle.timeout-per-shutdown-phase

付録 C　HMP-PCTO の分散トレース取得項目

付録 C.1　Span の一覧

HMP-PCTO の各コンポーネントで取得する Span の一覧を次の表に示します。

表の各列の記載ルールを次に示します。

スパン取得レベル

「必須情報」：必ず取得する情報です。

「デバッグ情報」：jp.co.Hitachi.soft.hmppcto.tracing.debug パラメタが true の場合だけ取得する情報です。

情報説明名

スパンイベントの場合に該当の情報を記載し、スパンタグの場合は「－」を記載しています。

key（情報名）

スパンタグの場合と、スパンイベントで情報説明に追加情報がある（「情報説明の詳細」列が「－」ではない）場合に、該当の情報を記載しています。スパンイベントで追加情報がない場合は、「－」を記載しています。

value（情報値）

次のすべての条件を満たす場合にだけ該当の情報を記載しています。1 つでも条件を満たしていない場合は、「－」を記載しています。

- スパンイベントである。
- 情報説明に追加情報がある（「情報説明の詳細」列が「－」ではない）。
- 設定する情報値が固定値である（条件によって情報値が変化するものは対象外）。

表 C-1　コンポーネント共通

コンポーネント	オペレーション名（短縮）	オペレーション処理概要	スパン取得レベル	スパンタグ/スパンイベント	情報説明		key（情報名）または value（情報値）		
					情報説明名	情報説明の詳細	key（情報名）	value（情報値）	key（情報名）または value（情報値）の詳細
Orchestrator	Recv:j.c.H.s.h.i.TxInterceptor:innerIntercept	トランザクションを開始または終了し、トランザクションの結	必須情報	スパンタグ	－	－	root-xid	－	(a)　スパンタグ

コンポーネント	オペレーション名（短縮）	オペレーション処理概要	スパン取得レベル	スパンタグ/スパンイベント	情報説明		key（情報名）または value（情報値）		
					情報説明名	情報説明の詳細	key（情報名）	value（情報値）	key（情報名）または value（情報値）の詳細
		果を AP に通知します。		スパンタグ	－	－	branch-xid	－	(a) スパンタグ
				スパンタグ	－	－	result	－	(a) スパンタグ
				スパンタグ	－	－	prepare	－	(a) スパンタグ
				スパンタグ	－	－	thread-name	－	(a) スパンタグ
				スパンタグ	－	－	otel.library.name	－	(a) スパンタグ
				スパンタグ	－	－	otel.library.version	－	(a) スパンタグ
				スパンタグ	－	－	otel.scope.name	－	(a) スパンタグ
				スパンタグ	－	－	otel.scope.version	－	(a) スパンタグ
				スパンイベント	start:service	(b) スパンイベント	－	－	－
				スパンイベント	end:service	(b) スパンイベント	－	－	－
				スパンイベント	start:branch	(b) スパンイベント	－	－	－

コンポーネント	オペレーション名（短縮）	オペレーション処理概要	スパン取得レベル	スパンタグ/スパンイベント	情報説明		key（情報名）または value（情報値）		
					情報説明名	情報説明の詳細	key（情報名）	value（情報値）	key（情報名）または value（情報値）の詳細
				スパンイベント	end:branch	(b) スパンイベント	−	−	−
				スパンイベント	start:AP	(b) スパンイベント	−	−	−
				スパンイベント	end:AP	(b) スパンイベント	−	−	−
				スパンイベント	start:jdbc <JDBC Driverのクラス名>.<JDBC Driverのメソッド名>	(b) スパンイベント	−	−	−
				スパンイベント	end:jdbc	(b) スパンイベント	−	−	−
				スパンイベント	start:xafunction	(b) スパンイベント	xa-name	xa_start	(b) スパンイベント
				スパンイベント	end:xafunction	(b) スパンイベント	xa-name	xa_start	(b) スパンイベント
				スパンイベント	failed:xafunction	(b) スパンイベント	xa-name	xa_start	(b) スパンイベント
				スパンイベント			message	−	(b) スパンイベント

コンポーネント	オペレーション名（短縮）	オペレーション処理概要	スパン取得レベル	スパンタグ/スパンイベント	情報説明		key（情報名）または value（情報値）		
					情報説明名	情報説明の詳細	key（情報名）	value（情報値）	key（情報名）または value（情報値）の詳細
Orchestrator/Entity-Module	j.c.H.s.h.f.p.TxPropagationRestTemplateInterceptor:innerRequestFilter	APによるREST通信の送信前後に呼び出されるフィルタの処理です。	必須情報	スパンタグ	−	−	thread-name	−	(a) スパンタグ
				スパンタグ	−	−	branch-xid	−	(a) スパンタグ
				スパンタグ	−	−	http-method	−	(a) スパンタグ
				スパンタグ	−	−	http-url	−	(a) スパンタグ
				スパンタグ	−	−	routed-http-url	−	(a) スパンタグ
				スパンタグ	−	−	http-status-code	−	(a) スパンタグ
				スパンタグ	−	−	otel.library.name	−	(a) スパンタグ
				スパンタグ	−	−	otel.library.version	−	(a) スパンタグ
				スパンタグ	−	−	otel.scope.name	−	(a) スパンタグ
				スパンタグ	−	−	otel.scope.version	−	(a) スパンタグ
				スパンイベント	start:execute	(b) スパンイベントおよび(b) ス	−	−	−

コンポーネント	オペレーション名（短縮）	オペレーション処理概要	スパン取得レベル	スパンタグ/スパンイベント	情報説明		key（情報名）または value（情報値）		
					情報説明名	情報説明の詳細	key（情報名）	value（情報値）	key（情報名）または value（情報値）の詳細
						パンイベント			
				スパンイベント	end:execute	(b) スパンイベントおよび (b) スパンイベント	－	－	－
				スパンイベント	response:get-status-code	(b) スパンイベントおよび (b) スパンイベント	－	－	－
Orchestrator/Entity-Module	j.c.H.s.h.f.p.TxPropagationWebclientInterceptor:innerRequestFilter	APによるREST通信の送信前後に呼び出されるフィルタの処理（WebClient使用時）です。	必須情報	スパンタグ	－	－	thread-name	－	(a) スパンタグ
				スパンタグ	－	－	thread-receive	－	(a) スパンタグ
				スパンタグ	－	－	branch-xid	－	(a) スパンタグ
				スパンタグ	－	－	http-method	－	(a) スパンタグ
				－	－	－	http-url	－	(a) スパンタグ
				－	－	－	routed-http-url	－	(a) スパンタグ

コンポーネント	オペレーション名（短縮）	オペレーション処理概要	スパン取得レベル	スパンタグ/スパンイベント	情報説明		key（情報名）または value（情報値）		
					情報説明名	情報説明の詳細	key（情報名）	value（情報値）	key（情報名）または value（情報値）の詳細
				－	－	－	http-status-code	－	(a) スパンタグ
				－	－	－	otel.library.name	－	(a) スパンタグ
				－	－	－	otel.library.version	－	(a) スパンタグ
				－	－	－	otel.scope.name	－	(a) スパンタグ
				－	－	－	otel.scope.version	－	(a) スパンタグ
				スパンイベント	start:execute	(b) スパンイベントおよび (b) スパンイベント	－	－	－
				スパンイベント	end:execute	(b) スパンイベントおよび (b) スパンイベント	－	－	－
				スパンイベント	response:get-status-code	(b) スパンイベントおよび (b) スパンイベント	－	－	－

コンポーネント	オペレーション名（短縮）	オペレーション処理概要	スパン取得レベル	スパンタグ/スパンイベント	情報説明		key（情報名）または value（情報値）		
					情報説明名	情報説明の詳細	key（情報名）	value（情報値）	key（情報名）または value（情報値）の詳細
Orchestrator	hmppcto.mediator.TerminationReceiverService/commit	Mediator へのトランザクション決着指示要求（gRPC 送信）です。	必須情報（通信処理のため）	スパンタグ	−	−	network.peer.address	−	(a) スパンタグ
				スパンタグ	−	−	network.peer.port	−	(a) スパンタグ
				スパンタグ	−	−	network.type	−	(a) スパンタグ
				スパンタグ	−	−	rpc.grpc.status_code	−	(a) スパンタグ
				スパンタグ	−	−	rpc.method	−	(a) スパンタグ
				スパンタグ	−	−	rpc.service	−	(a) スパンタグ
				スパンタグ	−	−	rpc.system	−	(a) スパンタグ
				スパンタグ	−	−	server.address	−	(a) スパンタグ
				スパンタグ	−	−	server.port	−	(a) スパンタグ
				スパンタグ	−	−	span.kind	−	(a) スパンタグ
				スパンタグ	−	−	otel.library.name	−	(a) スパンタグ

コンポーネント	オペレーション名（短縮）	オペレーション処理概要	スパン取得レベル	スパンタグ/スパンイベント	情報説明		key（情報名）または value（情報値）		
					情報説明名	情報説明の詳細	key（情報名）	value（情報値）	key（情報名）または value（情報値）の詳細
				スパンタグ	−	−	otel.library.version	−	(a) スパンタグ
				スパンタグ	−	−	otel.scope.name	−	(a) スパンタグ
				スパンタグ	−	−	otel.scope.version	−	(a) スパンタグ
				スパンイベント	sent message(hmppcto)	(b) スパンイベント	message-type	−	(b) スパンイベント
				スパンイベント			json-convert	−	(b) スパンイベント
				スパンイベント			message-value	−	(b) スパンイベント
				スパンイベント			message-size	−	(b) スパンイベント
				スパンイベント			thread-name	−	(b) スパンイベント
				スパンイベント	message	(b) スパンイベント	message.id	−	(b) スパンイベント
				スパンイベント			message.type	SENT	(b) スパンイベント
				スパンイベント	received message(hmppcto)	(b) スパンイベント	message-type	−	(b) スパンイベント

コンポーネント	オペレーション名（短縮）	オペレーション処理概要	スパン取得レベル	スパンタグ/スパンイベント	情報説明		key（情報名）または value（情報値）		
					情報説明名	情報説明の詳細	key（情報名）	value（情報値）	key（情報名）または value（情報値）の詳細
				スパンイベント			json-convert	－	(b) スパンイベント
				スパンイベント			message-value	－	(b) スパンイベント
				スパンイベント			message-size	－	(b) スパンイベント
				スパンイベント			thread-name	－	(b) スパンイベント
				スパンイベント	message	(b) スパンイベント	message.id	－	(b) スパンイベント
				スパンイベント			message.type	RECEIVED	(b) スパンイベント
Mediator	hmppcto.mediator.TerminationReceiverService/commit	Orchestratorからのトランザクション決着指示要求（gRPC受信処理）です。	必須情報（通信処理のため）	スパンタグ	－	－	network.peer.address	－	(a) スパンタグ
				スパンタグ	－	－	network.peer.port	－	(a) スパンタグ
				スパンタグ	－	－	network.type	－	(a) スパンタグ
				スパンタグ	－	－	rpc.grpc.status_code	－	(a) スパンタグ
				スパンタグ	－	－	rpc.method	－	(a) スパンタグ

コンポーネント	オペレーション名（短縮）	オペレーション処理概要	スパン取得レベル	スパンタグ/スパンイベント	情報説明		key（情報名）または value（情報値）		
					情報説明名	情報説明の詳細	key（情報名）	value（情報値）	key（情報名）または value（情報値）の詳細
				スパンタグ	－	－	rpc.service	－	(a) スパンタグ
				スパンタグ	－	－	rpc.system	－	(a) スパンタグ
				スパンタグ	－	－	server.address	－	(a) スパンタグ
				スパンタグ	－	－	server.port	－	(a) スパンタグ
				スパンタグ	－	－	span.kind	－	(a) スパンタグ
				スパンタグ	－	－	otel.library.name	－	(a) スパンタグ
				スパンタグ	－	－	otel.library.version	－	(a) スパンタグ
				スパンタグ	－	－	otel.scope.name	－	(a) スパンタグ
				スパンタグ	－	－	otel.scope.version	－	(a) スパンタグ
				スパンイベント	sent message(hmppcto)	(b) スパンイベント	message-type	－	(b) スパンイベント
				スパンイベント			json-convert	－	(b) スパンイベント

コンポーネント	オペレーション名（短縮）	オペレーション処理概要	スパン取得レベル	スパンタグ/スパンイベント	情報説明		key（情報名）または value（情報値）		
					情報説明名	情報説明の詳細	key（情報名）	value（情報値）	key（情報名）または value（情報値）の詳細
				スパンイベント			message-value	－	(b) スパンイベント
				スパンイベント			message-size	－	(b) スパンイベント
				スパンイベント			thread-name	－	(b) スパンイベント
				スパンイベント	message	(b) スパンイベント	message.id	－	(b) スパンイベント
				スパンイベント			message.type	SENT	(b) スパンイベント
				スパンイベント	received message(hmppcto)	(b) スパンイベント	message-type	－	(b) スパンイベント
				スパンイベント			json-convert	－	(b) スパンイベント
				スパンイベント			message-value	－	(b) スパンイベント
				スパンイベント			message-size	－	(b) スパンイベント
				スパンイベント			thread-name	－	(b) スパンイベント
				スパンイベント	message	(b) スパンイベント	message.id	－	(b) スパンイベント

コンポーネント	オペレーション名（短縮）	オペレーション処理概要	スパン取得レベル	スパンタグ/スパンイベント	情報説明		key（情報名）または value（情報値）		
					情報説明名	情報説明の詳細	key（情報名）	value（情報値）	key（情報名）または value（情報値）の詳細
				スパンイベント			message.type	RECEIVED	(b) スパンイベント
Orchestrator	hmppcto.mediator.TerminationReceiverService/rollback	Mediator へのトランザクションロールバック指示要求（gRPC 送信）です。	必須情報（通信処理のため）	スパンタグ	−	−	network.peer.address	−	(a) スパンタグ
				スパンタグ	−	−	network.peer.port	−	(a) スパンタグ
				スパンタグ	−	−	network.type	−	(a) スパンタグ
				スパンタグ	−	−	rpc.grpc.status_code	−	(a) スパンタグ
				スパンタグ	−	−	rpc.method	−	(a) スパンタグ
				スパンタグ	−	−	rpc.service	−	(a) スパンタグ
				スパンタグ	−	−	rpc.system	−	(a) スパンタグ
				スパンタグ	−	−	server.address	−	(a) スパンタグ
				スパンタグ	−	−	server.port	−	(a) スパンタグ
				スパンタグ	−	−	span.kind	−	(a) スパンタグ

コンポーネント	オペレーション名（短縮）	オペレーション処理概要	スパン取得レベル	スパンタグ/スパンイベント	情報説明		key（情報名）または value（情報値）		
					情報説明名	情報説明の詳細	key（情報名）	value（情報値）	key（情報名）または value（情報値）の詳細
				スパンタグ	−	−	otel.library.name	−	(a) スパンタグ
				スパンタグ	−	−	otel.library.version	−	(a) スパンタグ
				スパンタグ	−	−	otel.scope.name	−	(a) スパンタグ
				スパンタグ	−	−	otel.scope.version	−	(a) スパンタグ
				スパンイベント	sent message(hmppcto)	(b) スパンイベント	message-type	−	(b) スパンイベント
				スパンイベント			json-convert	−	(b) スパンイベント
				スパンイベント			message-value	−	(b) スパンイベント
				スパンイベント			message-size	−	(b) スパンイベント
				スパンイベント			thread-name	−	(b) スパンイベント
				スパンイベント	message	(b) スパンイベント	message.id	−	(b) スパンイベント
				スパンイベント			message.type	SENT	(b) スパンイベント

コンポーネント	オペレーション名（短縮）	オペレーション処理概要	スパン取得レベル	スパンタグ/スパンイベント	情報説明		key（情報名）または value（情報値）		
					情報説明名	情報説明の詳細	key（情報名）	value（情報値）	key（情報名）または value（情報値）の詳細
				スパンイベント	received message(hmppcto)	(b) スパンイベント	message-type	−	(b) スパンイベント
				スパンイベント			json-convert	−	(b) スパンイベント
				スパンイベント			message-value	−	(b) スパンイベント
				スパンイベント			message-size	−	(b) スパンイベント
				スパンイベント			thread-name	−	(b) スパンイベント
				スパンイベント	message	(b) スパンイベント	message.id	−	(b) スパンイベント
				スパンイベント			message.type	RECEIVED	(b) スパンイベント
Mediator	hmppcto.mediator.TerminationReceiverService/rollback	Orchestratorからのトランザクションロールバック指示要求（gRPC 受信処理）です。	必須情報（通信処理のため）	スパンタグ	−	−	network.peer.address	−	(a) スパンタグ
				スパンタグ	−	−	network.peer.port	−	(a) スパンタグ
				スパンタグ	−	−	network.type	−	(a) スパンタグ
				スパンタグ	−	−	rpc.grpc.status_code	−	(a) スパンタグ

コンポーネント	オペレーション名（短縮）	オペレーション処理概要	スパン取得レベル	スパンタグ/スパンイベント	情報説明		key（情報名）または value（情報値）		
					情報説明名	情報説明の詳細	key（情報名）	value（情報値）	key（情報名）または value（情報値）の詳細
				スパンタグ	－	－	rpc.method	－	(a) スパンタグ
				スパンタグ	－	－	rpc.service	－	(a) スパンタグ
				スパンタグ	－	－	rpc.system	－	(a) スパンタグ
				スパンタグ	－	－	server.address	－	(a) スパンタグ
				スパンタグ	－	－	server.port	－	(a) スパンタグ
				スパンタグ	－	－	span.kind	－	(a) スパンタグ
				スパンタグ	－	－	otel.library.name	－	(a) スパンタグ
				スパンタグ	－	－	otel.library.version	－	(a) スパンタグ
				スパンタグ	－	－	otel.scope.name	－	(a) スパンタグ
				スパンタグ	－	－	otel.scope.version	－	(a) スパンタグ
				スパンイベント	sent message(hmppcto)	(b) スパンイベント	message-type	－	(b) スパンイベント

コンポーネント	オペレーション名（短縮）	オペレーション処理概要	スパン取得レベル	スパンタグ/スパンイベント	情報説明		key（情報名）または value（情報値）		
					情報説明名	情報説明の詳細	key（情報名）	value（情報値）	key（情報名）または value（情報値）の詳細
				スパンイベント			json-convert	－	(b) スパンイベント
				スパンイベント			message-value	－	(b) スパンイベント
				スパンイベント			message-size	－	(b) スパンイベント
				スパンイベント			thread-name	－	(b) スパンイベント
				スパンイベント	message	(b) スパンイベント	message.id	－	(b) スパンイベント
				スパンイベント			message.type	SENT	(b) スパンイベント
				スパンイベント	received message(hmppcto)	(b) スパンイベント	message-type	－	(b) スパンイベント
				スパンイベント			json-convert	－	(b) スパンイベント
				スパンイベント			message-value	－	(b) スパンイベント
				スパンイベント			message-size	－	(b) スパンイベント
				スパンイベント			thread-name	－	(b) スパンイベント

コンポーネント	オペレーション名（短縮）	オペレーション処理概要	スパン取得レベル	スパンタグ/スパンイベント	情報説明		key（情報名）または value（情報値）		
					情報説明名	情報説明の詳細	key（情報名）	value（情報値）	key（情報名）または value（情報値）の詳細
				スパンイベント	message	(b) スパンイベント	message.id	−	(b) スパンイベント
				スパンイベント			message.type	RECEIVED	(b) スパンイベント
Entity-Module	Recv:j.c.H.s.h.f.TxServletFilter:innerIntercept	リクエスト受信時に実行する処理です。トランザクションを伝搬し、トランザクションブランチを開始します。AP の実行結果からトランザクションブランチをロールバックするかを決定し、Orchestrator に応答を返します。	必須情報	スパンタグ	−	−	branch-xid	−	(a) スパンタグ
				スパンタグ	−	−	thread-name	−	(a) スパンタグ
				スパンタグ	−	−	result	−	(a) スパンタグ
				スパンタグ	−	−	prepare	−	(a) スパンタグ
				スパンタグ	−	−	otel.library.name	−	(a) スパンタグ
				スパンタグ	−	−	otel.library.version	−	(a) スパンタグ
				スパンタグ	−	−	otel.scope.name	−	(a) スパンタグ
				スパンタグ	−	−	otel.scope.version	−	(a) スパンタグ
				スパンイベント	start:service	(b) スパンイベント	−	−	−

コンポーネント	オペレーション名（短縮）	オペレーション処理概要	スパン取得レベル	スパンタグ/スパンイベント	情報説明		key（情報名）または value（情報値）		
					情報説明名	情報説明の詳細	key（情報名）	value（情報値）	key（情報名）または value（情報値）の詳細
				スパンイベント	end:service	(b) スパンイベント	－	－	－
				スパンイベント	start:try	(b) スパンイベント	－	－	－
				スパンイベント	end:try	(b) スパンイベント	－	－	－
				スパンイベント	start:AP	(b) スパンイベント	－	－	－
				スパンイベント	end:AP	(b) スパンイベント	－	－	－
				スパンイベント	start:jdbc	(b) スパンイベント	－	－	－
				スパンイベント	end:jdbc	(b) スパンイベント	－	－	－
				スパンイベント	start:termination-judge	(b) スパンイベント	－	－	－
				スパンイベント	end:termination-judge	(b) スパンイベント	－	－	－
				スパンイベント	start:TP1-Converter	(b) スパンイベント	－	－	－
				スパンイベント	end:TP1-Converter	(b) スパンイベント	－	－	－

コンポーネント	オペレーション名（短縮）	オペレーション処理概要	スパン取得レベル	スパンタグ/スパンイベント	情報説明		key（情報名）または value（情報値）		
					情報説明名	情報説明の詳細	key（情報名）	value（情報値）	key（情報名）または value（情報値）の詳細
				スパンイベント	start:TP1/Client/J(<TP1/Client/Jのメソッド名>)	(b) スパンイベント	−	−	−
				スパンイベント	end:TP1/Client/J(<TP1/Client/Jのメソッド名>)	(b) スパンイベント	−	−	−
				スパンイベント	start:xa function	(b) スパンイベント	xa-name	xa_start	(b) スパンイベント
				スパンイベント	end:xa function	(b) スパンイベント	xa-name	xa_start	(b) スパンイベント
				スパンイベント	failed:xa function	(b) スパンイベント	xa-name	xa_start	(b) スパンイベント
				スパンイベント			message	−	(b) スパンイベント
Participant	j.c.H.s.h.p.s.GrpcTerminationJudgeRequestSender	TerminationJudge の DB 更新可否を Mediator に投票します。	デバッグ情報	スパンタグ	−	−	thread-name	−	(a) スパンタグ
				スパンタグ	−	−	root-xid	−	(a) スパンタグ

コンポーネント	オペレーション名（短縮）	オペレーション処理概要	スパン取得レベル	スパンタグ/スパンイベント	情報説明		key（情報名）または value（情報値）		
					情報説明名	情報説明の詳細	key（情報名）	value（情報値）	key（情報名）または value（情報値）の詳細
				スパンタグ	－	－	branch-xid	－	(a) スパンタグ
				スパンタグ	－	－	mediator-id	－	(a) スパンタグ
				スパンタグ	－	－	proposer-id	－	(a) スパンタグ
				スパンタグ	－	－	propose-values	－	(a) スパンタグ
				スパンタグ	－	－	otel.library.name	－	(a) スパンタグ
				スパンタグ	－	－	otel.library.version	－	(a) スパンタグ
				スパンタグ	－	－	otel.scope.name	－	(a) スパンタグ
				スパンタグ	－	－	otel.scope.version	－	(a) スパンタグ
Mediator	j.c.H.s.h.m.j.TerminationJudgeImpl$AsyncAcceptRequestTask:run	Participant からの投票内容を集計し ConsensusLog へ書き込みます。Orchestrator からの仕掛中トランザクションのキャンセル要求によって Rollback として	必須情報（プロポーザルログに関するため）	スパンタグ	－	－	thread-name	－	(a) スパンタグ
				スパンタグ	－	－	branch-xid	－	(a) スパンタグ
				スパンタグ	－	－	otel.library.name	－	(a) スパンタグ

コンポーネント	オペレーション名（短縮）	オペレーション処理概要	スパン取得レベル	スパンタグ/スパンイベント	情報説明		key（情報名）または value（情報値）		
					情報説明名	情報説明の詳細	key（情報名）	value（情報値）	key（情報名）または value（情報値）の詳細
		ConsensusLog へ書き込みます。		スパンタグ	−	−	otel.library.version	−	(a) スパンタグ
				スパンタグ	−	−	otel.scope.name	−	(a) スパンタグ
				スパンタグ	−	−	otel.scope.version	−	(a) スパンタグ
				スパンイベント	start:proposallog-add-propose	(b) スパンイベント	propose-values	−	(b) スパンイベント
				スパンイベント	end:proposallog-add-propose	(b) スパンイベント	−	−	−
				スパンイベント	start:proposallog-add-propose-participant	(b) スパンイベント	participants-value	−	(b) スパンイベント
				スパンイベント	end:proposallog-add-propose-participant	(b) スパンイベント	−	−	−

コンポーネント	オペレーション名（短縮）	オペレーション処理概要	スパン取得レベル	スパンタグ/スパンイベント	情報説明		key（情報名）または value（情報値）		
					情報説明名	情報説明の詳細	key（情報名）	value（情報値）	key（情報名）または value（情報値）の詳細
Mediator	j.c.H.s.h.m.j.TerminationJudgeImpl$AsyncTerminationRequestTask:run	DB 更新可否を決着します。	デバッグ情報	スパンタグ	－	－	thread-name	－	(a) スパンタグ
				スパンタグ	－	－	root-xid	－	(a) スパンタグ
				スパンタグ	－	－	prepare-priority	－	(a) スパンタグ
				スパンタグ	－	－	otel.library.name	－	(a) スパンタグ
				スパンタグ	－	－	otel.library.version	－	(a) スパンタグ
				スパンタグ	－	－	otel.scope.name	－	(a) スパンタグ
				スパンタグ	－	－	otel.scope.version	－	(a) スパンタグ
				スパンイベント	start:proposallog-add-all-participants	(b) スパンイベント	all-participants	－	(b) スパンイベント
				スパンイベント	end:proposallog-add-all-participants	(b) スパンイベント	－	－	－

コンポーネント	オペレーション名（短縮）	オペレーション処理概要	スパン取得レベル	スパンタグ/スパンイベント	情報説明		key（情報名）または value（情報値）		
					情報説明名	情報説明の詳細	key（情報名）	value（情報値）	key（情報名）または value（情報値）の詳細
Participant	j.c.H.s.h.p.c.o.ConsensusJudgeObserverImpl:observe	Consensus Learner の合意処理を実施します。	必須情報（補助合意結果）	スパンタグ	−	−	thread-name	−	(a) スパンタグ
				スパンタグ	−	−	otel.library.name	−	(a) スパンタグ
				スパンタグ	−	−	otel.library.version	−	(a) スパンタグ
				スパンタグ	−	−	otel.scope.name	−	(a) スパンタグ
				スパンタグ	−	−	otel.scope.version	−	(a) スパンタグ
				スパンイベント	learning log size m/n	(b) スパンイベント	−	−	−
				スパンイベント	fast consensus	(b) スパンイベント	−	−	−
				スパンイベント	skip consensus	(b) スパンイベント	−	−	−
				スパンイベント	external consensus	(b) スパンイベント	−	−	−
Participant	j.c.H.s.h.l.t.ConsensusLearningTimeoutJob	補助合意試行間隔タイマ処理です。	必須情報（タイマ発火）	スパンタグ	−	−	thread-name	−	(a) スパンタグ
				スパンタグ	−	−	otel.library.name	−	(a) スパンタグ

コンポーネント	オペレーション名（短縮）	オペレーション処理概要	スパン取得レベル	スパンタグ/スパンイベント	情報説明		key（情報名）または value（情報値）		
					情報説明名	情報説明の詳細	key（情報名）	value（情報値）	key（情報名）または value（情報値）の詳細
				スパンタグ	−	−	otel.library.version	−	(a) スパンタグ
				スパンタグ	−	−	otel.scope.name	−	(a) スパンタグ
				スパンタグ	−	−	otel.scope.version	−	(a) スパンタグ
				スパンイベント	consensus finished	(b) スパンイベント	−	−	−
				スパンイベント	learning log size m/n	(b) スパンイベント	−	−	−
				スパンイベント	fast consensus	(b) スパンイベント	−	−	−
				スパンイベント	skip consensus	(b) スパンイベント	−	−	−
				スパンイベント	external consensus	(b) スパンイベント	−	−	−
				スパンイベント	The consensus has not been terminated for a long time.	(b) スパンイベント	−	−	−

コンポーネント	オペレーション名（短縮）	オペレーション処理概要	スパン取得レベル	スパンタグ/スパンイベント	情報説明		key（情報名）または value（情報値）		
					情報説明名	情報説明の詳細	key（情報名）	value（情報値）	key（情報名）または value（情報値）の詳細
SQL-Participant/TP1-Bridge	j.c.H.s.h.t.o.OscProposalController#terminate:observe	合意結果に基づきトランザクションを決着します（commit/rollback）。	デバッグ情報	スパンタグ	−	−	thread-name	−	(a) スパンタグ
				スパンタグ	−	−	otel.library.name	−	(a) スパンタグ
				スパンタグ	−	−	otel.library.version	−	(a) スパンタグ
				スパンタグ	−	−	otel.scope.name	−	(a) スパンタグ
				スパンタグ	−	−	otel.scope.version	−	(a) スパンタグ
Participant	j.c.H.s.h.p.s.GrpcNotifyTerminationResultRequestSender	Mediator へ合意結果を送信します。	必須情報	スパンタグ	−	−	thread-name	−	(a) スパンタグ
				スパンタグ	−	−	root-xid	−	(a) スパンタグ
				スパンタグ	−	−	mediator-id	−	(a) スパンタグ
				スパンタグ	−	−	proposer-id	−	(a) スパンタグ
				スパンタグ	−	−	consensus-value	−	(a) スパンタグ
				スパンタグ	−	−	otel.library.name	−	(a) スパンタグ

コンポーネント	オペレーション名（短縮）	オペレーション処理概要	スパン取得レベル	スパンタグ/スパンイベント	情報説明		key（情報名）または value（情報値）		
					情報説明名	情報説明の詳細	key（情報名）	value（情報値）	key（情報名）または value（情報値）の詳細
				スパンタグ	－	－	otel.library.version	－	(a) スパンタグ
				スパンタグ	－	－	otel.scope.name	－	(a) スパンタグ
				スパンタグ	－	－	otel.scope.version	－	(a) スパンタグ
SQL-Participant/TP1-Bridge	j.c.H.s.h.t.o.t.TrnTerminationTimeoutJob:callObserver	トランザクション決着監視タイマ処理です。	必須情報（タイマ発火）	スパンタグ	－	－	thread-name	－	(a) スパンタグ
				スパンタグ	－	－	otel.library.name	－	(a) スパンタグ
				スパンタグ	－	－	otel.library.version	－	(a) スパンタグ
				スパンタグ	－	－	otel.scope.name	－	(a) スパンタグ
				スパンタグ	－	－	otel.scope.version	－	(a) スパンタグ
SQL-Participant/TP1-Bridge	j.c.H.s.h.t.o.t.TrnTerminationReplayTimeoutJob:callObserver	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処理です。	必須情報（タイマ発火）	スパンタグ	－	－	thread-name	－	(a) スパンタグ
				スパンタグ	－	－	otel.library.name	－	(a) スパンタグ
				スパンタグ	－	－	otel.library.version	－	(a) スパンタグ

コンポーネント	オペレーション名（短縮）	オペレーション処理概要	スパン取得レベル	スパンタグ/スパンイベント	情報説明		key（情報名）または value（情報値）		
					情報説明名	情報説明の詳細	key（情報名）	value（情報値）	key（情報名）または value（情報値）の詳細
				スパンタグ	－	－	otel.scope.name	－	(a) スパンタグ
				スパンタグ	－	－	otel.scope.version	－	(a) スパンタグ
Participant	j.c.H.s.h.p.c.o.TerminationProposalObserverImpl:observe	再投票処理を実施します。	必須情報	スパンタグ	－	－	thread-name	－	(a) スパンタグ
				スパンタグ	－	－	branch-xid	－	(a) スパンタグ
				スパンタグ	－	－	otel.library.name	－	(a) スパンタグ
				スパンタグ	－	－	otel.library.version	－	(a) スパンタグ
				スパンタグ	－	－	otel.scope.name	－	(a) スパンタグ
				スパンタグ	－	－	otel.scope.version	－	(a) スパンタグ
				スパンイベント	start:termination-judge	(b) スパンイベント	－	－	－
				スパンイベント	end:termination-judge	(b) スパンイベント	－	－	－
SQL-Participant/TP1-Bridge	j.c.H.s.h.t.o.OscInitializer:innerInit	Participant の初期化時のメソッドです。 Prepare 済みのトランザクショ	必須情報（DBアクセス（XA））	スパンタグ	－	－	thread-name	－	(a) スパンタグ

コンポーネント	オペレーション名（短縮）	オペレーション処理概要	スパン取得レベル	スパンタグ/スパンイベント	情報説明		key（情報名）または value（情報値）		
					情報説明名	情報説明の詳細	key（情報名）	value（情報値）	key（情報名）または value（情報値）の詳細
		ンの XID を取得します。		スパンタグ	－	－	otel.library.name	－	(a) スパンタグ
				スパンタグ	－	－	otel.library.version	－	(a) スパンタグ
				スパンタグ	－	－	otel.scope.name	－	(a) スパンタグ
				スパンタグ	－	－	otel.scope.version	－	(a) スパンタグ
				スパンイベント	start:xa function	(b) スパンイベント	xa-name	xa_recover	(b) スパンイベント
				スパンイベント	end:xa function	(b) スパンイベント	xa-name	xa_recover	(b) スパンイベント
				スパンイベント	failed:xa function	(b) スパンイベント	xa-name	xa_recover	(b) スパンイベント
				スパンイベント			message	－	(b) スパンイベント
Participant	j.c.H.s.h.l.ConsensusLearningProcess:executeExtConsToEADS	分散合意での補助合意時に EADS API を呼び出します。	必須情報（EADS との通信のため）	スパンタグ	－	－	thread-name	－	(a) スパンタグ
				スパンタグ	－	－	root-xid	－	(a) スパンタグ
				スパンタグ	－	－	otel.library.name	－	(a) スパンタグ

コンポーネント	オペレーション名（短縮）	オペレーション処理概要	スパン取得レベル	スパンタグ/スパンイベント	情報説明		key（情報名）または value（情報値）		
					情報説明名	情報説明の詳細	key（情報名）	value（情報値）	key（情報名）または value（情報値）の詳細
				スパンタグ	−	−	otel.library.version	−	(a) スパンタグ
				スパンタグ	−	−	otel.scope.name	−	(a) スパンタグ
				スパンタグ	−	−	otel.scope.version	−	(a) スパンタグ
				スパンイベント	start:EADS API	(b) スパンイベント	API-name	create	(b) スパンイベント
				スパンイベント			consensus	−	(b) スパンイベント
				スパンイベント	end:EADS API	(b) スパンイベント	API-name	create	(b) スパンイベント
				スパンイベント	failed:EADS API	(b) スパンイベント	API-name	create	(b) スパンイベント
				スパンイベント			message	−	(b) スパンイベント
				スパンイベント	start:EADS API	(b) スパンイベント	API-name	get	(b) スパンイベント
				スパンイベント	end:EADS API	(b) スパンイベント	API-name	get	(b) スパンイベント
				スパンイベント			get	−	(b) スパンイベント

コンポーネント	オペレーション名（短縮）	オペレーション処理概要	スパン取得レベル	スパンタグ/スパンイベント	情報説明		key（情報名）または value（情報値）		
					情報説明名	情報説明の詳細	key（情報名）	value（情報値）	key（情報名）または value（情報値）の詳細
				スパンイベント	failed:EADS API	(b) スパンイベント	API-name	get	(b) スパンイベント
				スパンイベント			message	例外の詳細メッセージ文字列	(b) スパンイベント
SQL-Participant/TP1-Bridge	j.c.H.s.h.t.o.s.RmProposalStateReducer$ProposalStateReducerProcess:prepareBranch	Participant のトランザクションステータスの状態遷移（投票送信に係る指示）です。	必須情報（DBアクセス（XA））	スパンタグ	−	−	thread-name	−	(a) スパンタグ
				スパンタグ	−	−	branch-xid	−	(a) スパンタグ
				スパンタグ	−	−	before-transaction-status	−	(a) スパンタグ
				スパンタグ	−	−	after-transaction-status	−	(a) スパンタグ
				スパンタグ	−	−	otel.library.name	−	(a) スパンタグ
				スパンタグ	−	−	otel.library.version	−	(a) スパンタグ
				スパンタグ	−	−	otel.scope.name	−	(a) スパンタグ
				スパンタグ	−	−	otel.scope.version	−	(a) スパンタグ
				スパンイベント	start:xa function	(b) スパンイベント	xa-name	xa_end	(b) スパンイベント

コンポーネント	オペレーション名（短縮）	オペレーション処理概要	スパン取得レベル	スパンタグ/スパンイベント	情報説明		key（情報名）または value（情報値）		
					情報説明名	情報説明の詳細	key（情報名）	value（情報値）	key（情報名）または value（情報値）の詳細
				スパンイベント	end:xa function	(b) スパンイベント	xa-name	xa_end	(b) スパンイベント
				スパンイベント	failed:xa function	(b) スパンイベント	xa-name	xa_end	(b) スパンイベント
				スパンイベント			message	−	(b) スパンイベント
				スパンイベント	start:xa function	(b) スパンイベント	xa-name	xa_prepare	(b) スパンイベント
				スパンイベント	end:xa function	(b) スパンイベント	xa-name	xa_prepare	(b) スパンイベント
				スパンイベント	failed:xa function	(b) スパンイベント	xa-name	xa_prepare	(b) スパンイベント
				スパンイベント			message	−	(b) スパンイベント
SQL-Participant/TP1-Bridge	j.c.H.s.h.t.o.s.RmTerminationStateReducer$TerminationStateReducerProcess:innerReduce	Participant のトランザクションステータスの状態遷移（RDBMS への commit/rollback の実行）です。	必須情報（DB アクセス（XA））	スパンタグ	−	−	thread-name	−	(a) スパンタグ
				スパンタグ	−	−	branch-xid	−	(a) スパンタグ
				スパンタグ	−	−	before-transaction-status	−	(a) スパンタグ
				スパンタグ	−	−	after-transaction-status	−	(a) スパンタグ

コンポーネント	オペレーション名（短縮）	オペレーション処理概要	スパン取得レベル	スパンタグ/スパンイベント	情報説明		key（情報名）または value（情報値）		
					情報説明名	情報説明の詳細	key（情報名）	value（情報値）	key（情報名）または value（情報値）の詳細
				スパンタグ	−	−	otel.library.name	−	(a) スパンタグ
				スパンタグ	−	−	otel.library.version	−	(a) スパンタグ
				スパンタグ	−	−	otel.scope.name	−	(a) スパンタグ
				スパンタグ	−	−	otel.scope.version	−	(a) スパンタグ
				スパンイベント	start:xa function	(b) スパンイベント	xa-name	xa_end	(b) スパンイベント
				スパンイベント	end:xa function	(b) スパンイベント	xa-name	xa_end	(b) スパンイベント
				スパンイベント	failed:xa function	(b) スパンイベント	xa-name	xa_end	(b) スパンイベント
				スパンイベント			message	−	(b) スパンイベント
				スパンイベント	start:xa function	(b) スパンイベント	xa-name	xa_commit	(b) スパンイベント
				スパンイベント	end:xa function	(b) スパンイベント	xa-name	xa_commit	(b) スパンイベント
				スパンイベント	failed:xa function	(b) スパンイベント	xa-name	xa_commit	(b) スパンイベント

コンポーネント	オペレーション名（短縮）	オペレーション処理概要	スパン取得レベル	スパンタグ/スパンイベント	情報説明		key（情報名）または value（情報値）		
					情報説明名	情報説明の詳細	key（情報名）	value（情報値）	key（情報名）または value（情報値）の詳細
				スパンイベント			message	－	(b) スパンイベント
				スパンイベント	start:xa function	(b) スパンイベント	xa-name	xa_rollback	(b) スパンイベント
				スパンイベント	end:xa function	(b) スパンイベント	xa-name	xa_rollback	(b) スパンイベント
				スパンイベント	failed:xa function	(b) スパンイベント	xa-name	xa_rollback	(b) スパンイベント
				スパンイベント			message	－	(b) スパンイベント
Mediator	j.c.H.s.h.m.a.i.d.DirectAccessConsensuslogReaderWriter:writeDirectIOConsensuslog	ConsensusLog ファイルへのアクセス（I/O）です。	必須情報（ファイル I/O のため）	スパンタグ	－	－	thread-name	－	(a) スパンタグ
				スパンタグ	－	－	otel.library.name	－	(a) スパンタグ
				スパンタグ	－	－	otel.library.version	－	(a) スパンタグ
				スパンタグ	－	－	otel.scope.name	－	(a) スパンタグ
				スパンタグ	－	－	otel.scope.version	－	(a) スパンタグ
				スパンイベント	start:consensuslog	(b) スパンイベント	－	－	－

コンポーネント	オペレーション名（短縮）	オペレーション処理概要	スパン取得レベル	スパンタグ/スパンイベント	情報説明		key（情報名）または value（情報値）		
					情報説明名	情報説明の詳細	key（情報名）	value（情報値）	key（情報名）または value（情報値）の詳細
				スパンイベント	end:consensuslog <追加情報>	(b) スパンイベント	−	−	−
				スパンイベント	consensusid:<ConsensusIdの値>	(b) スパンイベント	−	−	−
Mediator	j.c.H.s.h.m.a.i.d.DirectAccessConsensuslogReaderWriter:removeDirectIOConsensuslog	ConsensusLogファイルへのアクセス（I/O）です。	必須情報（ファイルI/Oのため）	スパンタグ	−	−	thread-name	−	(a) スパンタグ
				スパンタグ	−	−	otel.library.name	−	(a) スパンタグ
				スパンタグ	−	−	otel.library.version	−	(a) スパンタグ
				スパンタグ	−	−	otel.scope.name	−	(a) スパンタグ
				スパンタグ	−	−	otel.scope.version	−	(a) スパンタグ
				スパンイベント	start:consensuslog	(b) スパンイベント	−	−	−
				スパンイベント	end:consensuslog <追加情報>	(b) スパンイベント	−	−	−

コンポーネント	オペレーション名（短縮）	オペレーション処理概要	スパン取得レベル	スパンタグ/スパンイベント	情報説明		key（情報名）または value（情報値）		
					情報説明名	情報説明の詳細	key（情報名）	value（情報値）	key（情報名）または value（情報値）の詳細
				スパンイベント	consensusid:<ConsensusIdの値>	(b) スパンイベント	−	−	−
Mediator	j.c.H.s.h.m.a.i.d.DirectAccessConsensuslogReaderWriter:readDirectIOConsensuslog	ConsensusLogファイルへのアクセス（I/O）です。	必須情報（ファイル I/O のため）	スパンタグ	−	−	thread-name	−	(a) スパンタグ
				スパンタグ	−	−	otel.library.name	−	(a) スパンタグ
				スパンタグ	−	−	otel.library.version	−	(a) スパンタグ
				スパンタグ	−	−	otel.scope.name	−	(a) スパンタグ
				スパンタグ	−	−	otel.scope.version	−	(a) スパンタグ
				スパンイベント	start:consensuslog	(b) スパンイベント	−	−	−
				スパンイベント	end:consensuslog	(b) スパンイベント	−	−	−
				スパンイベント	consensusid:<ConsensusIdの値>	(b) スパンイベント	−	−	−
Mediator	j.c.H.s.h.m.a.i.d.DirectAccessConsensuslogR	ConsensusLogファイルへのアクセス（I/O）です。	必須情報（ファイ	スパンタグ	−	−	thread-name	−	(a) スパンタグ

コンポーネント	オペレーション名（短縮）	オペレーション処理概要	スパン取得レベル	スパンタグ/スパンイベント	情報説明		key（情報名）または value（情報値）		
					情報説明名	情報説明の詳細	key（情報名）	value（情報値）	key（情報名）または value（情報値）の詳細
	eaderWriter:updateDirectIOConsensuslog		ル I/O のため）	スパンタグ	－	－	otel.library.name	－	(a) スパンタグ
				スパンタグ	－	－	otel.library.version	－	(a) スパンタグ
				スパンタグ	－	－	otel.scope.name	－	(a) スパンタグ
				スパンタグ	－	－	otel.scope.version	－	(a) スパンタグ
				スパンイベント	start:consensuslog	(b) スパンイベント	－	－	－
				スパンイベント	end:consensuslog	(b) スパンイベント	－	－	－
				スパンイベント	consensusid:<ConsensusIdの値>	(b) スパンイベント	－	－	－
Participant	j.c.H.s.h.p.c.o.TerminatedTransactionNotify:sendTransactionNotify	Mediator へのトランザクション完了通知です。	デバッグ情報	スパンタグ	－	－	thread-name	－	(a) スパンタグ
				スパンタグ	－	－	otel.library.name	－	(a) スパンタグ
				スパンタグ	－	－	otel.library.version	－	(a) スパンタグ
				スパンタグ	－	－	otel.scope.name	－	(a) スパンタグ

コンポーネント	オペレーション名（短縮）	オペレーション処理概要	スパン取得レベル	スパンタグ/スパンイベント	情報説明		key（情報名）または value（情報値）		
					情報説明名	情報説明の詳細	key（情報名）	value（情報値）	key（情報名）または value（情報値）の詳細
				スパンタグ	－	－	otel.scope.version	－	(a) スパンタグ
Participant	j.c.H.s.h.p.s.GrpcTerminatedTransactionNotifyRequestSender	Mediator へのトランザクション完了通知です。	デバッグ情報	スパンタグ	－	－	thread-name	－	(a) スパンタグ
				スパンタグ	－	－	root-xid	－	(a) スパンタグ
				スパンタグ	－	－	mediator-id	－	(a) スパンタグ
				スパンタグ	－	－	proposer-id	－	(a) スパンタグ
				スパンタグ	－	－	otel.library.name	－	(a) スパンタグ
				スパンタグ	－	－	otel.library.version	－	(a) スパンタグ
				スパンタグ	－	－	otel.scope.name	－	(a) スパンタグ
				スパンタグ	－	－	otel.scope.version	－	(a) スパンタグ
Mediator	j.c.H.s.h.m.j.TerminationJudgeImpl$AsyncRemoveRequestTask:run	トランザクション完了通知を受け、Proposal Log の削除判定または削除を実行します。	デバッグ情報	スパンタグ	－	－	thread-name	－	(a) スパンタグ
				スパンタグ	－	－	otel.library.name	－	(a) スパンタグ

コンポーネント	オペレーション名（短縮）	オペレーション処理概要	スパン取得レベル	スパンタグ/スパンイベント	情報説明		key（情報名）または value（情報値）		
					情報説明名	情報説明の詳細	key（情報名）	value（情報値）	key（情報名）または value（情報値）の詳細
				スパンタグ	－	－	otel.library.version	－	(a) スパンタグ
				スパンタグ	－	－	otel.scope.name	－	(a) スパンタグ
				スパンタグ	－	－	otel.scope.version	－	(a) スパンタグ
				スパンイベント	start:proposallog-delete	(b) スパンイベント	proposal-log-before-size	－	(b) スパンイベント
				スパンイベント	end:proposallog-delete	(b) スパンイベント	proposal-log-after-size	－	(b) スパンイベント
Mediator	j.c.H.s.h.m.j.TerminationJudgeImpl$AsyncTerminatedTransactionVerifyRequestTask:run	Participant へトランザクション完了確認通知を送信します。	デバッグ情報	スパンタグ	－	－	thread-name	－	(a) スパンタグ
				スパンタグ	－	－	otel.library.name	－	(a) スパンタグ
				スパンタグ	－	－	otel.library.version	－	(a) スパンタグ
				スパンタグ	－	－	otel.scope.name	－	(a) スパンタグ
				スパンタグ	－	－	otel.scope.version	－	(a) スパンタグ
Mediator	j.c.H.s.h.m.a.ConsensusAccepto	Participant へトランザクショ	デバッグ情報	スパンタグ	－	－	thread-name	－	(a) スパンタグ

コンポーネント	オペレーション名（短縮）	オペレーション処理概要	スパン取得レベル	スパンタグ/スパンイベント	情報説明		key（情報名）または value（情報値）		
					情報説明名	情報説明の詳細	key（情報名）	value（情報値）	key（情報名）または value（情報値）の詳細
	rImpl:terminatedTransactionVerify	ン完了確認通知を送信します。		スパンタグ	－	－	otel.library.name	－	(a) スパンタグ
				スパンタグ	－	－	otel.library.version	－	(a) スパンタグ
				スパンタグ	－	－	otel.scope.name	－	(a) スパンタグ
				スパンタグ	－	－	otel.scope.version	－	(a) スパンタグ
				スパンイベント	start:terminated-transaction-verify	(b) スパンイベント	consensus log size	－	(b) スパンイベント
				スパンイベント	end:terminated-transaction-verify	(b) スパンイベント	－	－	－
Participant	j.c.H.s.h.p.c.o.TerminatedTransactionVerifyNotifyObserverImpl:observe	Mediator からのトランザクション完了確認通知の受信です。	デバッグ情報	スパンタグ	－	－	thread-name	－	(a) スパンタグ
				スパンタグ	－	－	otel.library.name	－	(a) スパンタグ
				スパンタグ	－	－	otel.library.version	－	(a) スパンタグ

コンポーネント	オペレーション名（短縮）	オペレーション処理概要	スパン取得レベル	スパンタグ/スパンイベント	情報説明		key（情報名）または value（情報値）		
					情報説明名	情報説明の詳細	key（情報名）	value（情報値）	key（情報名）または value（情報値）の詳細
				スパンタグ	－	－	otel.scope.name	－	(a) スパンタグ
				スパンタグ	－	－	otel.scope.version	－	(a) スパンタグ
TCC-Participant	j.c.H.s.h.t.o.t.TransactionTerminationTimeoutJob	トランザクション決着監視タイマ処理です。	必須情報（タイマ発火）	スパンタグ	－	－	thread-name	－	(a) スパンタグ
				スパンタグ	－	－	otel.library.name	－	(a) スパンタグ
				スパンタグ	－	－	otel.library.version	－	(a) スパンタグ
				スパンタグ	－	－	otel.scope.name	－	(a) スパンタグ
				スパンタグ	－	－	otel.scope.version	－	(a) スパンタグ
				スパンイベント	start:termination-judge	(b) スパンイベント	－	－	－
				スパンイベント	end:termination-judge	(b) スパンイベント	－	－	－
TCC-Participant	j.c.H.s.h.t.o.t.ConsensusIntervalTimeoutJob	分散合意リトライインターバルタイマ処理です。	必須情報（タイマ発火）	スパンタグ	－	－	thread-name	－	(a) スパンタグ
				スパンタグ	－	－	otel.library.name	－	(a) スパンタグ

コンポーネント	オペレーション名（短縮）	オペレーション処理概要	スパン取得レベル	スパンタグ/スパンイベント	情報説明		key（情報名）または value（情報値）		
					情報説明名	情報説明の詳細	key（情報名）	value（情報値）	key（情報名）または value（情報値）の詳細
				スパンタグ	－	－	otel.library.version	－	(a) スパンタグ
				スパンタグ	－	－	otel.scope.name	－	(a) スパンタグ
				スパンタグ	－	－	otel.scope.version	－	(a) スパンタグ
				スパンイベント	start:termination-judge	(b) スパンイベント	－	－	－
				スパンイベント	end:termination-judge	(b) スパンイベント	－	－	－
Mediator	j.c.H.s.h.m.a.ConsensusAcceptorImpl:accept	ConsensusAcceptor への更新可否の決着結果の受信です。	必須情報（プロポーザルログに関するため）	スパンタグ	－	－	thread-name	－	(a) スパンタグ
				スパンタグ	－	－	root-xid	－	(a) スパンタグ
				スパンタグ	－	－	consensus-value	－	(a) スパンタグ
				スパンタグ	－	－	otel.library.name	－	(a) スパンタグ
				スパンタグ	－	－	otel.library.version	－	(a) スパンタグ
				スパンタグ	－	－	otel.scope.name	－	(a) スパンタグ

コンポーネント	オペレーション名（短縮）	オペレーション処理概要	スパン取得レベル	スパンタグ/スパンイベント	情報説明		key（情報名）または value（情報値）		
					情報説明名	情報説明の詳細	key（情報名）	value（情報値）	key（情報名）または value（情報値）の詳細
				スパンタグ	－	－	otel.scope.version	－	(a) スパンタグ
				スパンイベント	send-consensus-value:<更新可否の決着結果>	(b) スパンイベント	－	－	－
TCC-Participant	j.c.H.s.h.p.t.i.TccOscController:terminate	分散合意結果通知送信処理です。	必須情報	スパンタグ	－	－	thread-name	－	(a) スパンタグ
				スパンタグ	－	－	otel.library.name	－	(a) スパンタグ
				スパンタグ	－	－	otel.library.version	－	(a) スパンタグ
				スパンタグ	－	－	otel.scope.name	－	(a) スパンタグ
				スパンタグ	－	－	otel.scope.version	－	(a) スパンタグ
Orchestrator	hmppcto.mediator.TerminationReceiverService/recover	Mediator へのトランザクションキャンセル要求（gRPC 送信）です。	必須情報（通信処理のため）	スパンタグ	－	－	network.peer.address	－	(a) スパンタグ
				スパンタグ	－	－	network.peer.port	－	(a) スパンタグ
				スパンタグ	－	－	network.type	－	(a) スパンタグ

コンポーネント	オペレーション名（短縮）	オペレーション処理概要	スパン取得レベル	スパンタグ/スパンイベント	情報説明		key（情報名）または value（情報値）		
					情報説明名	情報説明の詳細	key（情報名）	value（情報値）	key（情報名）または value（情報値）の詳細
				スパンタグ	－	－	rpc.grpc.status_code	－	(a) スパンタグ
				スパンタグ	－	－	rpc.method	－	(a) スパンタグ
				スパンタグ	－	－	rpc.service	－	(a) スパンタグ
				スパンタグ	－	－	rpc.system	－	(a) スパンタグ
				スパンタグ	－	－	server.address	－	(a) スパンタグ
				スパンタグ	－	－	server.port	－	(a) スパンタグ
				スパンタグ	－	－	span.kind	－	(a) スパンタグ
				スパンタグ	－	－	otel.library.name	－	(a) スパンタグ
				スパンタグ	－	－	otel.library.version	－	(a) スパンタグ
				スパンタグ	－	－	otel.scope.name	－	(a) スパンタグ
				スパンタグ	－	－	otel.scope.version	－	(a) スパンタグ

コンポーネント	オペレーション名（短縮）	オペレーション処理概要	スパン取得レベル	スパンタグ/スパンイベント	情報説明		key（情報名）または value（情報値）		
					情報説明名	情報説明の詳細	key（情報名）	value（情報値）	key（情報名）または value（情報値）の詳細
				スパンイベント	sent message(hmppcto)	(b) スパンイベント	message-type	－	(b) スパンイベント
				スパンイベント			json-convert	－	(b) スパンイベント
				スパンイベント			message-value	－	(b) スパンイベント
				スパンイベント			message-size	－	(b) スパンイベント
				スパンイベント			thread-name	－	(b) スパンイベント
				スパンイベント	message	(b) スパンイベント	message.id	－	(b) スパンイベント
				スパンイベント			message.type	SENT	(b) スパンイベント
				スパンイベント	received message(hmppcto)	(b) スパンイベント	message-type	－	(b) スパンイベント
				スパンイベント			json-convert	－	(b) スパンイベント
				スパンイベント			message-value	－	(b) スパンイベント
				スパンイベント			message-size	－	(b) スパンイベント

コンポーネント	オペレーション名（短縮）	オペレーション処理概要	スパン取得レベル	スパンタグ/スパンイベント	情報説明		key（情報名）または value（情報値）		
					情報説明名	情報説明の詳細	key（情報名）	value（情報値）	key（情報名）または value（情報値）の詳細
				スパンイベント			thread-name	－	(b) スパンイベント
				スパンイベント	message	(b) スパンイベント	message.id	－	(b) スパンイベント
				スパンイベント			message.type	RECEIVED	(b) スパンイベント
Mediator	hmppcto.mediator.TerminationReceiverService/recover	Orchestratorからのトランザクションキャンセル要求（gRPC 受信処理）です。	必須情報（通信処理のため）	スパンタグ	－	－	network.peer.address	－	(a) スパンタグ
				スパンタグ	－	－	network.peer.port	－	(a) スパンタグ
				スパンタグ	－	－	network.type	－	(a) スパンタグ
				スパンタグ	－	－	rpc.grpc.status_code	－	(a) スパンタグ
				スパンタグ	－	－	rpc.method	－	(a) スパンタグ
				スパンタグ	－	－	rpc.service	－	(a) スパンタグ
				スパンタグ	－	－	rpc.system	－	(a) スパンタグ
				スパンタグ	－	－	server.address	－	(a) スパンタグ

コンポーネント	オペレーション名（短縮）	オペレーション処理概要	スパン取得レベル	スパンタグ/スパンイベント	情報説明		key（情報名）または value（情報値）		
					情報説明名	情報説明の詳細	key（情報名）	value（情報値）	key（情報名）または value（情報値）の詳細
				スパンタグ	−	−	server.port	−	(a) スパンタグ
				スパンタグ	−	−	span.kind	−	(a) スパンタグ
				スパンタグ	−	−	otel.library.name	−	(a) スパンタグ
				スパンタグ	−	−	otel.library.version	−	(a) スパンタグ
				スパンタグ	−	−	otel.scope.name	−	(a) スパンタグ
				スパンタグ	−	−	otel.scope.version	−	(a) スパンタグ
				スパンイベント	sent message(hmppcto)	(b) スパンイベント	message-type	−	(b) スパンイベント
				スパンイベント			json-convert	−	(b) スパンイベント
				スパンイベント			message-value	−	(b) スパンイベント
				スパンイベント			message-size	−	(b) スパンイベント
				スパンイベント			thread-name	−	(b) スパンイベント

コンポーネント	オペレーション名（短縮）	オペレーション処理概要	スパン取得レベル	スパンタグ/スパンイベント	情報説明		key（情報名）または value（情報値）		
					情報説明名	情報説明の詳細	key（情報名）	value（情報値）	key（情報名）または value（情報値）の詳細
				スパンイベント	message	(b) スパンイベント	message.id	－	(b) スパンイベント
				スパンイベント			message.type	SENT	(b) スパンイベント
				スパンイベント	received message(hmppcto)	(b) スパンイベント	message-type	－	(b) スパンイベント
				スパンイベント			json-convert	－	(b) スパンイベント
				スパンイベント			message-value	－	(b) スパンイベント
				スパンイベント			message-size	－	(b) スパンイベント
				スパンイベント			thread-name	－	(b) スパンイベント
				スパンイベント	message	(b) スパンイベント	message.id	－	(b) スパンイベント
				スパンイベント			message.type	RECEIVED	(b) スパンイベント
SQL-Participant/TP1-Bridge	j.c.H.s.h.t.o.ParticipantProposalEventObserver:observe	Entity-Moduleからの先行プリペア要求またはキャンセル要求を実行します。	必須情報	スパンタグ	－	－	thread-name	－	(a) スパンタグ
				スパンタグ	－	－	prepare	－	(a) スパンタグまたは (a) ス

コンポーネント	オペレーション名（短縮）	オペレーション処理概要	スパン取得レベル	スパンタグ/スパンイベント	情報説明		key（情報名）または value（情報値）		
					情報説明名	情報説明の詳細	key（情報名）	value（情報値）	key（情報名）または value（情報値）の詳細
									パンタグ
				スパンタグ	−	−	cancel	−	(a) スパンタグまたは (a) スパンタグ
				スパンタグ	−	−	otel.library.name	−	(a) スパンタグ
				スパンタグ	−	−	otel.library.version	−	(a) スパンタグ
				スパンタグ	−	−	otel.scope.name	−	(a) スパンタグ
				スパンタグ	−	−	otel.scope.version	−	(a) スパンタグ
				スパンイベント	start:termination-judge	(b) スパンイベント	−	−	−
				スパンイベント	end:termination-judge	(b) スパンイベント	−	−	−
Mediator	j.c.H.s.h.m.j.t.PrepareDelayTimeoutJob:callObserver	投票要求送信監視タイマ処理です。	必須情報（タイマ発火）	スパンタグ	−	−	thread-name	−	(a) スパンタグ
				スパンタグ	−	−	otel.library.name	−	(a) スパンタグ

コンポーネント	オペレーション名（短縮）	オペレーション処理概要	スパン取得レベル	スパンタグ/スパンイベント	情報説明		key（情報名）または value（情報値）		
					情報説明名	情報説明の詳細	key（情報名）	value（情報値）	key（情報名）または value（情報値）の詳細
				スパンタグ	－	－	otel.library.version	－	(a) スパンタグ
				スパンタグ	－	－	otel.scope.name	－	(a) スパンタグ
				スパンタグ	－	－	otel.scope.version	－	(a) スパンタグ
Mediator	j.c.H.s.h.m.j.t.PrepareRetryTimeoutJob:callObserver	投票要求再送監視タイマ処理です。	必須情報（タイマ発火）	スパンタグ	－	－	thread-name	－	(a) スパンタグ
				スパンタグ	－	－	otel.library.name	－	(a) スパンタグ
				スパンタグ	－	－	otel.library.version	－	(a) スパンタグ
				スパンタグ	－	－	otel.scope.name	－	(a) スパンタグ
				スパンタグ	－	－	otel.scope.version	－	(a) スパンタグ
TP1-Bridge	j.c.H.s.h.a.TP1ConnectionPool:destroy	TP1-Bridge 終了時の TP1-Client/J のインスタンス破棄処理です。	必須情報	スパンタグ	－	－	thread-name	－	(a) スパンタグ
				スパンタグ	－	－	otel.library.version	－	(a) スパンタグ
				スパンタグ	－	－	otel.scope.name	－	(a) スパンタグ

コンポーネント	オペレーション名（短縮）	オペレーション処理概要	スパン取得レベル	スパンタグ/スパンイベント	情報説明		key（情報名）または value（情報値）		
					情報説明名	情報説明の詳細	key（情報名）	value（情報値）	key（情報名）または value（情報値）の詳細
				スパンタグ	－	－	otel.scope.version	－	(a) スパンタグ
				スパンイベント	start:TP1/Client/J(<TP1/Client/Jのメソッド名>)	(b) スパンイベント	－	－	－
				スパンイベント	end:TP1/Client/J(<TP1/Client/Jのメソッド名>)	(b) スパンイベント	－	－	－
SQL-Participant/TP1-Bridge	j.c.H.s.h.t.o.t.CommitRetryTimeoutJob:callObserver	コミットリトライタイマ処理です。	必須情報（タイマ発火）	スパンタグ	－	－	thread-name	－	(a) スパンタグ
				スパンタグ	－	－	otel.library.name	－	(a) スパンタグ
				スパンタグ	－	－	otel.library.version	－	(a) スパンタグ
				スパンタグ	－	－	otel.scope.name	－	(a) スパンタグ
				スパンタグ	－	－	otel.scope.version	－	(a) スパンタグ

コンポーネント	オペレーション名（短縮）	オペレーション処理概要	スパン取得レベル	スパンタグ/スパンイベント	情報説明		key（情報名）または value（情報値）		
					情報説明名	情報説明の詳細	key（情報名）	value（情報値）	key（情報名）または value（情報値）の詳細
SQL-Participant/TP1-Bridge	j.c.H.s.h.t.o.t.RollbackRetryTimeoutJob:callObserver	ロールバックリトライタイマ処理です。	必須情報（タイマ発火）	スパンタグ	－	－	thread-name	－	(a) スパンタグ
				スパンタグ	－	－	otel.library.name	－	(a) スパンタグ
				スパンタグ	－	－	otel.library.version	－	(a) スパンタグ
				スパンタグ	－	－	otel.scope.name	－	(a) スパンタグ
				スパンタグ	－	－	otel.scope.version	－	(a) スパンタグ
Participant	j.c.H.s.h.p.c.o.AlreadyTerminatedTransactionNotify:sendAlreadyTransactionNotify	Mediator へのトランザクション完了済み通知です。	デバッグ情報	スパンタグ	－	－	thread-name	－	(a) スパンタグ
				スパンタグ	－	－	otel.library.name	－	(a) スパンタグ
				スパンタグ	－	－	otel.library.version	－	(a) スパンタグ
				スパンタグ	－	－	otel.scope.name	－	(a) スパンタグ
				スパンタグ	－	－	otel.scope.version	－	(a) スパンタグ
Participant	j.c.H.s.h.p.s.GrpcAlreadyTerminatedTransactionNotify	Mediator へのトランザクション完了済み通知です。	デバッグ情報	スパンタグ	－	－	thread-name	－	(a) スパンタグ

コンポーネント	オペレーション名（短縮）	オペレーション処理概要	スパン取得レベル	スパンタグ/スパンイベント	情報説明		key（情報名）または value（情報値）		
					情報説明名	情報説明の詳細	key（情報名）	value（情報値）	key（情報名）または value（情報値）の詳細
	RequestSender			スパンタグ	－	－	root-xid	－	(a) スパンタグ
				スパンタグ	－	－	mediator-id	－	(a) スパンタグ
				スパンタグ	－	－	proposer-id	－	(a) スパンタグ
				スパンタグ	－	－	otel.library.name	－	(a) スパンタグ
				スパンタグ	－	－	otel.library.version	－	(a) スパンタグ
				スパンタグ	－	－	otel.scope.name	－	(a) スパンタグ
				スパンタグ	－	－	otel.scope.version	－	(a) スパンタグ
Mediator	j.c.H.s.h.m.j.TerminationJudgeImpl$AsyncNotifyAlreadyTerminatedAndRemoveRequestTask:run	トランザクション完了済み通知を受け、Proposal Logの削除判定または削除を実行します。	デバッグ情報	スパンタグ	－	－	thread-name	－	(a) スパンタグ
				スパンタグ	－	－	otel.library.name	－	(a) スパンタグ
				スパンタグ	－	－	otel.library.version	－	(a) スパンタグ
				スパンタグ	－	－	otel.scope.name	－	(a) スパンタグ

コンポーネント	オペレーション名（短縮）	オペレーション処理概要	スパン取得レベル	スパンタグ/スパンイベント	情報説明		key（情報名）または value（情報値）		
					情報説明名	情報説明の詳細	key（情報名）	value（情報値）	key（情報名）または value（情報値）の詳細
				スパンタグ	−	−	otel.scope.version	−	(a) スパンタグ
				スパンイベント	start:proposallog-delete	(b) スパンイベント	proposal-log-before-size	−	(b) スパンイベント
				スパンイベント	end:proposallog-delete	(b) スパンイベント	proposal-log-after-size	−	(b) スパンイベント
SQL-Participant	j.c.H.s.h.t.o.t.PeriodicallyRecoveryJob:execute	Recovery 処理の定期実行処理です。	必須情報	スパンタグ	−	−	thread-name	−	(a) スパンタグ
				スパンタグ	−	−	otel.library.name	−	(a) スパンタグ
				スパンタグ	−	−	otel.library.version	−	(a) スパンタグ
				スパンタグ	−	−	otel.scope.name	−	(a) スパンタグ
				スパンタグ	−	−	otel.scope.version	−	(a) スパンタグ
				スパンタグ	−	−	recover-pod-hostname	−	(a) スパンタグ
				スパンタグ	−	−	recover-pod-ip	−	(a) スパンタグ

コンポーネント	オペレーション名（短縮）	オペレーション処理概要	スパン取得レベル	スパンタグ/スパンイベント	情報説明		key（情報名）または value（情報値）		
					情報説明名	情報説明の詳細	key（情報名）	value（情報値）	key（情報名）または value（情報値）の詳細
				スパンイベント	start:recover-job	(b) スパンイベント	−	−	−
				スパンイベント	counter	(b) スパンイベント	−	−	−
				スパンイベント	pod-info-list	(b) スパンイベント	−	−	−
				スパンイベント	end:recover-job	(b) スパンイベント	−	−	−
Participant	hmppcto.mediator.TerminationReceiverService/bindInTransaction	Participant と Mediator 間（トランザクション内）のストリームです。	必須情報（通信処理のため）	スパンタグ	−	−	network.peer.address	−	(a) スパンタグ
				スパンタグ	−	−	network.peer.port	−	(a) スパンタグ
				スパンタグ	−	−	network.type	−	(a) スパンタグ
				スパンタグ	−	−	rpc.grpc.status_code	−	(a) スパンタグ
				スパンタグ	−	−	rpc.method	−	(a) スパンタグ
				スパンタグ	−	−	rpc.service	−	(a) スパンタグ
				スパンタグ	−	−	rpc.system	−	(a) スパンタグ

コンポーネント	オペレーション名（短縮）	オペレーション処理概要	スパン取得レベル	スパンタグ/スパンイベント	情報説明		key（情報名）または value（情報値）		
					情報説明名	情報説明の詳細	key（情報名）	value（情報値）	key（情報名）または value（情報値）の詳細
				スパンタグ	−	−	server.address	−	(a) スパンタグ
				スパンタグ	−	−	server.port	−	(a) スパンタグ
				スパンタグ	−	−	span.kind	−	(a) スパンタグ
				スパンタグ	−	−	otel.library.name	−	(a) スパンタグ
				スパンタグ	−	−	otel.library.version	−	(a) スパンタグ
				スパンタグ	−	−	otel.scope.name	−	(a) スパンタグ
				スパンタグ	−	−	otel.scope.version	−	(a) スパンタグ
		Mediator ヘストリームを開始します（gRPC 送信）。		スパンイベント	sent message(hmppcto)	(b) スパンイベント	message-type	mediator.MediatorMessage:bindRequest	(b) スパンイベント
				スパンイベント			json-convert	−	(b) スパンイベント
				スパンイベント			message-value	−	(b) スパンイベント
				スパンイベント			message-size	−	(b) スパンイベント

コンポーネント	オペレーション名（短縮）	オペレーション処理概要	スパン取得レベル	スパンタグ/スパンイベント	情報説明		key（情報名）または value（情報値）		
					情報説明名	情報説明の詳細	key（情報名）	value（情報値）	key（情報名）または value（情報値）の詳細
				スパンイベント			thread-name	－	(b) スパンイベント
				スパンイベント	message	(b) スパンイベント	message.id	－	(b) スパンイベント
				スパンイベント			message.type	SENT	(b) スパンイベント
		Termination Judge の DB 更新可否を Mediator に投票します（gRPC 送信）。		スパンイベント	sent message(hmppcto)	(b) スパンイベント	message-type	mediator.ParticipantMessage:consensusResultRequest	(b) スパンイベント
				スパンイベント			json-convert	－	(b) スパンイベント
				スパンイベント			message-value	－	(b) スパンイベント
				スパンイベント			message-size	－	(b) スパンイベント
				スパンイベント			thread-name	－	(b) スパンイベント
				スパンイベント	message	(b) スパンイベント	message.id	－	(b) スパンイベント
				スパンイベント			message.type	SENT	(b) スパンイベント

コンポーネント	オペレーション名（短縮）	オペレーション処理概要	スパン取得レベル	スパンタグ/スパンイベント	情報説明		key（情報名）または value（情報値）		
					情報説明名	情報説明の詳細	key（情報名）	value（情報値）	key（情報名）または value（情報値）の詳細
		Mediator へ合意結果を送信します（gRPC 送信）。		スパンイベント	sent message(hmppcto)	(b) スパンイベント	message-type	mediator.MediatorMessage:notifyTerminationResultRequest	(b) スパンイベント
				スパンイベント			json-convert	－	(b) スパンイベント
				スパンイベント			message-value	－	(b) スパンイベント
				スパンイベント			message-size	－	(b) スパンイベント
				スパンイベント			thread-name	－	(b) スパンイベント
				スパンイベント	message	(b) スパンイベント	message.id	－	(b) スパンイベント
				スパンイベント			message.type	SENT	(b) スパンイベント
		トランザクション完了通知を送信します（gRPC 送信）。		スパンイベント	sent message(hmppcto)	(b) スパンイベント	message-type	mediator.MediatorMessage:notifyTerminatedTransactionRequest	(b) スパンイベント
				スパンイベント			json-convert	－	(b) スパンイベント

コンポーネント	オペレーション名（短縮）	オペレーション処理概要	スパン取得レベル	スパンタグ/スパンイベント	情報説明		key（情報名）または value（情報値）		
					情報説明名	情報説明の詳細	key（情報名）	value（情報値）	key（情報名）または value（情報値）の詳細
				スパンイベント			message-value	－	(b)　スパンイベント
				スパンイベント			message-size	－	(b)　スパンイベント
				スパンイベント			thread-name	－	(b)　スパンイベント
				スパンイベント	message	(b)　スパンイベント	message.id	－	(b)　スパンイベント
				スパンイベント			message.type	SENT	(b)　スパンイベント
		Mediator からのトランザクション決着指示要求（gRPC 受信）です。		スパンイベント	received message(hmppcto)	(b)　スパンイベント	message-type	mediator.ParticipantMessage:consensusResultRequest	(b)　スパンイベント
				スパンイベント			json-convert	－	(b)　スパンイベント
				スパンイベント			message-value	－	(b)　スパンイベント
				スパンイベント			message-size	－	(b)　スパンイベント
				スパンイベント			thread-name	－	(b)　スパンイベント

コンポーネント	オペレーション名（短縮）	オペレーション処理概要	スパン取得レベル	スパンタグ/スパンイベント	情報説明		key（情報名）または value（情報値）		
					情報説明名	情報説明の詳細	key（情報名）	value（情報値）	key（情報名）または value（情報値）の詳細
				スパンイベント	message	(b) スパンイベント	message.id	−	(b) スパンイベント
				スパンイベント			message.type	RECEIVED	(b) スパンイベント
		Mediator からの投票依頼（gRPC 受信）です。		スパンイベント	received message(hmppcto)	(b) スパンイベント	message-type	mediator.ParticipantMessage:proposalRequest	(b) スパンイベント
				スパンイベント			json-convert	−	(b) スパンイベント
				スパンイベント			message-value	−	(b) スパンイベント
				スパンイベント			message-size	−	(b) スパンイベント
				スパンイベント			thread-name	−	(b) スパンイベント
				スパンイベント	message	(b) スパンイベント	message.id	−	(b) スパンイベント
				スパンイベント			message.type	RECEIVED	(b) スパンイベント
Mediator	hmppcto.mediator.TerminationReceiverService/	Participant と Mediator 間（トランザクション内）のストリームです。	必須情報（通信処理のため）	スパンタグ	−	−	network.peer.address	−	(a) スパンタグ
				スパンタグ	−	−	network.peer.port	−	(a) スパンタグ

コンポーネント	オペレーション名（短縮）	オペレーション処理概要	スパン取得レベル	スパンタグ/スパンイベント	情報説明		key（情報名）または value（情報値）		
					情報説明名	情報説明の詳細	key（情報名）	value（情報値）	key（情報名）または value（情報値）の詳細
	bindInTransaction			スパンタグ	－	－	network.type	－	(a) スパンタグ
				スパンタグ	－	－	rpc.grpc.status_code	－	(a) スパンタグ
				スパンタグ	－	－	rpc.method	－	(a) スパンタグ
				スパンタグ	－	－	rpc.service	－	(a) スパンタグ
				スパンタグ	－	－	rpc.system	－	(a) スパンタグ
				スパンタグ	－	－	server.address	－	(a) スパンタグ
				スパンタグ	－	－	server.port	－	(a) スパンタグ
				スパンタグ	－	－	span.kind	－	(a) スパンタグ
				スパンタグ	－	－	otel.library.name	－	(a) スパンタグ
				スパンタグ	－	－	otel.library.version	－	(a) スパンタグ
				スパンタグ	－	－	otel.scope.name	－	(a) スパンタグ

コンポーネント	オペレーション名（短縮）	オペレーション処理概要	スパン取得レベル	スパンタグ/スパンイベント	情報説明		key（情報名）または value（情報値）		
					情報説明名	情報説明の詳細	key（情報名）	value（情報値）	key（情報名）または value（情報値）の詳細
				スパンタグ	－	－	otel.scope.version	－	(a) スパンタグ
		Participant からストリームを開始します（gRPC 受信）。		スパンイベント	received message(hmppcto)	(b) スパンイベント	message-type	mediator.MediatorMessage:bindRequest	(b) スパンイベント
				スパンイベント			json-convert	－	(b) スパンイベント
				スパンイベント			message-value	－	(b) スパンイベント
				スパンイベント			message-size	－	(b) スパンイベント
				スパンイベント			thread-name	－	(b) スパンイベント
				スパンイベント	message	(b) スパンイベント	message.id	－	(b) スパンイベント
				スパンイベント			message.type	RECEIVED	(b) スパンイベント
		Participant からの Termination Judge の DB 更新可否の受信（gRPC 受信）です。		スパンイベント	received message(hmppcto)	(b) スパンイベント	message-type	mediator.ParticipantMessage:consensusResultRequest	(b) スパンイベント
				スパンイベント			json-convert	－	(b) スパンイベント

コンポーネント	オペレーション名（短縮）	オペレーション処理概要	スパン取得レベル	スパンタグ/スパンイベント	情報説明		key（情報名）または value（情報値）		
					情報説明名	情報説明の詳細	key（情報名）	value（情報値）	key（情報名）または value（情報値）の詳細
				スパンイベント			message-value	－	(b)　スパンイベント
				スパンイベント			message-size	－	(b)　スパンイベント
				スパンイベント			thread-name	－	(b)　スパンイベント
				スパンイベント	message	(b)　スパンイベント	message.id	－	(b)　スパンイベント
				スパンイベント			message.type	RECEIVED	(b)　スパンイベント
		Participant からの合意結果の受信（gRPC 受信）です。		スパンイベント	received message(hmppcto)	(b)　スパンイベント	message-type	mediator.MediatorMessage:notifyTerminationResultRequest	(b)　スパンイベント
				スパンイベント			json-convert	－	(b)　スパンイベント
				スパンイベント			message-value	－	(b)　スパンイベント
				スパンイベント			message-size	－	(b)　スパンイベント
				スパンイベント			thread-name	－	(b)　スパンイベント

コンポーネント	オペレーション名（短縮）	オペレーション処理概要	スパン取得レベル	スパンタグ/スパンイベント	情報説明		key（情報名）または value（情報値）		
					情報説明名	情報説明の詳細	key（情報名）	value（情報値）	key（情報名）または value（情報値）の詳細
				スパンイベント	message	(b) スパンイベント	message.id	－	(b) スパンイベント
				スパンイベント			message.type	RECEIVED	(b) スパンイベント
		トランザクション完了通知の受信（gRPC 受信）です。		スパンイベント	received message(hmppcto)	(b) スパンイベント	message-type	mediator.MediatorMessage:notifyTerminatedTransactionRequest	(b) スパンイベント
				スパンイベント			json-convert	－	(b) スパンイベント
				スパンイベント			message-value	－	(b) スパンイベント
				スパンイベント			message-size	－	(b) スパンイベント
				スパンイベント			thread-name	－	(b) スパンイベント
				スパンイベント	message	(b) スパンイベント	message.id	－	(b) スパンイベント
				スパンイベント			message.type	RECEIVED	(b) スパンイベント
		Participant へのトランザクション決着指示		スパンイベント	sent message(hmppcto)	(b) スパンイベント	message-type	mediator.ParticipantMessage:consensusR	(b) スパンイベント

コンポーネント	オペレーション名（短縮）	オペレーション処理概要	スパン取得レベル	スパンタグ/スパンイベント	情報説明		key（情報名）または value（情報値）		
					情報説明名	情報説明の詳細	key（情報名）	value（情報値）	key（情報名）または value（情報値）の詳細
		要求（gRPC 送信）です。						esultRequest	
				スパンイベント			json-convert	－	(b) スパンイベント
				スパンイベント			message-value	－	(b) スパンイベント
				スパンイベント			message-size	－	(b) スパンイベント
				スパンイベント			thread-name	－	(b) スパンイベント
				スパンイベント	message	(b) スパンイベント	message.id	－	(b) スパンイベント
				スパンイベント			message.type	SENT	(b) スパンイベント
		Participant への投票依頼（gRPC 送信）です。		スパンイベント	sent message(hmppcto)	(b) スパンイベント	message-type	mediator.ParticipantMessage:proposalRequest	(b) スパンイベント
				スパンイベント			json-convert	－	(b) スパンイベント
				スパンイベント			message-value	－	(b) スパンイベント
				スパンイベント			message-size	－	(b) スパンイベント

コンポーネント	オペレーション名（短縮）	オペレーション処理概要	スパン取得レベル	スパンタグ/スパンイベント	情報説明		key（情報名）または value（情報値）		
					情報説明名	情報説明の詳細	key（情報名）	value（情報値）	key（情報名）または value（情報値）の詳細
				スパンイベント			thread-name	－	(b) スパンイベント
				スパンイベント	message	(b) スパンイベント	message.id	－	(b) スパンイベント
				スパンイベント			message.type	SENT	(b) スパンイベント
Participant	hmppcto.mediator.TerminationReceiverService/bindOutTransaction	Participant と Mediator 間（トランザクション外）のストリームです。	必須情報（通信処理のため）	スパンタグ	－	－	network.peer.address	－	(a) スパンタグ
				スパンタグ	－	－	network.peer.port	－	(a) スパンタグ
				スパンタグ	－	－	network.type	－	(a) スパンタグ
				スパンタグ	－	－	rpc.grpc.status_code	－	(a) スパンタグ
				スパンタグ	－	－	rpc.method	－	(a) スパンタグ
				スパンタグ	－	－	rpc.service	－	(a) スパンタグ
				スパンタグ	－	－	rpc.system	－	(a) スパンタグ
				スパンタグ	－	－	server.address	－	(a) スパンタグ

コンポーネント	オペレーション名（短縮）	オペレーション処理概要	スパン取得レベル	スパンタグ/スパンイベント	情報説明		key（情報名）または value（情報値）		
					情報説明名	情報説明の詳細	key（情報名）	value（情報値）	key（情報名）または value（情報値）の詳細
				スパンタグ	−	−	server.port	−	(a) スパンタグ
				スパンタグ	−	−	span.kind	−	(a) スパンタグ
				スパンタグ	−	−	otel.library.name	−	(a) スパンタグ
				スパンタグ	−	−	otel.library.version	−	(a) スパンタグ
				スパンタグ	−	−	otel.scope.name	−	(a) スパンタグ
				スパンタグ	−	−	otel.scope.version	−	(a) スパンタグ
		Mediator ヘストリームを開始します（gRPC 送信）。		スパンイベント	sent message(hmppcto)	(b) スパンイベント	message-type	mediator.MediatorMessage:bindRequest	(b) スパンイベント
				スパンイベント			json-convert	−	(b) スパンイベント
				スパンイベント			message-value	−	(b) スパンイベント
				スパンイベント			message-size	−	(b) スパンイベント
				スパンイベント			thread-name	−	(b) スパンイベント

コンポーネント	オペレーション名（短縮）	オペレーション処理概要	スパン取得レベル	スパンタグ/スパンイベント	情報説明		key（情報名）または value（情報値）		
					情報説明名	情報説明の詳細	key（情報名）	value（情報値）	key（情報名）または value（情報値）の詳細
				スパンイベント	message	(b) スパンイベント	message.id	－	(b) スパンイベント
				スパンイベント			message.type	SENT	(b) スパンイベント
		トランザクション完了通知を送信します（gRPC 送信）。		スパンイベント	sent message(hmppcto)	(b) スパンイベント	message-type	mediator.MediatorMessage:notifyTerminatedTransactionRequest	(b) スパンイベント
				スパンイベント			json-convert	－	(b) スパンイベント
				スパンイベント			message-value	－	(b) スパンイベント
				スパンイベント			message-size	－	(b) スパンイベント
				スパンイベント			thread-name	－	(b) スパンイベント
				スパンイベント	message	(b) スパンイベント	message.id	－	(b) スパンイベント
				スパンイベント			message.type	SENT	(b) スパンイベント
		Mediator からのトランザクション決着指示		スパンイベント	received messag	(b) スパンイベント	message-type	mediator.ParticipantMessage:consensusR	(b) スパンイベント

コンポーネント	オペレーション名（短縮）	オペレーション処理概要	スパン取得レベル	スパンタグ/スパンイベント	情報説明		key（情報名）または value（情報値）		
					情報説明名	情報説明の詳細	key（情報名）	value（情報値）	key（情報名）または value（情報値）の詳細
		要求（gRPC 受信）です。			e(hmppcto)			esultRequest	
				スパンイベント			json-convert	－	(b) スパンイベント
				スパンイベント			message-value	－	(b) スパンイベント
				スパンイベント			message-size	－	(b) スパンイベント
				スパンイベント			thread-name	－	(b) スパンイベント
				スパンイベント	message	(b) スパンイベント	message.id	－	(b) スパンイベント
				スパンイベント			message.type	RECEIVED	(b) スパンイベント
		Mediator へのトランザクション完了済み通知を送信します（gRPC 送信）。		スパンイベント	sent message(hmppcto)	(b) スパンイベント	message-type	mediator.MediatorMessage:notifyAlreadyTerminatedTransactionRequest	(b) スパンイベント
				スパンイベント			json-convert	－	(b) スパンイベント
				スパンイベント			message-value	－	(b) スパンイベント

コンポーネント	オペレーション名（短縮）	オペレーション処理概要	スパン取得レベル	スパンタグ/スパンイベント	情報説明		key（情報名）または value（情報値）		
					情報説明名	情報説明の詳細	key（情報名）	value（情報値）	key（情報名）または value（情報値）の詳細
				スパンイベント			message-size	－	(b) スパンイベント
				スパンイベント			thread-name	－	(b) スパンイベント
				スパンイベント	message	(b) スパンイベント	message.id	－	(b) スパンイベント
				スパンイベント			message.type	SENT	(b) スパンイベント
		Mediator からの投票依頼（gRPC 受信）です。		スパンイベント	received message(hmppcto)	(b) スパンイベント	message-type	mediator.ParticipantMessage:proposalRequest	(b) スパンイベント
				スパンイベント			json-convert	－	(b) スパンイベント
				スパンイベント			message-value	－	(b) スパンイベント
				スパンイベント			message-size	－	(b) スパンイベント
				スパンイベント			thread-name	－	(b) スパンイベント
				スパンイベント	message	(b) スパンイベント	message.id	－	(b) スパンイベント
				スパンイベント			message.type	RECEIVED	(b) スパンイベント

コンポーネント	オペレーション名（短縮）	オペレーション処理概要	スパン取得レベル	スパンタグ/スパンイベント	情報説明		key（情報名）または value（情報値）		
					情報説明名	情報説明の詳細	key（情報名）	value（情報値）	key（情報名）または value（情報値）の詳細
		Mediator からのトランザクション完了確認通知の受信（gRPC 受信）です。		スパンイベント	received message(hmppcto)	(b) スパンイベント	message-type	mediator.ParticipantMessage:terminatedTransactionVerifyNotifyRequest	(b) スパンイベント
				スパンイベント			json-convert	−	(b) スパンイベント
				スパンイベント			message-value	−	(b) スパンイベント
				スパンイベント			message-size	−	(b) スパンイベント
				スパンイベント			thread-name	−	(b) スパンイベント
				スパンイベント	message	(b) スパンイベント	message.id	−	(b) スパンイベント
				スパンイベント		(b) スパンイベント	message.type	RECEIVED	(b) スパンイベント
Mediator	hmppcto.mediator.TerminationReceiverService/bindOutTransaction	Participant と Mediator 間（トランザクション外）のストリームです。	必須情報（通信処理のため）	スパンタグ	−	−	network.peer.address	−	(a) スパンタグ
				スパンタグ	−	−	network.peer.port	−	(a) スパンタグ
				スパンタグ	−	−	network.type	−	(a) スパンタグ

コンポーネント	オペレーション名（短縮）	オペレーション処理概要	スパン取得レベル	スパンタグ/スパンイベント	情報説明		key（情報名）または value（情報値）		
					情報説明名	情報説明の詳細	key（情報名）	value（情報値）	key（情報名）または value（情報値）の詳細
				スパンタグ	－	－	rpc.grpc.status_code	－	(a) スパンタグ
				スパンタグ	－	－	rpc.method	－	(a) スパンタグ
				スパンタグ	－	－	rpc.service	－	(a) スパンタグ
				スパンタグ	－	－	rpc.system	－	(a) スパンタグ
				スパンタグ	－	－	server.address	－	(a) スパンタグ
				スパンタグ	－	－	server.port	－	(a) スパンタグ
				スパンタグ	－	－	span.kind	－	(a) スパンタグ
				スパンタグ	－	－	otel.library.name	－	(a) スパンタグ
				スパンタグ	－	－	otel.library.version	－	(a) スパンタグ
				スパンタグ	－	－	otel.scope.name	－	(a) スパンタグ
				スパンタグ	－	－	otel.scope.version	－	(a) スパンタグ

コンポーネント	オペレーション名（短縮）	オペレーション処理概要	スパン取得レベル	スパンタグ/スパンイベント	情報説明		key（情報名）または value（情報値）		
					情報説明名	情報説明の詳細	key（情報名）	value（情報値）	key（情報名）または value（情報値）の詳細
		Participant からストリームを開始します（gRPC 受信）。		スパンイベント	received message(hmppcto)	(b) スパンイベント	message-type	mediator.MediatorMessage:bindRequest	(b) スパンイベント
				スパンイベント			json-convert	−	(b) スパンイベント
				スパンイベント			message-value	−	(b) スパンイベント
				スパンイベント			message-size	−	(b) スパンイベント
				スパンイベント			thread-name	−	(b) スパンイベント
				スパンイベント	message	(b) スパンイベント	message.id	−	(b) スパンイベント
				スパンイベント			message.type	RECEIVED	(b) スパンイベント
		トランザクション完了通知の受信（gRPC 受信）です。		スパンイベント	received message(hmppcto)	(b) スパンイベント	message-type	mediator.MediatorMessage:notifyTerminatedTransactionRequest	(b) スパンイベント
				スパンイベント			json-convert	−	(b) スパンイベント
				スパンイベント			message-value	−	(b) スパンイベント

コンポーネント	オペレーション名（短縮）	オペレーション処理概要	スパン取得レベル	スパンタグ/スパンイベント	情報説明		key（情報名）または value（情報値）		
					情報説明名	情報説明の詳細	key（情報名）	value（情報値）	key（情報名）または value（情報値）の詳細
				スパンイベント			message-size	－	(b) スパンイベント
				スパンイベント			thread-name	－	(b) スパンイベント
				スパンイベント	message	(b) スパンイベント	message.id	－	(b) スパンイベント
				スパンイベント			message.type	RECEIVED	(b) スパンイベント
		Participant へのトランザクション決着指示要求（gRPC 送信）です。		スパンイベント	sent message(hmppcto)	(b) スパンイベント	message-type	mediator.ParticipantMessage:consensusResultRequest	(b) スパンイベント
				スパンイベント			json-convert	－	(b) スパンイベント
				スパンイベント			message-value	－	(b) スパンイベント
				スパンイベント			message-size	－	(b) スパンイベント
				スパンイベント			thread-name	－	(b) スパンイベント
				スパンイベント	message	(b) スパンイベント	message.id	－	(b) スパンイベント

コンポーネント	オペレーション名（短縮）	オペレーション処理概要	スパン取得レベル	スパンタグ/スパンイベント	情報説明		key（情報名）または value（情報値）		
					情報説明名	情報説明の詳細	key（情報名）	value（情報値）	key（情報名）または value（情報値）の詳細
				スパンイベント			message.type	SENT	(b)　スパンイベント
		Participant からのトランザクション完了済み通知の受信（gRPC 受信）です。		スパンイベント	received message(hmppcto)	(b)　スパンイベント	message-type	mediator.MediatorMessage:notifyAlreadyTerminatedTransactionRequest	(b)　スパンイベント
				スパンイベント			json-convert	−	(b)　スパンイベント
				スパンイベント			message-value	−	(b)　スパンイベント
				スパンイベント			message-size	−	(b)　スパンイベント
				スパンイベント			thread-name	−	(b)　スパンイベント
				スパンイベント	message	(b)　スパンイベント	message.id	−	(b)　スパンイベント
				スパンイベント			message.type	RECEIVED	(b)　スパンイベント
		Participant への投票依頼（gRPC 送信）です。		スパンイベント	sent message(hmppcto)	(b)　スパンイベント	message-type	mediator.ParticipantMessage:proposalRequest	(b)　スパンイベント

コンポーネント	オペレーション名（短縮）	オペレーション処理概要	スパン取得レベル	スパンタグ/スパンイベント	情報説明		key（情報名）または value（情報値）		
					情報説明名	情報説明の詳細	key（情報名）	value（情報値）	key（情報名）または value（情報値）の詳細
				スパンイベント			json-convert	－	(b) スパンイベント
				スパンイベント			message-value	－	(b) スパンイベント
				スパンイベント			message-size	－	(b) スパンイベント
				スパンイベント			thread-name	－	(b) スパンイベント
				スパンイベント	message	(b) スパンイベント	message.id	－	(b) スパンイベント
				スパンイベント			message.type	SENT	(b) スパンイベント
		Participant へトランザクション完了確認通知を送信します（gRPC 送信）。		スパンイベント	sent message(hmppcto)	(b) スパンイベント	message-type	mediator.ParticipantMessage:terminatedTransactionVerifyNotifyRequest	(b) スパンイベント
				スパンイベント			json-convert	－	(b) スパンイベント
				スパンイベント			message-value	－	(b) スパンイベント
				スパンイベント			message-size	－	(b) スパンイベント

コンポーネント	オペレーション名（短縮）	オペレーション処理概要	スパン取得レベル	スパンタグ/スパンイベント	情報説明		key（情報名）または value（情報値）		
					情報説明名	情報説明の詳細	key（情報名）	value（情報値）	key（情報名）または value（情報値）の詳細
				スパンイベント			thread-name	－	(b) スパンイベント
				スパンイベント	message	(b) スパンイベント	message.id	－	(b) スパンイベント
				スパンイベント			message.type	SENT	(b) スパンイベント
SQL-Participant/TP1-Bridge	j.c.H.s.h.t.o.OscProposalController:init	Participant 再起動時の Recovery 処理です。	必須情報	スパンタグ	－	－	thread-name	－	(a) スパンタグ
				スパンタグ	－	－	branch-xid	－	(a) スパンタグ
				スパンタグ	－	－	otel.library.name	－	(a) スパンタグ
				スパンタグ	－	－	otel.library.version	－	(a) スパンタグ
				スパンタグ	－	－	otel.scope.name	－	(a) スパンタグ
				スパンタグ	－	－	otel.scope.version	－	(a) スパンタグ
Mediator	j.c.H.s.h.m.j.TerminationJudgeImpl:acceptTermination	分散合意結果通知を受け、Proposal Log の削除判定または削除を実行します。	デバッグ情報	スパンタグ	－	－	thread-name	－	(a) スパンタグ
				スパンタグ	－	－	otel.library.name	－	(a) スパンタグ

コンポーネント	オペレーション名（短縮）	オペレーション処理概要	スパン取得レベル	スパンタグ/スパンイベント	情報説明		key（情報名）または value（情報値）		
					情報説明名	情報説明の詳細	key（情報名）	value（情報値）	key（情報名）または value（情報値）の詳細
				スパンタグ	−	−	otel.library.version	−	(a) スパンタグ
				スパンタグ	−	−	otel.scope.name	−	(a) スパンタグ
				スパンタグ	−	−	otel.scope.version	−	(a) スパンタグ
				スパンイベント	start:proposallog-delete	(b) スパンイベント	proposal-log-before-size	−	(b) スパンイベント
				スパンイベント	end:proposallog-delete	(b) スパンイベント	proposal-log-after-size	−	(b) スパンイベント
SQL-Participant	j.c.H.s.h.t.o.t.PeriodicallyRecoveryJob:execute	スケーリング対応機能使用時の Recovery 処理です。	必須情報	スパンタグ	−	−	thread-name	−	(a) スパンタグ
				スパンタグ	−	−	branch-xid	−	(a) スパンタグ
				スパンタグ	−	−	otel.library.name	−	(a) スパンタグ
				スパンタグ	−	−	otel.library.version	−	(a) スパンタグ
				スパンタグ	−	−	otel.scope.name	−	(a) スパンタグ
				スパンタグ	−	−	otel.scope.version	−	(a) スパンタグ

コンポーネント	オペレーション名（短縮）	オペレーション処理概要	スパン取得レベル	スパンタグ/スパンイベント	情報説明		key（情報名）または value（情報値）		
					情報説明名	情報説明の詳細	key（情報名）	value（情報値）	key（情報名）または value（情報値）の詳細
				スパンイベント	old-pod-uuid	(b) スパンイベント	－	－	－
				スパンイベント	counter	(b) スパンイベント	－	－	－
TCC-Participant	j.c.H.s.h.p.c.o.s.ProposalStateReducer$ProposalStateReducerProcess:prepareBranch	TCC-Participant のトランザクションステータスの状態遷移（投票送信に係る指示）です。	必須情報	スパンタグ	－	－	thread-name	－	(a) スパンタグ
				スパンタグ	－	－	branch-xid	－	(a) スパンタグ
				スパンタグ	－	－	before-transaction-status	－	(a) スパンタグ
				スパンタグ	－	－	after-transaction-status	－	(a) スパンタグ
				スパンタグ	－	－	otel.library.name	－	(a) スパンタグ
				スパンタグ	－	－	otel.library.version	－	(a) スパンタグ
				スパンタグ	－	－	otel.scope.name	－	(a) スパンタグ
				スパンタグ	－	－	otel.scope.version	－	(a) スパンタグ
TCC-Participant	j.c.H.s.h.p.c.o.s.TerminationStat	TCC-Participant のトランザクショ	必須情報	スパンタグ	－	－	thread-name	－	(a) スパンタグ

コンポーネント	オペレーション名（短縮）	オペレーション処理概要	スパン取得レベル	スパンタグ/スパンイベント	情報説明		key（情報名）または value（情報値）		
					情報説明名	情報説明の詳細	key（情報名）	value（情報値）	key（情報名）または value（情報値）の詳細
	eReducer$TerminationStateReducerProcess:innerReduce	ンステータスの状態遷移（リソースへの Confirm または Cancel の実行）です。		スパンタグ	−	−	branch-xid	−	(a) スパンタグ
				スパンタグ	−	−	before-transaction-status	−	(a) スパンタグ
				スパンタグ	−	−	after-transaction-status	−	(a) スパンタグ
				スパンタグ	−	−	otel.library.name	−	(a) スパンタグ
				スパンタグ	−	−	otel.library.version	−	(a) スパンタグ
				スパンタグ	−	−	otel.scope.name	−	(a) スパンタグ
				スパンタグ	−	−	otel.scope.version	−	(a) スパンタグ
				スパンイベント	start:confirm	(b) スパンイベント	−	−	−
				スパンイベント	end:confirm	(b) スパンイベント	−	−	−
				スパンイベント	start:cancel	(b) スパンイベント	−	−	−
				スパンイベント	end:cancel	(b) スパンイベント	−	−	−

コンポーネント	オペレーション名（短縮）	オペレーション処理概要	スパン取得レベル	スパンタグ/スパンイベント	情報説明		key（情報名）または value（情報値）		
					情報説明名	情報説明の詳細	key（情報名）	value（情報値）	key（情報名）または value（情報値）の詳細
Orchestrator/Entity-Module	AP が実装した gRPC 送信クラス名:メソッド名	AP による gRPC 通信です。	必須情報（通信処理のため）	スパンタグ	−	−	network.peer.address	−	(a) スパンタグ
				スパンタグ	−	−	network.peer.port	−	(a) スパンタグ
				スパンタグ	−	−	network.type	−	(a) スパンタグ
				スパンタグ	−	−	rpc.grpc.status_code	−	(a) スパンタグ
				スパンタグ	−	−	rpc.method	−	(a) スパンタグ
				スパンタグ	−	−	rpc.service	−	(a) スパンタグ
				スパンタグ	−	−	rpc.system	−	(a) スパンタグ
				スパンタグ	−	−	server.address	−	(a) スパンタグ
				スパンタグ	−	−	server.port	−	(a) スパンタグ
				スパンタグ	−	−	span.kind	−	(a) スパンタグ
				スパンタグ	−	−	otel.library.name	−	(a) スパンタグ

コンポーネント	オペレーション名（短縮）	オペレーション処理概要	スパン取得レベル	スパンタグ/スパンイベント	情報説明		key（情報名）または value（情報値）		
					情報説明名	情報説明の詳細	key（情報名）	value（情報値）	key（情報名）または value（情報値）の詳細
				スパンタグ	−	−	otel.library.version	−	(a) スパンタグ
				スパンタグ	−	−	otel.scope.name	−	(a) スパンタグ
				スパンタグ	−	−	otel.scope.version	−	(a) スパンタグ
				スパンタグ	−	−	routed-uri	−	(a) スパンタグ
				スパンイベント	start:pre process	(b) スパンイベントおよび (b) スパンイベント	−	−	−
				スパンイベント	end:pre process	(b) スパンイベントおよび (b) スパンイベント	−	−	−
				スパンイベント	return:InnerClientCall	(b) スパンイベントおよび (b) スパンイベント	−	−	−
				スパンイベント	start:InnerClientCall	(b) スパンイベントおよび	−	−	−

コンポーネント	オペレーション名（短縮）	オペレーション処理概要	スパン取得レベル	スパンタグ/スパンイベント	情報説明		key（情報名）または value（情報値）		
					情報説明名	情報説明の詳細	key（情報名）	value（情報値）	key（情報名）または value（情報値）の詳細
						(b) スパンイベント			
				スパンイベント	start:post process	(b) スパンイベントおよび(b) スパンイベント	−	−	−
				スパンイベント	end:post process	(b) スパンイベントおよび(b) スパンイベント	−	−	−
				スパンイベント	message	(b) スパンイベント	message.id	−	(b) スパンイベント
				スパンイベント			message.type	SENT	(b) スパンイベント
				スパンイベント	message	(b) スパンイベント	message.id	−	(b) スパンイベント
				スパンイベント			message.type	RECEIVED	(b) スパンイベント
Entity-Module	AP が実装した gRPC 受信クラス名:メソッド名	AP による gRPC 通信です。	必須情報（通信処理のため）	スパンタグ	−	−	branch-xid	−	(a) スパンタグ
				スパンタグ	−	−	thread-name	−	(a) スパンタグ

コンポーネント	オペレーション名（短縮）	オペレーション処理概要	スパン取得レベル	スパンタグ/スパンイベント	情報説明		key（情報名）または value（情報値）		
					情報説明名	情報説明の詳細	key（情報名）	value（情報値）	key（情報名）または value（情報値）の詳細
				スパンタグ	－	－	result	－	(a) スパンタグ
				スパンタグ	－	－	prepare	－	(a) スパンタグ
				スパンタグ	－	－	network.peer.address	－	(a) スパンタグ
				スパンタグ	－	－	network.peer.port	－	(a) スパンタグ
				スパンタグ	－	－	network.type	－	(a) スパンタグ
				スパンタグ	－	－	rpc.grpc.status_code	－	(a) スパンタグ
				スパンタグ	－	－	rpc.method	－	(a) スパンタグ
				スパンタグ	－	－	rpc.service	－	(a) スパンタグ
				スパンタグ	－	－	rpc.system	－	(a) スパンタグ
				スパンタグ	－	－	server.address	－	(a) スパンタグ
				スパンタグ	－	－	server.port	－	(a) スパンタグ

コンポーネント	オペレーション名（短縮）	オペレーション処理概要	スパン取得レベル	スパンタグ/スパンイベント	情報説明		key（情報名）または value（情報値）		
					情報説明名	情報説明の詳細	key（情報名）	value（情報値）	key（情報名）または value（情報値）の詳細
				スパンタグ	−	−	span.kind	−	(a) スパンタグ
				スパンタグ	−	−	otel.library.version	−	(a) スパンタグ
				スパンタグ	−	−	otel.scope.name	−	(a) スパンタグ
				スパンタグ	−	−	otel.scope.version	−	(a) スパンタグ
				スパンイベント	start:pre process	(b) スパンイベント	−	−	−
				スパンイベント	end:pre process	(b) スパンイベント	−	−	−
				スパンイベント	start:post process	(b) スパンイベント	−	−	−
				スパンイベント	end:post process	(b) スパンイベント	−	−	−
				スパンイベント	start:jdbc <JDBC Driverのクラス名>.<JDBC Driverのメソッド名>	(b) スパンイベント	−	−	−

コンポーネント	オペレーション名（短縮）	オペレーション処理概要	スパン取得レベル	スパンタグ/スパンイベント	情報説明		key（情報名）または value（情報値）		
					情報説明名	情報説明の詳細	key（情報名）	value（情報値）	key（情報名）または value（情報値）の詳細
				スパンイベント	end:jdbc	(b) スパンイベント	−	−	−
				スパンイベント	start:xa function	(b) スパンイベント	xa-name	xa_start	(b) スパンイベント
				スパンイベント	end:xa function	(b) スパンイベント	xa-name	xa_start	(b) スパンイベント
				スパンイベント	failed:xa function	(b) スパンイベント	xa-name	xa_start	(b) スパンイベント
				スパンイベント			message	JDBC Driver が送出した例外のメッセージ文字列	(b) スパンイベント
				スパンイベント	start:try	(b) スパンイベント	−	−	−
				スパンイベント	end:try	(b) スパンイベント	−	−	−
				スパンイベント	start:termination-judge	(b) スパンイベント	−	−	−
				スパンイベント	end:termination-judge	(b) スパンイベント	−	−	−
				スパンイベント	message	(b) スパンイベント	message.id	−	(b) スパンイベント
				スパンイベント			message.type	RECEIVED	(b) スパンイベント

コンポーネント	オペレーション名（短縮）	オペレーション処理概要	スパン取得レベル	スパンタグ/スパンイベント	情報説明		key（情報名）または value（情報値）		
					情報説明名	情報説明の詳細	key（情報名）	value（情報値）	key（情報名）または value（情報値）の詳細
				スパンイベント	message	(b) スパンイベント	message.id	−	(b) スパンイベント
				スパンイベント			message.type	SENT	(b) スパンイベント
Participant	j.c.H.s.h.g.s.GrpcMediatorTransactionStreamImpl:connectMediators	Participant と Mediator 間（トランザクション内）のストリームを開始します。	デバッグ情報	スパンタグ	−	−	thread-name	−	(a) スパンタグ
				スパンタグ	−	−	otel.library.name	−	(a) スパンタグ
				スパンタグ	−	−	otel.library.version	−	(a) スパンタグ
				スパンタグ	−	−	otel.scope.name	−	(a) スパンタグ
				スパンタグ	−	−	otel.scope.version	−	(a) スパンタグ
Participant	j.c.H.s.h.g.s.GrpcMediatorTransactionStreamImpl:connectAfterSend	Participant と Mediator 間（トランザクション内）のストリームの開始後に、メッセージを送信します。	デバッグ情報	スパンタグ	−	−	thread-name	−	(a) スパンタグ
				スパンタグ	−	−	otel.library.name	−	(a) スパンタグ
				スパンタグ	−	−	otel.library.version	−	(a) スパンタグ
				スパンタグ	−	−	otel.scope.name	−	(a) スパンタグ

コンポーネント	オペレーション名（短縮）	オペレーション処理概要	スパン取得レベル	スパンタグ/スパンイベント	情報説明		key（情報名）または value（情報値）		
					情報説明名	情報説明の詳細	key（情報名）	value（情報値）	key（情報名）または value（情報値）の詳細
				スパンタグ	−	−	otel.scope.version	−	(a) スパンタグ
Mediator	j.c.H.s.h.m.r.GrpcParticipantClient$AsyncSendTask:run	Participant への更新可否の決着結果通知（gRPC 送信）です。	必須情報（通信処理のため）	スパンタグ	−	−	thread-name	−	(a) スパンタグ
				スパンタグ	−	−	otel.library.name	−	(a) スパンタグ
				スパンタグ	−	−	otel.library.version	−	(a) スパンタグ
				スパンタグ	−	−	otel.scope.name	−	(a) スパンタグ
				スパンタグ	−	−	otel.scope.version	−	(a) スパンタグ
				スパンタグ	−	−	proposer-id	−	(a) スパンタグ
				スパンイベント	SentConsensusResultRequest.	(b) スパンイベント	−	−	−
				スパンイベント	UseOutboundStream.	(b) スパンイベント	−	−	−
				スパンイベント	UnknownMediatorId.Consen	(b) スパンイベント	−	−	−

コンポーネント	オペレーション名（短縮）	オペレーション処理概要	スパン取得レベル	スパンタグ/スパンイベント	情報説明		key（情報名）または value（情報値）		
					情報説明名	情報説明の詳細	key（情報名）	value（情報値）	key（情報名）または value（情報値）の詳細
					susResultRequest not sent.				
				スパンイベント	Stream closed. ConsensusResultRequest not sent.	(b) スパンイベント	−	−	−
Mediator	j.c.H.s.h.m.r.GrpcParticipantClient$AsyncProposeTask:run	Participant への更新可否の投票依頼（gRPC送信）です。	必須情報（通信処理のため）	スパンタグ	−	−	thread-name	−	(a) スパンタグ
				スパンタグ	−	−	otel.library.name	−	(a) スパンタグ
				スパンタグ	−	−	otel.library.version	−	(a) スパンタグ
				スパンタグ	−	−	otel.scope.name	−	(a) スパンタグ
				スパンタグ	−	−	otel.scope.version	−	(a) スパンタグ
				スパンタグ	−	−	proposer-id	−	(a) スパンタグ
				スパンイベント	Sent TerminationProposal	(b) スパンイベント	−	−	−

コンポーネント	オペレーション名（短縮）	オペレーション処理概要	スパン取得レベル	スパンタグ/スパンイベント	情報説明		key（情報名）または value（情報値）		
					情報説明名	情報説明の詳細	key（情報名）	value（情報値）	key（情報名）または value（情報値）の詳細
					Request.				
				スパンイベント	UseOutboundStream.	(b) スパンイベント	−	−	−
				スパンイベント	Stream closed. TerminationProposalRequest not sent.	(b) スパンイベント	−	−	−
Mediator	j.c.H.s.h.m.r.GrpcParticipantClient$AsyncTransactionVerifyNotifyTask:run	Participant へのトランザクション完了確認通知を送信します（gRPC 送信）。	必須情報（通信処理のため）	スパンタグ	−	−	thread-name	−	(a) スパンタグ
				スパンタグ	−	−	otel.library.name	−	(a) スパンタグ
				スパンタグ	−	−	otel.library.version	−	(a) スパンタグ
				スパンタグ	−	−	otel.scope.name	−	(a) スパンタグ
				スパンタグ	−	−	otel.scope.version	−	(a) スパンタグ
				スパンタグ	−	−	proposer-id	−	(a) スパンタグ
				スパンイベント	SentTransactionV	(b) スパンイベント	−	−	−

コンポーネント	オペレーション名（短縮）	オペレーション処理概要	スパン取得レベル	スパンタグ/スパンイベント	情報説明		key（情報名）または value（情報値）		
					情報説明名	情報説明の詳細	key（情報名）	value（情報値）	key（情報名）または value（情報値）の詳細
					erifyNotifyRequest.				
				スパンイベント	Stream closed. TransactionVerifyNotifyRequest not sent.	(b) スパンイベント	－	－	－

（凡例）－：該当しません。

付録 C.2　各スパンの詳細

HMP-PCTO が Span に付加する情報の詳細について説明します。

重要

分散トレース閲覧サービスで確認できる情報について

HMP-PCTO のコンポーネントが取得する付加情報が同一であっても、使用する分散トレース閲覧サービスによっては、名称が異なったり、確認できなかったり、HMP-PCTO が取得した付加情報以外の情報が取得されていたりする場合があります。

詳細については、使用する分散トレース閲覧サービスのドキュメントを参照してください。

Google Cloud Platform の Cloud Trace を使用する場合に取得される情報については、Google Cloud Platform の公式ドキュメントを参照してください。

New Relic を使用する場合に取得される情報については、New Relic の公式ドキュメントを参照してください。

(1) コンポーネント共通

(a) スパンタグ

表 C-2 コンポーネント共通の付加情報（スパンタグ）

項番	情報名	Jaeger を使用する場合	Google Cloud Platform の Cloud Trace を使用する場合	New Relic を使用する場合	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	○	○	○	Span が取得されたスレッド名を設定します。
4	otel.library.name	−	−	○	Span を取得する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 を取得するライブラリのバージョンを設定します。 HMP-PCTO のライブラリで取得された Span の場合、HMP-PCTO のライブラリのバージョン「V.R.S」が設定されます。
6	otel.scope.name	○	○	−	Span を取得するスコープの名称を設定します。 HMP-PCTO のスコープで取得された Span の場合、「HMP-PCTO」が設定されます。
7	otel.scope.version	○	○	−	Span を取得するスコープのバージョンを設定します。 HMP-PCTO のスコープで取得された Span の場合、HMP-PCTO のバージョン「V.R.S」が設定されます。

（凡例）

○：分散トレース閲覧サービスで確認できます。

−：分散トレース閲覧サービスで確認できません。

(b) スパンイベント

付加情報はありません。

(c) スパンコンテキスト

表 C-3 コンポーネント共通の付加情報（スパンコンテキスト）

項番	情報名	Jaeger を使用する場合	Google Cloud Platform の Cloud Trace を使用する場合	New Relic を使用する場合	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	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。
4	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。
5	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。
6	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 以降では、「1.38.0」が設定されます。

（凡例）

○：分散トレース閲覧サービスで確認できます。

−：分散トレース閲覧サービスで確認できません。

(2) 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。

(a) スパンタグ

表 C-4 REST 通信の付加情報（スパンタグ）

項番	情報名	Jaeger を使用する場合	Google Cloud Platform の Cloud Trace を使用する場合	New Relic を使用する場合	設定する情報値の説明
1	http-method	−	−	−	リクエストの HTTP メソッドを設定します。

項番	情報名	Jaeger を使用する場合	Google Cloud Platform の Cloud Trace を使用する場合	New Relic を使用する場合	設定する情報値の説明
					例：POST、GET など
2	http-url	−	−	−	リクエストの URL を設定します。
3	routed-http-url	−	−	−	スケーリング対応機能のルーティング処理による書き換え後のリクエスト URL を設定します。 スケーリング対応機能を使用する場合だけ、この項目を設定します。
4	http-status-code	−	−	−	レスポンスの HTTP ステータスコードを設定します。 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。

（凡例）

−：分散トレース閲覧サービスで確認できません。

(b) スパンイベント

付加情報はありません。

(c) スパンコンテキスト

付加情報はありません。

(3) 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。

(a) スパンタグ

表 C-5 gRPC 通信の付加情報（スパンタグ）

項番	項目名	Jaeger を使用する場合	Google Cloud Platform の Cloud Trace を使用する場合	New Relic を使用する場合	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。

項番	項目名	Jaeger を使用する場合	Google Cloud Platform の Cloud Trace を使用する場合	New Relic を使用する場合	説明
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」が設定されます。
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。 • server：gRPC サーバの場合に設定します。 • client：gRPC クライアントの場合に設定します。
11	otel.library.name	−	−	○	gRPC 通信に関する Span を取得するライブラリの名称を設定します。
12	otel.library.version	−	−	○	gRPC 通信に関する Span を取得するライブラリのバージョンを設定します。
13	otel.scope.name	○	○	−	gRPC 通信に関する Span を取得するスコープの名称を設定します。
14	otel.scope.version	○	○	−	gRPC 通信に関する Span を取得するスコープのバージョンを設定します。
15	routed-uri	○	○	−	スケーリング対応機能のルーティング機能による書き換え後のリクエスト URI を設定します。 スケーリング対応機能が有効の場合だけ、この項目を設定します。

（凡例）

○：分散トレース閲覧サービスで確認できます。

−：分散トレース閲覧サービスで確認できません。

(b) スパンイベント

表 C-6 gRPC 通信の付加情報（スパンイベント）

項番	情報説明名	意味	追加情報名	Jaeger を使用する場合	Google Cloud Platform の Cloud Trace を使用する場合	New Relic を使用する場合	設定する情報値の説明
1	message	gRPC のメッセージが送信さ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を設定します。
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 のメッセージが送信されたことを意味します。 jp.co.Hitachi.soft.hmppcto.tracing.grpc.debug パラメタが true の場合だけ取得されます。	message-type	○	○	○	電文種別を設定します。
4			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
5			message-value	○	○	○	電文内容を設定します。
6			thread-name	○	○	○	gRPC 通信のメッセージを送信したスレッド名を設定します。
7			message-size	○	○	○	gRPC 通信のメッセージのサイズを設定します。
8	sent message(hmppcto)	gRPC のメッセージが送信されたことを意味します。 jp.co.Hitachi.soft.hmppcto.tracing.grpc.debug パラメタが false の場合だけ取得されます。	message.type	○	○	○	電文種別を設定します。
9	received message(hmppcto)	gRPC のメッセージが受信されたことを意味します。 jp.co.Hitachi.soft.hmppcto.tracing.grpc.de	message-type	○	○	○	電文種別を設定します。
10			json-convert	○	○	○	JSON 形式での出力か、protobuf 形式での出力かを示します。

項番	情報説明名	意味	追加情報名	Jaegerを使用する場合	Google Cloud Platformの Cloud Traceを使用する場合	New Relicを使用する場合	設定する情報値の説明
		bugパラメタがtrueの場合だけ取得されます。					true：JSON形式 false：protobuf形式
11			message-value	○	○	○	電文内容を設定します。
12			thread-name	○	○	○	gRPC通信のメッセージを送信したスレッド名を設定します。
13			message-size	○	○	○	gRPC通信のメッセージのサイズを設定します。
14	received message(hmppcto)	gRPCのメッセージが受信されたことを意味します。 jp.co.Hitachi.soft.hmppcto.tracing.grpc.debugパラメタがfalseの場合だけ取得されます。	message.type	○	○	○	電文種別を設定します。

（凡例）

○：分散トレース閲覧サービスで確認できます。

(c) スパンコンテキスト

付加情報はありません。

(4) 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。

(a) スパンタグ

表 C-7 Orchestratorの付加情報（スパンタグ）

項番	情報名	Jaegerを使用する場合	Google Cloud Platformの Cloud Traceを使用する場合	New Relicを使用する場合	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を設定します。

項番	情報名	Jaeger を使用する場合	Google Cloud Platform の Cloud Trace を使用する場合	New Relic を使用する場合	設定する情報値の説明
					• 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
2	prepare	○	○	○	先行プリペアの実施有無を設定します。 • Y 実行 • N 未実行

（凡例）

○：分散トレース閲覧サービスで確認できます。

(b) スパンイベント

表 C-8 Orchestrator の付加情報（スパンイベント）

項番	情報説明名	意味	追加情報名	Jaeger を使用する場合	Google Cloud Platform の Cloud Trace を使用する場合	New Relic を使用する場合	設定する情報値の説明
1	start:service	Orchestrator の、HMP-PCTO のトランザクション処理の開始を意味します。	情報なし	○	○	○	−
2	end:service	Orchestrator の、HMP-PCTO のトランザクション処理の終了を意味します。	情報なし	○	○	○	−
3	start:branch	Orchestrator で @PctoTransactionBranch アノテーションを使用した際の、トランザクションブランチの開始を意味します。	情報なし	○	○	○	−

項番	情報説明名	意味	追加情報名	Jaegerを使用する場合	Google Cloud Platformの Cloud Traceを使用する場合	New Relicを使用する場合	設定する情報値の説明
4	end:branch	Orchestratorで@PctoTransactionBranchアノテーションを使用した際の、トランザクションブランチの終了を意味します。	情報なし	○	○	○	−
5	start:AP	Orchestratorのユーザアプリケーション開始を意味します。	情報なし	○	○	○	−
6	end:AP	Orchestratorのユーザアプリケーション終了を意味します。	情報なし	○	○	○	−
7	start:preprocess	Orchestratorでgrpcを通信手段とした際の、ユーザアプリケーション実行の前処理の開始を意味します。	情報なし	○	○	○	−
8	end:preprocess	Orchestratorでgrpcを通信手段とした際の、ユーザアプリケーション実行の前処理の終了を意味します。	情報なし	○	○	○	−
9	return:InnerClientCall	Orchestratorでgrpcを通信手段とした際のユーザアプリケーション実行の前処理で、InnerClientCallインスタンスの返却を意味します。	情報なし	○	○	○	−
10	start:InnerClientCall	Orchestratorでgrpcを通信手段とした際のユーザアプリケーション実行の前処理で、InnerClientCallのstartの実行を意味します。	情報なし	○	○	○	−
11	start:postprocess	Orchestratorでgrpcを通信手段とした際の、ユーザアプリケーション	情報なし	○	○	○	−

項番	情報説明名	意味	追加情報名	Jaeger を使用する場合	Google Cloud Platform の Cloud Trace を使用する場合	New Relic を使用する場合	設定する情報値の説明
		実行の後処理の開始を意味します。					
12	end:post process	Orchestrator で gRPC を通信手段とした際の、ユーザアプリケーション実行の後処理の終了を意味します。	情報なし	○	○	○	−
13	start:execute	Orchestrator のユーザアプリケーションの送信処理内で HMP-PCTO の前処理が完了し、送信処理を開始したことを意味します。	情報なし	○	○	○	−
14	end:execute	Orchestrator のユーザアプリケーションの送信処理内で送信処理が終了し、HMP-PCTO の後処理を開始したことを意味します。	情報なし	○	○	○	−
15	response:get-status-code	Orchestrator のユーザアプリケーションの送信処理内で、HMP-PCTO が応答のステータスコードを確認したことを意味します。	情報なし	○	○	○	−

（凡例）

○：分散トレース閲覧サービスで確認できます。

−：該当しません。

(c) スパンコンテキスト

付加情報はありません。

(5) Entity-Module

コンポーネント共通の情報に加えて、Entity-Module が Span に付加する情報の一覧を次に示します。

(a) スパンタグ

表 C-9 Entity-Module の付加情報（スパンタグ）

項番	情報名	Jaeger を使用する場合	Google Cloud Platform の Cloud Trace を使用する場合	New Relic を使用する場合	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を設定します。 • Commit コミット • Rollback ロールバック
2	prepare	○	○	○	先行プリペアの実施有無を設定します。 • Y 実行 • N 未実行

（凡例）

○：分散トレース閲覧サービスで確認できます。

(b) スパンイベント

表 C-10 Entity-Module の付加情報（スパンイベント）

項番	情報説明名	意味	追加情報名	Jaeger を使用する場合	Google Cloud Platform の Cloud Trace を使用する場合	New Relic を使用する場合	設定する情報値の説明
1	start:service	Entity-Module の、HMP-PCTO のトランザクション処理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-Module の、HMP-PCTO のトランザクション処理のアプリケーション終了を意味します。	情報なし	○	○	○	−
3	start:AP	Entity-Module のユーザアプリケーション開始を意味します。	情報なし	○	○	○	−
4	end:AP	Entity-Module のユーザアプリケーション終了を意味します。	情報なし	○	○	○	−

項番	情報説明名	意味	追加情報名	Jaeger を使用する場合	Google Cloud Platform の Cloud Trace を使用する場合	New Relic を使用する場合	設定する情報値の説明
5	start:pre process	Entity-Module で gRPC を通信手段とした際の、ユーザアプリケーション実行の前処理の開始を意味します。	情報なし	○	○	○	−
6	end:pre process	Entity-Module で gRPC を通信手段とした際の、ユーザアプリケーション実行の前処理の終了を意味します。	情報なし	○	○	○	−
7	return:InnerClientCall	Entity-Module で gRPC を通信手段とした際のユーザアプリケーション実行の前処理で、InnerClientCall インスタンスの返却を意味します。	情報なし	○	○	○	−
8	start:InnerClientCall	Entity-Module で gRPC を通信手段とした際のユーザアプリケーション実行の前処理で、InnerClientCall の start の実行を意味します。	情報なし	○	○	○	−
9	start:post process	Entity-Module で gRPC を通信手段とした際の、ユーザアプリケーション実行の後処理の開始を意味します。	情報なし	○	○	○	−
10	end:post process	Entity-Module で gRPC を通信手段とした際の、ユーザアプリケーション実行の後処理の終了を意味します。	情報なし	○	○	○	−
11	start:execute	Entity-Module のユーザアプリケーションの送信処理内で HMP-PCTO の前処理が完了し、送信処理を開始したことを意味します。	情報なし	○	○	○	−
12	end:execute	Entity-Module のユーザアプリケーションの送信処理内で送信処理が終了し、HMP-PCTO の後処理を開始したことを意味します。	情報なし	○	○	○	−
13	response:get-status-code	Entity-Module のユーザアプリケーションの送信処理内で、HMP-PCTO が応答のステータ	情報なし	○	○	○	−

項番	情報説明名	意味	追加情報名	Jaeger を使用する場合	Google Cloud Platform の Cloud Trace を使用する場合	New Relic を使用する場合	設定する情報値の説明
		スコードを確認したことを意味します。					

（凡例）

○：分散トレース閲覧サービスで確認できます。

－：該当しません。

(c) スパンコンテキスト

付加情報はありません。

(6) SQL Participant、TCC Participant および TP1-Bridge 共通

コンポーネント共通の情報に加えて SQL Participant、TCC Participant および TP1-Bridge 共通で Span に付加する情報の一覧を次に示します。

(a) スパンタグ

表 C–11 SQL Participant、TCC Participant および TP1-Bridge 共通の付加情報（スパンタグ）

項番	情報名	Jaeger を使用する場合	Google Cloud Platform の Cloud Trace を使用する場合	New Relic を使用する場合	設定する情報値の説明
1	consensus-value	○	○	○	分散合意結果を設定します。 • Commit コミット • Rollback ロールバック
2	mediator-id	○	○	○	Mediator の host 名と port 番号を設定します。
3	proposer-id	○	○	○	Participant 情報（<Service 名>.<Namespace 名>:IP アドレス:ホスト名）を設定します。
4	propose-values	○	○	○	投票内容を設定します。 • COMMIT_VOTE コミット投票 • ROLLBACK_VOTE ロールバック投票

項番	情報名	Jaeger を使用する場合	Google Cloud Platform の Cloud Trace を使用する場合	New Relic を使用する場合	設定する情報値の説明
5	before-transaction-status	○	○	○	遷移前のトランザクションブランチの状態を設定します。 • 0 NONE • 1 PREPARING • 2 COMMIT_VOTE • 3 READ_ONLY • 4 ROLLBACK_VOTE • 5 COMMITTING • 6 ROLLING_BACK • 7 COMMITTED • 8 ROLLED_BACK • 9 COMMIT_ERROR • 10 ROLLBACK_ERROR • 11 ROLLED_BACK_BY_RM
6	after-transaction-status	○	○	○	遷移後のトランザクションブランチの状態を設定します。 • 0 NONE • 1 PREPARING • 2 COMMIT_VOTE • 3 READ_ONLY • 4 ROLLBACK_VOTE

項番	情報名	Jaeger を使用する場合	Google Cloud Platform の Cloud Trace を使用する場合	New Relic を使用する場合	設定する情報値の説明
					• 5 COMMITTING • 6 ROLLING_BACK • 7 COMMITTED • 8 ROLLED_BACK • 9 COMMIT_ERROR • 10 ROLLBACK_ERROR • 11 ROLLED_BACK_BY_RM

（凡例）

○：分散トレース閲覧サービスで確認できます。

(b) スパンイベント

表 C-12 SQL Participant、TCC Participant および TP1-Bridge 共通の付加情報（スパンイベント）

項番	情報説明名	意味	追加情報名	Jaeger を使用する場合	Google Cloud Platform の Cloud Trace を使用する場合	New Relic を使用する場合	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	○	○	○	−

項番	情報説明名	意味	追加情報名	Jaeger を使用する場合	Google Cloud Platform の Cloud Trace を使用する場合	New Relic を使用する場合	設定する情報値の説明
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	The consensus has not been terminated for a long time.	分散合意の回収時間を経過したことを意味しています。	情報なし	○	○	○	−
7	start:termination-judge	Mediator への投票開始を意味します。	情報なし	○	○	○	−
8	end:termination-judge	Mediator への投票終了を意味します。	情報なし	○	○	○	−

（凡例）

○：分散トレース閲覧サービスで確認できます。

−：該当しません。

(c) スパンコンテキスト

付加情報はありません。

(7) SQL-Participant 固有

コンポーネント共通、SQL Participant、TCC Participant および TP1-Bridge 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。

(a) スパンタグ

表 C-13 SQL-Participant の付加情報（スパンタグ）

項番	情報名	Jaeger を使用する場合	Google Cloud Platform の Cloud Trace を使用する場合	New Relic を使用する場合	設定する情報値の説明
1	recover-pod-hostname※	○	○	○	recover 担当となった Pod のホスト名を設定します。
2	recover-pod-ip※	○	○	○	recover 担当となった Pod の IP を設定します。
3	prepare	○	○	○	null

項番	情報名	Jaeger を使用する場合	Google Cloud Platform の Cloud Trace を使用する場合	New Relic を使用する場合	設定する情報値の説明
					Entity-Module からトランザクションブランチの Prepare 要求を受けたことを意味します。
4	cancel	○	○	○	null Entity-Module からトランザクションブランチのキャンセル要求を受けたことを意味します。

（凡例）

○：分散トレース閲覧サービスで確認できます。

注※

この付加情報はスケーリング対応機能を使用する場合に、一定間隔で繰り返し実行される定期的な Recovery 機能で出力します。出力契機は次のとおりです。

jp.co.Hitachi.soft.hmppcto.tracing.debug パラメタが false の場合：

・jp.co.Hitachi.soft.hmppcto.recover.interval-in-millis パラメタが 30 分未満のとき、最後の出力から 30 分経過後の定期実行時に出力

・jp.co.Hitachi.soft.hmppcto.recover.interval-in-millis パラメタが 30 分以上のとき、定期実行のたびに常に出力

・定期実行時に Recovery 対象のトランザクションがあるときに、必ず出力

jp.co.Hitachi.soft.hmppcto.tracing.debug パラメタが true の場合：

定期実行のたびに常に出力

(b) スパンイベント

表 C–14 SQL-Participant の付加情報（スパンイベント）

項番	情報説明名	意味	追加情報名	Jaeger を使用する場合	Google Cloud Platform の Cloud Trace を使用する場合	New Relic を使用する場合	設定する情報値の説明
1	start:jdbc <JDBC Driver のクラス名>.<JDBC Driver のメソッド名>	Participant での JDBC メソッド開始を意味します。 jp.co.Hitachi.soft.hmppcto.tracing.jdbc.debug パラメタが true の場合だけ取得されます。	情報なし	○	○	○	−
2	end:jdbc	Participant での JDBC メソッド終了を意味します。 jp.co.Hitachi.soft.hmppcto.tracing.jdbc.debug パラメタが true の場合だけ取得されます。	情報なし	○	○	○	−

項番	情報説明名	意味	追加情報名	Jaeger を使用する場合	Google Cloud Platform の Cloud Trace を使用する場合	New Relic を使用する場合	設定する情報値の説明
3	start:recover-job※	スケーリング対応機能を使用する場合の、未決着トランザクションの Recovery 機能の処理開始を意味します。	情報なし	○	○	○	−
4	end:recover-job※	スケーリング対応機能を使用する場合の、未決着トランザクションの Recovery 機能の処理終了を意味します。	情報なし	○	○	○	−
5	old-pod-uuid	トランザクションの XID に付与していた Pod の UUID です。	情報なし	○	○	○	−
6	counter※	スケーリング対応機能を使用する場合の、未決着トランザクションの Recovery 機能の実行回数です。 この付加情報は複数のスパンで同一の値を出力します。この値とトレースを出力している Pod のホスト名を用いてスパン同士を一意に紐づけする目的で出力しています。	情報なし	○	○	○	上限を 9,999 とし、超えた場合は 1 に戻ります。
7	pod-info-list※	スケーリング対応機能を使用する場合の、未決着トランザクションの Recovery 機能の実行時に Kubernetes の API によって Pod を取得した結果です。	情報なし	○	○	○	取得した Pod のホスト名と UUID の一覧です。

（凡例）

○：分散トレース閲覧サービスで確認できます。

−：該当しません。

注※

この付加情報はスケーリング対応機能を使用する場合に、一定間隔で繰り返し実行される定期的な Recovery 機能で出力します。出力契機は次のとおりです。

jp.co.Hitachi.soft.hmppcto.tracing.debug パラメタが false の場合：

・jp.co.Hitachi.soft.hmppcto.recover.interval-in-millis パラメタが 30 分未満のとき、最後の出力から 30 分経過後の定期実行時に出力

・jp.co.Hitachi.soft.hmppcto.recover.interval-in-millis パラメタが 30 分以上のとき、定期実行のたびに常に出力

・定期実行時に Recovery 対象のトランザクションがあるときに、必ず出力

jp.co.Hitachi.soft.hmppcto.tracing.debug パラメタが true の場合：

定期実行のたびに常に出力

(c) スパンコンテキスト

付加情報はありません。

(8) TCC-Participant 固有

コンポーネント共通、SQL Participant、TCC Participant および TP1-Bridge 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。

(a) スパンタグ

付加情報はありません。

(b) スパンイベント

表 C-15 TCC-Participant の付加情報（スパンイベント）

項番	情報説明名	意味	追加情報名	Jaeger を使用する場合	Google Cloud Platform の Cloud Trace を使用する場合	New Relic を使用する場合	設定する情報値の説明
1	start:try	Try 開始を意味します。	情報なし	○	○	○	−
2	end:try	Try 終了を意味します。	情報なし	○	○	○	−
3	start:confirm	Confirm 開始を意味します。	情報なし	○	○	○	−
4	end:confirm	Confirm 終了を意味します。	情報なし	○	○	○	−
5	start:cancel	Cancel 開始を意味します。	情報なし	○	○	○	−
6	end:cancel	Cancel 終了を意味します。	情報なし	○	○	○	−

（凡例）

○：分散トレース閲覧サービスで確認できます。

−：該当しません。

(c) スパンコンテキスト

付加情報はありません。

(9) TP1-Bridge 固有

コンポーネント共通、SQL Participant、TCC Participant および TP1-Bridge 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。

(a) スパンタグ

表 C–16 TP1-Bridge の付加情報（スパンタグ）

項番	情報名	Jaeger を使用する場合	Google Cloud Platform の Cloud Trace を使用する場合	New Relic を使用する場合	設定する情報値の説明
1	prepare	○	○	○	null Entity-Module からトランザクションブランチの Prepare 要求を受けたことを意味します。
2	cancel	○	○	○	null Entity-Module からトランザクションブランチのキャンセル要求を受けたことを意味します。

（凡例）

○：分散トレース閲覧サービスで確認できます。

－：該当しません。

(b) スパンイベント

表 C–17 TP1-Bridge の付加情報（スパンイベント）

項番	情報説明名	意味	追加情報名	Jaeger を使用する場合	Google Cloud Platform の Cloud Trace を使用する場合	New Relic を使用する場合	設定する情報値の説明
1	start:TP1/Client/J(<TP1/Client/J のメソッド名>)	TP1/Client/J のメソッドの呼び出し開始を意味します。	情報なし	○	○	○	－
2	end:TP1/Client/J(<TP1/Client/J のメソッド名>)	TP1/Client/J のメソッドの呼び出し終了を意味します。	情報なし	○	○	○	－
3	start:TP1-Converter	TP1-Converter の呼び出し開始を意味します。	情報なし	○	○	○	－
4	end:TP1-Converter	TP1-Converter の呼び出し終了を意味します。	情報なし	○	○	○	－

（凡例）

○：分散トレース閲覧サービスで確認できます。

－：該当しません。

(c) スパンコンテキスト

付加情報はありません。

(10) 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。

(a) スパンタグ

表 C-18 Mediator の付加情報（スパンタグ）

項番	情報名	Jaeger を使用する場合	Google Cloud Platform の Cloud Trace を使用する場合	New Relic を使用する場合	設定する情報値の説明
1	consensus-value	○	○	○	Participant での分散合意結果を設定します。 • Commit コミット • Rollback ロールバック
2	prepare-priority	○	○	○	投票要求を送信する際の優先度を設定します。
3	proposer-id	○	○	○	Participant 情報（<Service 名>.<Namespace 名>:IP アドレス:ホスト名）を意味します。

（凡例）

○：分散トレース閲覧サービスで確認できます。

(b) スパンイベント

表 C-19 Mediator の付加情報（スパンイベント）

項番	情報説明名	意味	追加情報名	Jaeger を使用する場合	Google Cloud Platform の Cloud Trace を使用する場合	New Relic を使用する場合	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	○	○	○	－

項番	情報説明名	意味	追加情報名	Jaeger を使用する場合	Google Cloud Platform の Cloud Trace を使用する場合	New Relic を使用する場合	設定する情報値の説明
2	end:consensuslog <追加情報>	ConsensusLog へのアクセス（I/O）終了を意味します。 ConsensusLog を追加する場合と削除する場合の、スパンイベント、出力内容および意味は次のとおりです。 • end:consensuslog ConsensusLog の追加または削除処理が終了しました。ファイルの読み込み前に例外が発生しました。 • end:consensuslog read ConsensusLog の追加または削除処理が終了しました。ファイルの書き込み前に読み込みをしました。 • end:consensuslog not read ConsensusLog の追加または削除処理が終了しました。ファイルの書き込み前に読み込みをしませんでした。 ConsensusLog を読み込む場合と ConsensusLog の Participant 情報の更新をする場合の、スパンイベント、出力内容および意味は次のとおりです。 • end:consensuslog ConsensusLog の読み込みまたは Participant 情報の更新処理が終了しました。	情報なし	○	○	○	−
3	consensusid:<ConsensusId の値>	ConsensusId の値を意味します。	情報なし	○	○	○	−

項番	情報説明名	意味	追加情報名	Jaeger を使用する場合	Google Cloud Platform の Cloud Trace を使用する場合	New Relic を使用する場合	設定する情報値の説明
4	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。 []で囲まれた中には分散トレース取得時に Participant から受信した投票内容を設定します。
5	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	○	○	○	−
6	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>.<Namespace 名>:IP アドレス:ホスト名）の一覧を設定します。 一覧は<Service 名>.<Namespace 名>:IP アドレス:ホスト名を","（コンマ）区切りで設定します。

項番	情報説明名	意味	追加情報名	Jaeger を使用する場合	Google Cloud Platform の Cloud Trace を使用する場合	New Relic を使用する場合	設定する情報値の説明
							[]で囲まれた中には分散トレース取得時に Participant から受信した投票内容を設定します。
7	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	○	○	○	−
8	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>.<Namespace 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>.<Namespace 名>）の一覧を設定します。 一覧は<Service 名>.<Namespace 名>を","（コンマ）区切りで設定します。
9	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>.<Namespace 名>））の一覧の保存の終了を意味します。	情報なし	○	○	○	−
10	start:proposallog-delete	Proposal Log の削除開始を意味します。	proposal-log-before-size	○	○	○	Proposal Log の削除前の件数を設定します。
11	end:proposallog-delete	Proposal Log の削除終了を意味します。	proposal-log-	○	○	○	Proposal Log の削除後の件数を設定します。

項番	情報説明名	意味	追加情報名	Jaeger を使用する場合	Google Cloud Platform の Cloud Trace を使用する場合	New Relic を使用する場合	設定する情報値の説明
			after-size				
12	start: terminated-transaction-verify	ConsensusLog のトランザクション完了確認通知の送信開始を意味します。	consensus log size	○	○	○	完了確認通知を送信した ConsensusLog の数を設定します。
13	end: terminated-transaction-verify	ConsensusLog のトランザクション完了確認通知の送信終了を意味します。	情報なし	○	○	○	−
14	send-consensus-value:<更新可否の決着結果>	Participant へ送信した更新可否の決着結果を示します。	情報なし	○	○	○	−
15	Sent ConsensusResultRequest.	Participant への更新可否の決着結果通知の送信を意味します。	情報なし	○	○	○	−
16	Unknown MediatorId. ConsensusResultRequest not sent.	Mediator のホスト名の解決に失敗したため、Participant への更新可否の決着結果通知の送信に失敗しました。	情報なし	○	○	○	−
17	Stream closed. ConsensusResultRequest not sent.	Participant へのストリームがクローズしているため、Participant への更新可否の決着結果通知の送信に失敗しました。	情報なし	○	○	○	−
18	Sent TerminationProposalRequest.	Participant への更新可否の投票依頼の送信を意味します。	情報なし	○	○	○	−
19	Use Outbound Stream.	Participant へのストリームがクローズしているため、異なるストリームを使用します。	情報なし	○	○	○	−
20	Stream closed. TerminationP	Participant へのストリームがクローズしているため、Participant への更新可否の	情報なし	○	○	○	−

項番	情報説明名	意味	追加情報名	Jaeger を使用する場合	Google Cloud Platform の Cloud Trace を使用する場合	New Relic を使用する場合	設定する情報値の説明
	roposalRequest not sent.	投票依頼の送信に失敗しました。					
21	Sent TransactionVerifyNotifyRequest.	Participant へトランザクション完了確認通知を送信します。	情報なし	○	○	○	−
22	Stream closed. TransactionVerifyNotifyRequest not sent.	Participant へのストリームがクローズしているため、Participant へのトランザクション完了確認通知の送信に失敗しました。	情報なし	○	○	○	−

（凡例）

○：分散トレース閲覧サービスで確認できます。

−：該当しません。

(c) スパンコンテキスト

付加情報はありません。

(11) EADS（通常版限定）

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。

(a) スパンタグ

付加情報はありません。

(b) スパンイベント

表 C-20 EADS の付加情報（スパンイベント）

項番	情報説明名	意味	追加情報名	Jaeger を使用する場合	Google Cloud Platform の Cloud Trace を使用する場合	New Relic を使用する場合	設定する情報値の説明
1	start:EADS API	EADS の API が呼び出されたこと	API-name	○	○	○	呼び出す EADS API 名を設定します。 • create

項番	情報説明名	意味	追加情報名	Jaeger を使用する場合	Google Cloud Platform の Cloud Trace を使用する場合	New Relic を使用する場合	設定する情報値の説明
		を意味します。					create の EADS API • get get の EADS API
2			consensus	○	○	○	補助合意提案値を、create の場合だけ設定します。 • Commit コミット • Rollback ロールバック
3	end:EADS API	EADS の API の実行が終了したことを意味します。	API-name	○	○	○	実行が終わった EADS API 名を設定します。 • create create の EADS API • get get の EADS API
4			get	○	○	○	取得した補助合意提案値を、get の場合だけ設定します。 • 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 名を設定します。 • 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を設定します。

（凡例）

○：分散トレース閲覧サービスで確認できます。

(c) スパンコンテキスト

付加情報はありません。

(12) XA 関数（SQL-Participant 限定）

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。

(a) スパンタグ

付加情報はありません。

(b) スパンイベント

表 C-21 XA 関数の付加情報（スパンイベント）

項番	情報説明名	意味	追加情報名	Jaeger を使用する場合	Google Cloud Platform の Cloud Trace を使用する場合	New Relic を使用する場合	設定する情報値の説明
1	start:xa function	XA 関数が呼び出されたことを意味します。	xa-name	○	○	○	呼び出す XA 関数名を設定します。 • 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
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 関数名を設定します。 • 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

項番	情報説明名	意味	追加情報名	Jaeger を使用する場合	Google Cloud Platform の Cloud Trace を使用する場合	New Relic を使用する場合	設定する情報値の説明
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 関数名を設定します。 • 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を設定します。

（凡例）

○：分散トレース閲覧サービスで確認できます。

(c) スパンコンテキスト

付加情報はありません。

付録 D　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 D-1　メトリクスの Meter 種別

Meter 種別	説明
Counter	単調に増加する値
FunctionCounter	単調増加関数を追跡するカウンター
Gauge	上下する値
MultiGauge	複数の Gauge
Timer	多数の短時間実行イベント

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 D-2　システムの利用状況、および Java VM の利用状況

Prometheus 上の表記	Meter 種別	tag※	単位	説明
jvm_gc_memory_allocated_bytes_total	Counter	−	bytes	ある GC の実施後から、次の GC の実施までの、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	Counter	−	bytes	ある GC の実施前と GC の実施後での、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。
jvm_memory_used_bytes	Gauge	−	bytes	Java VM の使用メモリ量です。
jvm_memory_max_bytes	Gauge	−	bytes	Java VM で使用できるメモリの最大量です。
process_files_open_files	Gauge	−	files	開いているファイル記述子の数です。
process_files_max_files	Gauge	−	files	ファイル記述子の最大数です。
system_cpu_usage	Gauge	−	−	システム（Kubernetes の場合、コンテナ環境）の最近の 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。
process_cpu_usage	Gauge	−	−	Java VM プロセスの最近の 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。

（凡例）−：該当しません。

注※　独自 Binder の場合だけ記載しています。

表 D-3　HMP-PCTO の性能を評価するための指標

Prometheus 上の表記	Meter 種別	tag※	単位	説明
hmppcto_mediator_consensus_logs	MultiGauge	id=＜ホスト名＞ status=＜状態＞	logs	ConsensusLog の状態ごとの件数です。
hmppcto_mediator_proposals_logs	MultiGauge	id=＜ホスト名＞ status=＜状態＞	logs	Proposal Log の状態ごとの件数です。
grpc_server_processing_duration_seconds_maxs	Timer	－	seconds	gRPC サーバが呼び出し完了に掛かった時間の統計情報です。
grpc_client_processing_duration_seconds_maxs	Timer	－	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の統計情報（ネットワーク遅延を含む）です。
hmppcto_asynctask_duration_seconds	Timer	name=＜クラス名＞	seconds	非同期タスク（HMP-PCTO 内部で実行する非同期送信処理）の所要時間の統計情報です。
hmppcto_asynctask_operations_total	FunctionCounter	name=＜クラス名＞ kind=unique	operations	作成された非同期タスク（HMP-PCTO 内部で実行する非同期送信処理）の数です。
hmppcto_asynctask_operations_total	FunctionCounter	name=＜クラス名＞ kind=success	operations	正常に終了した非同期タスク（HMP-PCTO 内部で実行する非同期送信処理）の数です。
hmppcto_asynctask_operations_total	FunctionCounter	name=＜クラス名＞ kind=failure	operations	再試行された非同期タスク（HMP-PCTO 内部で実行する非同期送信処理）の数と失敗した非同期タスクの数です。
hmppcto_asynctask_operations_total	FunctionCounter	name=＜クラス名＞ kind=retry	operations	実行中に例外が発生したため、再試行する予定の非同期タスク（HMP-PCTO 内部で実行する非同期送信処理）の数です。
hmppcto_asynctask_queued_operations	Gauge	name=＜クラス名＞	operations	キュー内の非同期タスク（HMP-PCTO 内部で実行する非同期送信処理）の数です。
hmppcto_participant_transactions_inProcess_operations	MultiGauge	id=＜ホスト名＞	operations	処理中のトランザクションブランチ数です。

Prometheus 上の表記	Meter 種別	tag※	単位	説明
		status=＜状態＞		
hmppcto_participant_transactions_terminated_operations	MultiGauge	id=＜ホスト名＞ status=＜状態＞	operations	完了したトランザクションブランチ数です。
hmppcto_mediator_consensus_duration_seconds	Timer	id=＜ホスト名＞	seconds	ConsensusLog の処理時間です。
hmppcto_participant_eads_duration_seconds_count	Timer	id=＜ホスト名＞	seconds	EADS API の処理時間です。Amazon Web Services 環境の場合、およびトライアル版の場合は出力できません。
hmppcto_participant_eads_duration_seconds_max	Timer	id=＜ホスト名＞	seconds	EADS API の処理時間です。トライアル版では出力できません。
hmppcto_participant_eads_duration_seconds_sum	Timer	id=＜ホスト名＞	seconds	EADS API の処理時間です。Amazon Web Services 環境の場合、およびトライアル版の場合は出力できません。
hmppcto_executor_completed_tasks_total	FunctionCounter	－	tasks	実行が完了したタスクの概算総数です。
hmppcto_executor_active_threads	Gauge	－	threads	アクティブにタスクを実行しているスレッドのおおよその数です。
hmppcto_executor_queued_tasks	Gauge	－	tasks	実行待ちのタスクのおおよその数です。
hmppcto_executor_queue_remaining_tasks	Gauge	－	tasks	このキューがブロックしないで完全に受け入れることができる追加要素の数です。
hmppcto_executor_pool_size_threads	Gauge	－	threads	プール内の現在のスレッド数です。
hmppcto_executor_pool_core_threads	Gauge	－	threads	プールのスレッドのコア数です。
hmppcto_executor_pool_max_threads	Gauge	－	threads	プール内のスレッドの最大許容数です。
hmppcto_executor_active	Gauge	－	threads	アクティブにタスクを実行しているスレッドのおおよその数です。
hmppcto_executor_queued	Gauge	－	tasks	実行待ちのタスクのおおよその数です。
hmppcto_executor_running	Gauge	－	threads	ブロックされていない、タスクの結合を待機しているスレッド、または他の管理された同期スレッドの推定数です。
hmppcto_executor_steals_total	FunctionCounter	－	tasks	1 つのスレッドの作業キューから盗まれたタスクの総数の推定値です。
tomcat_threads_busy_threads	Gauge	－	threads	Spring Boot 組み込み Tomcat で、ビジー状態のスレッド数です。

Prometheus 上の表記	Meter 種別	tag※	単位	説明
tomcat_threads_current_threads	Gauge	－	threads	Spring Boot 組み込み Tomcat で管理される現在のスレッド数です。
hmppcto_participant_transactions_inProcess_operations	MultiGauge	id=＜ホスト名＞ status=＜状態＞	operations	処理中のトランザクションブランチ数です。

（凡例）－：該当しません。

注※　独自 Binder の場合だけ記載しています。

表 D-4　Jaeger 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
otelcol_exporter_queue_size	Exporter の内部キュー内に存在するアイテム数です。	Jaeger
otelcol_exporter_enqueue_failed	Exporter 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 の数です。	Jaeger

表 D-5　Prometheus 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
up	スクレイプに成功したときは「1」、失敗したときは「0」を示すメトリックです。HMP-PCTO の Kubernetes アプリケーションの稼働監視に利用できます。	Prometheus-Server
scrape_duration_seconds	スクレイプに掛かった時間を示すメトリックです。通常の運用では使用しません。 スクレイプが想定した時間内に終わらなかった場合の調査などで使用します。	Prometheus-Server
scrape_samples_post_metric_relabeling	メトリックのリラベル後に残ったサンプル数を示すメトリックです。通常の運用では使用しません。 環境構築時のデータ数の確認で使用します。	Prometheus-Server
scrape_samples_scraped	HMP-PCTO の Kubernetes アプリケーションがスクレイプで返却したサンプル数を示すメトリックです。通常の運用では使用しません。 環境構築時のデータ数の確認で使用します。	Prometheus-Server
scrape_series_added	新しく生成した系列のおおよその数を示すメトリックです。通常の運用では使用しません。	Prometheus-Server

付録 E　システム運用時の注意事項

システム運用時の注意事項について説明します。

付録 E.1　システム時刻の補正時の注意事項

Pod 内のシステム時刻は動作するワーカーノードのシステム時刻に依存します。時刻を補正するソフトウェア（NTP など）を使用してワーカーノードのシステム時刻を補正する場合、次の点に注意して運用してください。システム時刻補正の詳細については、使用しているソフトウェアのマニュアルを参照してください。

- NTP プログラムの slew モードのように、時刻飛びが発生しないように時刻のずれを徐々に補正するようにしてください。
- 一度に秒単位、またはそれより大きな単位で時刻補正が必要な場合は、HMP-PCTO の Pod を停止してから時刻を補正してください。

また、時刻補正によってシステム時刻が戻ったり進んだりすると、次のような事象が発生することがあります。

- トランザクションや通信などで、タイムアウトが想定より遅くなったり早くなったりする
- ログやトレース情報の時系列が崩れる

付録 F　このマニュアルの参考情報

付録 F.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

HMP-PCTO

取扱説明書「Hitachi Microservices Platform - Paxos Commit Transaction Orchestrator　テクニカルガイド」

次の内容について説明しています。

- システムの概要
- Orchestrator/Entity-Module 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

関連製品

- 取扱説明書「Hitachi Elastic Application Data Store　コンテナオーケストレーションツール対応機能」
- 取扱説明書「Hitachi Microservices Platform - Application Distributed Integration Framework　ユーザガイド」

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 Elastic Application Data Store　コンテナオーケストレーションツール対応機能	EADS コンテナオーケストレーションツール対応機能
Hitachi Microservices Platform - Paxos Commit Transaction Orchestrator　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）	

マニュアル名	表記
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
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 F.2　マニュアルでの表記

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・サービス名
AKS	Azure Kubernetes Service
Amazon ECR	Amazon Elastic Container Registry
Amazon EFS	Amazon Elastic File System
Amazon EKS	Amazon Elastic Kubernetes Service
EADS	Hitachi Elastic Application Data Store
HMP-PCTO	Hitachi Microservices Platform - Paxos Commit Transaction Orchestrator
RHEL	Red Hat Enterprise Linux
RHEL/UBI	Red Hat Universal Base Image
UBI	
Tomcat	Apache Tomcat

付録 F.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
APM	Application Performance Monitoring
BLOB	Binary Large Object
CA	Certification Authority
CLI	Command Line Interface
CPU	Central Processing Unit
CSV	Comma Separated Values
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

英略語	英字での表記
KVS	Key-Value store
NFS	Network File System
NRQL	New Relic Query Language
NTP	Network Time Protocol
OS	Operating System
OTLP	OpenTelemetry Protocol
PEM	Privacy Enhanced Mail
PV	Persistent Volume
PVC	Persistent Volume Claim
RAS	RASIS
RASIS	Reliability Availability Serviceability Integrity Security
RDBMS	Relational Database Management System
REST	Representational State Transfer
RPC	Remote Procedure Call
RPM	RPM Package Manager
SCC	Security Context Constraints
SPP	Service Providing Program
TCC	Try、Confirm、Cancel
TCP	Transmission Control Protocol
TLS	Transport Layer Security
TPS	Transaction Per Second
URI	Uniform Resource Identifier

付録 F.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 G　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 G-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
Alternate-Service	呼び出し元に応じて、Orchestrator-Service か Entity-Service に役割が変わる Kubernetes アプリケーションおよびコンテナ
Alternate-Service(SQL)	Alternate-Service 上で JDBC Proxy Driver を使用して DB にアクセスする Kubernetes アプリケーションおよびコンテナ
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	Orchestrator、Participant と連携してトランザクションの伝搬をする HMP-PCTO の機能
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 Kubernetes アプリケーションおよびコンテナ
Entity-Service(SQL)	Entity-Service 上で JDBC Proxy Driver を使用して DB にアクセスする Kubernetes アプリケーションおよびコンテナ
Entity-Service(TCC)	Entity-Service 上で TCC パターンを使用してリソースにアクセスする Kubernetes アプリケーションおよびコンテナ
Ext Cons	• 補助合意のこと • External Consensus の略

用語	説明
External Consensus	補助合意のこと
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
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 HMP-PCTO 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	• Orchestrator、Participant と連携して Paxos Commit を実現する HMP-PCTO の機能。またはその機能を使った Kubernetes アプリケーションおよびコンテナ • 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	Mediator、Entity-Module と連携してトランザクションの制御をする HMP-PCTO の機能
Orchestrator-Service	Client システムからの要求を受け付けて、各サービスを呼び出して利用することで業務を実行する Kubernetes アプリケーションおよびコンテナ
Orchestrator-Service(SQL)	Orchestrator-Service 上で JDBC Proxy Driver を使用して DB にアクセスする Kubernetes アプリケーションおよびコンテナ
Participant	• Mediator、Entity-Module と連携して Paxos Commit を実現する HMP-PCTO の機能 • 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

用語	説明
Relay-Service	HMP-PCTO の機能を用いたリソースの更新は行わない、トランザクションの伝搬を中継するアプリケーション
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	• Participant の特性を持ち X/Open の XA 仕様に準拠したトランザクション制御を実現する HMP-PCTO の機能 • 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	• Participant の特性を持ち TCC パターンを実現する HMP-PCTO の機能 • Try、Confirm、Cancel、Recover に対応したインタフェースを提供。ユーザがインタフェースを実装することで、TCC-Participant はリソースを管理する
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。またはその機能を使った Kubernetes アプリケーションおよびコンテナ • Entity-Module、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

用語	説明
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
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
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
スパンイベント	• Span 内の構造化ログメッセージのこと • Span に付加されるイベント名、タイムスタンプ、およびスパンタグで構成されるメタデータ • Jaeger の「Logs」 • Google Cloud Platform の Cloud Trace の「Time Events」 • New Relic の「Span event」
スパンコンテキスト	• Span に関するコンテキストのこと • Span を生成したサービスやアプリケーションのプロセスに関する Key-Value のメタデータ • Jaeger の「Process」 • Google Cloud Platform の Cloud Trace の「トレースラベル」 • New Relic の「Attribute」
スパンタグ	• Span の属性（アトリビュート）のこと • Span に付加される Key-Value のメタデータ • Jaeger の「Tags」 • Google Cloud Platform の Cloud Trace の「トレースラベル」 • New Relic の「Attribute」
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
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
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

索引

記号

--storage.tsdb.retention.time パラメタ〔Prometheus〕 905

A

absolute(int row)メソッド〔PctoResultSet クラス〕 763

accesslog.dir パラメタ〔server.undertow〕 864

accesslog.enabled パラメタ〔server.tomcat〕 862

accesslog.enabled パラメタ〔server.undertow〕 863

accesslog.locale パラメタ〔server.tomcat〕 862

accesslog.max-days パラメタ〔server.tomcat〕 863

accesslog.pattern パラメタ〔server.tomcat〕 862

accesslog.pattern パラメタ〔server.undertow〕 864

add.class.path パラメタ〔uCosminexus Application Server の usrconf.cfg〕 917

addBatch()メソッド〔PctoPreparedStatement クラス〕 715

addBatch(java.lang.String sql)メソッド〔PctoStatement クラス〕 698

affinity パラメタ〔Mediator〕 870

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〕 914

B

basedir パラメタ〔server.tomcat〕 862

branch.qualifier.list パラメタ〔Entity-Module〕 846

branch.qualifier.list パラメタ〔Orchestrator〕 843

branch.qualifier.propagation-order パラメタ〔Entity-Module〕 846

C

caches.name パラメタ〔EADS〕 895

caches.pvc.accessModes パラメタ〔EADS〕 895

caches.pvc.storageClassName パラメタ〔EADS〕 895

client.proposer.type パラメタ〔Entity-Module〕 846

close()メソッド〔PctoConnection クラス〕 683

close()メソッド〔PctoResultSet クラス〕 764

close()メソッド〔PctoStatement クラス〕 699

closeOnCompletion()メソッド〔PctoStatement クラス〕 700

consensusLog.io.blockSize パラメタ〔Mediator〕 873

consensusLog.io.queueSize パラメタ〔Mediator〕 873

consensusLog.io.threadPoolSize パラメタ〔Mediator〕 873

consensusLog.io.watchInSeconds パラメタ〔Mediator〕 873

consensusLog.size パラメタ〔Mediator〕 872

consensusLog.storageClassName パラメタ〔Mediator〕 874

converter.connectTimeInSeconds パラメタ〔TP1-Bridge〕 883

converter.host パラメタ〔TP1-Bridge〕 882

converter.port パラメタ〔TP1-Bridge〕 882

converter.watchTimeInSeconds パラメタ〔TP1-Bridge〕 883

createArrayOf(java.lang.String typeName, java.lang.Object[] elements)メソッド〔PctoConnection クラス〕 684

createSQLXML()メソッド〔PctoConnection クラス〕 685

createStatement()メソッド〔PctoConnection クラス〕 686

createStatement(int resultSetType, int resultSetConcurrency)メソッド〔PctoConnection クラス〕 687

D

dependencyCheck.connectTimeoutSeconds パラメタ〔Ext-Cons〕 902

dependencyCheck.connectTimeoutSeconds パラメタ〔Mediator〕 868

dependencyCheck.connectTimeoutSeconds パラメタ〔TP1-Bridge〕 879

dependencyCheck.enabled パラメタ〔Mediator〕 867

dependencyCheck.imagePullPolicy パラメタ〔Ext-Cons〕 902

dependencyCheck.imagePullPolicy パラメタ〔Mediator〕 868

dependencyCheck.imagePullPolicy パラメタ〔TP1-Bridge〕 879

dependencyCheck.image パラメタ〔Ext-Cons〕 901

dependencyCheck.image パラメタ〔Mediator〕 867

dependencyCheck.image パラメタ〔TP1-Bridge〕 879

dependencyCheck.readTimeoutSeconds パラメタ〔Ext-Cons〕 902

dependencyCheck.readTimeoutSeconds パラメタ〔Mediator〕 868

dependencyCheck.readTimeoutSeconds パラメタ〔TP1-Bridge〕 879

dependencyCheck.retryCount パラメタ〔Ext-Cons〕 902

dependencyCheck.retryCount パラメタ〔Mediator〕 868

dependencyCheck.retryCount パラメタ〔TP1-Bridge〕 879

dependencyCheck.retryIntervalSeconds パラメタ〔Ext-Cons〕 902

dependencyCheck.retryIntervalSeconds パラメタ〔Mediator〕 868

dependencyCheck.retryIntervalSeconds パラメタ〔TP1-Bridge〕 879

dependencyCheck.targets パラメタ〔Ext-Cons〕 902

dependencyCheck.targets パラメタ〔Mediator〕 868

dependencyCheck.targets パラメタ〔TP1-Bridge〕 879

driver-class-name パラメタ〔JDBC Proxy Driver〕 840

E

eads.cache.disk.filenum パラメタ〔EADS〕 899

eads.cache.disk.filesize パラメタ〔EADS〕 899

eads.cache.disk.multipleWritePrevention.enable パラメタ〔EADS〕 899

eads.cache.eviction.keyCount パラメタ〔EADS〕 899

eads.cache.eviction.policy パラメタ〔EADS〕 899

eads.cache.key.maxsize パラメタ〔EADS〕 898

eads.cache.keyCount パラメタ〔EADS〕 898

eads.cache.type パラメタ〔EADS〕 899

eads.client.＜接続先 EADS サーバ＞.address パラメタ〔EADS〕 900

eads.client.＜接続先 EADS サーバ＞.port パラメタ〔EADS〕 900

eads.client.connect.servers パラメタ〔EADS〕 900

eads.client.connectionPool.poolsize パラメタ〔EADS〕 900

eads.client.logger.dir パラメタ〔EADS〕 900

eads.client.properties パラメタ〔TP1-Bridge〕 891

eads.failureDetector.connection.timeout パラメタ〔EADS〕 897

eads.failureDetector.heartbeat.timeout パラメタ〔EADS〕 897

eads.failureDetector.read.timeout パラメタ〔EADS〕 897

eads.java.external.heapsize パラメタ〔EADS〕 898

eads.java.heapsize パラメタ〔EADS〕 898

eads.java.nio.maxCachedBufferSize パラメタ〔EADS〕 897

eads.logger.message.console.enable パラメタ〔EADS〕 897

eads.logger.multipleWritePrevention.enable パラメタ〔EADS〕 897
eads.node.< EADS サーバ ID >.address パラメタ〔EADS〕 899
eads.replication.external.heapsize パラメタ〔EADS〕 898
eads.replication.factor パラメタ〔EADS〕 898
eads.server.cache.maxExecuteThreads パラメタ〔EADS〕 897
eads.server.connection.keepAlive.timeout パラメタ〔EADS〕 897
eads.server.maxConnections パラメタ〔EADS〕 896
ejbserver.application.userlog.CJLogHandler.conf1.level パラメタ〔uCosminexus Application Server の usrconf.properties〕 916
ejbserver.application.userlog.Logger.jp.co.Hitachi.soft.hmppcto.converter.handlers パラメタ〔uCosminexus Application Server の usrconf.properties〕 916
ejbserver.application.userlog.Logger.jp.co.Hitachi.soft.hmppcto.converter.level パラメタ〔uCosminexus Application Server の usrconf.properties〕 916
ejbserver.application.userlog.loggers パラメタ〔uCosminexus Application Server の usrconf.properties〕 916
execute()メソッド〔PctoPreparedStatement クラス〕 716
execute(java.lang.String sql, int autoGeneratedKeys)メソッド〔PctoStatement クラス〕 701
execute(java.lang.String sql)メソッド〔PctoStatement クラス〕 700
executeBatch()メソッド〔PctoStatement クラス〕 703
executeCancel(TccResponse<O> response)メソッド〔TccService<I, O extends java.io.Serializable>インタフェース〕 826
executeConfirm(TccResponse<O> response)メソッド〔TccService<I, O extends java.io.Serializable>インタフェース〕 825
executeQuery()メソッド〔PctoPreparedStatement クラス〕 717
executeQuery(java.lang.String sql)メソッド〔PctoStatement クラス〕 704
executeRecover()メソッド〔TccService<I, O extends java.io.Serializable>インタフェース〕 826
executeTry(I input)メソッド〔TccService<I, O extends java.io.Serializable>インタフェース〕 824
executeUpdate()メソッド〔PctoPreparedStatement クラス〕 717
executeUpdate(java.lang.String sql)メソッド〔PctoStatement クラス〕 705
exporters.jaeger_storage_exporter.queue.num_consumers パラメタ〔Jaeger〕 904
exporters.jaeger_storage_exporter.queue.queue_size パラメタ〔Jaeger〕 904
externalConsensus.eads.cache パラメタ〔TP1-Bridge〕 885
externalConsensus.intervalInMillis パラメタ〔TP1-Bridge〕 885
external-consensus.eads.cache パラメタ〔SQL-Participant〕 848
external-consensus.eads.cache パラメタ〔TCC-Participant〕 859
external-consensus.eads.properties パラメタ〔SQL-Participant〕 847
external-consensus.eads.properties パラメタ〔TCC-Participant〕 859
external-consensus.interval-in-millis パラメタ〔SQL-Participant〕 847
external-consensus.interval-in-millis パラメタ〔TCC-Participant〕 859
external-consensus.type パラメタ〔SQL-Participant〕 855
external-consensus.type パラメタ〔TCC-Participant〕 860

F

filebeat.enabled パラメタ〔TP1-Bridge〕 880
filebeat.imagePullPolicy パラメタ〔TP1-Bridge〕 880

filebeat.image パラメタ〔TP1-Bridge〕 880
filebeat.livenessProbe.failureThreshold パラメタ〔TP1-Bridge〕 880
filebeat.livenessProbe.initialDelaySeconds パラメタ〔TP1-Bridge〕 880
filebeat.livenessProbe.periodSeconds パラメタ〔TP1-Bridge〕 880
filebeat.livenessProbe.timeoutSeconds パラメタ〔TP1-Bridge〕 880
filebeat.output.logstash.namespace パラメタ〔TP1-Bridge〕 880
filebeat.readinessProbe.failureThreshold パラメタ〔TP1-Bridge〕 880
filebeat.readinessProbe.initialDelaySeconds パラメタ〔TP1-Bridge〕 880
filebeat.readinessProbe.periodSeconds パラメタ〔TP1-Bridge〕 880
filebeat.readinessProbe.timeoutSeconds パラメタ〔TP1-Bridge〕 881
free()メソッド〔PctoArray クラス〕 814
free()メソッド〔PctoSQLXML クラス〕 806

G

getArray()メソッド〔PctoArray クラス〕 815
getArray(int columnIndex)メソッド〔PctoResultSet クラス〕 764
getArray(int parameterIndex)メソッド〔PctoCallableStatement クラス〕 741
getArray(java.lang.String columnLabel)メソッド〔PctoResultSet クラス〕 765
getBigDecimal(int columnIndex)メソッド〔PctoResultSet クラス〕 766
getBigDecimal(int parameterIndex, int scale)メソッド〔PctoCallableStatement クラス〕 741
getBigDecimal(java.lang.String columnLabel)メソッド〔PctoResultSet クラス〕 767
getBinaryStream()メソッド〔PctoBlob クラス〕 809
getBlob(int columnIndex)メソッド〔PctoResultSet クラス〕 768
getBlob(int parameterIndex)メソッド〔PctoCallableStatement クラス〕 742
getBlob(java.lang.String columnLabel)メソッド〔PctoResultSet クラス〕 768
getBoolean(int columnIndex)メソッド〔PctoResultSet クラス〕 769
getBoolean(java.lang.String columnLabel)メソッド〔PctoResultSet クラス〕 770
getByte(int columnIndex)メソッド〔PctoResultSet クラス〕 771
getByte(java.lang.String columnLabel)メソッド〔PctoResultSet クラス〕 772
getBytes(int columnIndex)メソッド〔PctoResultSet クラス〕 773
getBytes(int parameterIndex)メソッド〔PctoCallableStatement クラス〕 743
getBytes(java.lang.String columnLabel)メソッド〔PctoResultSet クラス〕 773
getBytes(long pos, int length)メソッド〔PctoBlob クラス〕 810
getCharacterStream()メソッド〔PctoClob クラス〕 812
getClientInterceptor()メソッド〔GrpcTraceInterceptorFactory クラス〕 669
getClob(int columnIndex)メソッド〔PctoResultSet クラス〕 774
getClob(int parameterIndex)メソッド〔PctoCallableStatement クラス〕 744
getClob(java.lang.String columnLabel)メソッド〔PctoResultSet クラス〕 775
getColumnClassName(int column)メソッド〔PctoResultSetMetaData クラス〕 803
getColumnCount()メソッド〔PctoResultSetMetaData クラス〕 803
getColumnLabel(int column)メソッド〔PctoResultSetMetaData クラス〕 804
getColumnName(int column)メソッド〔PctoResultSetMetaData クラス〕 804
getColumnType(int column)メソッド〔PctoResultSetMetaData クラス〕 805
getConnection()メソッド〔PctoDataSource クラス〕 681
getConnection()メソッド〔PctoStatement クラス〕 706

getData()メソッド〔TccResponse<T extends java.io.Serializable>クラス〕 823
getDatabaseProductName()メソッド〔PctoDatabaseMetaData クラス〕 801
getDate(int columnIndex)メソッド〔PctoResultSet クラス〕 776
getDate(int parameterIndex)メソッド〔PctoCallableStatement クラス〕 745
getDate(java.lang.String columnLabel)メソッド〔PctoResultSet クラス〕 777
getDouble(int columnIndex)メソッド〔PctoResultSet クラス〕 777
getDouble(int parameterIndex)メソッド〔PctoCallableStatement クラス〕 746
getDouble(java.lang.String columnLabel)メソッド〔PctoResultSet クラス〕 778
getFloat(int columnIndex)メソッド〔PctoResultSet クラス〕 779
getFloat(int parameterIndex)メソッド〔PctoCallableStatement クラス〕 746
getFloat(java.lang.String columnLabel)メソッド〔PctoResultSet クラス〕 780
getGeneratedKeys()メソッド〔PctoStatement クラス〕 706
getInt(int columnIndex)メソッド〔PctoResultSet クラス〕 780
getInt(int parameterIndex)メソッド〔PctoCallableStatement クラス〕 747
getInt(java.lang.String columnLabel)メソッド〔PctoResultSet クラス〕 781
getLong(int columnIndex)メソッド〔PctoResultSet クラス〕 782
getLong(int parameterIndex)メソッド〔PctoCallableStatement クラス〕 748
getLong(java.lang.String columnLabel)メソッド〔PctoResultSet クラス〕 783
getMetaData()メソッド〔PctoConnection クラス〕 688
getMetaData()メソッド〔PctoResultSet クラス〕 784
getMoreResults()メソッド〔PctoStatement クラス〕 707
getNString(int columnIndex)メソッド〔PctoResultSet クラス〕 784
getNString(int parameterIndex)メソッド〔PctoCallableStatement クラス〕 749
getNString(java.lang.String columnLabel)メソッド〔PctoResultSet クラス〕 785
getObject(int columnIndex, Class<T> type)メソッド〔PctoResultSet クラス〕 788
getObject(int columnIndex)メソッド〔PctoResultSet クラス〕 786
getObject(int parameterIndex, Class<T> type)メソッド〔PctoCallableStatement クラス〕 750
getObject(int parameterIndex)メソッド〔PctoCallableStatement クラス〕 750
getObject(java.lang.String columnLabel, Class<T> type)メソッド〔PctoResultSet クラス〕 789
getObject(java.lang.String columnLabel)メソッド〔PctoResultSet クラス〕 787
getOperationId()メソッド〔TccService<I, O extends java.io.Serializable>インタフェース〕 827
getQueryTimeout()メソッド〔PctoStatement クラス〕 708
getResultSet()メソッド〔PctoStatement クラス〕 708
getServerInterceptor()メソッド〔GrpcTraceInterceptorFactory クラス〕 669
getShort(int columnIndex)メソッド〔PctoResultSet クラス〕 790
getShort(int parameterIndex)メソッド〔PctoCallableStatement クラス〕 751
getShort(java.lang.String columnLabel)メソッド〔PctoResultSet クラス〕 790
getSQLXML(int columnIndex)メソッド〔PctoResultSet クラス〕 791
getSQLXML(int parameterIndex)メソッド〔PctoCallableStatement クラス〕 752
getSQLXML(java.lang.String columnLabel)メソッド〔PctoResultSet クラス〕 792
getString()メソッド〔PctoSQLXML クラス〕 807
getString(int columnIndex)メソッド〔PctoResultSet クラス〕 793

getString(int parameterIndex)メソッド〔PctoCallableStatement クラス〕 753
getString(java.lang.String columnLabel)メソッド〔PctoResultSet クラス〕 794
getSubString(long pos, int length)メソッド〔PctoClob クラス〕 813
getTime(int columnIndex)メソッド〔PctoResultSet クラス〕 794
getTime(int parameterIndex)メソッド〔PctoCallableStatement クラス〕 754
getTime(java.lang.String columnLabel)メソッド〔PctoResultSet クラス〕 795
getTimestamp(int columnIndex)メソッド〔PctoResultSet クラス〕 796
getTimestamp(int parameterIndex)メソッド〔PctoCallableStatement クラス〕 755
getTimestamp(java.lang.String columnLabel)メソッド〔PctoResultSet クラス〕 797
getTransactionKey()メソッド〔TccResponse<T extends java.io.Serializable>クラス〕 823
getType()メソッド〔PctoResultSet クラス〕 797
getUpdateCount()メソッド〔PctoStatement クラス〕 709
getWarnings()メソッド〔PctoStatement クラス〕 710
global.scrape_configs パラメタ〔Prometheus〕 905
global.scrape_interval パラメタ〔Prometheus〕 905
grpc.keepalive.intervalInSeconds パラメタ〔Mediator〕 875
grpc.keepalive.intervalInSeconds パラメタ〔TP1-Bridge〕 890
grpc.keepalive.interval-in-seconds パラメタ〔SQL-Participant〕 855
grpc.keepalive.interval-in-seconds パラメタ〔TCC-Participant〕 861
grpc.keepalive.timeoutInSeconds パラメタ〔Mediator〕 875
grpc.keepalive.timeoutInSeconds パラメタ〔TP1-Bridge〕 890
grpc.keepalive.timeout-in-seconds パラメタ〔SQL-Participant〕 856
grpc.keepalive.timeout-in-seconds パラメタ〔TCC-Participant〕 861
grpc.port パラメタ〔Mediator〕 874
grpc.server.queueSize パラメタ〔Mediator〕 872
grpc.server.threadPoolSize パラメタ〔Mediator〕 872
grpc.tls.client.client-auth.certificate パラメタ〔SQL-Participant〕 855
grpc.tls.client.client-auth.certificate パラメタ〔TCC-Participant〕 860
grpc.tls.client.client-auth.key パラメタ〔SQL-Participant〕 855
grpc.tls.client.client-auth.key パラメタ〔TCC-Participant〕 860
grpc.tls.client.server-auth.certificate パラメタ〔SQL-Participant〕 855
grpc.tls.client.server-auth.certificate パラメタ〔TCC-Participant〕 860
grpc.tls.clientAuth.enabled パラメタ〔Mediator〕 874
grpc.tls.clientAuth.enabled パラメタ〔TP1-Bridge〕 889
grpc.tls.secretName パラメタ〔Mediator〕 874
grpc.tls.secretName パラメタ〔TP1-Bridge〕 889
grpc.tls.session.cacheSize パラメタ〔TP1-Bridge〕 889
grpc.tls.session.cache-size パラメタ〔SQL-Participant〕 855
grpc.tls.session.cache-size パラメタ〔TCC-Participant〕 860
grpc.tls.session.timeoutInSeconds パラメタ〔TP1-Bridge〕 890
grpc.tls.session.timeout-in-seconds パラメタ〔SQL-Participant〕 855
grpc.tls.session.timeout-in-seconds パラメタ〔TCC-Participant〕 861
grpc.useTls パラメタ〔Mediator〕 874
grpc.useTls パラメタ〔TP1-Bridge〕 889
grpc.use-tls パラメタ〔SQL-Participant〕 855
grpc.use-tls パラメタ〔TCC-Participant〕 860

H

http_config.enable_http2 パラメタ〔Prometheus〕 906
http.port パラメタ〔Ext-Cons〕 902
http.port パラメタ〔Mediator〕 874
http.port パラメタ〔TP1-Bridge〕 889

I

idle_in_transaction_session_timeout パラメタ〔PostgreSQL〕 909
imagePullPolicy パラメタ〔Ext-Cons〕 901
imagePullPolicy パラメタ〔Mediator〕 866
imagePullPolicy パラメタ〔TP1-Bridge〕 878
imagePullSecrets パラメタ〔Ext-Cons〕 901
imagePullSecrets パラメタ〔Mediator〕 866
imagePullSecrets パラメタ〔TP1-Bridge〕 878
image パラメタ〔Ext-Cons〕 901
image パラメタ〔Jaeger〕 903
image パラメタ〔Mediator〕 866
image パラメタ〔Metrics Server〕 908
image パラメタ〔Prometheus〕 905
image パラメタ〔TP1-Bridge〕 878
init.mode パラメタ〔JDBC Proxy Driver〕 840
instance-id パラメタ〔Orchestrator〕 841
intercept(org.springframework.http.HttpRequest request, byte[] body, org.springframework.http.client.ClientHttpRequestExecution execution)メソッド〔TxPropagationRestTemplateInterceptor クラス〕 663
interceptCall(io.grpc.ServerCall<ReqT,RespT> call, io.grpc.Metadata headers, io.grpc.ServerCallHandler<ReqT,RespT> next)メソッド〔TxPropagationGrpcServerInterceptor クラス〕 668
interceptCall io.grpc.ClientCall<ReqT,RespT> interceptCall(io.grpc.MethodDescriptor<ReqT, RespT> method,io.grpc.CallOptions callOptions, io.grpc.Channel next)メソッド〔TxPropagationGrpcClientInterceptor クラス〕 664
isClosed()メソッド〔PctoResultSet クラス〕 798
isPrePrepared()メソッド〔TxPropagationOptionalSettings クラス〕 666
isWrapperFor(Class<?> iface)メソッド〔PctoDataSource クラス〕 682

J

java.options パラメタ〔Ext-Cons〕 902
java.options パラメタ〔Mediator〕 876
java.options パラメタ〔TP1-Bridge〕 891

L

labels.appKubernetesIoName パラメタ〔TP1-Bridge〕 878
labels.hmppctoAppGroup パラメタ〔Mediator〕 877
labels.hmppctoAppGroup パラメタ〔TP1-Bridge〕 879
lb.list-timeout-in-seconds パラメタ〔Entity-Module〕 846
lb.list-timeout-in-seconds パラメタ〔Orchestrator〕 845
learner.consensus-type パラメタ〔Orchestrator〕 841
length()メソッド〔PctoBlob クラス〕 811
lifecycle.timeout-per-shutdown-phase パラメタ〔共通〕 833
livenessProbe.failureThreshold パラメタ〔Mediator〕 868
livenessProbe.failureThreshold パラメタ〔TP1-Bridge〕 881
livenessProbe.healthCheck.connectTimeoutSeconds パラメタ〔Mediator〕 869
livenessProbe.healthCheck.connectTimeoutSeconds パラメタ〔TP1-Bridge〕 881
livenessProbe.healthCheck.readTimeoutSeconds パラメタ〔Mediator〕 869
livenessProbe.healthCheck.readTimeoutSeconds パラメタ〔TP1-Bridge〕 881
livenessProbe.healthCheck.retryCount パラメタ〔Mediator〕 868

livenessProbe.healthCheck.retryIntervalSeconds パラメタ〔Mediator〕 869
livenessProbe.initialDelaySeconds パラメタ〔Mediator〕 868
livenessProbe.initialDelaySeconds パラメタ〔TP1-Bridge〕 881
livenessProbe.periodSeconds パラメタ〔Mediator〕 868
livenessProbe.periodSeconds パラメタ〔TP1-Bridge〕 881
livenessProbe.timeoutSeconds パラメタ〔Mediator〕 868
livenessProbe.timeoutSeconds パラメタ〔TP1-Bridge〕 881
logs.accessModes パラメタ〔EADS〕 896
logs.dirType パラメタ〔EADS〕 896
logs.storageClassName パラメタ〔EADS〕 896

M

max_connections パラメタ〔PostgreSQL〕 909
max_prepared_transactions パラメタ〔PostgreSQL〕 909
mediator.discovery-interval-in-millis パラメタ〔Orchestrator〕 843
mediator.discovery-retry-count パラメタ〔Orchestrator〕 843
mediator.judge.proposalLogLimitNum パラメタ〔Mediator〕 875
mediator.judge.queueSize パラメタ〔Mediator〕 871
mediator.judge.threadPoolSize パラメタ〔Mediator〕 871
mediator.minimum-start-size パラメタ〔Orchestrator〕 842
mediator.nameSuffix パラメタ〔TP1-Bridge〕 890
mediator.name パラメタ〔Orchestrator〕 841
mediator.name パラメタ〔SQL-Participant〕 856
mediator.name パラメタ〔TCC-Participant〕 860
mediator.name パラメタ〔TP1-Bridge〕 890
mediator.name-suffix パラメタ〔Orchestrator〕 841
mediator.name-suffix パラメタ〔SQL-Participant〕 856
mediator.name-suffix パラメタ〔TCC-Participant〕 860
mediator.port パラメタ〔Orchestrator〕 842
mediator.port パラメタ〔SQL-Participant〕 856
mediator.port パラメタ〔TCC-Participant〕 860
mediator.port パラメタ〔TP1-Bridge〕 890
mediator.size パラメタ〔Orchestrator〕 842
mediator.termination-thread-pool-size パラメタ〔Orchestrator〕 843
mediatorCommunication.queueSize パラメタ〔TP1-Bridge〕 889
mediatorCommunication.threadPoolSize パラメタ〔TP1-Bridge〕 889
mediatorProcessManagement.threadPoolSize パラメタ〔TP1-Bridge〕 888
mediator-communication.queue-size パラメタ〔SQL-Participant〕 854
mediator-communication.queue-size パラメタ〔TCC-Participant〕 859
mediator-communication.thread-pool-size パラメタ〔SQL-Participant〕 854
mediator-communication.thread-pool-size パラメタ〔TCC-Participant〕 859
mediator-process-management.thread-pool-size パラメタ〔SQL-Participant〕 854
mediator-process-management.thread-pool-size パラメタ〔TCC-Participant〕 859
metrics.collection.tomcat パラメタ〔共通〕 838

N

namespace パラメタ〔EADS〕 895
namespace パラメタ〔Ext-Cons〕 901
namespace パラメタ〔Mediator〕 866
namespace パラメタ〔TP1-Bridge〕 878
next()メソッド〔PctoResultSet クラス〕 799

O

options.server.record-request-start-time パラメタ〔server.undertow〕 864

orchestrator.grpc.keepalive.interval-in-seconds パラメタ〔Orchestrator〕 845
orchestrator.grpc.keepalive.timeout-in-seconds パラメタ〔Orchestrator〕 845
orchestrator.grpc.tls.client.client-auth.certificate パラメタ〔Orchestrator〕 844
orchestrator.grpc.tls.client.client-auth.key パラメタ〔Orchestrator〕 844
orchestrator.grpc.tls.client.server-auth.certificate パラメタ〔Orchestrator〕 844
orchestrator.grpc.tls.session.cache-size パラメタ〔Orchestrator〕 844
orchestrator.grpc.tls.session.timeout-in-seconds パラメタ〔Orchestrator〕 844
orchestrator.grpc.use-tls パラメタ〔Orchestrator〕 844
orchestrator.grpc.watchtime-in-seconds パラメタ〔Orchestrator〕 845

P

participant.sender.queueSize パラメタ〔Mediator〕 871
participant.sender.threadPoolSize パラメタ〔Mediator〕 871
password パラメタ〔JDBC Proxy Driver〕 840
pd_max_users オペランド〔HiRDB〕 911
pipeline.batch.size パラメタ〔Logstash〕 907
port パラメタ〔共通〕 832
postStart.healthCheck.connectTimeoutSeconds パラメタ〔Mediator〕 869
postStart.healthCheck.connectTimeoutSeconds パラメタ〔TP1-Bridge〕 881
postStart.healthCheck.readTimeoutSeconds パラメタ〔Mediator〕 869
postStart.healthCheck.readTimeoutSeconds パラメタ〔TP1-Bridge〕 881
postStart.healthCheck.retryCount パラメタ〔Mediator〕 869
postStart.healthCheck.retryCount パラメタ〔TP1-Bridge〕 881
postStart.healthCheck.retryIntervalSeconds パラメタ〔Mediator〕 869
postStart.healthCheck.retryIntervalSeconds パラメタ〔TP1-Bridge〕 881
prepareCall(java.lang.String sql, int resultSetType, int resultSetConcurrency)メソッド〔PctoConnection クラス〕 690
prepareCall(java.lang.String sql)メソッド〔PctoConnection クラス〕 689
prepareStatement(java.lang.String sql, int autoGeneratedKeys)メソッド〔PctoConnection クラス〕 692
prepareStatement(java.lang.String sql, int resultSetType, int resultSetConcurrency)メソッド〔PctoConnection クラス〕 693
prepareStatement(java.lang.String sql, java.lang.String[] columnNames)メソッド〔PctoConnection クラス〕 694
prepareStatement(java.lang.String sql)メソッド〔PctoConnection クラス〕 691
preStop.waitTransactions.connectTimeoutSeconds パラメタ〔Mediator〕 870
preStop.waitTransactions.connectTimeoutSeconds パラメタ〔TP1-Bridge〕 882
preStop.waitTransactions.readTimeoutSeconds パラメタ〔Mediator〕 870
preStop.waitTransactions.readTimeoutSeconds パラメタ〔TP1-Bridge〕 882
preStop.waitTransactions.retryCount パラメタ〔Mediator〕 869
preStop.waitTransactions.retryCount パラメタ〔TP1-Bridge〕 881
preStop.waitTransactions.retryIntervalSeconds パラメタ〔Mediator〕 869
preStop.waitTransactions.retryIntervalSeconds パラメタ〔TP1-Bridge〕 881
processors.batch. send_batch_max_size パラメタ〔Jaeger〕 904
processors.batch.send_batch_size パラメタ〔Jaeger〕 904
processors.batch.timeout パラメタ〔Jaeger〕 904

Q

queue.disk.max_size パラメタ〔Filebeat〕 903

queue.disk.max_size パラメタ〔Metricbeat〕 906
queue.disk.read_ahead パラメタ〔Filebeat〕 903
queue.disk.read_ahead パラメタ〔Metricbeat〕 907
queue.max_bytes パラメタ〔Logstash〕 907
queue.max_events パラメタ〔Logstash〕 907

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〕 913
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〕 913
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〕 913
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〕 913
recover.interval-in-millis パラメタ〔SQL-Participant〕 854
registerOutParameter(int parameterIndex, int sqlType, int scale)メソッド〔PctoCallableStatement クラス〕 756
registerOutParameter(int parameterIndex, int sqlType, java.lang.String typeName)メソッド〔PctoCallableStatement クラス〕 758
registerOutParameter(int parameterIndex, int sqlType)メソッド〔PctoCallableStatement クラス〕 755
release.server パラメタ〔EADS〕 895
release.service パラメタ〔EADS〕 895
remote_write.queue_config.capacity パラメタ〔Prometheus〕 905
remote_write.queue_config.max_samples_per_send パラメタ〔Prometheus〕 906
remote_write.queue_config.max_shards パラメタ〔Prometheus〕 906
replicas パラメタ〔Mediator〕 866
resources パラメタ〔Mediator〕 867
resources パラメタ〔TP1-Bridge〕 879
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〕 914

S

securityContext パラメタ〔Mediator〕 870
securityContext パラメタ〔TP1-Bridge〕 882
server.tomcat.maxSwallowSize パラメタ〔TP1-Bridge〕 882
server.tomcat.threads.max パラメタ〔TP1-Bridge〕 882
service.port パラメタ〔Metrics Server〕 908
service.port パラメタ〔Prometheus〕 905
SESSIONS パラメタ〔Oracle Database〕 912
setArray(int parameterIndex,java.sql.Array x)メソッド〔PctoPreparedStatement クラス〕 718
setBigDecimal(int parameterIndex, java.math.BigDecimal x)メソッド〔PctoPreparedStatement クラス〕 719
setBinaryStream(int parameterIndex, java.io.InputStream x, int length)メソッド〔PctoPreparedStatement クラス〕 720
setBlob(int parameterIndex, java.io.InputStream inputStream)メソッド〔PctoPreparedStatement クラス〕 721
setBoolean(int parameterIndex, boolean x)メソッド〔PctoPreparedStatement クラス〕 722
setByte(int parameterIndex, byte x)メソッド〔PctoPreparedStatement クラス〕 723
setBytes(int parameterIndex, byte[] x)メソッド〔PctoPreparedStatement クラス〕 724
setCharacterStream(int parameterIndex, java.io.Reader reader, int length)メソッド〔PctoPreparedStatement クラス〕 724
setClob(int parameterIndex, java.io.Reader reader)メソッド〔PctoPreparedStatement クラス〕 725
setDate(int parameterIndex, java.sql.Date x)メソッド〔PctoPreparedStatement クラス〕 727
setDouble(int parameterIndex, double x)メソッド〔PctoPreparedStatement クラス〕 727
setEscapeProcessing(boolean enable)メソッド〔PctoStatement クラス〕 711
setFetchSize(int rows)メソッド〔PctoStatement クラス〕 712

setFloat(int parameterIndex, float x)メソッド〔PctoPreparedStatement クラス〕 728
setInt(int parameterIndex, int x)メソッド〔PctoPreparedStatement クラス〕 729
setLong(int parameterIndex, long x)メソッド〔PctoPreparedStatement クラス〕 730
setNString(int parameterIndex, java.lang.String value)メソッド〔PctoPreparedStatement クラス〕 731
setNull(int parameterIndex, int sqlType)メソッド〔PctoPreparedStatement クラス〕 732
setObject(int parameterIndex, java.lang.Object x, int targetSqlType)メソッド〔PctoPreparedStatement クラス〕 734
setObject(int parameterIndex, java.lang.Object x)メソッド〔PctoPreparedStatement クラス〕 733
setPrePrepared(boolean prePreparedState)メソッド〔TxPropagationOptionalSettings クラス〕 667
setQueryTimeout(int seconds)メソッド〔PctoStatement クラス〕 712
setShort(int parameterIndex, short x)メソッド〔PctoPreparedStatement クラス〕 735
setString(int parameterIndex, java.lang.String x)メソッド〔PctoPreparedStatement クラス〕 736
setString(java.lang.String value)メソッド〔PctoSQLXML クラス〕 808
setTime(int parameterIndex, java.sql.Time x)メソッド〔PctoPreparedStatement クラス〕 736
setTimestamp(int parameterIndex, java.sql.Timestamp x)メソッド〔PctoPreparedStatement クラス〕 737
setTransactionIsolation(int level)メソッド〔PctoConnection クラス〕 696
set-routing パラメタ〔Entity-Module〕 847
set-routing パラメタ〔Orchestrator〕 845
set-routing パラメタ〔SQL-Participant〕 856
shutdown パラメタ〔共通〕 832
sql.datasource.class-name パラメタ〔SQL-Participant〕 849
sql.datasource.property.delimiter パラメタ〔SQL-Participant〕 851
sql.datasource.property.values パラメタ〔SQL-Participant〕 849
sql.db-type パラメタ〔SQL-Participant〕 855
sql.jdbc.driver.additional-library-paths パラメタ〔SQL-Participant〕 848
sql.jdbc.driver.path パラメタ〔SQL-Participant〕 848
sql.participant-processing.queue-size パラメタ〔SQL-Participant〕 854
sql.participant-processing.thread-pool-size パラメタ〔SQL-Participant〕 853
sql.pod.service-name パラメタ〔SQL-Participant〕 848
sql.pool.allocation-wait-time-in-millis パラメタ〔SQL-Participant〕 848
sql.pool.connection-retry-interval-in-seconds パラメタ〔SQL-Participant〕 852
sql.pool.connection-retry-limit パラメタ〔SQL-Participant〕 852
sql.pool.max-pool-size パラメタ〔SQL-Participant〕 851
sql.pool.transaction-timeout-in-seconds パラメタ〔SQL-Participant〕 852
sql.transaction-termination.consensus.replay-interval-in-millis パラメタ〔SQL-Participant〕 853
sql.transaction-termination.timeout-in-millis パラメタ〔SQL-Participant〕 852
store.storageClassName パラメタ〔EADS〕 896
supportsMultipleResultSets()メソッド〔PctoDatabaseMetaData クラス〕 801

T

tcc.consensus.interval-in-millis パラメタ〔TCC-Participant〕 858
tcc.participant-processing.queue-size パラメタ〔TCC-Participant〕 858
tcc.participant-processing.thread-pool-size パラメタ〔TCC-Participant〕 858
tcc.pod.service-name パラメタ〔TCC-Participant〕 856
tcc.recover.enabled パラメタ〔TCC-Participant〕 857

tcc.recover.order パラメタ〔TCC-Participant〕 857
tcc.transaction-termination.timeout-in-millis パラメタ〔TCC-Participant〕 857
TccResponse(final T data)メソッド〔TccResponse<T extends java.io.Serializable>クラス〕 822
terminationGracePeriodSeconds パラメタ〔Mediator〕 870
terminationGracePeriodSeconds パラメタ〔TP1-Bridge〕 882
timezone パラメタ〔Ext-Cons〕 902
timezone パラメタ〔Mediator〕 876
timezone パラメタ〔TP1-Bridge〕 891
tp1.cltj.dccltdelay パラメタ〔TP1-Bridge〕 883
tp1.cltj.dccltrecvbuffsize パラメタ〔TP1-Bridge〕 885
tp1.cltj.dccltrpcmaxmsgsize パラメタ〔TP1-Bridge〕 884
tp1.cltj.dccltsendbuffsize パラメタ〔TP1-Bridge〕 884
tp1.cltj.dcdatatracemaxsize パラメタ〔TP1-Bridge〕 884
tp1.cltj.dcdatatracesize パラメタ〔TP1-Bridge〕 884
tp1.cltj.dcdatatrace パラメタ〔TP1-Bridge〕 884
tp1.cltj.dcerrtracesize パラメタ〔TP1-Bridge〕 884
tp1.cltj.dcerrtrace パラメタ〔TP1-Bridge〕 884
tp1.cltj.dchost パラメタ〔TP1-Bridge〕 883
tp1.cltj.dcmethodtracesize パラメタ〔TP1-Bridge〕 884
tp1.cltj.dcmethodtrace パラメタ〔TP1-Bridge〕 884
tp1.cltj.dcrapport パラメタ〔TP1-Bridge〕 883
tp1.cltj.dcuaptracesize パラメタ〔TP1-Bridge〕 884
tp1.cltj.dcuaptrace パラメタ〔TP1-Bridge〕 884
tp1.cltj.dcwatchtiminherit パラメタ〔TP1-Bridge〕 883
tp1.cltj.dcwatchtimrpcinherit パラメタ〔TP1-Bridge〕 883
tp1.cltj.dcwatchtim パラメタ〔TP1-Bridge〕 883
tp1.data-conversion.definition-file.directory パラメタ〔uCosminexus Application Server の usrconf.properties〕 915
tp1.participantProcessing.queueSize パラメタ〔TP1-Bridge〕 888
tp1.participantProcessing.threadPoolSize パラメタ〔TP1-Bridge〕 888
tp1.pod.serviceName パラメタ〔TP1-Bridge〕 886
tp1.pool.allocationWaitTimeInMillis パラメタ〔TP1-Bridge〕 885
tp1.pool.maxPoolSize パラメタ〔TP1-Bridge〕 886
tp1.transactionTermination.consensus.replayIntervalInMillis パラメタ〔TP1-Bridge〕 887
tp1.transactionTermination.timeoutInMillis パラメタ〔TP1-Bridge〕 886
tracing.batchSize パラメタ〔Mediator〕 876
tracing.batchSize パラメタ〔TP1-Bridge〕 890
tracing.batch-size パラメタ〔共通〕 837
tracing.debug パラメタ〔Mediator〕 876
tracing.debug パラメタ〔TP1-Bridge〕 890
tracing.debug パラメタ〔共通〕 836
tracing.delayTimeInMillis パラメタ〔Mediator〕 876
tracing.delayTimeInMillis パラメタ〔TP1-Bridge〕 890
tracing.delay-time-in-millis パラメタ〔共通〕 836
tracing.grpc.debug パラメタ〔Mediator〕 876
tracing.grpc.debug パラメタ〔TP1-Bridge〕 891
tracing.grpc.debug パラメタ〔共通〕 837
tracing.instanceId パラメタ〔Mediator〕 876
tracing.instanceId パラメタ〔TP1-Bridge〕 890
tracing.instance-id パラメタ〔共通〕 833
tracing.jdbc.debug パラメタ〔共通〕 838
tracing.queueSize パラメタ〔Mediator〕 876
tracing.queueSize パラメタ〔TP1-Bridge〕 890
tracing.queue-size パラメタ〔共通〕 836
tracing.url パラメタ〔Mediator〕 876
tracing.url パラメタ〔TP1-Bridge〕 890

tracing.url パラメタ〔共通〕 834
tracing.useTrace パラメタ〔Mediator〕 876
tracing.useTrace パラメタ〔TP1-Bridge〕 890
tracing.use-trace パラメタ〔共通〕 835
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〕 914
type パラメタ〔JDBC Proxy Driver〕 840

U

ucars.enabled パラメタ〔Mediator〕 877
ucars.snapshots.claimName パラメタ〔Mediator〕 877
ucars.snapshots.claimName パラメタ〔TP1-Bridge〕 891
url パラメタ〔JDBC Proxy Driver〕 840
user connections オプション〔SQL Server〕 910
username パラメタ〔JDBC Proxy Driver〕 840
use-scaling パラメタ〔Entity-Module〕 846
use-scaling パラメタ〔Mediator〕 877
use-scaling パラメタ〔Orchestrator〕 845
use-scaling パラメタ〔SQL-Participant〕 856
use-scaling パラメタ〔TCC-Participant〕 861
use-scaling パラメタ〔TP1-Bridge〕 892

W

wasNull()メソッド〔PctoCallableStatement クラス〕 759
wasNull()メソッド〔PctoResultSet クラス〕 800
webClientFilter()メソッド〔TxPropagationOptionalSettings クラス〕 665
webserver.connector.nio_http.limit.max_request_body パラメタ〔uCosminexus Application Server の usrconf.properties〕 916
webserver.connector.nio_http.max_servlet_execute_threads パラメタ〔ucosminexusapplicationserver の usrconf.properties〕 916
webserver.connector.nio_http.max_threads パラメタ〔uCosminexus Application Server の usrconf.properties〕 915
webserver.connector.nio_http.port パラメタ〔uCosminexus Application Server の usrconf.properties〕 916

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