HITACHI
Inspire the Next

Cosminexus V11 BPM/ESB 基盤 サービスプラットフォーム 解説

解説書

3021-3-J42-80

前書き

■ 対象製品

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

■ 輸出時の注意

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

なお、不明な場合は、弊社担当営業にお問い合わせください。

■ 商標類

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

Eclipse は，Eclipse Foundation, Inc.の商標です。

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

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

Microsoft，Excel，Windows，Windows Server は，マイクロソフト 企業グループの商標です。

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

Salesforce は，salesforce.com,Inc.の商標です。

本書に記載される SAP 及びその他の SAP の製品やサービス，並びにそれらの個々のロゴは，ドイツ及びその他の国における SAP SE（又は SAP の関連会社）の商標若しくは登録商標です。

UNIX は，The Open Group の登録商標です。

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

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

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

本ソフトウェアに含まれる第三者ソフトウェアに関する情報を次に示します。

第三者ソフトウェア：Trang

著作権者：Thai Open Source Software Center Ltd

当該ソフトウェアのライセンス条文：以下の通り。

Copyright (c) 2002, 2003, 2008 Thai Open Source Software Center Ltd

All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

Neither the name of the Thai Open Source Software Center Ltd nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE REGENTS OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

■ ランゲージパックの使用について

このマニュアルでは，Eclipse Babel Project が提供する BABEL 日本語ランゲージパックを適用した Eclipse メニュー表記を使用しています。使用するランゲージパックのバージョンによっては，マニュアル内の表記と異なる場合があります。

■ 発行

2026 年 1 月 3021-3-J42-80

■ 著作権

All Rights Reserved. Copyright (C) 2020, 2026, Hitachi, Ltd.

変更内容

変更内容(3021-3-J42-80) uCosminexus Service Architect 11-70，uCosminexus Service Platform 11-70

追加・変更内容	変更個所
Service Platform および Service Architect から次の適用 OS を削除した。 • Windows 10 x64 • Red Hat Enterprise Linux 8（AMD/Intel 64）	−
XML スキーマを利用した JSON-XML 変換（拡張モード）ができるようにした。 これに伴い，次の点を修正した。 • 拡張モードに対し，既存の JSON-XML 変換を標準モードとして区別した。 • httprecp.json-xml-convert.mode プロパティと httprecp.json-transfer.allow-partial-conversion プロパティの説明を追加した。 • cscjson2xsdadv コマンドと cscjsonxmlconvadv コマンドの説明を追加した。	2.15.10，2.15.10(2)，2.15.10(3)，2.15.10(4)，2.15.12(2)，4.11.2，4.12，付録 B.3，付録 B.3(4)
繰り返しアクティビティのリスト指定方式で，永続化するビジネスプロセスを指定できるようにした。 これに伴い，永続化するビジネスプロセスについての注意事項を削除した。	3.3.1
スケジュール駆動受付で，イベント開始時刻を指定できるようにした。 これに伴い，開始時刻を指定してスケジュール処理を開始する設定についての説明を追加した。	3.6.1
FTP アダプタに RENAME オペレーションを追加し，ファイルのリネームができるようにした。	8.6，8.6.1，8.6.2，8.6.3(1)，8.6.5
FTP アダプタのファイル転送の性能向上のため，ログインセッション維持機能を追加した。 これに伴い，次の点を修正した。 • MGET オペレーション，MPUT オペレーション，LOGIN オペレーション，および LOGOUT オペレーションの説明を追加した。 • LIST コマンドおよび NLST コマンドを実行する場合の注意事項を追加した。	8.6，8.6.2，8.6.6，8.6.7，8.6.7(3)，8.6.7(8)，8.9.3(3)，8.12.2

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

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

目次

前書き　2
変更内容　4
はじめに　5

第 1 編　機能解説編

1　サービスプラットフォームの機能概要　18
1.1　開発環境の機能概要　19
1.1.1　電文フォーマットの作成機能　19
1.1.2　ユーザ定義受付の作成機能　20
1.1.3　サービスアダプタの作成機能　22
1.1.4　ビジネスプロセスの作成機能　23
1.1.5　データ変換定義の作成機能　26
1.1.6　配備定義機能　27
1.1.7　パッケージング機能　28
1.1.8　配備から開始または停止から削除の一括実行機能　29
1.1.9　サービスリクエスタの作成機能　30
1.1.10　ビジネスプロセスのデバッグ機能　30
1.2　実行環境の機能概要　33
1.2.1　サービス振り分け実行機能　33
1.2.2　各種サービス呼び出し機能　33
1.2.3　ビジネスプロセス実行機能　34
1.3　運用環境の機能概要　35
1.3.1　実行環境のセットアップ機能　35
1.3.2　HCSC コンポーネントの配備機能　36
1.3.3　実行環境での運用を管理する機能　37
1.3.4　実行履歴の管理機能　39
1.4　HCSC サーバのクラスタ構成　40
1.4.1　ロードバランス機能を利用した HCSC サーバの冗長構成　40
1.4.2　クラスタソフトウェアを利用した HCSC サーバの冗長構成　41
1.4.3　N 対 1 クラスタ構成を利用した HCSC サーバの冗長構成　43
1.5　クラスタ構成における運用環境の構成パターン　44
1.6　分散開発時のリポジトリのインポート　47
1.6.1　コンポーネントインポート機能を利用した分散開発の流れ　47
1.6.2　分散開発時の注意事項　49

2 さまざまなシステムと接続するための機能 50

2.1 サービス部品呼び出しの基本的な流れ 51
2.1.1 基本的な構造 51
2.1.2 サービス部品を呼び出す流れ 54
2.1.3 サービス部品側と異なる電文フォーマットでサービス部品を呼び出す場合の流れ 59
2.1.4 サービス部品呼び出し処理で与えられる識別情報 61
2.2 Web サービス（SOAP 通信）を使用したサービスとの接続 64
2.2.1 Web サービスの通信基盤との関係 64
2.2.2 サービス部品呼び出し時のユーザ電文と WSDL の関係 65
2.2.3 Web サービス（SOAP 通信）の場合のトランザクション 73
2.2.4 Web サービス（SOAP 通信）の場合の Cookie 情報 73
2.2.5 Web サービス（SOAP 通信）の場合の HTTP ヘッダ引き継ぎ 77
2.2.6 Web サービス（SOAP 通信）の場合の SOAP メッセージの構成 81
2.2.7 Web サービス（SOAP 通信）の場合の接続先動的変更 84
2.2.8 Web サービス（SOAP 通信）の場合の添付ファイル 91
2.3 SessionBean を使用したサービスとの接続 102
2.3.1 アプリケーションサーバ（EJB コンテナ）との関係 102
2.3.2 サービス部品呼び出し時のユーザ電文とサービス部品のスタブとの関係 102
2.3.3 SessionBean 使用時のトランザクション 104
2.3.4 SessionBean アダプタ使用時の注意事項 106
2.4 WS-R（WS-Reliability）を使用したサービスとの接続 107
2.4.1 Reliable Messaging との関係 107
2.4.2 サービス部品呼び出し時のメッセージの流れ 107
2.4.3 同期のサービス部品を呼び出す場合の応答電文の扱い 111
2.4.4 MDB（WS-R）使用時のトランザクション 113
2.4.5 デッドメッセージキュー 118
2.4.6 MDB（WS-R）で使用するキューの名称と個数 118
2.5 DB キューを使用したサービスとの接続 119
2.5.1 Reliable Messaging との関係 119
2.5.2 サービス部品呼び出し時のメッセージの流れ 120
2.5.3 同期のサービス部品を呼び出す場合の応答電文の扱い 125
2.5.4 MDB（DB キュー）使用時のトランザクション 127
2.5.5 デッドメッセージキュー 127
2.5.6 MDB（DB キュー）で使用するキューの名称と個数 128
2.6 データベースとの接続 129
2.6.1 DB アダプタを使ったデータベースへのアクセス 129
2.6.2 DB アダプタが使用する DB Connector の設定 132
2.6.3 DB アダプタ使用時のトランザクション 134
2.6.4 DB アダプタの接続先情報の動的変更 135

2.7 OpenTP1 との接続 141
2.7.1 TP1 から非同期の標準受付を使った HCSC サーバのサービス部品の呼び出し 141
2.7.2 TP1 から TP1/RPC 受付を使った HCSC サーバのサービス部品の呼び出し 141
2.7.3 HCSC サーバから TP1 アダプタを使った TP1 サービス部品の呼び出し 147
2.7.4 HCSC サーバからの MDB（DB キュー）サービスアダプタを使った TP1 サービス部品の呼び出し 165
2.7.5 TP1 アダプタのオペレーション動的変更 165
2.7.6 TP1 アダプタの接続先動的変更 165
2.7.7 TP1 アダプタの接続先情報の動的変更 169
2.8 ファイルを扱うシステムとの接続 175
2.8.1 ファイルアダプタを使ったファイルへのアクセス 175
2.8.2 ファイルアダプタを使ったファイルの読み込みと書き込み 176
2.8.3 ファイルアダプタで使用できるファイル形式 178
2.8.4 ファイル操作が行われるときのモード 178
2.8.5 ファイルの読み込みおよび書き込み時の排他制御 179
2.9 Object Wrapper システムとの接続 181
2.9.1 Object Access アダプタを使った Object Wrapper システムへのアクセス 181
2.9.2 接続時の電文の形式変換 184
2.10 メッセージキューとの接続 187
2.10.1 Message Queue 受付を使用したサービス部品の呼び出し 187
2.10.2 Message Queue アダプタを使ったメッセージキューへのアクセス 202
2.11 FTP クライアントおよび FTP サーバとの接続 208
2.11.1 FTP と連携したファイルの転送 208
2.12 Kafka との接続 211
2.12.1 Kafka 受付の機能 211
2.12.2 Kafka 受付の JSON-XML 変換機能 216
2.12.3 Kafka インバウンドアダプタの機能 217
2.12.4 Kafka インバウンドアダプタのタイムアウト監視 232
2.12.5 Kafka アダプタの機能 238
2.12.6 Kafka アダプタのタイムアウト監視 243
2.12.7 Kafka アダプタの JSON-XML 変換機能 246
2.13 gRPC プロトコルを利用した接続 248
2.13.1 gRPC プロトコルを利用したシステム 248
2.13.2 gRPC 電文（Protocol Buffers 形式）と XML 形式の変換 254
2.13.3 gRPC 受付の機能 266
2.13.4 gRPC インバウンドアダプタの機能 270
2.13.5 gRPC アダプタの機能 272
2.14 メールサーバとの接続 281
2.14.1 前提条件 281

2.14.2 メールアダプタを使用したシステム 281
2.14.3 メールアダプタ使用時のトランザクション 287
2.14.4 メールアダプタで設定できるメール情報 287
2.14.5 メールアダプタのタイマ管理 289
2.14.6 メールアダプタからメールを送信するときの認証 290
2.15 HTTP 通信を使用したクライアントとの接続 292
2.15.1 HTTP 受付を使用したサービス部品の呼び出し 292
2.15.2 HTTP 受付を使用したオンラインシステムの例 293
2.15.3 HTTP 受付の構成例 299
2.15.4 HTTP 受付のリクエスト処理 304
2.15.5 HTTP 受付のレスポンス処理 312
2.15.6 HTTP 受付のファイル管理 322
2.15.7 HTTP リクエストと HTTP レスポンスのエンコーディング/デコーディング 325
2.15.8 HTTP 受付使用時のトランザクション 326
2.15.9 HTTP 受付のタイマ管理 327
2.15.10 HTTP 受付の JSON-XML 変換機能 327
2.15.11 HTTP 受付の呼び出しサービス選択機能 335
2.15.12 HTTP 受付のルーティング機能 336
2.15.13 HTTP 受付の電文自動変換機能 339
2.15.14 OpenAPI 仕様書から HCSC コンポーネントを作成する OpenAPI 仕様書インポート機能 340
2.15.15 HTTP 受付を使用するときの注意 351
2.16 HTTP 通信を使用したサービスとの接続 354
2.16.1 前提条件 354
2.16.2 HTTP アダプタを使用したサービス部品の呼び出し 354
2.16.3 HTTP リクエストと要求電文の関係 355
2.16.4 HTTP レスポンスと応答電文の関係 379
2.16.5 HTTP アダプタを使用した Cookie 情報の引き継ぎ 388
2.16.6 プロキシサーバを経由した通信 392
2.16.7 コネクション継続（Keep-Alive）を使用した通信 393
2.16.8 チャンク送信を使用した通信 394
2.16.9 HTTP ボディの圧縮／解凍 395
2.16.10 HTTP アダプタのタイマ管理 396
2.16.11 HTTP アダプタのリダイレクト 398
2.16.12 ベーシック認証機能を使用した接続 399
2.16.13 HTTP アダプタを利用したシステムの例 404
2.16.14 要求電文（ヘッダ）と HTTP アダプタ実行環境（共通）プロパティファイルのパラメタの対応関係 406
2.16.15 HTTP アダプタの JSON-XML 変換機能 413
2.16.16 WWW 認証を使用したセキュアな接続 418

2.16.17 OAuth 2.0 に対応したサービスへのアクセス 419
2.16.18 署名付き AWS API リクエストの署名情報付与 435
2.17 外部コマンドの実行によるシステムとの接続 444
2.17.1 前提条件 444
2.17.2 コマンドアダプタを使用した外部コマンドの実行 444
2.17.3 要求電文を利用した外部コマンド情報の指定 445
2.17.4 応答電文で取得できる外部コマンドプロセスの終了情報 447
2.17.5 外部コマンドプロセスに関する動作オプションの設定 447
2.17.6 実行許可コマンド定義ファイルによる実行制限 451
2.18 SFTP サーバとの接続 453
2.18.1 前提条件 453
2.18.2 SFTP アダプタを使用したシステム 453
2.18.3 SFTP アダプタで設定できるファイル転送の機能 456
2.18.4 SFTP アダプタのファイルアクセス方式 458
2.18.5 SFTP アダプタと SFTP サーバ間の通信 459
2.18.6 SFTP アダプタのタイムアウトを監視するポイント 466
2.18.7 SFTP アダプタの排他管理 468
2.18.8 SFTP アダプタが使用する IP アドレスおよびポート番号の指定 469
2.18.9 SFTP アダプタからファイルを送信するときの認証 470
2.18.10 SFTP アダプタの要求電文に指定できるリモートのファイルパス 472
2.19 ファイルイベント連携を使用したファイルの監視 474
2.19.1 ファイルイベント連携を使用したシステム間連携 474
2.19.2 ファイルイベント連携で監視するフォルダおよびファイル 474
2.20 カスタム受付を使用した場合のシステムとの接続 477
2.20.1 カスタム受付の構成 477
2.20.2 カスタム受付フレームワークの概要 477
2.21 汎用カスタムアダプタを使用した場合のシステムとの接続 480
2.21.1 汎用カスタムアダプタの概要 480
2.21.2 カスタムアダプタ開発フレームワークの概要 481
2.22 TP1 アダプタおよび Message Queue アダプタのオペレーション動的変更機能 484
2.22.1 オペレーション動的変更機能を利用した処理の流れ 484
2.22.2 ヘッダ割当変数の定義 485
2.22.3 オペレーション動的変更機能を使ったサービス呼び出し 486

3 ビジネスプロセスの機能 488
3.1 ビジネスプロセスの概要 489
3.1.1 受付・応答 489
3.1.2 サービス呼び出し 490
3.1.3 Java 呼び出し 491

3.1.4 データ変換 492
3.1.5 代入 495
3.1.6 繰り返し 496
3.1.7 分岐 498
3.1.8 並列処理 499
3.1.9 待機処理 500
3.2 ビジネスプロセスの仕組み 501
3.2.1 ビジネスプロセスの処理の流れ 501
3.2.2 ビジネスプロセスの通信モデル 501
3.2.3 プロセスインスタンス 502
3.2.4 アクティビティ 505
3.2.5 変数の種類と役割 509
3.2.6 相関セットの役割 509
3.3 永続化するビジネスプロセスと永続化しないビジネスプロセス 510
3.3.1 永続化するビジネスプロセス 510
3.3.2 永続化しないビジネスプロセス 510
3.4 ビジネスプロセスのトランザクション 512
3.4.1 ビジネスプロセスの状態の互換性の指定を ON にした場合のトランザクション 513
3.4.2 ビジネスプロセスの状態の互換性の指定を OFF，サービス呼出アクティビティの状態の互換性の指定を ON にした場合のトランザクション 521
3.4.3 ビジネスプロセスの状態の互換性の指定を OFF，サービス呼出アクティビティの状態の互換性の指定を OFF にした場合のトランザクション 529
3.4.4 サービス呼出アクティビティのトランザクション 533
3.4.5 スコープの開始時および終了時にコミットする設定を選択した場合のトランザクション 535
3.4.6 スコープアクティビティのトランザクション 541
3.4.7 応答アクティビティのトランザクション 542
3.4.8 途中終了アクティビティのトランザクション 549
3.5 ビジネスプロセスの再実行 551
3.5.1 運用環境の画面からの再実行 551
3.5.2 コマンドからの再実行 552
3.5.3 サービスリクエスタからの再実行（SOAP 通信） 552
3.5.4 サービスリクエスタからの再実行（SessionBean） 553
3.6 ビジネスプロセスのスケジュール設定 554
3.6.1 スケジュール駆動受付でのスケジュールの設定 554
3.6.2 スケジュール駆動受付でのタスク情報管理機能 558
3.6.3 スケジュール駆動受付でのタスクの呼び出し機能 559
3.6.4 スケジュール駆動受付の作業フォルダ 564
3.6.5 スケジュール駆動受付のタイマ監視 565
3.6.6 スケジュール駆動受付のトランザクション 565
3.7 ビジネスプロセスを使用した待機処理 567

3.7.1 待機アクティビティでの待機時間の指定 568
3.7.2 待機アクティビティを実行するための条件 569
3.7.3 待機アクティビティに関する多重度 569
3.8 応答アクティビティのあとに定義されたアクティビティの非同期実行 570
3.8.1 非同期 EJB を使用して非同期実行する場合の設定項目 570
3.8.2 非同期 EJB の実行で例外が発生した場合の再実行方法 571
3.9 ログ出力アクティビティでのログ出力 573
3.9.1 設定に応じたログファイルの出力内容 573
3.9.2 ログ出力アクティビティのログファイルの取得方法 576
3.9.3 ログ出力アクティビティのログファイルの出力 578

4 システム動作を円滑にするための機能 580
4.1 HCSC サーバに関連するタイマの設定 581
4.1.1 HCSC サーバで設定できるタイマ 581
4.1.2 サービス部品のタイムアウト値の変更 588
4.1.3 タイマ設定時の注意事項 588
4.2 サービス部品の接続先変更 589
4.3 SOAP アダプタを利用したフォルトのキャッチ 590
4.3.1 SOAP アダプタのフォルト判別の概要 590
4.3.2 SOAP Fault の詳細項目によるフォルト判別 592
4.3.3 SOAP Fault の faultcode によるフォルト判別 593
4.3.4 システム例外のフォルト変換 594
4.3.5 SOAP Fault 動作定義ファイルによるフォルトのキャッチ 594
4.4 アプリケーションからの稼働状態の確認 597
4.4.1 稼働状態の確認の概要 597
4.4.2 稼働状態の確認の流れ 597
4.4.3 稼働状態の確認時の注意事項 598
4.5 XML 電文内の属性値に指定した名前空間接頭辞の補完 600
4.6 任意形式（any 形式）の電文フォーマットの定義 601
4.6.1 任意形式（any 形式）を設定した場合の処理の流れ 601
4.6.2 データ変換をしない場合の変数（メッセージ型）の組み合わせ 602
4.7 システム例外をフォルトに変換する汎用フォルト 608
4.7.1 システム例外をフォルトに変換する汎用フォルトの概要 608
4.7.2 汎用フォルトを使用してシステム例外発生個所を特定する場合の流れ 610
4.7.3 汎用フォルト電文とメッセージログのマッピング 611
4.8 例外発生時にサービスアダプタが取得するエラーの選択 612
4.9 UOC を利用したユーザ電文の値の変更 613
4.9.1 コンポーネント共通 UOC の概要 613
4.9.2 コンポーネント共通 UOC の使用例 617

4.10 バイナリデータの変換エラー時のデバッグ情報の出力 621
4.10.1 デバッグ情報出力機能の概要 621
4.10.2 デバッグ情報の出力例 621
4.11 API を使用したデータ変換 625
4.11.1 データ変換 API を使用したデータ変換 625
4.11.2 JSON-XML 変換 API を使用した変換 626
4.12 JSON 変換ツールを使用した変換 629
4.13 サービスのエミュレーション機能の概要 633
4.13.1 サービスのエミュレーション機能の設定 633
4.13.2 サービスのエミュレーション機能を使用する場合の注意 635

5 性能向上のための機能 637
5.1 サービス部品呼び出し処理の多重度 638
5.1.1 HCSC サーバ内の多重度 638
5.1.2 データベースへのアクセスに関する多重度 641
5.1.3 Web サービス（SOAP 通信）に関する多重度 644
5.1.4 XML 解析処理に関する多重度 647
5.2 XML 電文のキャッシュ機能 649
5.2.1 XML 電文のキャッシュ機能の概要 649
5.2.2 XML 電文のキャッシュ機能の設定 649
5.3 ビジネスプロセスの事前キャッシュ機能 650
5.4 データ変換処理の XML パーサプール機能 651
5.5 データ変換のキャッシュ機能 654
5.5.1 キャッシュの登録および削除のタイミング 654
5.5.2 キャッシュ機能を利用したオブジェクトの再利用 654
5.5.3 サービスアダプタでのキャッシュの注意事項 655

6 障害調査のための機能 656
6.1 障害情報の取得・障害回復・障害解析機能 657
6.1.1 障害情報の取得機能 657
6.1.2 障害回復機能 657
6.1.3 障害解析機能 657
6.2 データ検証機能 658
6.2.1 データ検証機能の概要 658
6.2.2 データ検証機能の設定 659
6.2.3 データ検証機能の有効範囲 660
6.3 ユーザ電文トレース機能 661
6.3.1 ユーザ電文トレース機能の概要 661
6.3.2 ユーザ電文トレース機能の設定 661

6.3.3 ユーザ電文トレース機能使用時の注意事項 662
6.4 アクティビティトレース機能 663

7 他製品と連携して利用する機能 664
7.1 JP1 と連携した運用管理 665
7.1.1 JP1/IM との連携による障害検知 665
7.1.2 JP1 によるプロセス監視 666
7.2 接続できるデータベース 670
7.3 HiRDB と連携した実行履歴の管理 671
7.3.1 実行履歴の分割格納 671

第 2 編 設計解説編

8 FTP と連携してファイルを転送する機能（FTP 連携） 674
8.1 FTP 連携の概要 675
8.1.1 FTP 連携の目的 675
8.1.2 FTP 連携の特長 676
8.2 FTP 連携を利用するためのシステムの構成 677
8.3 FTP 連携を利用したファイル転送パターンの例 678
8.3.1 FTP クライアントから FTP サーバへのファイル転送 678
8.3.2 FTP サーバから FTP クライアントへのファイル転送 681
8.3.3 FTP クライアントから別の FTP クライアントへのファイル転送 684
8.3.4 FTP サーバのファイルリストの取得 687
8.4 FTP 受付の機能 690
8.4.1 FTP 受付でサポートするオペレーションと FTP コマンド 690
8.4.2 FTP 受付で使用できる通信モデルおよびビジネスプロセス 692
8.4.3 FTP 受付使用時のトランザクション 692
8.4.4 リクエスト ID の生成 694
8.4.5 作業フォルダと中間ファイルの作成 695
8.4.6 ビジネスプロセスのアクセス制御 696
8.4.7 転送前／転送後またはリスト前／リスト後に実行する FTP コマンドの管理 698
8.4.8 リストコマンドオプションの設定による LIST／NLST コマンドの実行 699
8.4.9 タイムアウトの監視 700
8.4.10 共通フォルダと中間ファイルの作成 700
8.4.11 リストコマンドパスのデータ抽出機能 702
8.5 FTP インバウンドアダプタの機能 705
8.5.1 送受信後のファイルサイズ確認 705
8.5.2 タイムアウトの監視 706
8.6 FTP アダプタの機能 714
8.6.1 FTP アダプタの処理の流れ 716

8.6.2 FTP コマンドの実行 717
8.6.3 処理に使用するフォルダの指定 720
8.6.4 送受信後のファイルサイズ確認 721
8.6.5 ファイルのリネーム 722
8.6.6 ログインセッションの維持 722
8.6.7 タイマ管理 726
8.7 ファイル操作アダプタの機能 735
8.7.1 ファイル操作アダプタでサポートするオペレーション 735
8.7.2 各オペレーションの要求電文および応答電文 739
8.7.3 ファイル変換オペレーション 740
8.7.4 ファイル複製オペレーション 758
8.7.5 ファイル移動オペレーション 759
8.7.6 ファイル・フォルダ削除オペレーション 760
8.7.7 ファイル圧縮オペレーション 762
8.7.8 ファイル伸張オペレーション 764
8.7.9 ファイル一覧取得オペレーション 766
8.7.10 ファイル一覧書き込みオペレーション 767
8.7.11 ファイル読み込みオペレーション 770
8.7.12 ファイル書き込みオペレーション 771
8.7.13 各オペレーションで使用するフォルダおよびファイルの管理 773
8.8 作業フォルダおよび共通フォルダを利用したファイルアクセス 775
8.8.1 作業フォルダを利用したファイルアクセス 775
8.8.2 共通フォルダを利用したファイルアクセス 775
8.9 FTP 連携の適用範囲 776
8.9.1 FTP 連携でサポートする FTP コマンド 776
8.9.2 転送コマンドの指定方法 779
8.9.3 FTP 連携を利用するときの注意 780
8.10 FTP 連携システムの開発の手順 782
8.10.1 FTP 連携システムの開発の流れ 782
8.11 FTP 受付および FTP アダプタの実行環境に依存する設定の定義 785
8.11.1 FTP 受付の実行環境に依存する設定の定義 785
8.11.2 FTP アダプタの実行環境に依存する設定の定義 790
8.12 障害対策のための設計 794
8.12.1 エラー設計の概要 794
8.12.2 タイマ設計 795
8.12.3 トランザクション設計 796
8.12.4 トラブルシュート設計 803

9 ファイルイベント連携を使用したファイルの監視 806
9.1 ファイル連携の機能の概要 807
9.1.1 ファイル監視機能 807
9.1.2 監視対象のファイルの状態管理機能 811
9.1.3 ファイルイベントトレース機能 812
9.1.4 ファイルイベント連携使用時の注意事項 812
9.2 ファイルイベント連携を使用するためのシステムの構成例 814
9.2.1 サービスプラットフォームのサーバマシンのローカルディスクを使用する構成例 814
9.2.2 ファイルサーバの共有ディスクを使用する構成例 815
9.3 ファイルイベント連携で必要な設定 816
9.3.1 ファイルイベント連携の設定の流れ 816
9.3.2 ファイルイベント連携機能で設計が必要な項目 817
9.3.3 ファイルイベント受付が監視するフォルダの設定 818
9.3.4 ファイルイベント連携時に作成するファイル 820
9.3.5 ファイルイベント連携に関する実行環境の設定 821
9.3.6 ファイルイベント連携のエラー設計の概要 823

10 作業フォルダおよび共通フォルダの機能 825
10.1 作業フォルダおよび共通フォルダにアクセスできるコンポーネント 826
10.2 作業フォルダを利用したファイルアクセス 828
10.2.1 作業フォルダルートの定義 828
10.2.2 作業フォルダとリクエスト ID の生成 828
10.2.3 作業フォルダ内のファイルの生成 830
10.2.4 作業フォルダ内のファイルの排他制御 831
10.2.5 作業フォルダおよび作業フォルダに格納された中間ファイルの削除 831
10.3 共通フォルダを利用したファイルアクセス 832
10.3.1 共通フォルダ定義名と対応するパスの定義 832
10.3.2 共通フォルダ内のファイルの生成 832
10.3.3 共通フォルダ内のファイルの排他制御 834
10.3.4 共通フォルダ内のファイルの削除 834

付録 836
付録 A TP1 アダプタと TP1/Client/J の関係 837
付録 A.1 TP1/Client/J の機能 837
付録 A.2 TP1/Client/J の定義 854
付録 A.3 TP1/Client/J の障害対策 886
付録 B データ変換機能の処理概要 903
付録 B.1 データ変換に必要な定義ファイルと作成方法 903
付録 B.2 データ変換処理の概要 905

付録 B.3　データ変換機能のエミュレーション　909
付録 C　IPv6 対応　911
付録 C.1　サポート範囲　911
付録 C.2　制限事項　912
付録 D　用語解説　913

索引　914

1 サービスプラットフォームの機能概要

この章では，サービスプラットフォームで SOA を適用したシステムを実現するための機能の概要について説明します。

1.1　開発環境の機能概要

サービスプラットフォームの開発環境は，実行環境で動かすサービスアダプタやビジネスプロセスなどの定義をするための環境です。開発環境では，主に次の機能を提供しています。

- 電文フォーマットの作成機能
- ユーザ定義受付の作成機能
- サービスアダプタの作成機能
- ビジネスプロセスの作成機能
- データ変換定義の作成機能
- ビジネスプロセスのデバッグ機能

これらの機能は，開発環境の画面を利用して定義します。

また，開発環境の画面を利用して作成した定義の整合性を検証する機能も備えています。HCSC コンポーネントの検証機能は，HCSC コンポーネントの作成中や作成後など，任意の段階で実行できます。

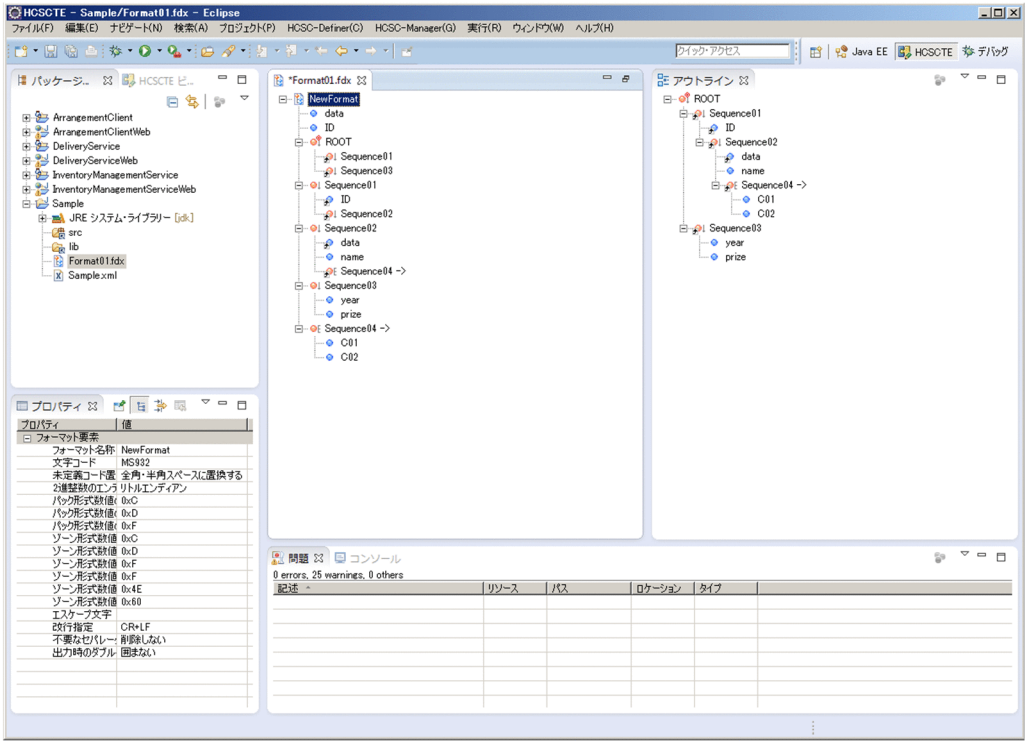
各機能の概要を次に説明します。

1.1.1　電文フォーマットの作成機能

電文フォーマットとは，サービスリクエスタ，サービスアダプタ，およびサービス部品の間でやり取りされる電文の形式です。利用するデータの形式が XML 形式の場合は XML 形式の電文フォーマットを，バイナリ形式（XML 形式以外）の場合はバイナリ形式の電文フォーマットを作成します。バイナリ形式の電文フォーマットの場合は，画面を利用して作成できます。

バイナリ形式の電文フォーマットの作成に利用する画面の例を次の図に示します。

図 1-1　バイナリ形式の電文フォーマットを作成する画面の例



バイナリ形式の電文フォーマットは，ウィザードで新規追加したあと，バイナリ形式の電文フォーマットを作成する画面で，電文フォーマットを構成する要素の詳細な内容を定義できます。

なお，すでに作成されているバイナリ形式の電文フォーマットを流用して，新たなバイナリ形式の電文フォーマットを作成する場合には，既存のバイナリ形式の電文フォーマットを複製して使用することもできます。

1.1.2　ユーザ定義受付の作成機能

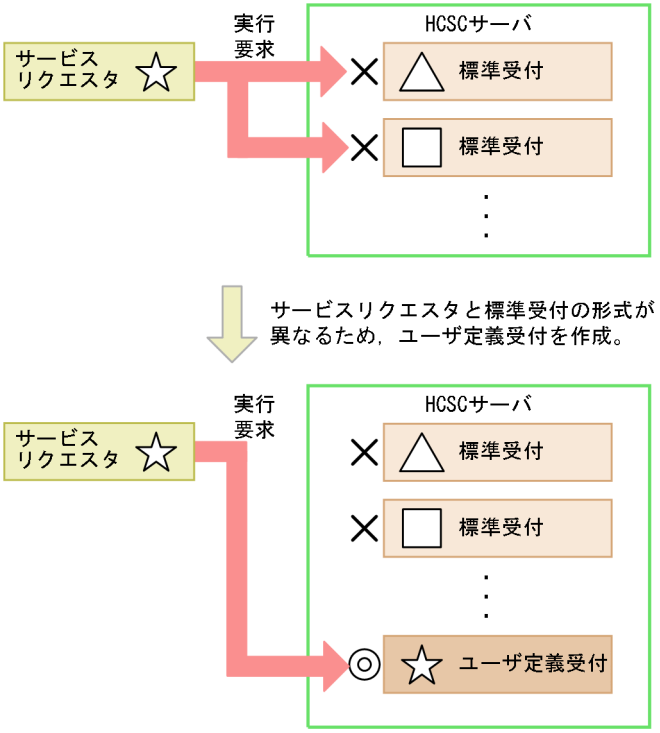
サービスリクエスタからの実行要求を受け付けるリクエスト受付には，標準受付とユーザ定義受付があります。

標準受付は，サービスリクエスタからのサービス部品の実行要求を受け付ける固定のインターフェースです。ユーザ定義受付は，サービスリクエスタからのサービス部品の実行要求に合わせて任意の形式を定義できるインターフェースです。

標準受付を利用するには，それぞれの標準受付の形式に合わせてサービスリクエスタを作成する必要があります。そのため，既存のサービスリクエスタを再利用する場合，標準受付のインターフェースの形式と合わないときには，サービスリクエスタを作り直したり，大きく改造したりする必要がありました。

このような場合，既存のサービスリクエスタからの実行要求を受け付けられる形式の**ユーザ定義受付**を作成できます。ユーザ定義受付の作成について，次の図に示します。

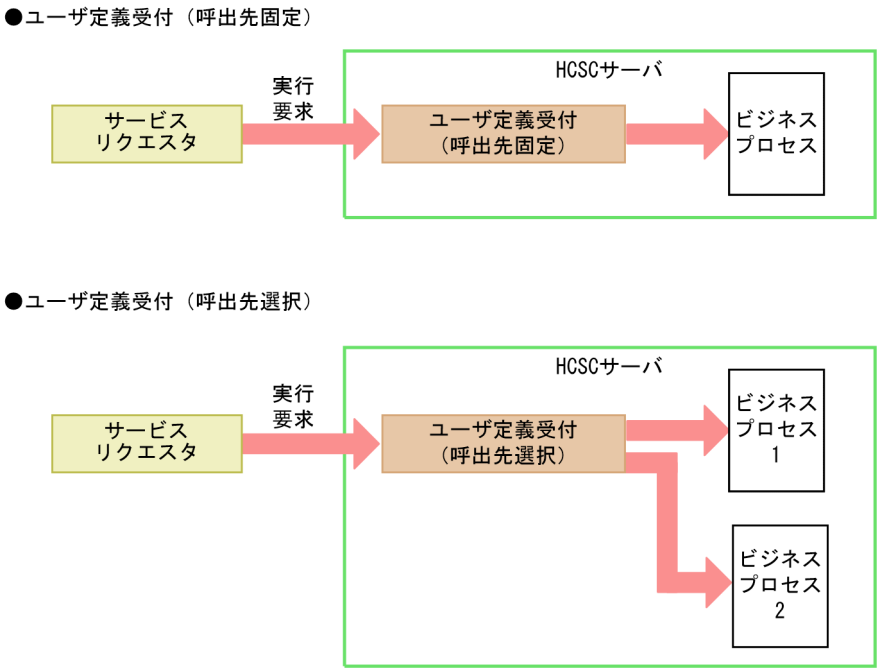
図 1-2　ユーザ定義受付の作成



また，ユーザ定義受付には，特定のビジネスプロセスを呼び出すユーザ定義受付（呼出先固定）と，呼び出すビジネスプロセスを選択できるユーザ定義受付（呼出先選択）があります。2 つを区別しない場合は，ユーザ定義受付と表記します。

ユーザ定義受付（呼出先固定）とユーザ定義受付（呼出先選択）の作成について，次の図に示します。

図 1-3　ユーザ定義受付（呼出先固定）とユーザ定義受付（呼出先選択）の作成



なお，ユーザ定義受付（呼出先選択）は HTTP 受付の場合だけ使用できます。

サービスプラットフォームには，ユーザ定義受付を効率的に作成するためのウィザードが用意されています。

1.1.3　サービスアダプタの作成機能

サービスアダプタとは，サービス部品を呼び出すための HCSC コンポーネントの一つです。

サービスプラットフォームでは，他システム，データベース，ファイルなど用途に合わせてさまざまなサービスアダプタを提供しています。また，提供しているサービスアダプタでは直接対応できないプロトコルを持つシステムにも対応できるように，カスタムアダプタ開発フレームワークを使用して開発する汎用カスタムアダプタを提供しています。

サービスプラットフォームで利用できるサービスアダプタの種類を次の表に示します。

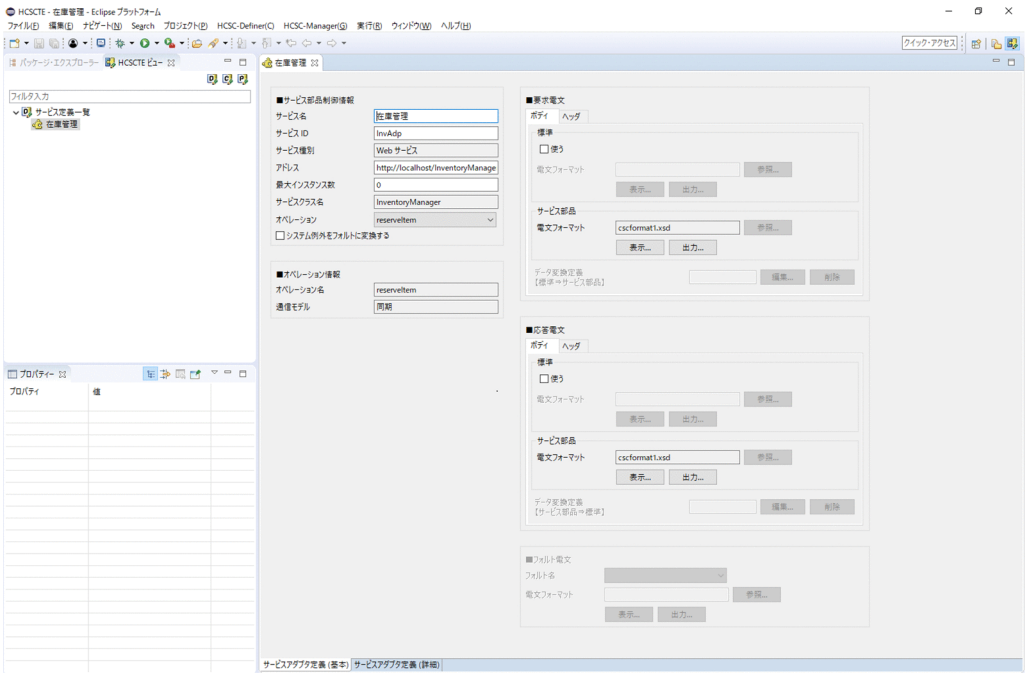
表 1-1　サービスアダプタの種類

種別	用途
SOAP アダプタ	Web サービス（SOAP 通信）を利用してサービス部品を呼び出す場合
SessionBean アダプタ	SessionBean を利用してサービス部品を呼び出す場合
MDB（WS-R）アダプタ	WS-R（WS-Reliability）を利用して MDB（Message Driven Bean）で作成したサービス部品を呼び出す場合
MDB（DB キュー）アダプタ	DB キューを利用して非同期のサービス部品を呼び出す場合
DB アダプタ	データベースの操作をサービス部品として利用する場合
TP1 アダプタ	既存の OpenTP1 システム内にあるサービス部品を呼び出す場合
ファイルアダプタ	HCSC サーバ上のローカルディスクに対して直接ファイルの入出力をする場合
Object Access アダプタ	既存の TPBroker システム（Object Wrapper システム）のサービス部品を呼び出す場合
Message Queue アダプタ	既存のメッセージキュー（IBM MQ システム）に対してメッセージの送受信をする場合
FTP アダプタ	サービスプラットフォームと FTP サーバとの間でファイルの送受信をする場合
ファイル操作アダプタ	ファイルのレイアウト変換，文字コード変換，複製，移動，削除，圧縮，伸張，および一覧取得をする場合
メールアダプタ	SMTP プロトコルをサポートしたメールサーバをサービスプラットフォーム上でサービス部品として呼び出す場合
HTTP アダプタ	Web サーバ上に公開されているリソースや，REST スタイルで公開されている Web サービス（RESTful Web サービス）を呼び出す場合
コマンドアダプタ	オペレーティングシステム上で動作する外部コマンドを起動し，実行結果を取得する場合
SFTP アダプタ	サービスプラットフォームと SFTP サーバとの間で SFTP プロトコルによるファイル転送をする場合
Kafka アダプタ	サービスプラットフォームから Kafka にメッセージを送信する場合

種別	用途
gRPC アダプタ	gRPC で作成されたサービスにメッセージを送信する場合
汎用カスタムアダプタ	サービスプラットフォームで提供するサービスアダプタでは対応できないプロトコルを持つシステムのサービス部品を呼び出す場合

サービスアダプタの作成に利用する画面の例を次の図に示します。

図 1-4　サービスアダプタを作成する画面の例



ウィザードでサービスアダプタを新規追加したあとに，これらの画面で詳細な内容を定義します。なお，すでに作成されているサービスアダプタを流用して，新たなサービスアダプタを作成する場合には，既存のサービスアダプタを複製して使用することもできます。

サービスアダプタの定義の詳細については，マニュアル「サービスプラットフォーム 開発ガイド 受付・アダプタ定義編」の「3.　サービスアダプタの定義」を参照してください。

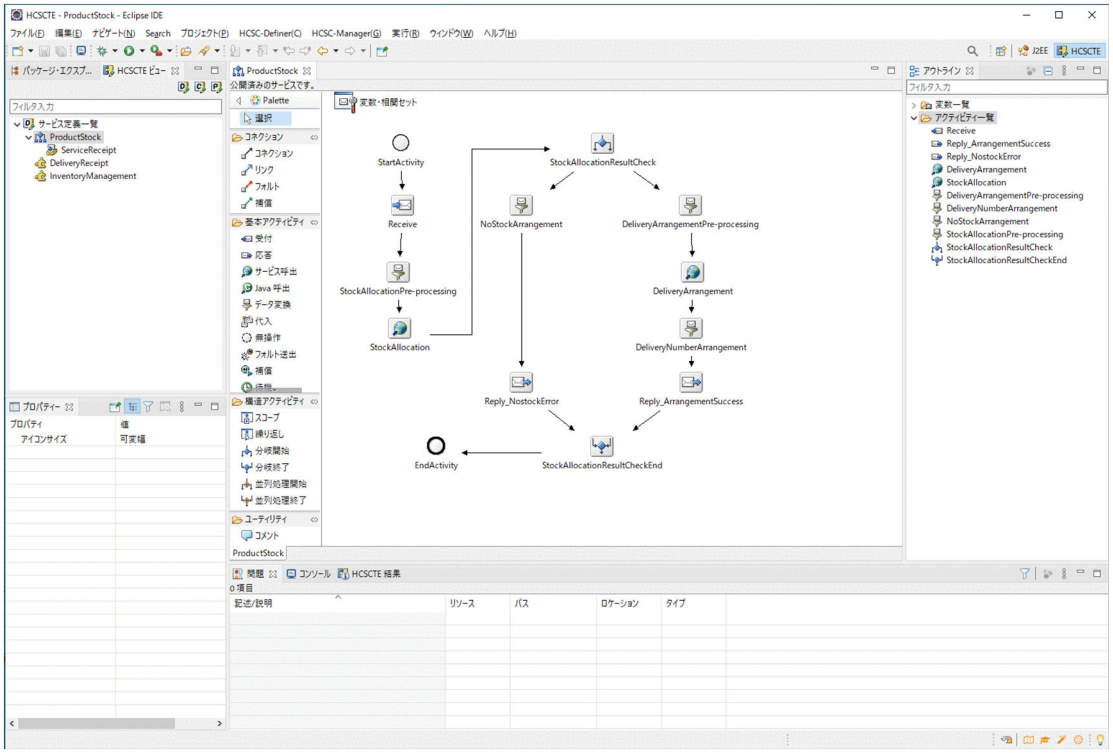
1.1.4　ビジネスプロセスの作成機能

ビジネスプロセスとは，複数のサービスの処理の順番や条件などを定義して，一連の業務の流れとして定義したものです。

画面で定義したビジネスプロセスは BPEL として記述されます。

ビジネスプロセスの作成に利用する画面の例を次の図に示します。

図 1-5　ビジネスプロセスを作成する画面の例



ビジネスプロセスを作成する画面では，アクティビティと，アクティビティ同士の関係を規定する連結線，および変数・相関セットを利用してビジネスプロセスを定義します。

アクティビティとは，ビジネスプロセスの構成要素となる部品であり，処理の構造も表現したものです。複数のアクティビティを連結して，ビジネスプロセスのフローを規定します。ビジネスプロセスは，アクティビティを画面上で配置・連結して定義します。

また，ビジネスプロセスと各アクティビティには，変数と相関セットを定義できます。

変数は，ビジネスプロセスの処理の中で，条件式の項の値などに変数値を利用する場合に宣言します。

相関セットは，サービスリクエスタから送信された要求電文を一意に識別するために利用する文字列です。

なお，すでに作成されているビジネスプロセスを流用して，新たなビジネスプロセスを作成する場合には，既存のビジネスプロセスを複製して使用することもできます。

さらに，サービスプラットフォームでは，ビジネスプロセスを作成するときに，ビジネスプロセスの実行状態の管理を設定できます。また，より高度なビジネスプロセスとして，複数のビジネスプロセスを階層化したビジネスプロセスを作成できます。

(1)　ビジネスプロセスの実行状態の管理

ビジネスプロセスを新規追加する時には，実行環境でのビジネスプロセスの実行状態を管理するかどうか設定する機能があります。

ビジネスプロセスの実行状態の管理を設定すると，ビジネスプロセスの実行途中に障害が発生した場合に，ビジネスプロセスのどの段階で障害が発生し，どこまでが実行されたのか把握できます。停止したビジネスプロセスの状況を把握することで，ビジネスプロセス内の特定の処理から再度実行できます。

ビジネスプロセスの実行状態の管理については，「1.3.4　実行履歴の管理機能」を参照してください。

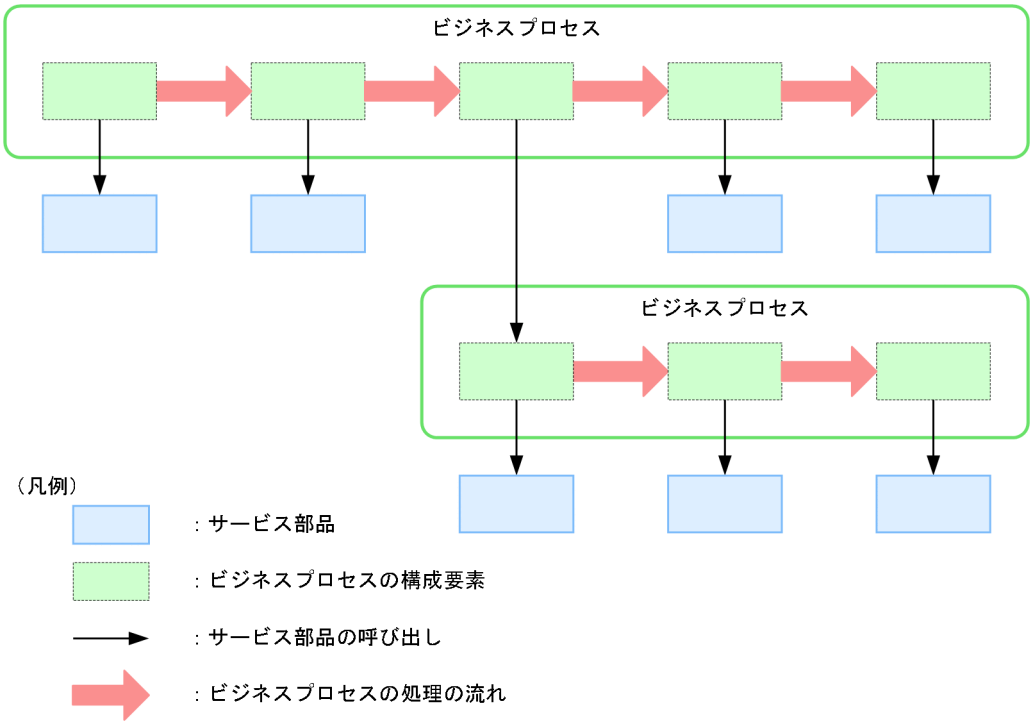
なお，ビジネスプロセスの実行状態を管理するかどうかが選択できるので，処理の速度などの性能を重視するビジネスプロセスでは，ビジネスプロセスの実行状態を管理しない設定もできます。

(2)　ビジネスプロセスの階層化

あるビジネスプロセスからほかのビジネスプロセスを呼び出すように定義して，ビジネスプロセスを階層化できます。

ビジネスプロセスの階層化の概要を次の図に示します。

図 1-6　ビジネスプロセスの階層化



「図 1-6　ビジネスプロセスの階層化」のように，ビジネスプロセスから各サービスを呼び出す一連の流れの中に，ビジネスプロセスを呼び出す処理を定義できます。

ビジネスプロセスで呼び出す処理としてビジネスプロセスを部品化できるため，より複雑な業務処理もビジネスプロセス化できます。

(3)　上流工程で作成したビジネスプロセス定義のインポート

ビジネスプロセスの設計時に，設計ツール（BPMN を用いた上流設計ツール）を使用して作成した BPEL 定義を，サービスプラットフォームの開発環境（Service Architect）へインポートできます。インポート

したBPEL定義は，ビジネスプロセス定義として変換され，画面上にアクティビティとして表示されます。メッセージやプロセスの詳細化などの編集ができます。

1.1.5　データ変換定義の作成機能

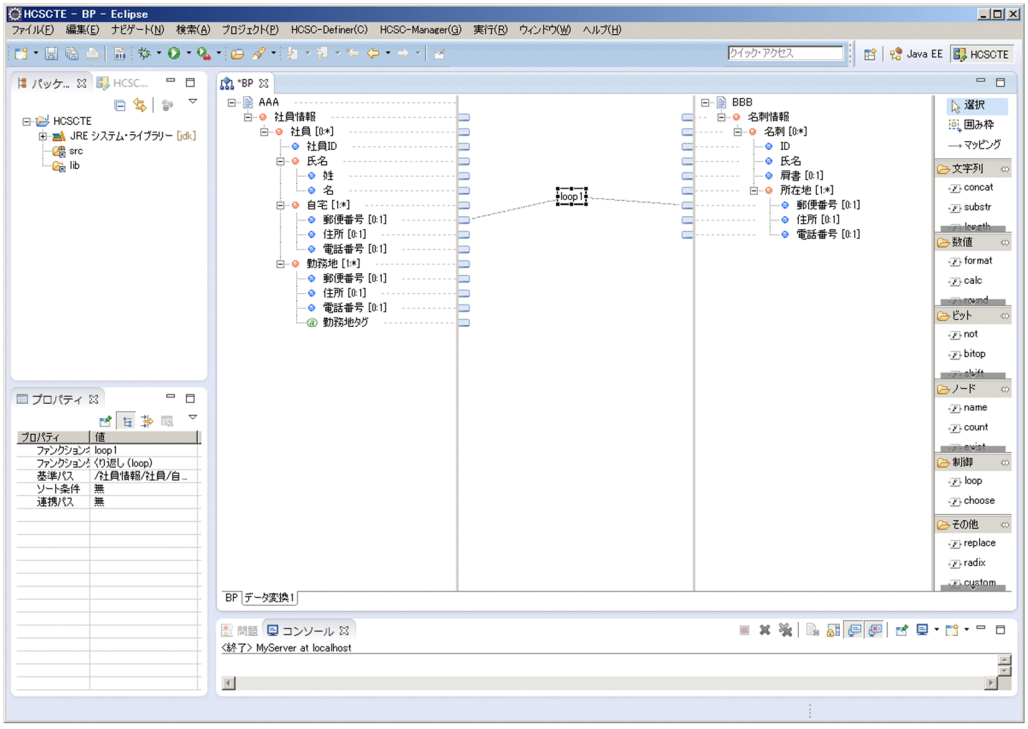
データ変換定義とは，実行環境でサービスリクエスタから送信される電文データの変換方法や，サービス部品間で受け渡すデータのフォーマットの変換方法について定義したものです。

変換元／変換先のスキーマファイルを作成したあと，作成したスキーマファイルを基に，変換元の要素や属性を変換先のどの要素や属性に変換（マッピング）するかなど，変換時の処理について定義します。

あらかじめExcelで変換元と変換先の組み合わせを定義したファイルを作成しておくことで，マッピングを画面上で一括して設定することもできます。接続数が多い場合や複数のビジネスプロセスで同じようにマッピングする場合，作成したファイルを利用することで開発効率を向上できます。

データ変換定義の作成に利用する画面の例を次の図に示します。

図 1-7　データ変換定義を作成する画面の例

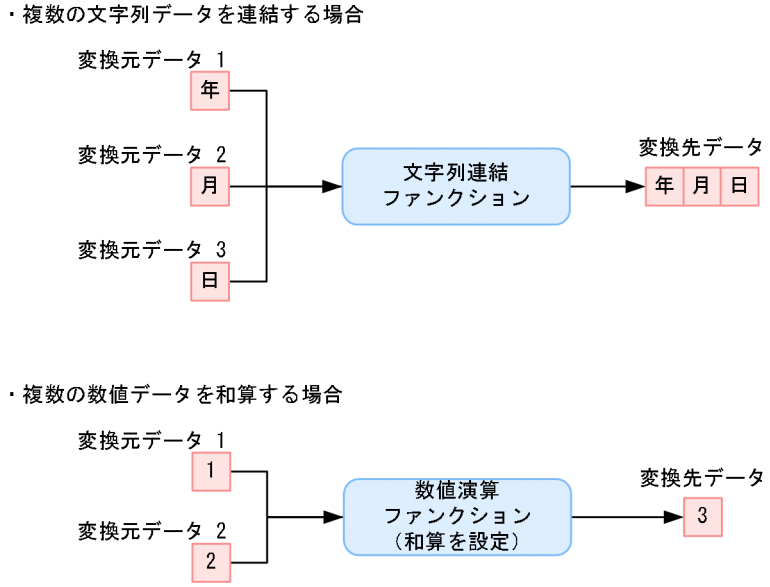


データ変換を定義する画面では，変換元と変換先を接続（マッピング）し，変換元から変換先への値の代入をビジュアルに定義できます。

また，変換元のデータを加工してから変換先に代入したい場合は，ファンクションを設定できます。ファンクションには，文字列の連結や数値演算などの機能があります。

ファンクションを利用したデータの加工の例を次の図に示します。

図 1-8　ファンクションを利用したデータの加工の例



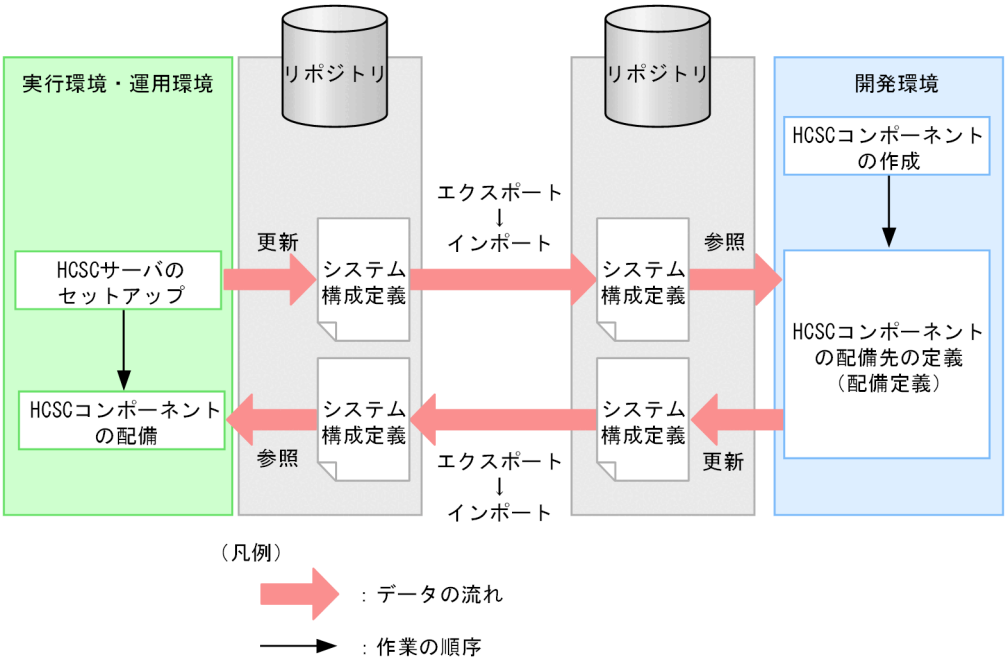
また，変換元データが複数ある場合のくり返し処理や，一定の条件に合致する場合にだけデータを代入するなどの設定もできます。

1.1.6　配備定義機能

HCSC コンポーネントの配備は，開発環境と運用環境でリポジトリを介して相互に情報をやり取りしながら実施します。

HCSC コンポーネントの配備の概要について，次の図に示します。

図 1-9　HCSC コンポーネントの配備



運用環境では，HCSC サーバおよびクラスタのセットアップ時に，リポジトリにある実行環境の構成情報（システム構成定義）が更新されます。更新されたシステム構成定義は，リポジトリの管理機能でエクスポートします。

開発環境では，運用環境でエクスポートした HCSC サーバおよびクラスタのセットアップ情報を含むシステム構成定義をリポジトリにインポートし，システム構成定義に HCSC コンポーネントをどの HCSC サーバに配備するかを定義（配備定義）して更新し，更新したシステム構成定義をエクスポートします。

ポイント

開発環境で配備定義するときは，サービス部品をクラスタ，または単一の HCSC サーバ単位で配置します。

そのあと，運用環境では，開発環境で更新したシステム構成定義をリポジトリにインポートします。システム構成定義の内容に従って，実行環境へ HCSC コンポーネントを配備します。

リポジトリのエクスポートについては，マニュアル「サービスプラットフォーム システム構築・運用ガイド」の「4.2　リポジトリのエクスポート」を参照してください。リポジトリのインポートについては，マニュアル「サービスプラットフォーム システム構築・運用ガイド」の「4.3　リポジトリのインポート」を参照してください。

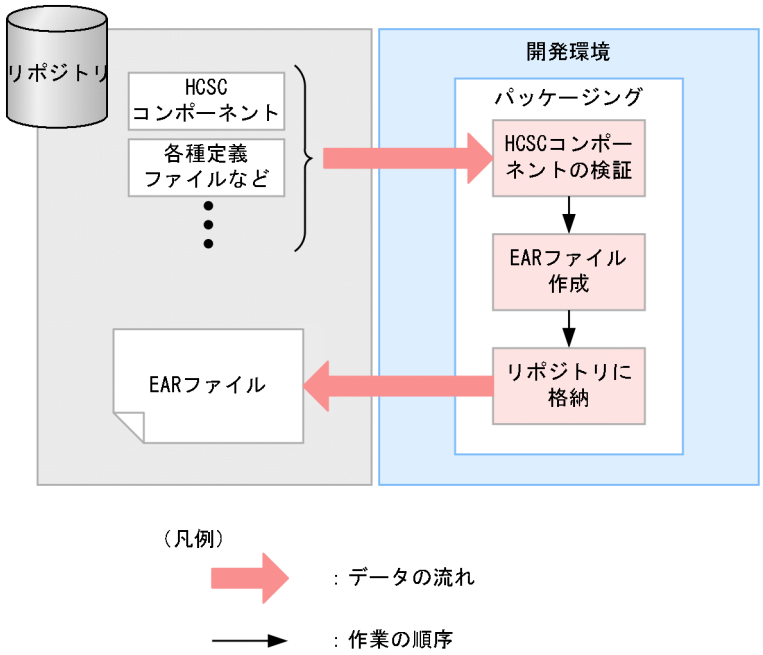
1.1.7　パッケージング機能

パッケージングとは，開発環境で作成した HCSC コンポーネントと必要な情報を EAR ファイル形式で組み立てて，リポジトリに格納することをいいます。

開発環境では，画面を利用してパッケージングを実行できます。

パッケージングの概要を次の図に示します。

図 1-10　パッケージング機能



開発環境からパッケージング機能を実行すると，パッケージングに必要な情報がリポジトリから収集され，HCSC コンポーネントが検証されます。

検証の結果，HCSC コンポーネントに必要な定義がすべて定義されている場合，EAR ファイルが作成されます。

HCSC コンポーネントに不備がある場合，パッケージングは中止されます。

作成された EAR ファイルは，リポジトリに格納されます。運用環境では，リポジトリに格納された EAR ファイル形式の HCSC コンポーネントを実行環境に配備します。

1.1.8　配備から開始または停止から削除の一括実行機能

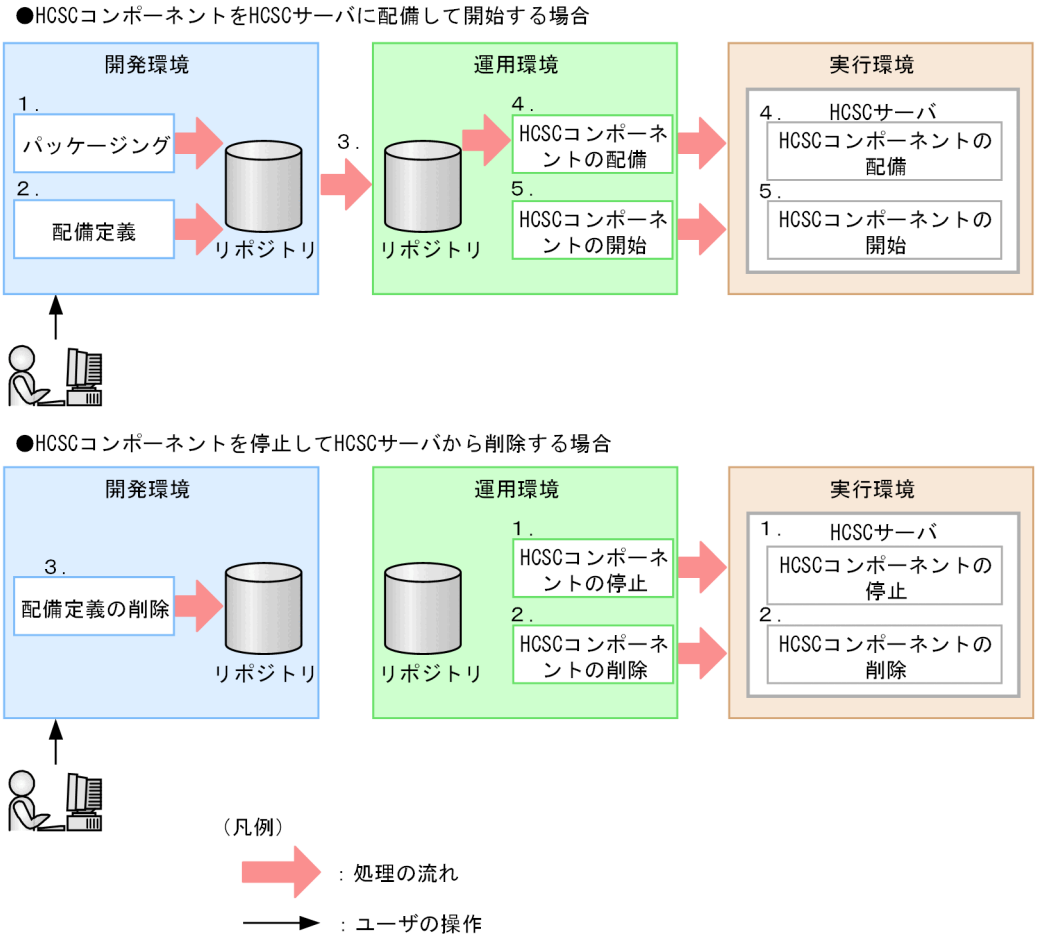
開発環境では，次の一連の処理を一括で実行できます。

- HCSC コンポーネントを HCSC サーバに配備して開始する
- HCSC コンポーネントを停止して HCSC サーバから削除する

開発環境または運用環境で一つ一つ実施していた作業を開発環境で一括実行するため，ユーザの操作負担を軽減できます。なお，一括実行は，システム開発時，または単体テストから結合テスト時を対象としています。

一括実行の処理の流れを次の図に示します。図中の番号は，開発環境でユーザが操作したあとに，自動的に実行される処理の順番を示しています。

図 1-11　一括実行の処理の流れ



1.1.9　サービスリクエスタの作成機能

サービスリクエスタとは，業務担当者からのサービス部品の実行要求を受け付け，各 HCSC コンポーネントにサービス部品の実行要求を送信するアプリケーションです。

サービスリクエスタは，開発環境で作成し，実行環境に組み込んで利用します。

サービスプラットフォームの開発環境の画面には，サービスリクエスタの作成に必要な WSDL やスタブを取得するなど，サービスリクエスタの作成を支援する機能があります。

1.1.10　ビジネスプロセスのデバッグ機能

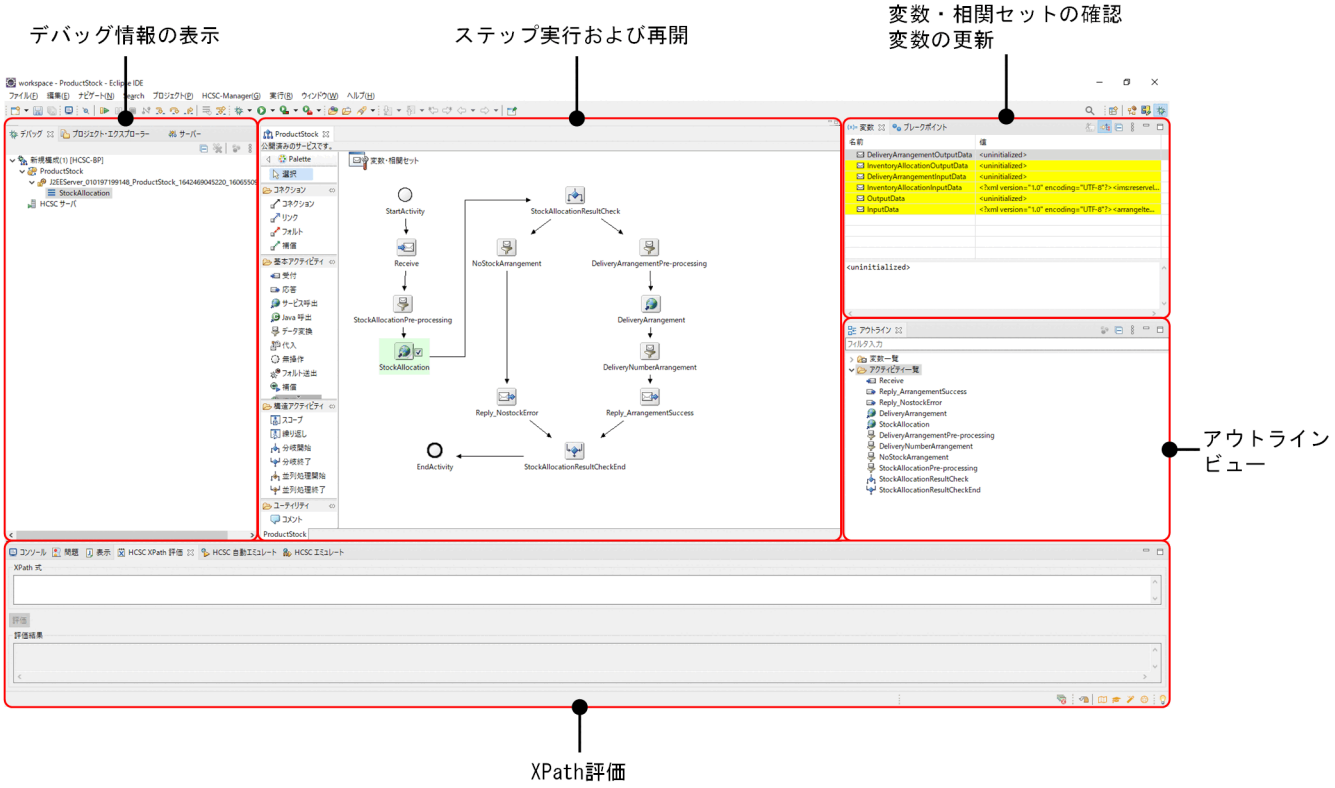
ビジネスプロセスのデバッグ機能とは，ビジネスプロセスに着目したテストおよびデバッグを実行できる機能です。

ビジネスプロセスのデバッグ機能を利用することで，HCSC コンポーネントを実行環境に配備しなくても，開発環境上でビジネスプロセスをデバッグできます。ビジネスプロセスの定義およびデバッグが同じ環境上でできるため，開発効率を向上できます。

ビジネスプロセスのデバッグでは，ビジネスプロセスの処理がアクティビティ間をどのように進んでいるかを確認したり，アクティビティで使用している変数の中身を確認・更新したりできます。

ビジネスプロセスのデバッグで利用する画面の例と利用できる機能を次の図に示します。

図 1-12　ビジネスプロセスのデバッグで利用する画面の例と利用できる機能



ビジネスプロセスのデバッグで利用できる機能を次に示します。

- ステップ実行および再開

  アクティビティにブレークポイントを設定することで，ビジネスプロセスの処理を任意のタイミングで中断できます。中断しているビジネスプロセスでは，アクティビティ単位で中断しながら，逐次処理を進めることができます。

- デバッグ情報の表示

  デバッグ中のビジネスプロセスの名称，およびアクティビティの名称を表示します。

- 変数・相関セットの確認

  ビジネスプロセスで使用している変数および相関セットの値を確認します。

- 変数の更新

  ビジネスプロセスで使用している変数の値を変更し，ビジネスプロセスに反映します。

- XPath 評価
  分岐アクティビティや代入アクティビティに指定した条件式の妥当性を評価します。
- アウトラインビューの表示
  デバッグ中のビジネスプロセスで定義した変数の名称，およびアクティビティの名称を表示します。

ビジネスプロセスのデバッグ機能では，サービスおよびサービスリクエスタをエミュレートして利用することもできます。

1.2　実行環境の機能概要

サービスプラットフォームの実行環境は，SOA を実現するための環境です。実行環境では，主に次の機能を提供しています。

- サービス振り分け実行機能
- 各種サービス呼び出し機能
- ビジネスプロセス実行機能

各機能の概要を次に説明します。

1.2.1　サービス振り分け実行機能

サービスプラットフォームでは，要求を行う利用者側のシステムを「サービスリクエスタ」と呼び，機能を提供する側の業務処理を「サービス部品」と呼びます。また，サービス連携を実現するエンタープライズサービスバス（ESB：Enterprise Service Bus）や BPEL 準拠のビジネスプロセスを管理する SOA 環境を「HCSC サーバ」と呼びます。利用者がサービス部品の実行要求をすると，SOA は HCSC サーバに配備したビジネスプロセスやサービスアダプタの中から，適切なサービス部品を呼び出して実行します。また，業務・機能を“サービス”として管理し，要求に対してサービス振り分けを実行します。

サービス振り分け実行機能の詳細については，「2.　さまざまなシステムと接続するための機能」を参照してください。

1.2.2　各種サービス呼び出し機能

サービスプラットフォームでは，さまざまな種類のサービスを呼び出すためのサービスアダプタを提供しています。

次に示すサービスアダプタを利用することで，柔軟にシステムを構築でき，既存システムのデータをサービス部品として有効に活用できます。

- Web サービス（SOAP 通信）を利用するための SOAP アダプタ
- SessionBean を利用するための SessionBean アダプタ
- WS-R を利用して MDB で作成したサービス部品を呼び出すための MDB（WS-R）アダプタ
- DB キューを利用して非同期のサービス部品を呼び出すための MDB（DB キュー）アダプタ
- データベースをサービス部品として利用するための DB アダプタ
- OpenTP1 や XDM/DCCM3 上のシステムを利用するための TP1 アダプタ
- ファイルを介した業務処理システムを利用するためのファイルアダプタ
- メインフレームなどの基幹システムを利用するための Object Access アダプタ

- 既存のメッセージキュー（IBM MQ システム）に対してメッセージの送受信をするための Message Queue アダプタ
- FTP サーバと接続してファイル転送をするための FTP アダプタ
- ファイルのフォーマット変換，複製，削除などをするためのファイル操作アダプタ
- SMTP プロトコルをサポートしたメールサーバをサービスプラットフォーム上で利用するためのメールアダプタ
- Web サーバ上に公開されているリソースや，REST スタイルで公開されている Web サービス（RESTful Web サービス）を利用するための HTTP アダプタ
- オペレーティングシステム上で動作する外部コマンドを起動し，実行結果を取得するためのコマンドアダプタ
- サービスプラットフォームと SFTP サーバとの間で SFTP プロトコルによるファイル転送をするための SFTP アダプタ
- サービスプラットフォームから Kafka にメッセージを送信するための Kafka アダプタ
- gRPC で作成されたサービスにメッセージを送信するための gRPC アダプタ
- サービスプラットフォームで提供する上記サービスアダプタでは対応できないプロトコルを持つシステムと接続してサービスを呼び出すための汎用カスタムアダプタ

サービスアダプタの詳細については「2.　さまざまなシステムと接続するための機能」を参照してください。

1.2.3　ビジネスプロセス実行機能

ビジネスプロセスとは，複数のサービス部品の処理の順番や条件などを定義して，一連の業務の流れとして定義したものです。

サービスプラットフォームでは，ビジネスプロセスに定義されている順に，連続してサービス部品を呼び出して実行できます。

ビジネスプロセス実行機能は，サービス呼び出しの流れを制御するためのビジネスプロセスの開発，および実行を行います。

ビジネスプロセスの詳細については，「3.　ビジネスプロセスの機能」を参照してください。

1.3　運用環境の機能概要

サービスプラットフォームの運用環境は，実行環境の構築，必要な情報の配置，運用管理，および実行履歴の管理をする環境です。運用環境では，主に次の機能を提供しています。

- 実行環境のセットアップ機能
- HCSC コンポーネントの配備機能
- 実行環境での運用を管理する機能
- 実行履歴の管理機能

これらの機能は，コマンドまたは運用環境の画面を利用して実行します。

各機能の概要を次に説明します。

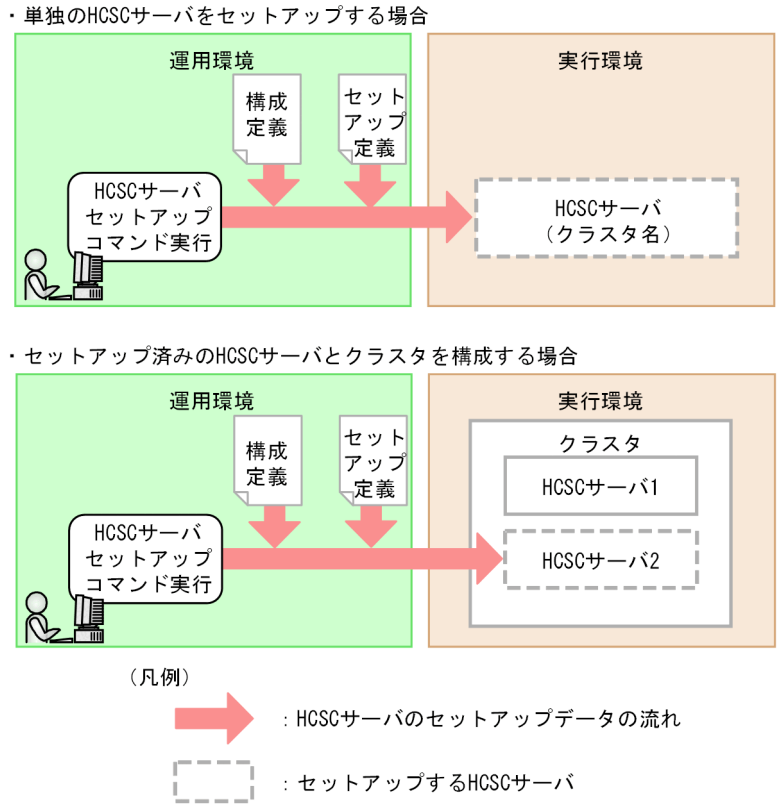
1.3.1　実行環境のセットアップ機能

運用環境では，コマンドを利用して，実行環境の HCSC サーバをセットアップできます。また，HCSC サーバをセットアップするときに，あわせてクラスタもセットアップできます。

(1)　HCSC サーバおよびクラスタのセットアップ機能

HCSC サーバのセットアップの概要について，次の図に示します。

図 1-13　HCSC サーバのセットアップの概要



運用環境からコマンドを実行すると，あらかじめ作成していた HCSC サーバの構成定義ファイルとセットアップ定義ファイルからセットアップに必要な情報を取得して HCSC サーバがセットアップされます。

単独の HCSC サーバをセットアップする場合

コマンド実行時にクラスタ名を指定しますが，この段階では，単独の HCSC サーバであるため，クラスタの構成にはなりません。

ほかの HCSC サーバをセットアップするときに，クラスタ名を指定してセットアップすることで，クラスタを構成できます。

セットアップ済みの HCSC サーバとクラスタを構成する場合

コマンド実行時に，ほかの HCSC サーバをセットアップしたときに指定したクラスタ名を指定すると，すでにセットアップされている HCSC サーバとクラスタを構成する HCSC サーバがセットアップできます。

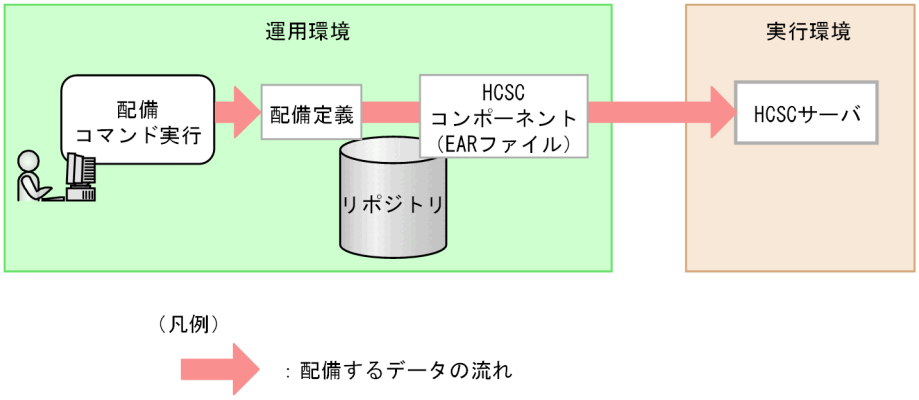
1.3.2　HCSC コンポーネントの配備機能

配備とは，HCSC コンポーネントを実行環境で利用できるよう，実行環境に必要な情報を配置することです。

HCSC コンポーネントは，運用環境でコマンドを実行して実行環境に配備できます。

配備の概要について，次の図に示します。

図 1-14　配備の概要



HCSC コンポーネントを配備する場合，開発環境で作成してリポジトリに格納されている配備定義の情報に従って，HCSC コンポーネント（EAR ファイル）を実行環境（HCSC サーバ）に配備します。

1.3.3　実行環境での運用を管理する機能

運用環境では，実行環境での業務の運用を管理できます。

実行環境での業務を開始する場合や，実行環境の構成を変更するために業務を停止する場合，また障害の発生によって業務を停止する場合など，必要に応じて運用環境で実行環境の業務の運用状態を確認し，操作できます。

運用環境から実施できる実行環境の操作について，次の表に示します。

表 1-2　運用環境から実施できる実行環境の操作

操作内容	実行インターフェース	
	画面	コマンド
HCSC サーバのセットアップ	－	○
HCSC サーバのアンセットアップ	－	○
HCSC サーバの起動	○	○
HCSC サーバの起動（クラスタ内一括起動）	－	○
HCSC サーバの停止	○	○
HCSC サーバの停止（クラスタ内一括停止）	－	○
HCSC サーバの情報の参照※	○	○
HCSC サーバのセットアップ情報の参照	－	○
HCSC サーバの定義	－	○
HCSC サーバの定義内容の確認※	○	○

操作内容	実行インターフェース	
	画面	コマンド
HCSC サーバの定義の変更	−	○
HCSC サーバのバージョンアップ	−	○
標準受付の開始	○	○
標準受付の停止	○	○
ユーザ定義受付の配備	−	○
配備済みユーザ定義受付の削除	−	○
ユーザ定義受付の開始	○	○
ユーザ定義受付の停止	○	○
ユーザ定義受付の情報の参照※	○	○
ユーザ定義受付の定義情報の確認	○	○
ユーザ定義受付の定義情報の変更	−	○
サービスアダプタの配備	−	○
配備済みサービスアダプタの削除	−	○
サービスアダプタの開始	○	○
サービスアダプタの停止	○	○
サービスアダプタの情報の参照※	○	○
ビジネスプロセスの配備	−	○
配備済みビジネスプロセスの削除	−	○
ビジネスプロセスの開始	○	○
ビジネスプロセスの停止	○	○
ビジネスプロセスの情報の参照※	○	○
HCSC サーバ，標準受付，および HCSC コンポーネントの状態の確認	○	○
HCSC サーバのリソースの稼働状態の表示	○	○
アプリケーションの操作	−	○
HCSC-Manager 環境のバックアップ	−	○
HCSC-Manager 環境のリストア	−	○
サービス情報の変更	−	○
サービス情報の確認	○	○

（凡例）

○：利用できます。

－：利用できません。

注※

画面とコマンドで参照できる情報が異なります。

1.3.4　実行履歴の管理機能

運用環境では，ビジネスプロセスのプロセスインスタンスの実行履歴を管理できます。

(1)　プロセスインスタンスの実行履歴の管理

プロセスインスタンスの実行履歴は，ビジネスプロセスのプロセスインスタンスの実行状況を履歴としてデータベースに記録（永続化）して管理します。

(a)　プロセスインスタンスの実行履歴の検索

条件を指定して，プロセスインスタンスの実行履歴を検索できます。相関セット，開始日時または状態を検索キーとして利用できます。

(b)　プロセスインスタンスの実行履歴の詳細情報の参照

プロセスインスタンスの実行履歴の詳細情報を参照できます。次に示す情報を参照できます。

- 相関セット定義名および相関セット値
- 各プロセスインスタンスが保持する変数の変数定義名，および変数値
- ビジネスプロセス内に定義された各アクティビティの状態

(c)　プロセスインスタンスの実行履歴の削除

ビジネスプロセスの確認のために使用するデータベースから不要なプロセスインスタンスの実行履歴を削除できます。1 件ずつ削除することも，状態を指定して一括して削除することもできます。

(d)　実行中のプロセスインスタンスの再実行

フォルトが発生するなどの理由で処理できなかったビジネスプロセスのプロセスインスタンスを確認して，実行が中断したプロセスインスタンスを再実行できます。プロセスインスタンスの再実行は，1 件ずつ再実行することも一括して再実行することもできます。

1.4　HCSC サーバのクラスタ構成

複数の HCSC サーバを組み合わせて，より冗長なクラスタ構成の環境を構築できます。

1.4.1　ロードバランス機能を利用した HCSC サーバの冗長構成

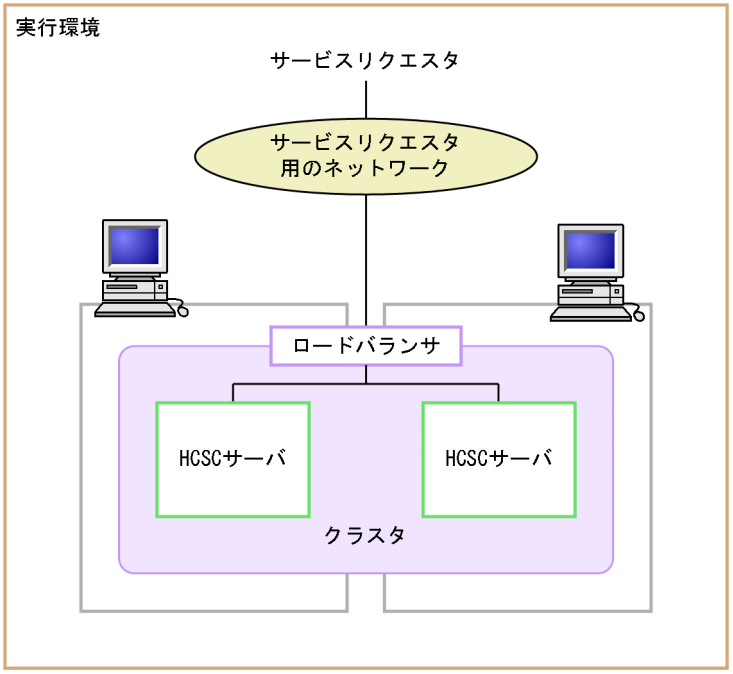
ロードバランス機能とは，複数台の HCSC サーバを並列に並べることで，サービスリクエスタからのサービスの実行要求の一点集中による J2EE サーバの負荷を軽減するための機能です。サービスプラットフォームでは，次に示すロードバランサを使用して負荷を分散できます。

- 汎用ロードバランサ
  同期（Web サービス）の標準受付の場合に使用します。
- CTM
  同期（SessionBean）の標準受付の場合に使用します。

サービスプラットフォームの実行環境では，ロードバランサを使用して，複数の HCSC サーバを組み合わせて，より冗長なクラスタ構成の環境を構築できます。

HCSC サーバのロードバランスクラスタ構成について，次の図に示します。

図 1-15　HCSC サーバのロードバランスクラスタ構成



同期（Web サービス，SessionBean）の標準受付の場合だけ，ロードバランサを使用してサービスリクエスタからのサービス部品の実行要求を，複数の HCSC サーバに分散して送信できます。

使用するロードバランサによっては，運用を停止しないで，ロードバランスクラスタを構成する HCSC サーバを追加したり，削除したりできます。また，HCSC サーバに配備済みのサービスアダプタやビジネスプロセスを停止しないで，サービスアダプタやビジネスプロセスを新規追加することもできます。

1.4.2　クラスタソフトウェアを利用した HCSC サーバの冗長構成

クラスタソフトウェアとは，システムの信頼性および稼働率の向上を目的とした，サーバプログラムを含めたシステムの切り替えを実現するプログラムです。サービスプラットフォームでは，次に示すクラスタソフトウェアと連携できます。

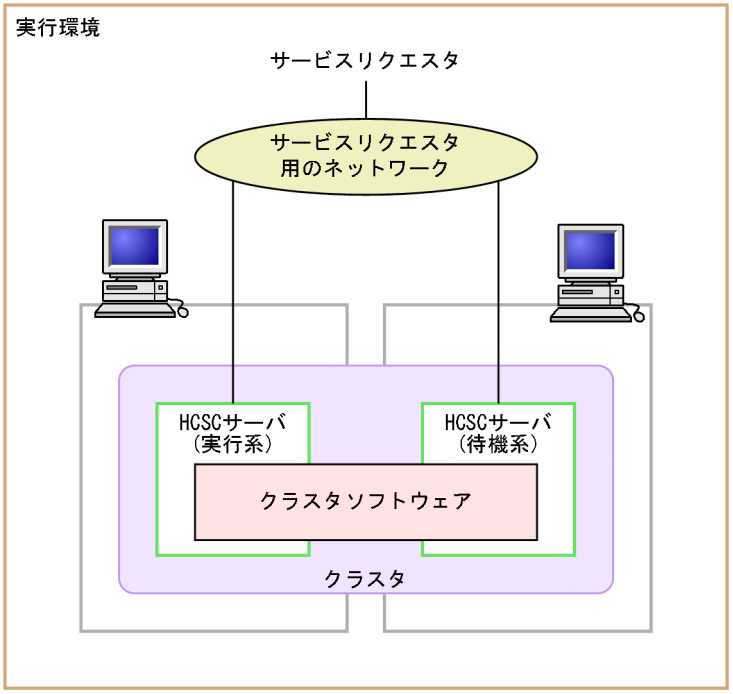
- Windows Server Failover Cluster（Windows の場合）
- HA モニタ（UNIX の場合）

サービスプラットフォームの実行環境では，クラスタソフトウェアと連携して，同じ内容の HCSC サーバを 2 つ組み合わせて，より冗長なクラスタ構成（コールドスタンバイ構成）の環境を構築できます。

クラスタを構成する 2 つの HCSC サーバのセットアップ情報や，配備される HCSC コンポーネントは，同じ内容である必要があります。

HCSC サーバのクラスタ構成について，次の図に示します。

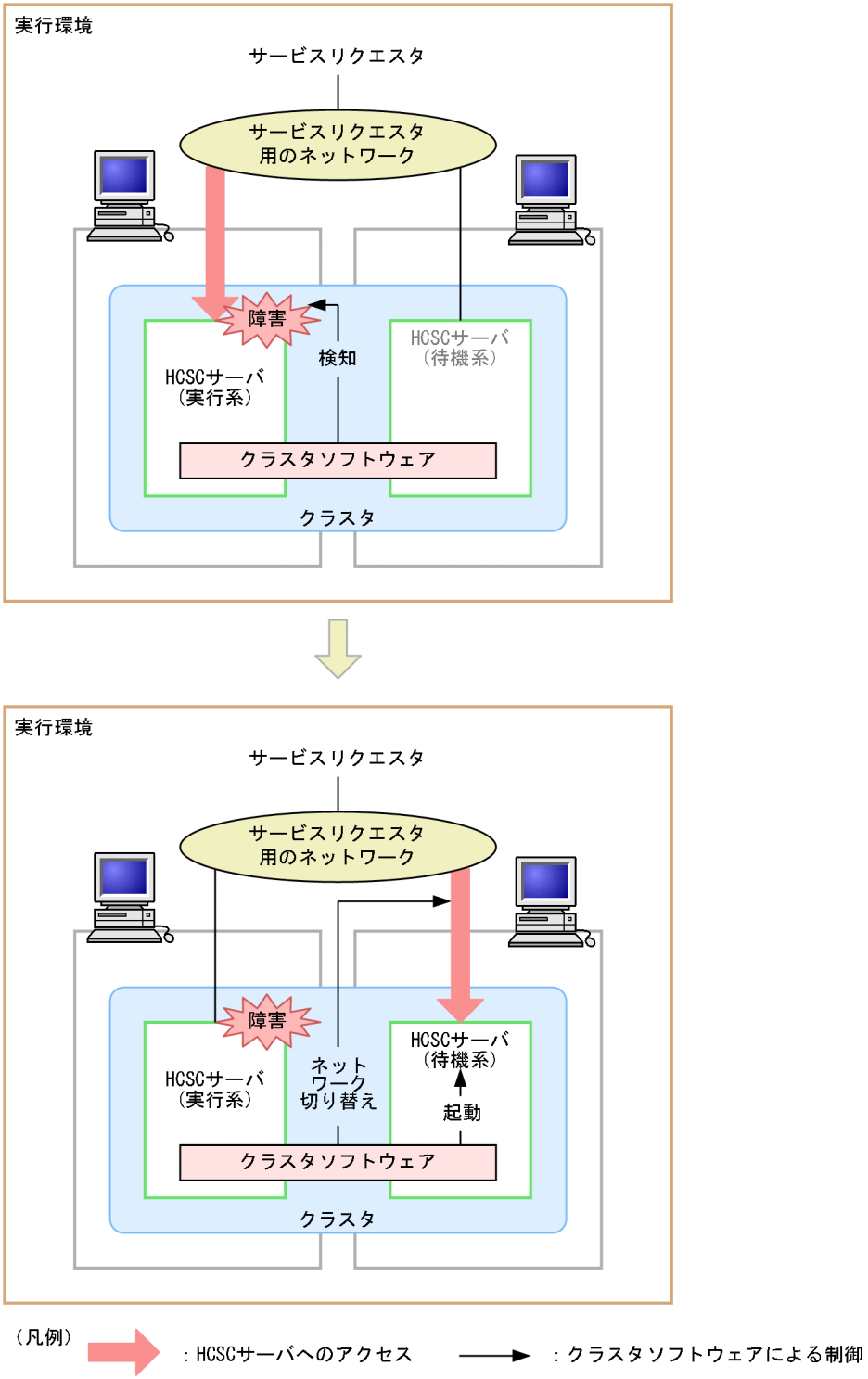
図 1-16　HCSC サーバのクラスタ構成



サービスプラットフォームの実行環境では，クラスタソフトウェアと連携して，実行系と待機系を 1 対 1 で運用するシステム（1:1 系切り替えシステム）を構築できます。

1:1 系切り替えシステムでは，実行系に何らかの障害が発生すると，クラスタソフトウェアがこれを検知して，自動的に待機している系に切り替えて業務を続行します。障害発生時の系の切り替えを次の図に示します。

図 1-17　障害発生時の系の切り替え



障害の発生から系の切り替えまでの流れは次のとおりです。

1. 実行系 HCSC サーバで障害が発生します。

2. クラスタソフトウェアが実行系 HCSC サーバの障害を検知します。

3. 実行系 HCSC サーバのあるマシンへのネットワークが切断されます。

4. クラスタソフトウェアがネットワークを待機系 HCSC サーバのあるマシンに切り替えます。

5. クラスタソフトウェアが待機系 HCSC サーバを起動します。

なお，クラスタソフトウェアには，障害を検知した場合に待機系 HCSC サーバを自動的に起動するよう設定しておきます。

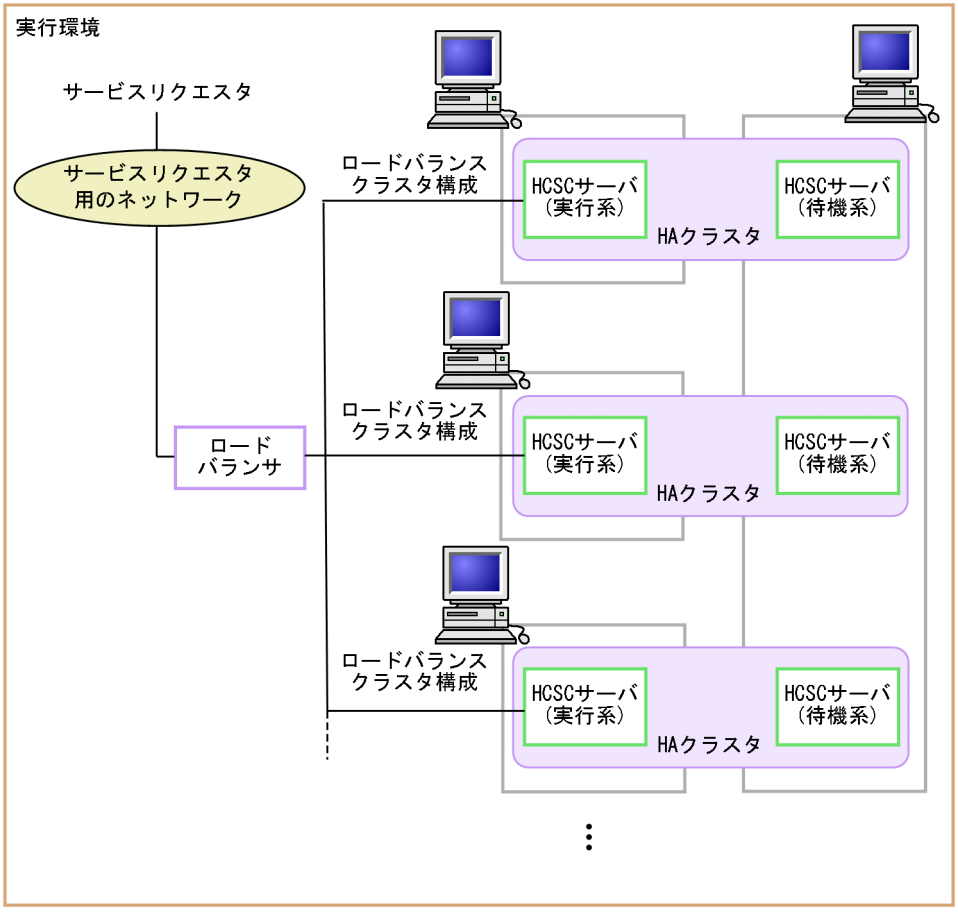
1.4.3　N 対 1 クラスタ構成を利用した HCSC サーバの冗長構成

N 対 1 クラスタ構成とは，ロードバランスクラスタを構成するそれぞれの HCSC サーバに対して HA クラスタ相当の構成（コールドスタンバイ構成）にしたものです。

すべての待機系サーバを 1 台のマシン上で稼働させることで，実行系のマシン N 台に対して待機系のマシン 1 台で構成を構築できます。そのため，HA クラスタ構成の HCSC サーバの組を並列に並べることができ，負荷分散できます。

N 対 1 クラスタ構成について，次の図に示します。

図 1-18　N 対 1 クラスタの構成



1.5　クラスタ構成における運用環境の構成パターン

クラスタ構成にする場合に実現できる運用環境と実行環境の構成パターンには，実行環境の数に関係なく運用環境を 1 台とする構成のほかに，運用環境と実行環境を 1 対 1 に配置する構成があります。

運用環境と実行環境を 1 対 1 に配置する構成のメリットとデメリットを次に示します。

メリット

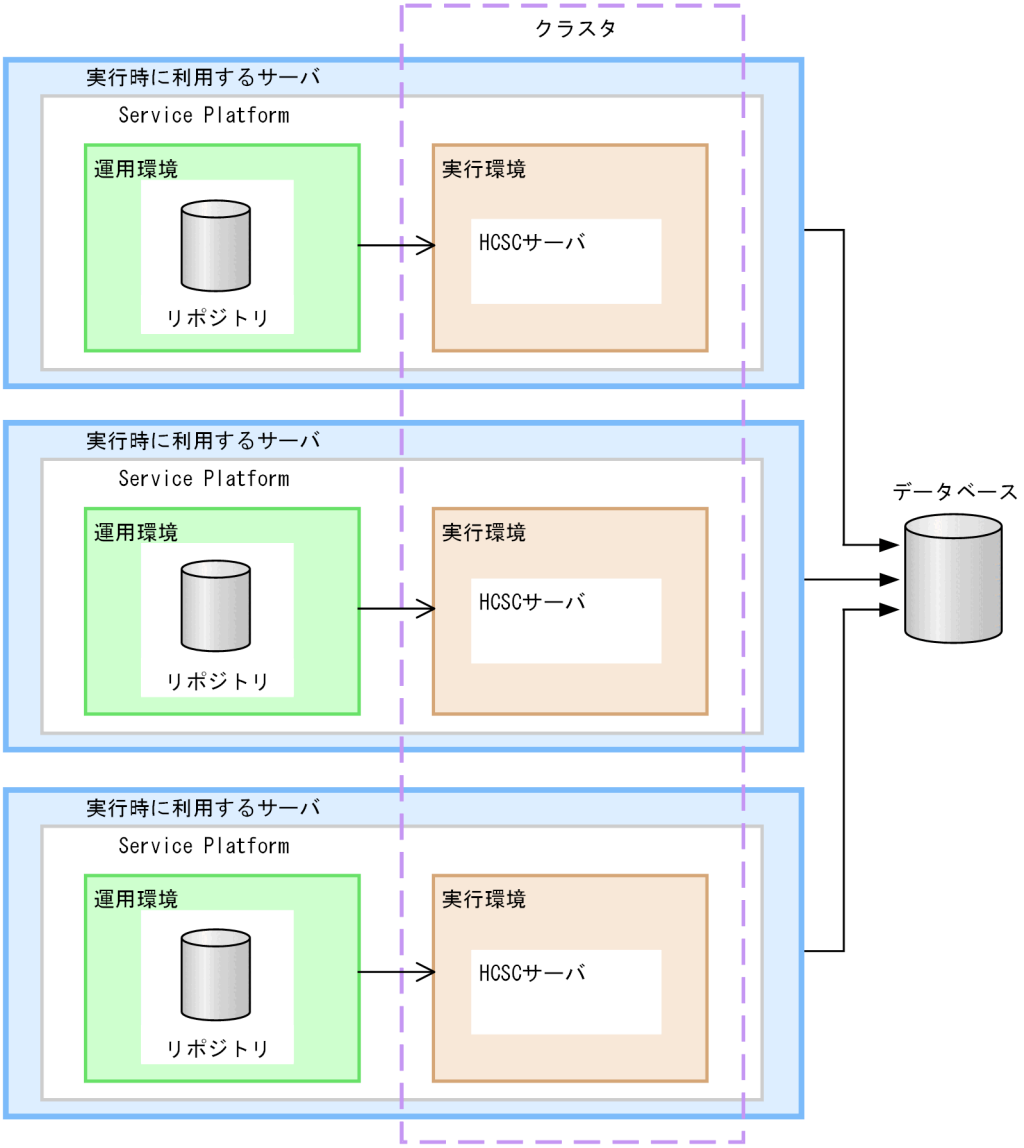
- 実行環境と運用環境を 1 台のマシンにまとめられます。
- 運用環境が 1 台ダウンしても，エラーになったリクエストをほかの運用環境から再実行できます。
- 運用環境用のマシンが不要になるため，運用監視ソフトのライセンス数を低減できます。
- 予備の運用環境が不要になります。

デメリット

- UNIX 上に実行環境を構築する場合，運用環境の GUI を利用できません。
- クラスタ単位の一括操作ができません。

運用・実行 1 対 1 構成の構成例を次の図に示します。

図 1-19　運用・実行 1 対 1 構成の構成例



（凡例）

：起動・停止命令の流れ　　：データの流れ

注意事項

運用・実行 1 対 1 構成以外の構成から運用・実行 1 対 1 構成に移行する場合の注意事項

運用環境と実行環境の構成については，マニュアル「サービスプラットフォーム システム構築・運用ガイド」の「1.2　システム全体と運用環境・実行環境の関係」を参照してください。

- 移行前の環境が 08-70 以前にセットアップした環境の場合，クラスタに含まれる HCSC サーバをいったんすべてアンセットアップする必要があります。このため，データベースに格納されているプロセスインスタンスや実行履歴など，クラスタで共有する情報は移行できません。

- データベースを利用していない運用・実行 1 対 1 構成からデータベースを利用した運用・実行 1 対 1 構成には移行できません。

1.6　分散開発時のリポジトリのインポート

複数の開発環境で分散開発する場合，開発したリポジトリを運用環境で使用するために，リポジトリを集約する必要があります。リポジトリを集約するには，次のどちらかを実施してください。

- 開発環境でサービスインポートを実行する。
- 運用環境でコンポーネントインポートを実行する。

開発環境でリポジトリを集約する場合は，インポート対象の取捨選択やサービス ID の衝突回避など，運用環境で集約するよりも詳細に条件を設定できます。

運用環境でリポジトリを集約する場合は，リポジトリを集約するための開発環境が不要になるため，マシンの台数を削減できます。

サービスインポートの詳細についてはマニュアル「サービスプラットフォーム 開発ガイド 基本開発編」の「3.2.3　リポジトリをインポートする」を，コンポーネントインポートの詳細についてはマニュアル「サービスプラットフォーム システム構築・運用ガイド」の「4.6　コンポーネント単位でのリポジトリの追加，更新，および削除」を参照してください。

1.6.1　コンポーネントインポート機能を利用した分散開発の流れ

コンポーネントインポート機能を利用すると，開発環境を用意しないで，運用コマンドだけでリポジトリを集約できます。このため，開発環境（マスタ）とテスト環境を 1 台のマシンに集約でき，マシンの台数を削減できます。

開発環境でリポジトリを集約する場合の流れを「図 1-20　開発環境でリポジトリを集約する場合の流れ」に，運用環境でリポジトリを集約する場合の流れを「図 1-21　運用環境でリポジトリを集約する場合の流れ」にそれぞれ示します。

図 1-20　開発環境でリポジトリを集約する場合の流れ

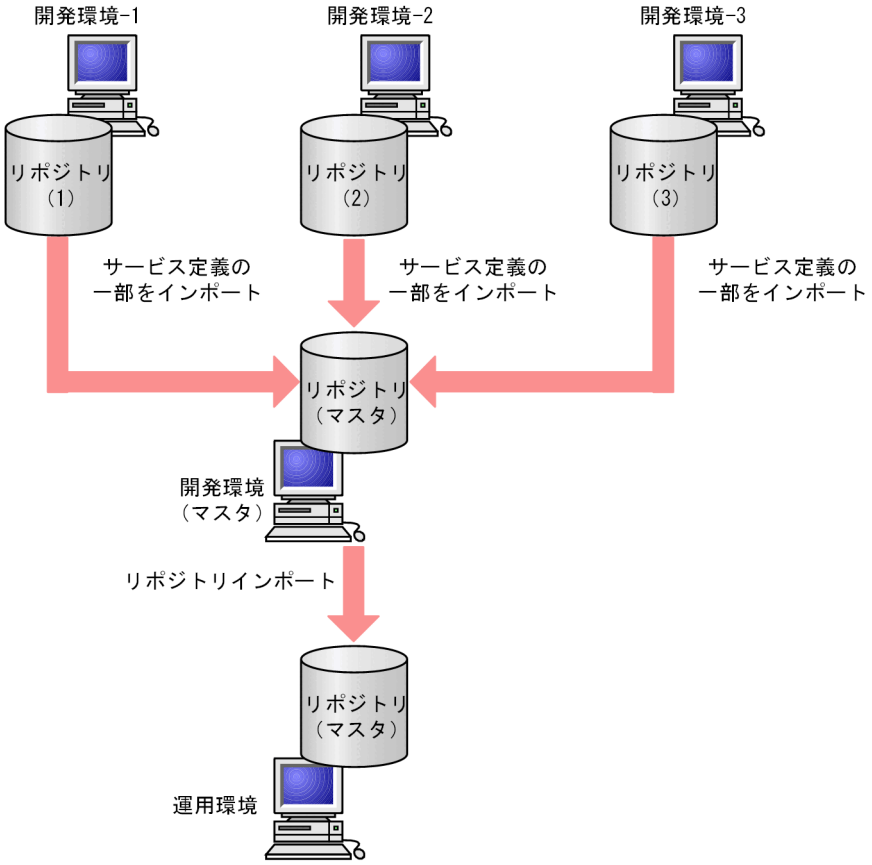
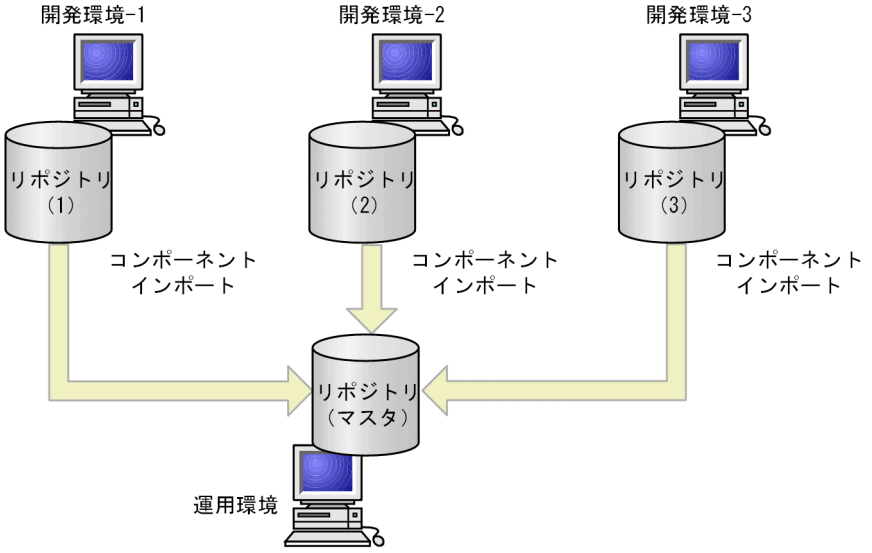
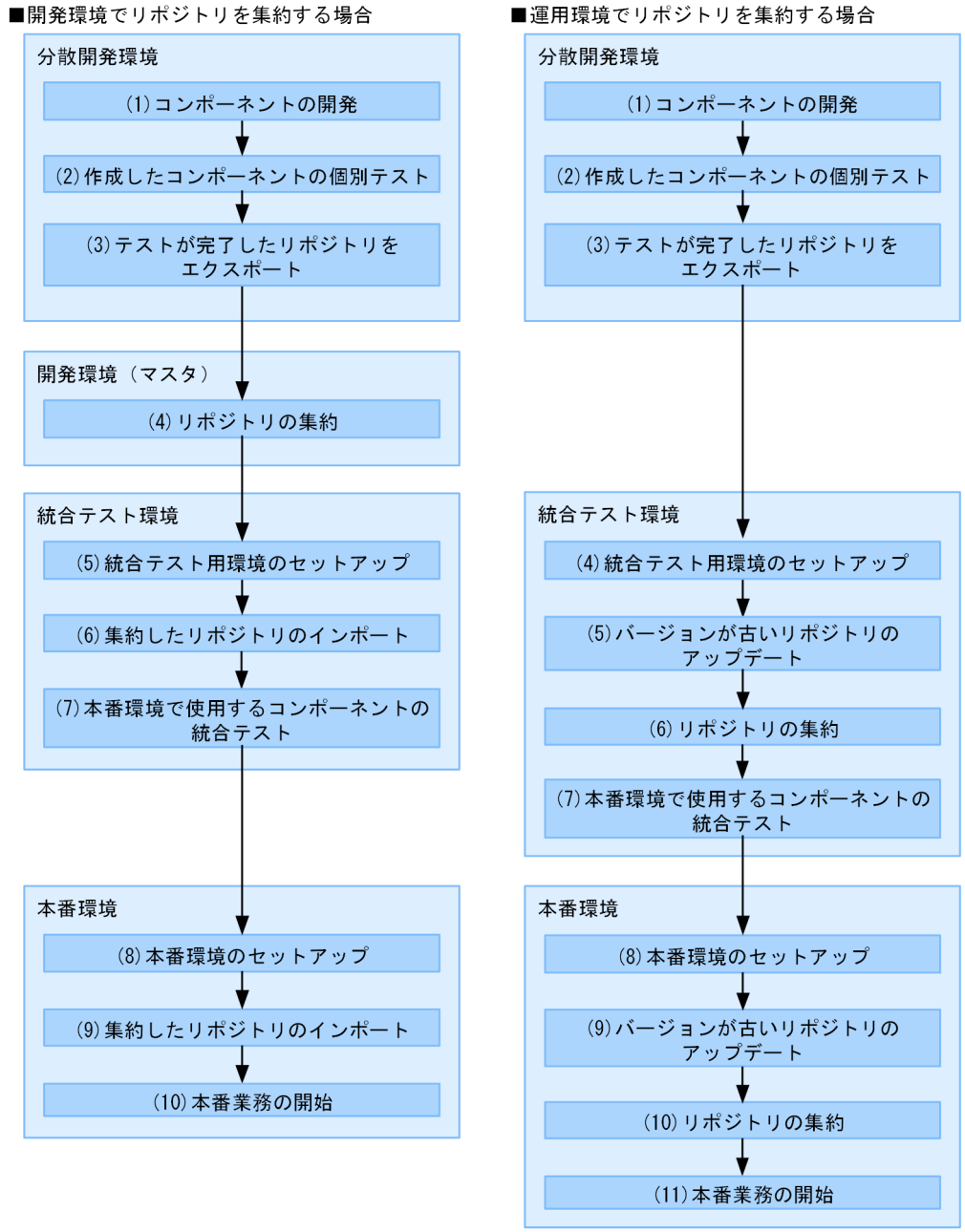


図 1-21　運用環境でリポジトリを集約する場合の流れ



また，開発環境でリポジトリを集約する場合と，運用環境でリポジトリを集約する場合の環境の移行手順を次の図に示します。

図 1-22　環境の移行手順



1.6.2　分散開発時の注意事項

分散開発時の注意事項を次に示します。

- サービス ID，サービス名，コンテキストルートおよび UOC ファイルは，リポジトリの集約時に重複しないよう，分散開発環境全体で一意となるように命名してください。
- UOC ファイルの共通パッケージは，同期を取ってください。

2 さまざまなシステムと接続するための機能

この章では，サービス部品を呼び出す基本的な流れや，サービスプラットフォームの実行環境の仕組みについて説明します。また，各種通信サービスや他システムなどに接続したときの，サービス部品の呼び出しについても説明します。なお，開発環境の定義は，運用環境を通じて実行環境で動作しますが，ここでは運用環境の説明は省略します。

2.1　サービス部品呼び出しの基本的な流れ

サービスプラットフォームを使ってシステム構築や運用をする場合，サービスリクエスタから HCSC サーバを経由してサービス部品を呼び出す流れを理解することで，より良いシステムが実現でき，SOA を適用することによる効果が得られます。

また，SOA を適用したシステム開発をする場合も，どのように運用されるのか理解した上で，サービス部品呼び出し処理やエラー時の処理を設計することが大切です。

サービス部品を呼び出す流れには，プロトコルで共通な部分と，プロトコルごとに異なる部分があります。ここでは，プロトコルで共通するサービス部品を呼び出す基本的な流れについて説明します。

2.1.1　基本的な構造

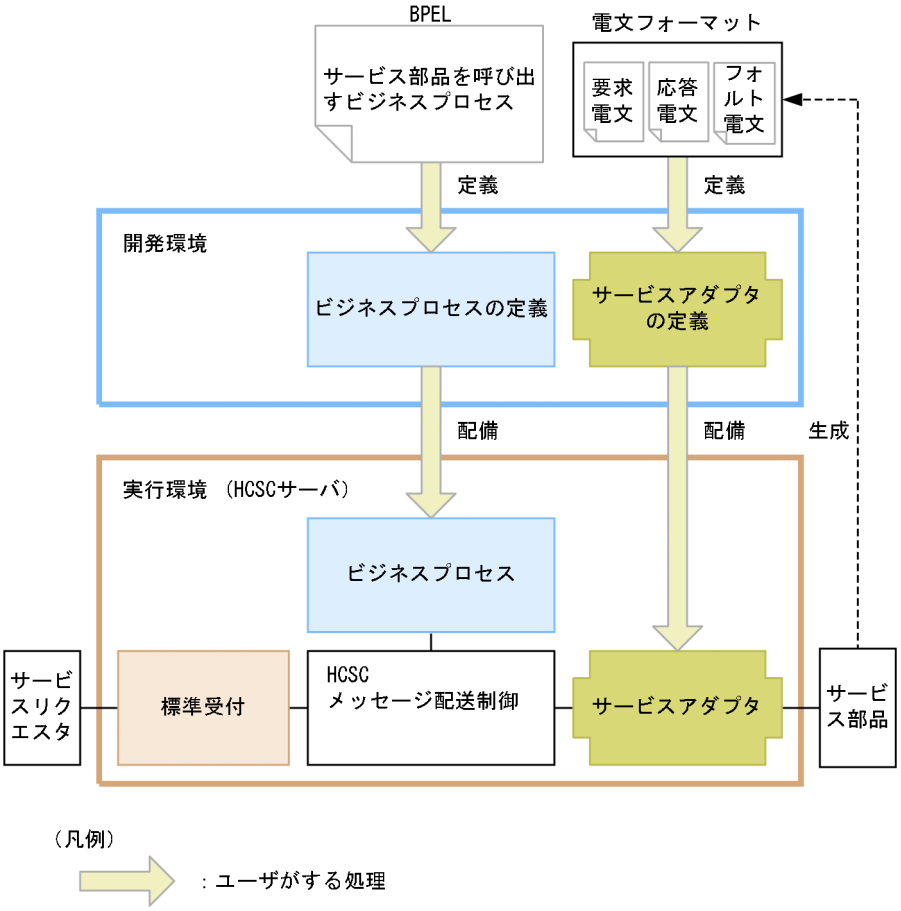
サービス部品を呼び出すためには，基本的にサービスリクエスタ，受付，HCSC メッセージ配送制御，サービスアダプタ，およびサービス部品が必要です。受付は，サービス部品を呼び出すための要求を受け付けるところで，標準受付とユーザ定義受付があります。

標準受付とは，HCSC サーバが標準提供している，サービスリクエスタからの要求電文を受け付けるための機能（インターフェース）です。ユーザ定義受付とは，開発環境でユーザが定義したものを HCSC サーバに配備してから，サービスリクエスタからの実行要求を受け付けるための機能（インターフェース）で，ユーザが定義した任意のインターフェースで受け付けることができます。

(1)　標準受付の場合

標準受付を使用する場合の構造と開発の流れとの関係を次の図に示します。

図 2-1　標準受付を使用する場合の構造と開発の流れとの関係



サービス部品を呼び出すサービスリクエスタと HCSC サーバ，およびサービス部品の関係は，次のようになっています。開発の流れに沿って，それぞれの関係を説明します。

- サービス部品側の開発の流れ

  最初に呼び出すサービス部品を作成します。次に，呼び出すサービス部品に合わせた形式の電文フォーマットやビジネスプロセス定義（BPEL）を作成し，開発環境で電文フォーマットからサービスアダプタを定義したり，ビジネスプロセスを定義したりします（ビジネスプロセスにも電文フォーマットを定義します）。呼び出すサービス部品に合わせた形式の電文フォーマットを作成する方法は，使用するプロトコル（SOAP，RMI-IIOP，WS-R，および JMS）によって異なります。定義したサービスアダプタやビジネスプロセスは実行環境（HCSC サーバ）へ配備します。電文フォーマットについては，マニュアル「サービスプラットフォーム 開発ガイド 基本開発編」の「4.　電文フォーマットの作成」を参照してください。ビジネスプロセス定義（BPEL）については，マニュアル「サービスプラットフォーム 開発ガイド 基本開発編」の「5.　ビジネスプロセスの定義」を参照してください。

- サービスリクエスタ側の開発の流れ

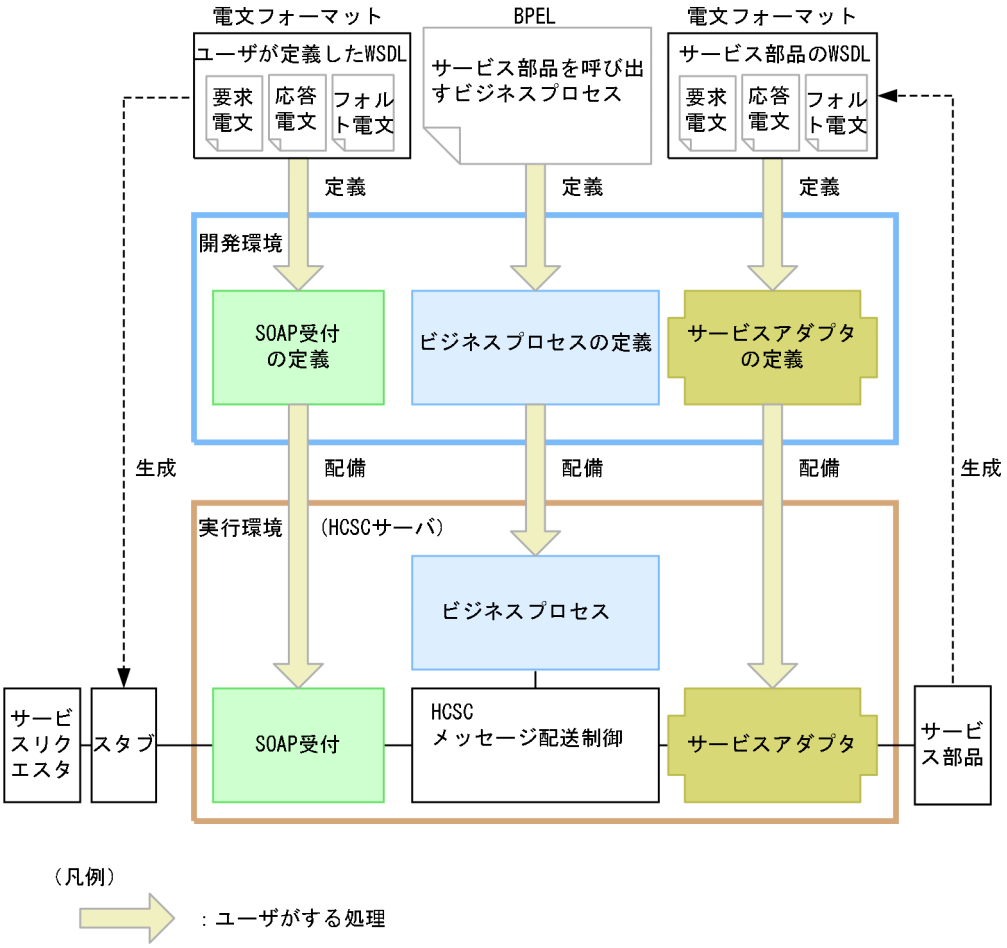
  サービス部品の作成後，定義したサービスアダプタやビジネスプロセスを呼び出すサービスリクエスタを作成します。サービスリクエスタを作成する方法については，使用する標準受付の種別によって異なります。サービスリクエスタの作成については，マニュアル「サービスプラットフォーム 開発ガイド 基本開発編」の「9.　サービスリクエスタの作成」を参照してください。

(2)　ユーザ定義受付の場合

ユーザ定義受付を使用する場合の構造と開発の流れとの関係を次の図に示します。

なお，ここでは Web サービス（SOAP 通信）を使用した場合の例を使用して説明します。

図 2-2　ユーザ定義受付を使用する場合の構造と開発の流れとの関係



Web サービス（SOAP 通信）を使用する場合，サービス部品を呼び出すサービスリクエスタと HCSC サーバ，およびサービス部品の関係は，次のようになっています。開発の流れに沿って，それぞれの関係を説明します。

- サービス部品側の開発の流れ

  最初に呼び出すサービス部品を作成します。次に，呼び出すサービス部品に合わせた形式の電文フォーマットやビジネスプロセス定義（BPEL）を作成し，開発環境で電文フォーマットからサービスアダプタを定義したり，ビジネスプロセスを定義したりします（ビジネスプロセスにも電文フォーマットを定義します）。定義したサービスアダプタ，ビジネスプロセス，およびユーザ定義受付は実行環境（HCSC サーバ）へ配備します。電文フォーマットについては，マニュアル「サービスプラットフォーム 開発ガイド 基本開発編」の「4.　電文フォーマットの作成」を参照してください。ビジネスプロセス定義（BPEL）については，マニュアル「サービスプラットフォーム 開発ガイド 基本開発編」の「5.　ビジネスプロセスの定義」を参照してください。

- 受付側の開発の流れ

  サービス部品の作成後，呼び出すビジネスプロセスの電文フォーマットに合わせた WSDL を作成し，開発環境でユーザ定義受付を定義します。ユーザ定義受付の定義については，マニュアル「サービスプラットフォーム 開発ガイド 受付・アダプタ定義編」の「2. ユーザ定義受付の定義」を参照してください。

- サービスリクエスタ側の開発の流れ

  ユーザ定義受付の作成後，定義したユーザ定義受付を呼び出すサービスリクエスタを作成します。ユーザ定義受付を定義するときに使用した WSDL からスタブを生成し，スタブを呼び出すサービスリクエスタを作成します。サービスリクエスタの作成については，マニュアル「サービスプラットフォーム 開発ガイド 基本開発編」の「9. サービスリクエスタの作成」を参照してください。

2.1.2 サービス部品を呼び出す流れ

サービスリクエスタからサービス部品を呼び出す流れは，受付の種類と呼び出す HCSC コンポーネントの組み合わせによって動作が異なります。受付の種類と呼び出す HCSC コンポーネントの組み合わせを次に示します。

- 標準受付からサービスアダプタを直接呼び出してサービス部品を実行する場合
- 標準受付からビジネスプロセスを呼び出してサービス部品を実行する場合
- ユーザ定義受付からビジネスプロセスを呼び出してサービス部品を実行する場合

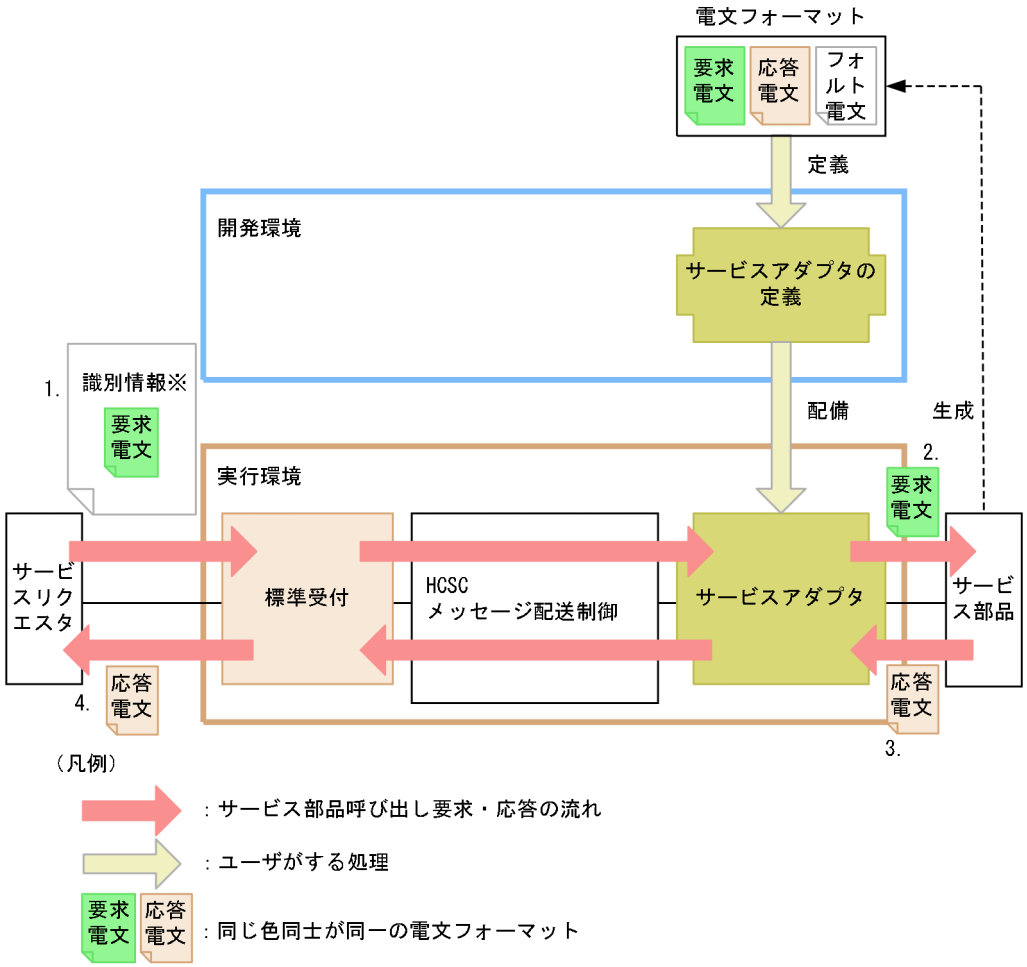
それぞれの場合の電文の流れを，次に説明します。エラー時の処理の流れについては，マニュアル「サービスプラットフォーム システム構築・運用ガイド」の「7. 障害対策」を参照してください。

(1) 標準受付からサービスアダプタを直接呼び出してサービス部品を実行する場合

標準受付からサービスアダプタを直接呼び出してサービス部品を実行する場合，サービスアダプタで定義したサービス名をサービスリクエスタで指定して実行します。実行すると，サービスアダプタで定義したサービス部品を呼び出します。

サービスアダプタを直接呼び出す場合の電文の流れを次の図に示します。

図 2-3　サービスアダプタを直接呼び出す場合の電文の流れ



注※　識別情報とは，サービス名，クライアント相関IDなどです。

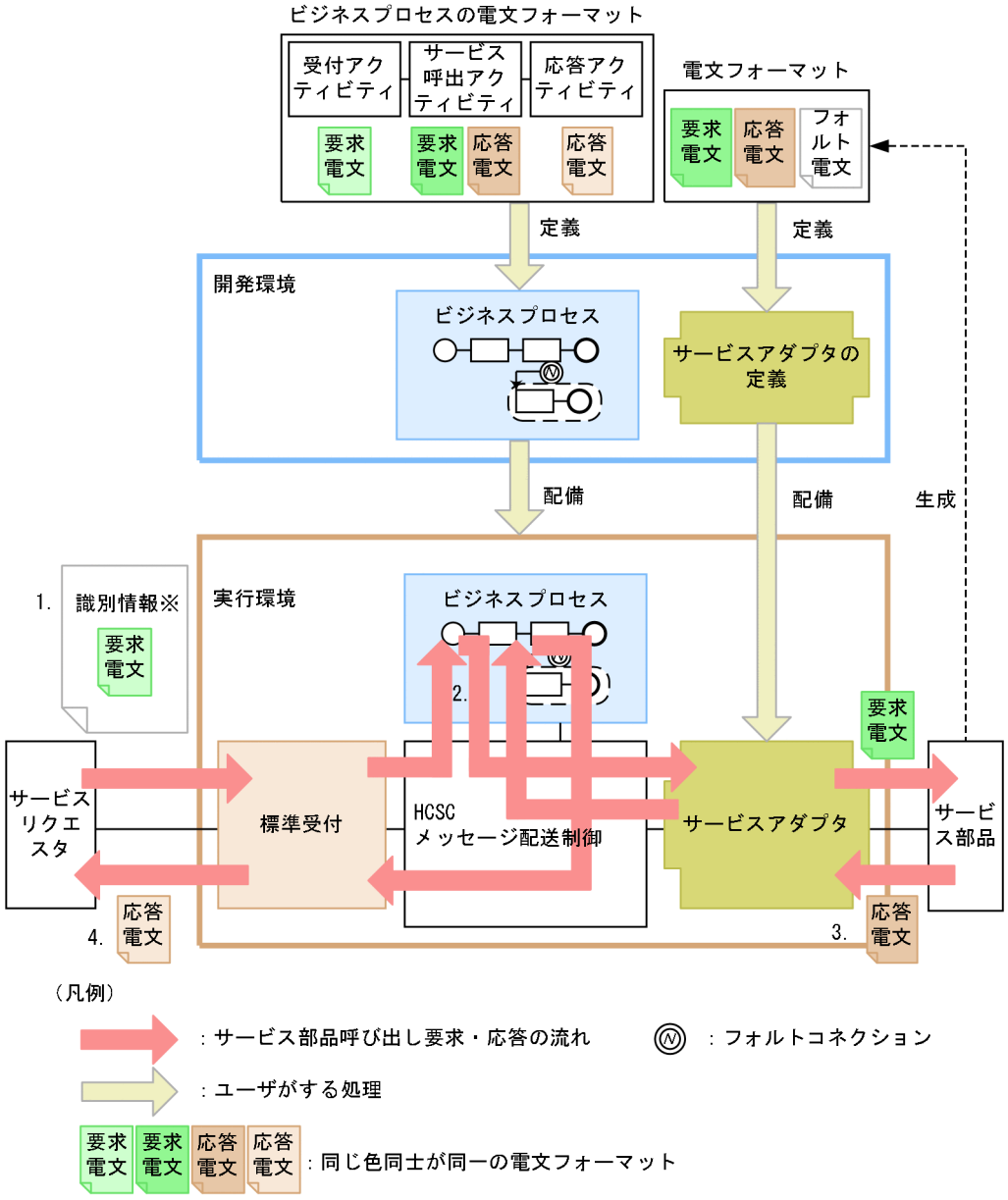
1. サービスリクエスタから標準受付に送信する要求電文は，開発環境でサービスアダプタを定義したときに設定した（サービス部品の形式に合わせた）電文フォーマットを使用します。そのため，サービスリクエスタ内で，サービスアダプタで定義した電文フォーマットに合わせた要求電文を作成し，作成した電文を標準受付のパラメタ（ユーザ電文のパラメタ）に設定して実行します。
2. サービスアダプタからサービス部品を呼び出します。呼び出すときは，サービスアダプタで定義した要求電文を使用します。このように，サービスリクエスタから送信する要求電文とサービス部品が受け取る要求電文は同一になります。なお，サービス部品の電文フォーマット以外の電文フォーマットでも，サービスアダプタでデータ変換定義をすれば，サービスリクエスタから要求できます。
3. サービス部品からサービスアダプタに応答電文を送信します。応答電文は，サービスアダプタで定義した応答電文と同じ電文フォーマットを使用します。
4. サービスリクエスタへ応答します。応答には，サービスアダプタで定義した応答電文の電文フォーマットを使用します。このように，サービス部品から返す応答電文とサービスリクエスタで受け取る応答電文は同一になります。

データ変換定義については，マニュアル「サービスプラットフォーム 開発ガイド 基本開発編」の「7.3 データ変換定義」を参照してください。

(2)　標準受付からビジネスプロセスを呼び出してサービス部品を実行する場合

標準受付からビジネスプロセスを呼び出してサービス部品を実行する場合，ビジネスプロセスで定義したサービス名（ビジネスプロセス名）をサービスリクエスタで指定して実行します。そして，ビジネスプロセスに定義したプロセスに従って，処理を実行します。プロセスはアクティビティから構成されています。その中の，サービス呼出アクティビティでは，定義されたサービスアダプタを呼び出して，サービスアダプタに定義されたサービス部品を実行します。ビジネスプロセスを呼び出す場合の電文の流れを次の図に示します。

図 2-4　ビジネスプロセスを呼び出す場合の電文の流れ



注※　識別情報とは，サービス名，クライアント相関IDなどです。

1. サービスリクエスタ内で，ビジネスプロセスの受付アクティビティで定義した要求電文の電文フォーマットに合わせた要求電文を作成し，作成した電文を標準受付のパラメタ（ユーザ電文のパラメタ）に

設定して実行します。サービスリクエスタで指定する要求電文は，ビジネスプロセスの受付アクティビティに定義した電文フォーマットを使用します。

2. ビジネスプロセスからサービス部品を呼び出します。呼び出すときは，サービス呼出アクティビティで定義した要求電文を使用します。この要求電文の電文フォーマットは，呼び出すサービス部品のサービスアダプタで定義した要求電文の形式と同じである必要があります。なお，サービス部品の電文フォーマット以外の電文フォーマットでも，サービスアダプタでデータ変換定義をすれば，サービス呼出アクティビティから要求できます。
3. サービス部品からサービスアダプタに応答電文を送信します。応答電文は，ビジネスプロセスのサービス呼出アクティビティの応答電文と同じ電文フォーマットを定義します。
4. サービスリクエスタへ応答します。応答には，ビジネスプロセスの応答アクティビティで定義した応答電文の電文フォーマットを使用します。

データ変換定義については，マニュアル「サービスプラットフォーム 開発ガイド 基本開発編」の「7.3 データ変換定義」を参照してください。

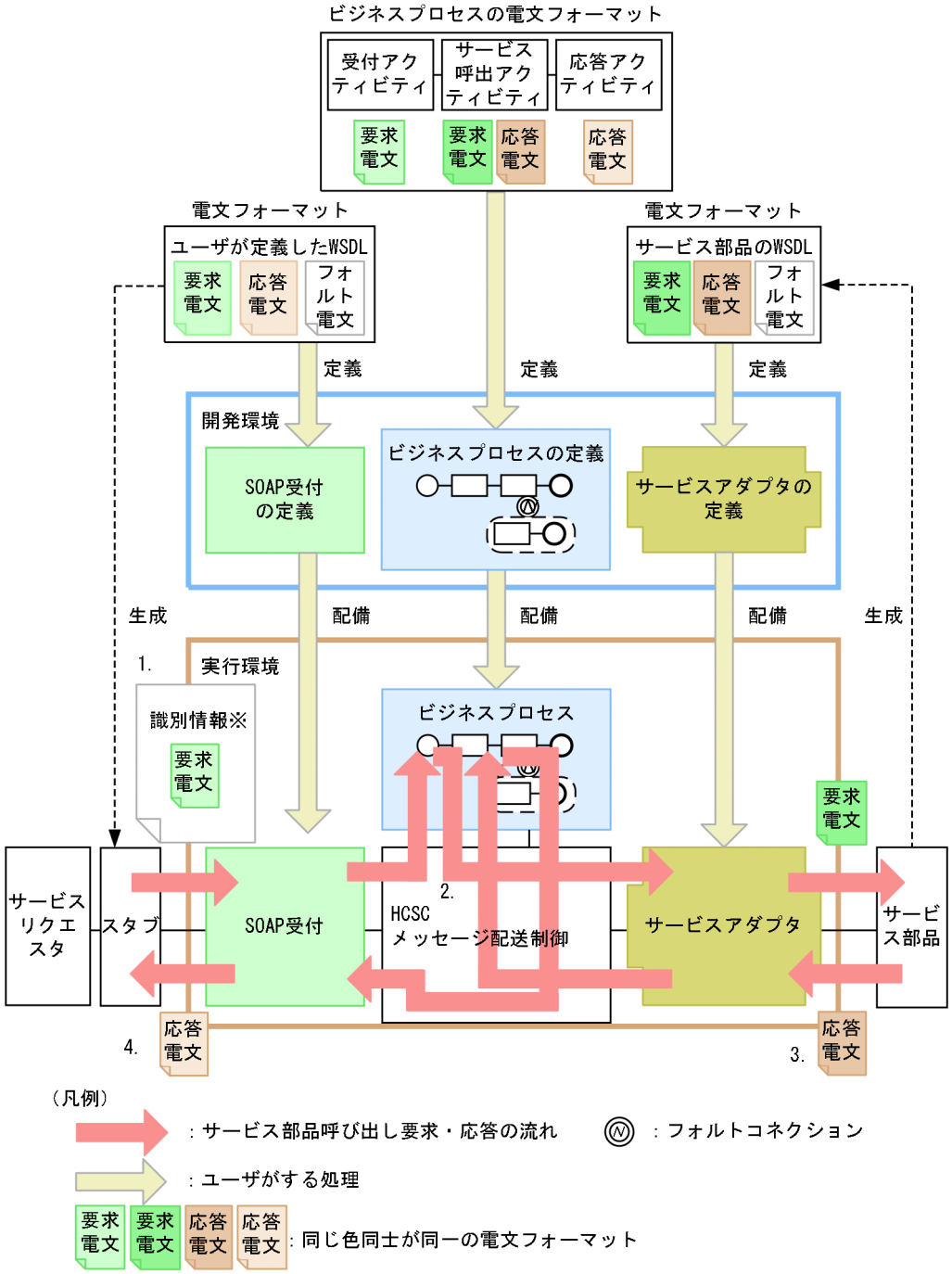
(3) ユーザ定義受付からビジネスプロセスを呼び出してサービス部品を実行する場合

ユーザ定義受付は，ユーザが定義した任意の形式でサービス部品実行要求を受け付け，応答を返すインターフェースです。

開発環境での定義に従ってユーザ定義受付からビジネスプロセスを呼び出し，ビジネスプロセスに定義したプロセスに従って，処理を実行します。サービス呼出アクティビティでは，定義されたサービスアダプタを呼び出し，サービスアダプタに定義されたサービス部品を実行します。ユーザ定義受付を使う場合の電文の流れを次の図に示します。

なお，ここでは SOAP 通信基盤を使用した場合の例を使用して説明します。

図 2-5　ユーザ定義受付を使う場合の電文の流れ（SOAP 通信基盤の例）



注※　識別情報とは，サービス名，クライアント相関IDなどです。

1. サービスリクエスタからの要求電文は，ビジネスプロセスの受付アクティビティに定義した電文フォーマットと同一のものになります。そのため，ビジネスプロセスの受付アクティビティに定義した電文フォーマットに合わせた形式の WSDL を開発環境のユーザ定義受付定義で設定する必要があります。サービスリクエスタから SOAP 受付を呼び出すときは，開発環境で SOAP 受付に定義した WSDL を使用します。SOAP1.1（SOAP 通信基盤または JAX-WS エンジン）に対応したサービスリクエスタを使用する場合は，WSDL から WSDL2Java コマンドを使用してスタブを生成し，スタブを呼び出すようにサービスリクエスタを実装します。

SOAP1.2（JAX-WS エンジン）に対応したサービスリクエスタを使用する場合は，WSDL から cjwsimport コマンドを使用してサービスクラスを生成し，サービスクラスを呼び出すようにサービスリクエスタを実装します。

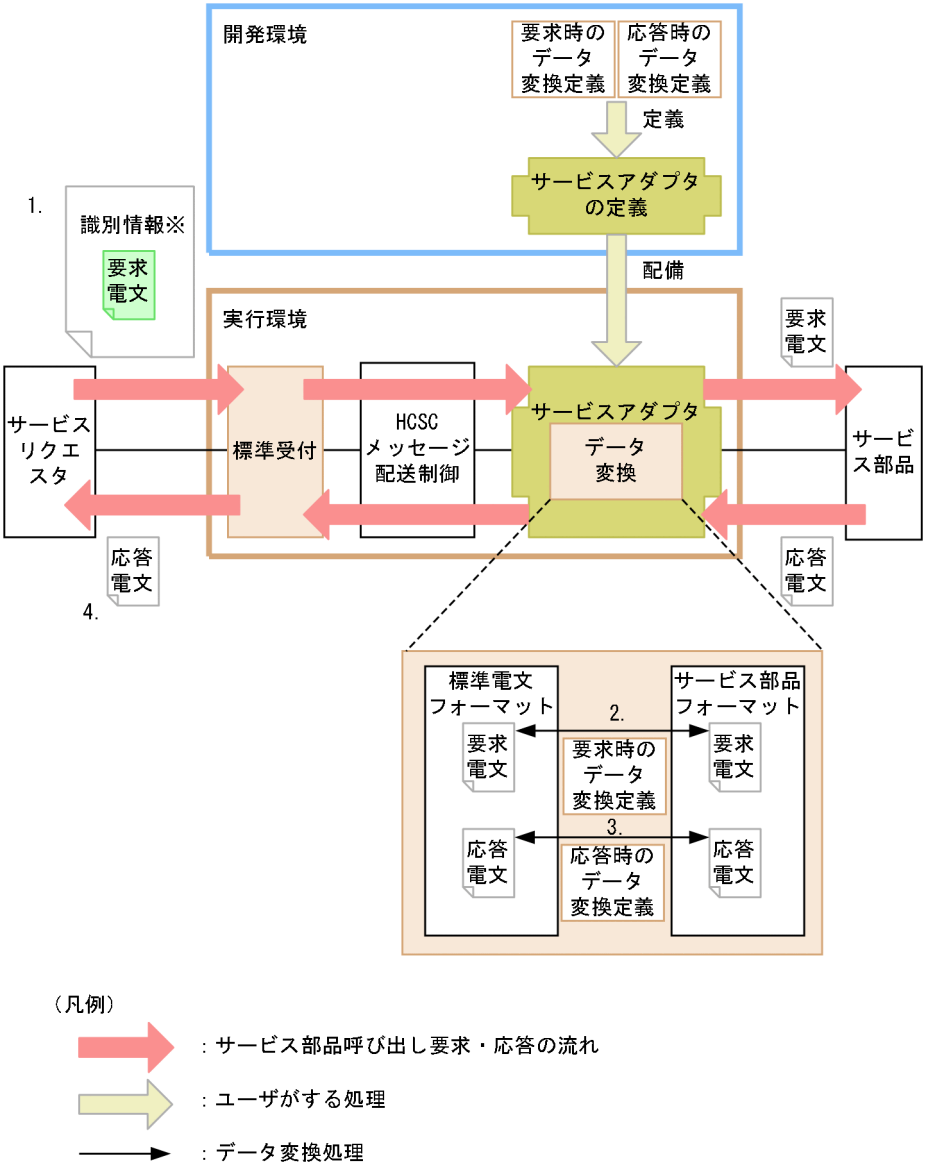
2. ビジネスプロセスからサービス部品を呼び出すときは，サービス呼出アクティビティで定義した要求電文を使用します。この要求電文の電文フォーマットは，呼び出すサービス部品のサービスアダプタで定義した要求電文の形式と同じである必要があります。
3. サービス部品からの応答電文もビジネスプロセスのサービス呼出アクティビティの応答電文で同じ電文フォーマットを定義します。なお，サービス部品の電文フォーマット以外の電文フォーマットでも，サービスアダプタでデータ変換定義をすれば，サービス呼出アクティビティから要求できます。
4. サービスリクエスタへの応答は，ビジネスプロセスの応答アクティビティで定義した応答電文の電文フォーマットを使用します。この電文フォーマットは開発環境のユーザ定義受付定義で設定する WSDL の応答電文と同じ形式である必要があります。

データ変換定義については，マニュアル「サービスプラットフォーム 開発ガイド 基本開発編」の「7.3 データ変換定義」を参照してください。

2.1.3 サービス部品側と異なる電文フォーマットでサービス部品を呼び出す場合の流れ

サービス部品側の電文フォーマットと異なる電文でサービス部品を呼び出す場合，またはサービス部品側の電文フォーマットと異なる電文を応答として受信する場合は，開発環境で標準電文フォーマットを定義します。また，標準電文フォーマットとサービス部品電文フォーマットの変換ルールを定義したデータ変換定義は，開発環境で定義します。標準電文フォーマットの電文をサービス部品電文フォーマットの電文に変換することでサービス部品を呼び出せます。標準電文フォーマットとデータ変換定義の関係を次の図に示します。

図 2-6　標準電文フォーマットとデータ変換定義の関係

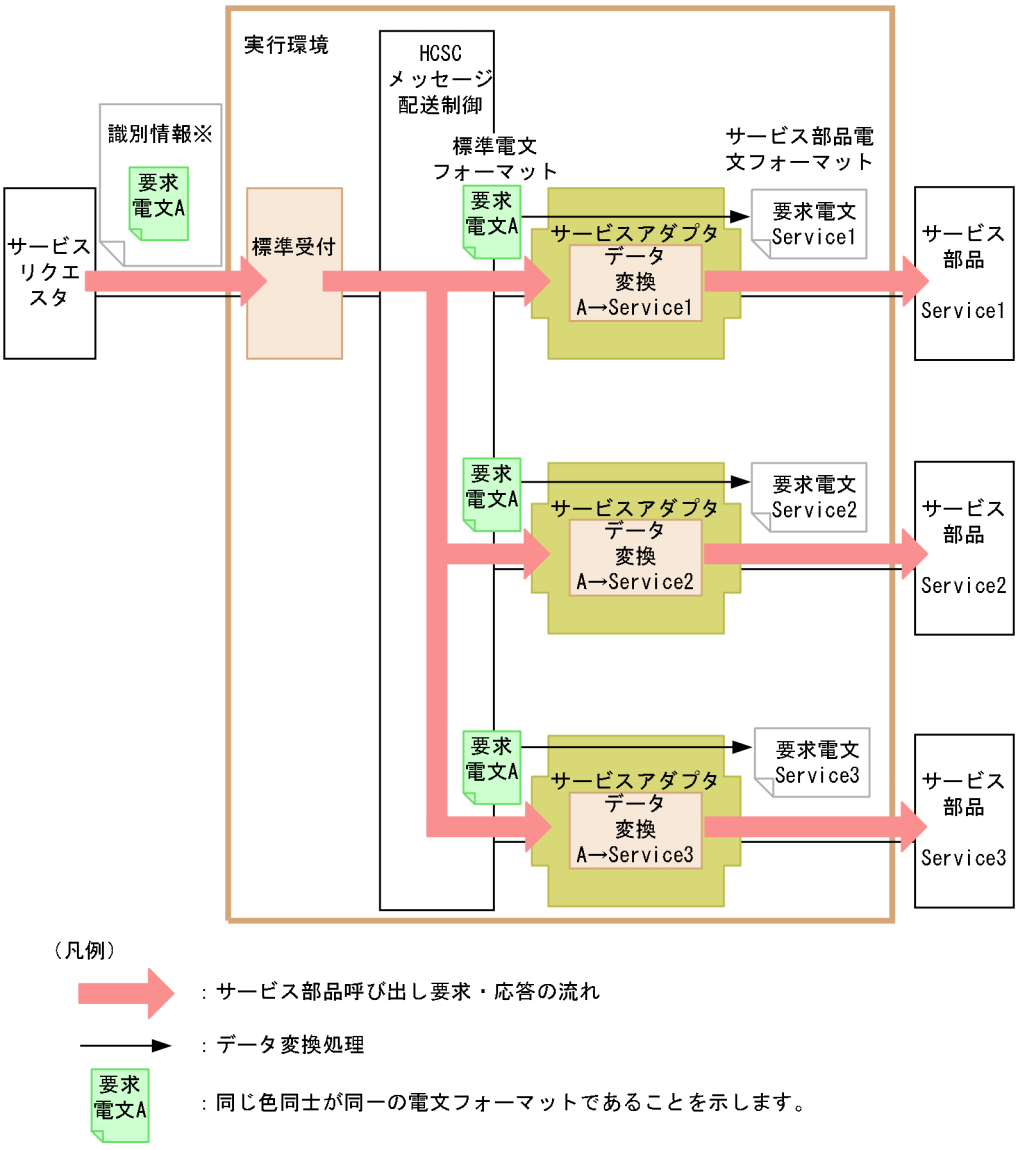


注※　識別情報とは，サービス名，クライアント相関IDなどです。

1. サービスリクエスタからは，標準電文フォーマットの形式に合わせた電文フォーマットを要求電文に指定して，サービス部品の呼び出しを要求します。
2. サービスアダプタでは標準電文フォーマットからサービス部品電文フォーマットにデータを変換し，変換後の電文フォーマットでサービス部品を呼び出します。
3. サービス部品からの応答についても，サービスアダプタで，サービス部品電文フォーマットから標準電文フォーマットにデータを変換します。
4. サービスリクエスタに標準電文フォーマットを応答します。

複数のサービスアダプタがある場合も，次の図に示すように Service1 用のサービスアダプタ，Service2 用のサービスアダプタ，Service3 用のサービスアダプタで，それぞれに同一の標準電文フォーマット A と，それぞれのデータ変換定義を定義すると，サービスリクエスタ側から要求するときに，要求時のサービス名（Service1 など）を変更すれば，同じ要求電文で要求できます。

図 2-7　複数のサービスアダプタで同一の標準電文フォーマットを定義した場合の流れ



注※　識別情報とは，サービス名，クライアント相関IDなどです。

標準電文フォーマットおよびサービス部品電文フォーマットについては，マニュアル「サービスプラットフォーム 開発ガイド 基本開発編」の「4.　電文フォーマットの作成」を参照してください。データ変換定義については，マニュアル「サービスプラットフォーム 開発ガイド 基本開発編」の「7.3　データ変換定義」を参照してください。

2.1.4　サービス部品呼び出し処理で与えられる識別情報

HCSC サーバ内のサービス部品呼び出し要求とその流れを区別するために，各要求電文にメッセージ共通 ID とサービスリクエスト ID が割り当てられます。これらの ID はメッセージログ，リクエストトレース，および性能解析トレースで出力されます。

メッセージ共通 ID とサービスリクエスト ID をたどることで，サービス部品呼び出しの処理の流れを追跡できます。電文の追跡については，マニュアル「サービスプラットフォーム システム構築・運用ガイド」の「7.7　サービス部品呼び出し要求時の障害対策」を参照してください。

なお，HCSC サーバに割り当てられる識別情報のほかに，ユーザが任意の値を設定できるクライアント相関 ID があります。

(1)　識別情報の種類

- メッセージ共通 ID

  サービスリクエスタからサービス部品呼び出し要求を受け取ったタイミングで自動的に割り当てられます。応答まで同じメッセージ共通 ID を使用します。

  サービスリクエスタからビジネスプロセスを呼び出す場合でも，ビジネスプロセス内の個々のサービス部品呼び出し処理を同一の処理として識別できます。メッセージ共通 ID の形式を次に示します。

  ```
  CSC_[サーバ名]_[受付時刻(ミリ秒)]_[通番]
  ```

  注　末尾に与えられる通番は，「1」から順番に割り当てられる番号です。

  （例）CSC_CSCServerName_2008-04-20_11:32:18.360_1

- サービスリクエスト ID

  HCSC サーバ内の個別のサービス部品呼び出し処理ごとに割り当てられる識別情報です。

  サービスリクエスタからのサービス部品呼び出し要求を受け取ったタイミングだけでなく，ビジネスプロセスからのサービス部品呼び出し処理を実施するタイミングでも割り当てられます。サービスリクエスト ID の形式を次に示します。

  ```
  MSG_[サーバ名]_[受付種別]_[受付時刻(ミリ秒)]_[通番]
  ```

  注　末尾に与えられる通番は，「1」から順番に割り当てられる番号です。

  （例）MSG_CSCServerName_MDBWSR_2008-04-20_11:32:18.360_1

- ビジネスプロセスインスタンス ID

  ビジネスプロセスの個々のプロセスインスタンスを特定するための識別情報（プロセスインスタンスの識別子）です。サービスリクエスタからビジネスプロセスの再実行を要求する場合や，プロセスインスタンスの実行履歴を削除するときに使用します。

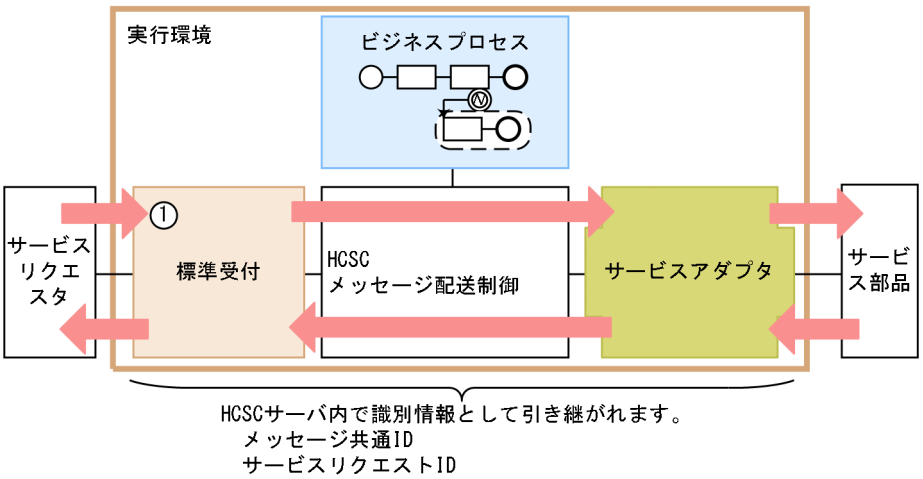
(2)　識別情報が割り当てられるタイミング

標準受付から直接サービスアダプタを呼び出す場合は，メッセージ共通 ID およびサービスリクエスト ID は，HCSC サーバ内で識別情報として引き継がれます。標準受付やユーザ定義受付からビジネスプロセスを経由してサービスアダプタを呼び出す場合は，メッセージ共通 ID は，HCSC サーバ内で識別情報として引き継がれます。しかし，サービスリクエスト ID は，サービス部品呼び出し処理ごとの識別情報として引き継がれます。また，ビジネスプロセスインスタンス ID は，ビジネスプロセスが最初に要求を受け付けるときに割り当てます。

メッセージ共通 ID とサービスリクエスト ID が割り当てられるタイミングを次の図に示します。

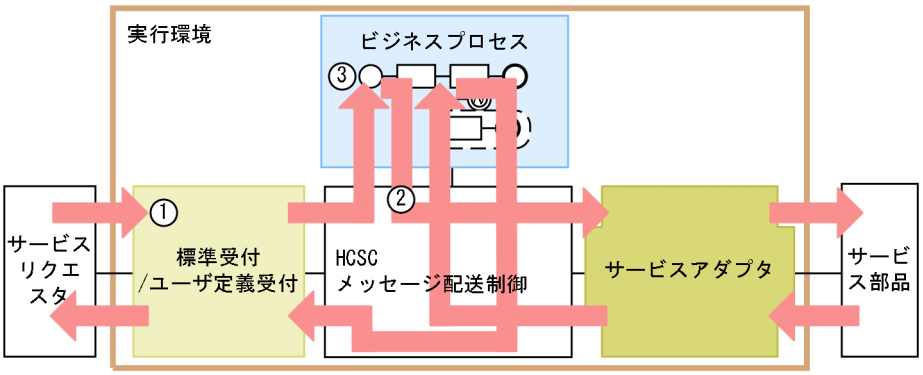
図 2-8　識別情報割り当てのタイミング

●標準受付から直接サービスアダプタを呼び出す場合



HCSCサーバ内で識別情報として引き継がれます。
　メッセージ共通ID
　サービスリクエストID

●ビジネスプロセスを呼び出す場合



サービスリクエストID：
　サービス呼び出し処理ごとの識別情報として引き継がれます。

メッセージ共通ID：
　HCSCサーバ内で識別情報として引き継がれます。

（凡例）

：サービス部品呼び出し要求・応答の流れ

① ：メッセージ共通ID，およびサービスリクエストIDの割り当て

② ：サービスリクエストIDの割り当て

③ ：ビジネスプロセスインスタンスIDの割り当て

Ⓝ ：フォルトコネクション

2.2 Web サービス（SOAP 通信）を使用したサービスとの接続

2.2.1 Web サービスの通信基盤との関係

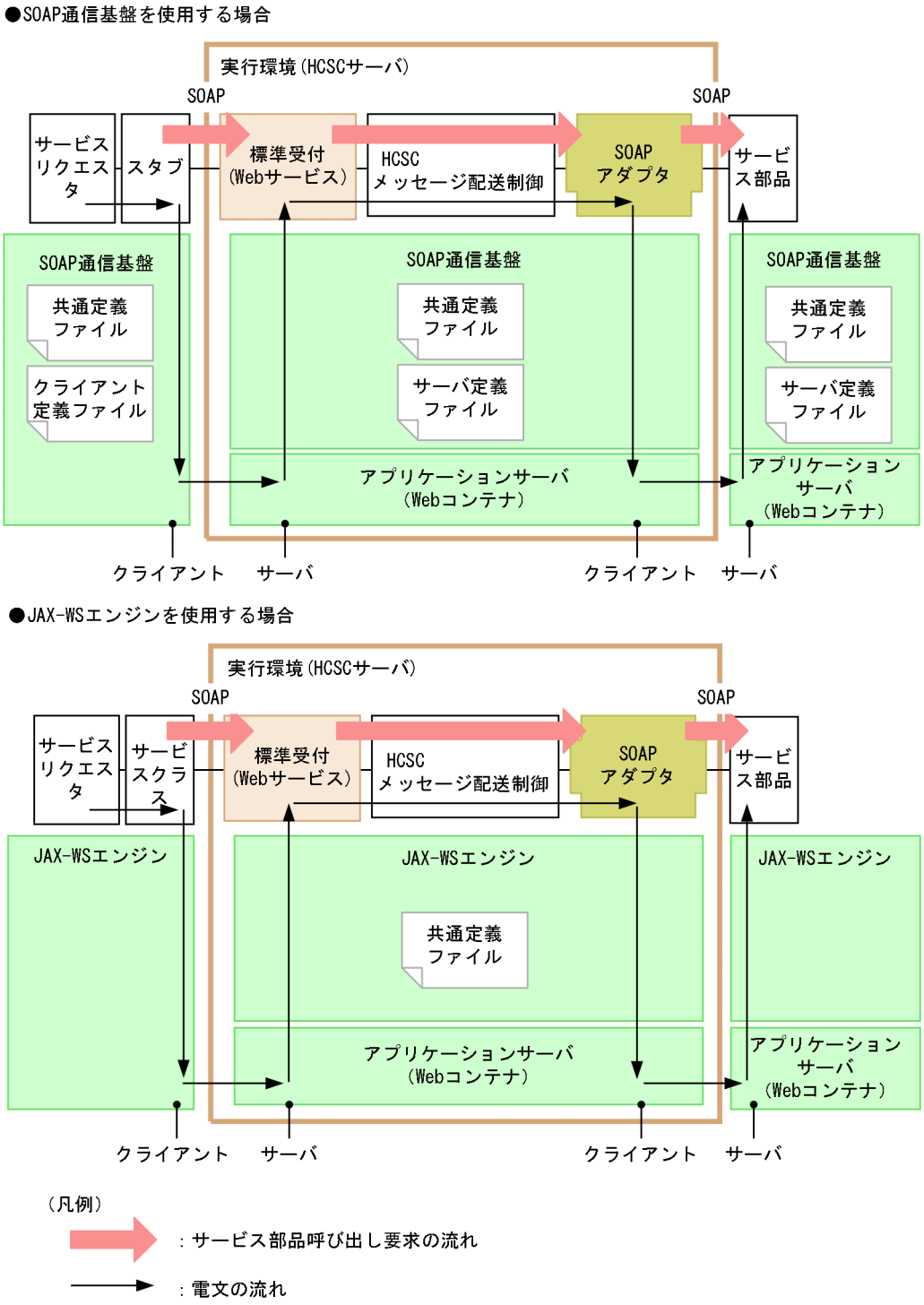
実行環境の HCSC サーバは，Web サービスの通信基盤および Web サーバ（HTTP サーバ）上で動作します。そのため，Web サービス（SOAP 通信）を使用する場合は，Web サービスを実行するための通信基盤や Web サーバの設定が必要です。

サービスリクエスタと HCSC サーバの標準受付（Web サービス）との関係は，クライアントとサーバの関係であり，HCSC サーバのサービスアダプタとサービス部品の関係もクライアントとサーバの関係になります。

サービスリクエスタ側およびサービス部品側の通信基盤は，SOAP 通信基盤を使用する場合と JAX-WS エンジンを使用する場合があります。SOAP1.1 に対応したサービスリクエスタおよびサービス部品を使用する場合は，SOAP 通信基盤または JAX-WS エンジンを使用します。また，SOAP1.2 に対応したサービスリクエスタおよびサービス部品を使用する場合は，JAX-WS エンジンを使用します。

サービスリクエスタおよびサービス部品と，SOAP 通信基盤の関係を次の図に示します。なお，サービスリクエスタ側とサービス部品側で，片方が SOAP 通信基盤，もう片方が JAX-WS エンジンを使用する構成も可能です。

図 2-9　SOAP 通信基盤との関係



2.2.2　サービス部品呼び出し時のユーザ電文と WSDL の関係

サービス部品呼び出し時のユーザ電文と WSDL の関係は，標準受付の場合とユーザ定義受付の場合とで異なります。

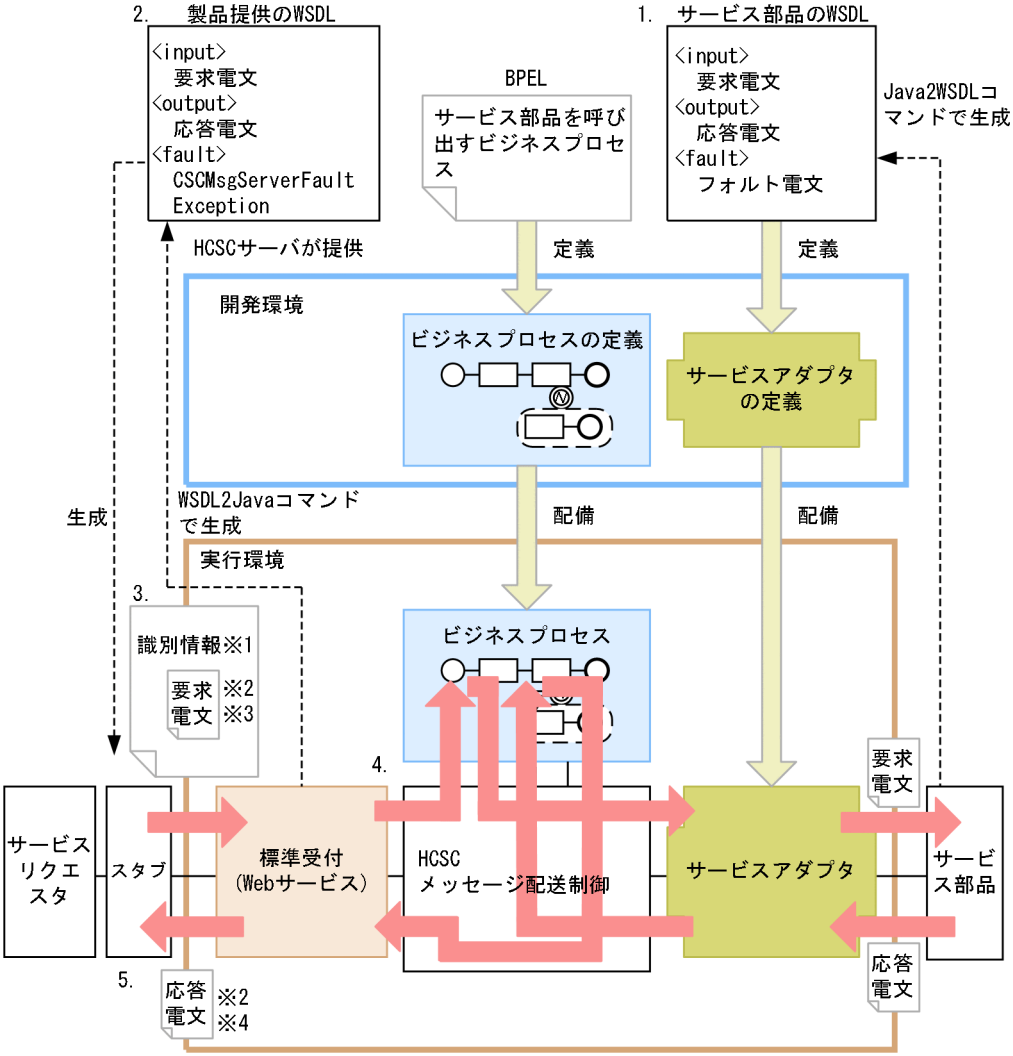
(1)　標準受付の場合

SOAP 通信時のユーザ電文と WSDL の関係（標準受付（Web サービス））を次に説明します。

(a)　SOAP 通信基盤を使用するとき

通信基盤に SOAP 通信基盤を使用するときの，SOAP 通信時のユーザ電文と WSDL の関係（標準受付（Web サービス））を次の図に示します。なお，サービスリクエスタ側とサービス部品側で，片方が SOAP 通信基盤，もう片方が JAX-WS エンジンを使用する構成も可能です。JAX-WS エンジンを使用する側の詳細は，「(b)　JAX-WS エンジンを使用するとき」を参照してください。

図 2-10　SOAP 通信基盤を使用するときの SOAP 通信時のユーザ電文と WSDL の関係（標準受付（Web サービス））



注※1　識別情報とは，サービス名，クライアント相関IDなどです。
注※2　サービス部品のWSDLに定義されている要求電文または応答電文です。
注※3　ビジネスプロセスを呼び出す場合は，ビジネスプロセスの受付アクティビティに定義した要求電文です。
注※4　ビジネスプロセスを呼び出す場合は，ビジネスプロセスの応答アクティビティに定義した応答電文です。

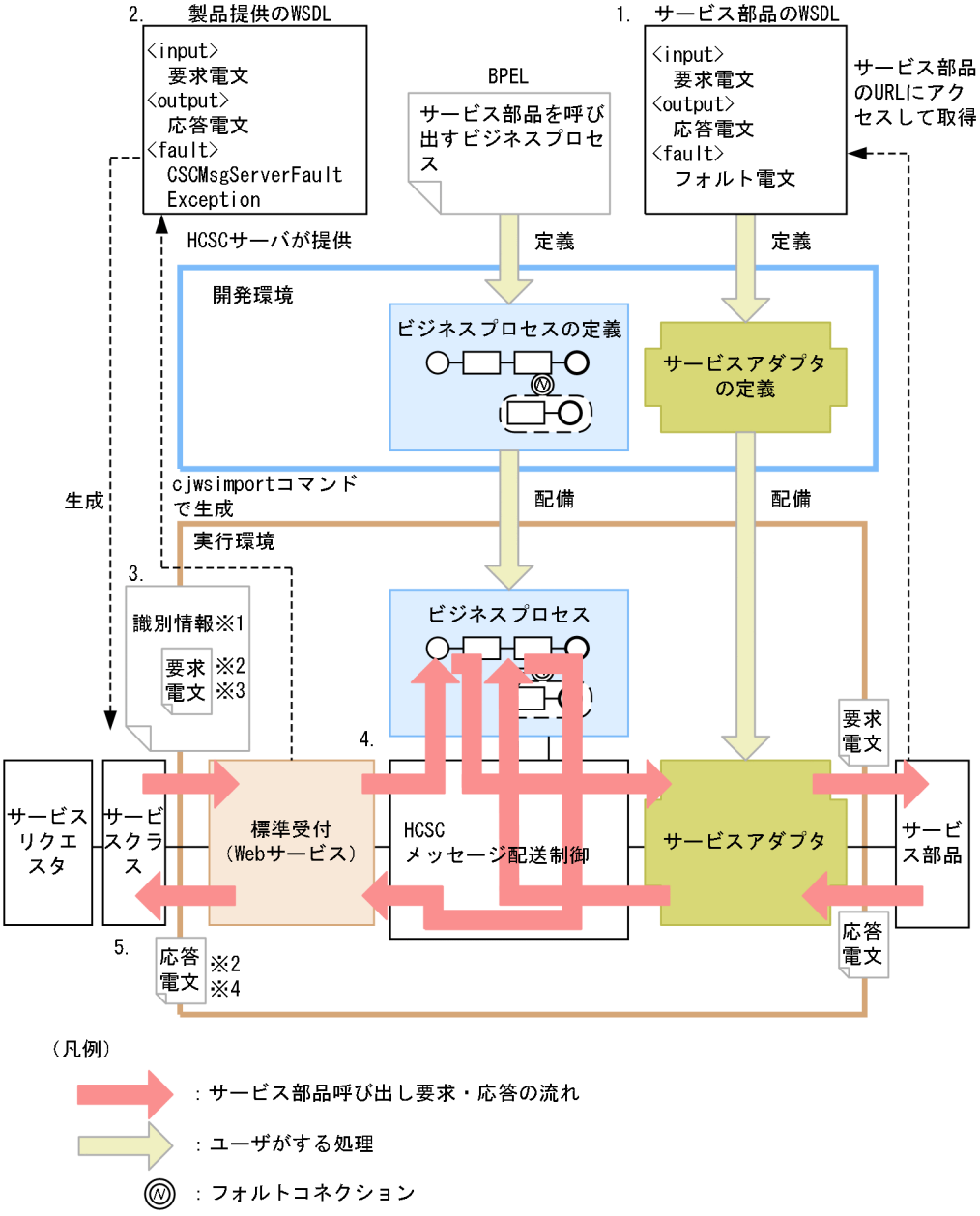
1. サービスアダプタを定義するときは，サービス部品側の電文フォーマットとしてサービス部品の WSDL を定義します（サービス部品の WSDL には，要求電文，応答電文およびフォルト電文の形式が定義されています）。WSDL を作成するには，SOAP 通信基盤が提供する Java2WSDL コマンドを使用します。
2. サービスリクエスタから標準受付（Web サービス）を呼び出してサービス部品を呼び出すときは，HCSC サーバが提供する WSDL を使用します。HCSC サーバが提供する WSDL を基に WSDL2Java コマンドを使用してスタブを作成できます。

3. サービスリクエスタから標準受付（Web サービス）を呼び出すときは，サービス部品側の電文フォーマットに合わせた要求電文を事前に作成します。そして，HCSC サーバが提供する WSDL に定義されている標準受付（Web サービス）のパラメタ（ユーザ電文のパラメタ）に設定して実行します。
4. サービスリクエスタからビジネスプロセスを呼び出すときは，要求電文はビジネスプロセスの受付アクティビティに定義した電文フォーマットになります。サービスリクエスタ内でビジネスプロセスの受付アクティビティで定義されている要求電文の電文フォーマットに合わせた電文を作成し，作成した電文を標準受付（Web サービス）のパラメタ（ユーザ電文のパラメタ）に設定して実行します。
5. サービスリクエスタへの応答は，サービスアダプタで定義した応答電文の電文フォーマットを使用します。

(b)　JAX-WS エンジンを使用するとき

通信基盤に JAX-WS エンジンを使用するときの，SOAP 通信時のユーザ電文と WSDL の関係（標準受付（Web サービス））を次の図に示します。なお，サービスリクエスタ側とサービス部品側で，片方が JAX-WS エンジン，もう片方が SOAP 通信基盤を使用する構成も可能です。SOAP 通信基盤を使用する側の詳細は，「(a)　SOAP 通信基盤を使用するとき」を参照してください。

図 2-11　JAX-WS エンジンを使用するときの SOAP 通信時のユーザ電文と WSDL の関係（標準受付（Web サービス））



注※1　識別情報とは，サービス名，クライアント相関IDなどです。
注※2　サービス部品のWSDLに定義されている要求電文または応答電文です。
注※3　ビジネスプロセスを呼び出す場合は，ビジネスプロセスの受付アクティビティに定義した要求電文です。
注※4　ビジネスプロセスを呼び出す場合は，ビジネスプロセスの応答アクティビティに定義した応答電文です。

1. サービスアダプタを定義するときは，サービス部品側の電文フォーマットとしてサービス部品の WSDL を定義します（サービス部品の WSDL には，要求電文，応答電文およびフォルト電文の形式が定義されています）。サービス部品の WSDL は，サービス部品側が用意している URL にアクセスして WSDL を取得します。

2. サービスリクエスタから標準受付（Web サービス）を呼び出してサービス部品を呼び出すときは，HCSC サーバが提供する WSDL を使用します。HCSC サーバが提供する WSDL を基に cjwsimport コマンドを使用してサービスクラスを作成できます。
3. サービスリクエスタから標準受付（Web サービス）を呼び出すときは，サービス部品側の電文フォーマットに合わせた要求電文を事前に作成します。そして，HCSC サーバが提供する WSDL に定義されている標準受付（Web サービス）のパラメタ（ユーザ電文のパラメタ）に設定して実行します。
4. サービスリクエスタからビジネスプロセスを呼び出すときは，要求電文はビジネスプロセスの受付アクティビティに定義した電文フォーマットになります。サービスリクエスタ内でビジネスプロセスの受付アクティビティで定義されている要求電文の電文フォーマットに合わせた電文を作成し，作成した電文を標準受付（Web サービス）のパラメタ（ユーザ電文のパラメタ）に設定して実行します。
5. サービスリクエスタへの応答は，サービスアダプタで定義した応答電文の電文フォーマットを使用します。

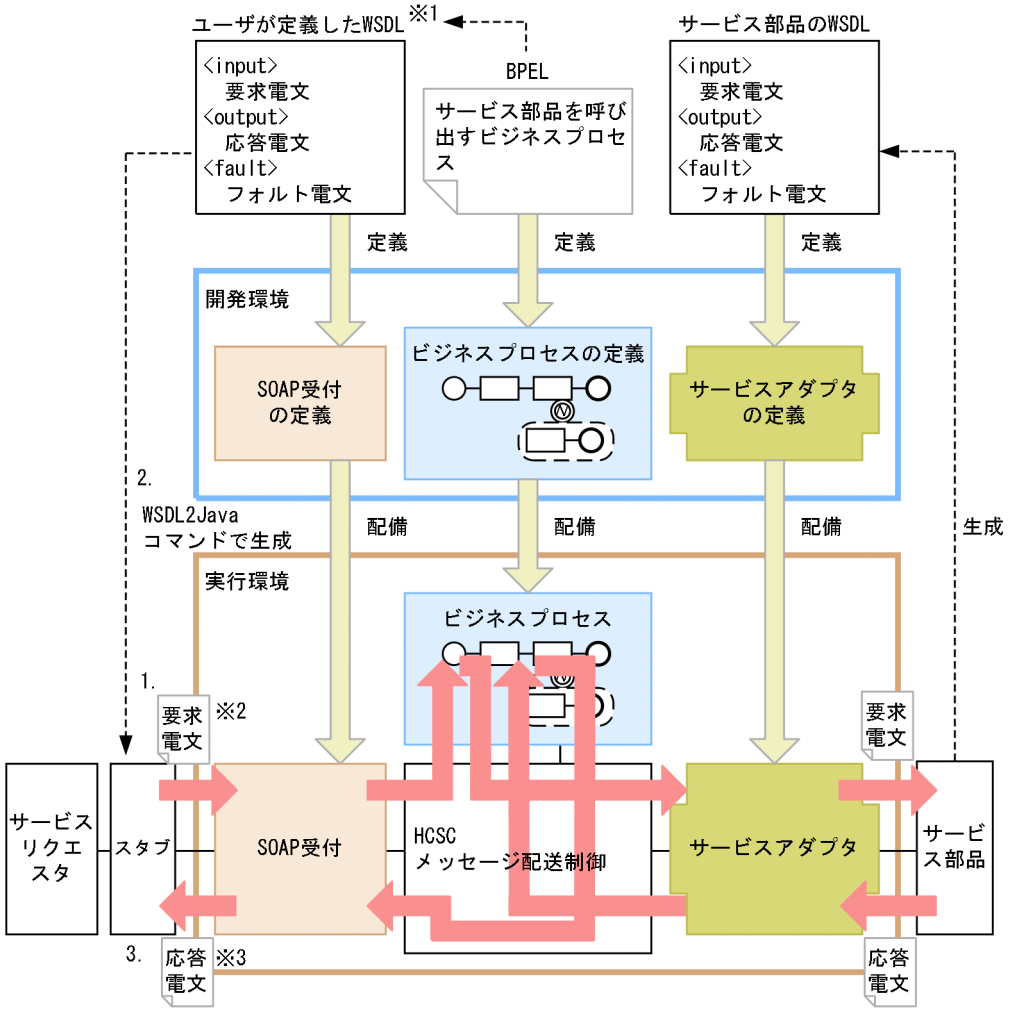
(2) ユーザ定義受付の場合

SOAP 通信時のユーザ電文と WSDL の関係（SOAP 受付）を次に説明します。

(a) SOAP 通信基盤を使用するとき

通信基盤に SOAP 通信基盤を使用するときの，SOAP 通信時のユーザ電文と WSDL の関係（SOAP 受付）を次の図に示します。

図 2-12　SOAP 通信基盤を使用するときの SOAP 通信時のユーザ電文と WSDL の関係（SOAP 受付）



注※1　ビジネスプロセスの受付アクティビティや応答アクティビティに定義した形式に合わせた WSDL です。
注※2　ユーザが定義した WSDL に定義されている要求電文です。
注※3　ユーザが定義した WSDL に定義されている応答電文です。

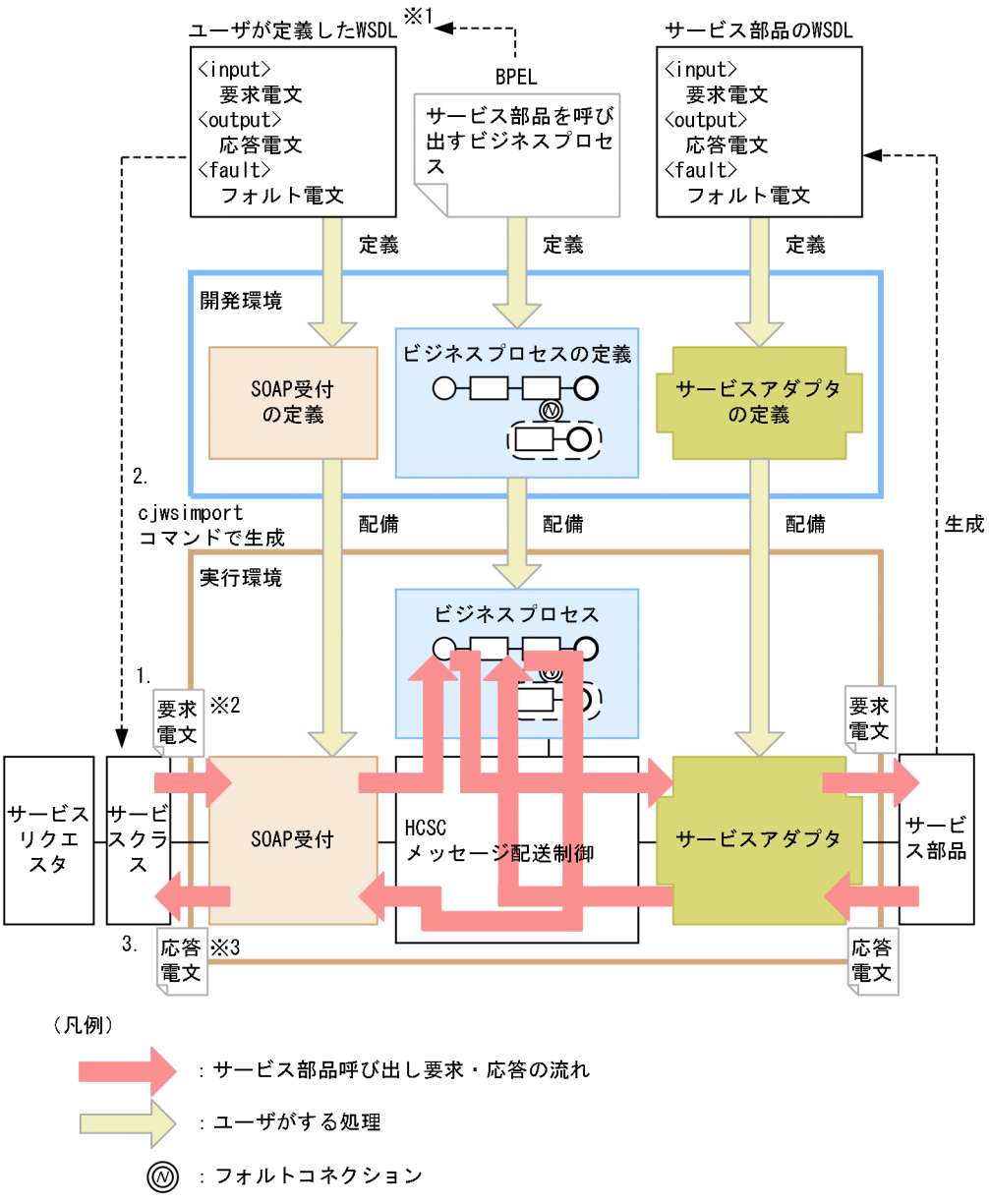
1. SOAP 受付の場合，サービスリクエスタで指定する要求電文は，ビジネスプロセスの受付アクティビティに定義した任意の電文フォーマットの形式で要求します（受付アクティビティに定義した電文フォーマットに合わせた WSDL を作成し，SOAP 受付に定義します）。
2. スタブは，ユーザが定義した WSDL を基に，SOAP アプリケーション開発支援機能が提供する WSDL2Java コマンドを使用して作成できます。標準受付（Web サービス）の場合と異なり，サービスリクエスタ側のプログラムで要求電文の XML を作成する必要はありません。サービスリクエスタではスタブに要求を出すときに，それぞれのパラメタに値をセットして実行します（スタブの中で SOAP メッセージを作成します）。

3. サービスリクエスタへの応答は，ビジネスプロセスの応答アクティビティで定義した応答電文の電文フォーマットで応答します（応答アクティビティに定義した電文フォーマットに合わせた WSDL を作成し，SOAP 受付に定義します）。

(b) JAX-WS エンジンを使用するとき

通信基盤に JAX-WS エンジンを使用するときの，SOAP 通信時のユーザ電文と WSDL の関係（SOAP 受付）を次の図に示します。

図 2-13 JAX-WS エンジンを使用するときの SOAP 通信時のユーザ電文と WSDL の関係（SOAP 受付）



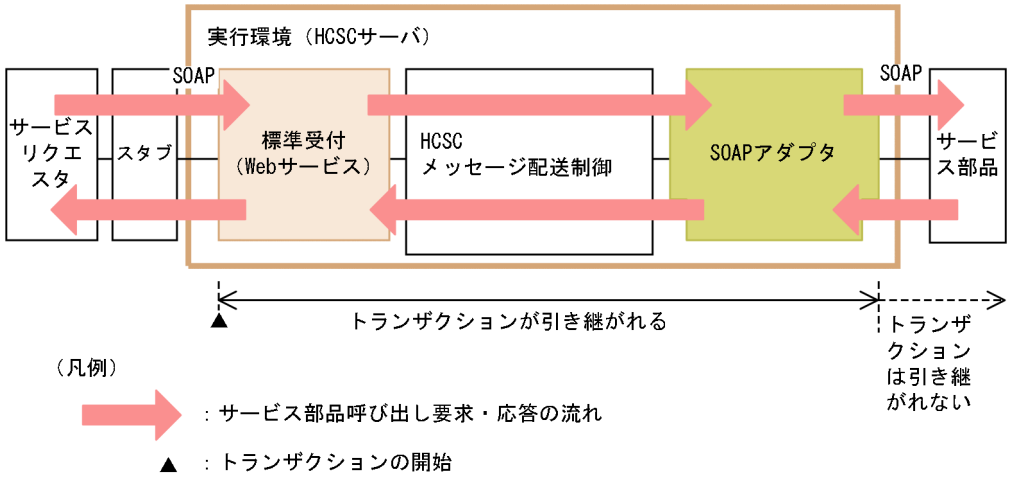
注※1　ビジネスプロセスの受付アクティビティや応答アクティビティに定義した形式に合わせたWSDLです。
注※2　ユーザが定義したWSDLに定義されている要求電文です。
注※3　ユーザが定義したWSDLに定義されている応答電文です。

1. SOAP 受付の場合，サービスリクエスタで指定する要求電文は，ビジネスプロセスの受付アクティビティに定義した任意の電文フォーマットの形式で要求します（受付アクティビティに定義した電文フォーマットに合わせた WSDL を作成し，SOAP 受付に定義します）。
2. サービスクラスは，ユーザが定義した WSDL を基に，JAX-WS 機能が提供する cjwsimport コマンドを使用して作成できます。標準受付（Web サービス）の場合と異なり，サービスリクエスタ側のプログラムで要求電文の XML を作成する必要はありません。サービスリクエスタではサービスクラスに要求を出すときに，それぞれのパラメタに値をセットして実行します（サービスクラスの中で SOAP メッセージを作成します）。
3. サービスリクエスタへの応答は，ビジネスプロセスの応答アクティビティで定義した応答電文の電文フォーマットで応答します（応答アクティビティに定義した電文フォーマットに合わせた WSDL を作成し，SOAP 受付に定義します）。

2.2.3 Web サービス（SOAP 通信）の場合のトランザクション

Web サービス（SOAP 通信）の場合，HCSC サーバ内で管理しているトランザクションは，サービス側には引き継がれません。Web サービス（SOAP 通信）の場合のトランザクションを次の図に示します。

図 2-14 Web サービス（SOAP 通信）の場合のトランザクション



2.2.4 Web サービス（SOAP 通信）の場合の Cookie 情報

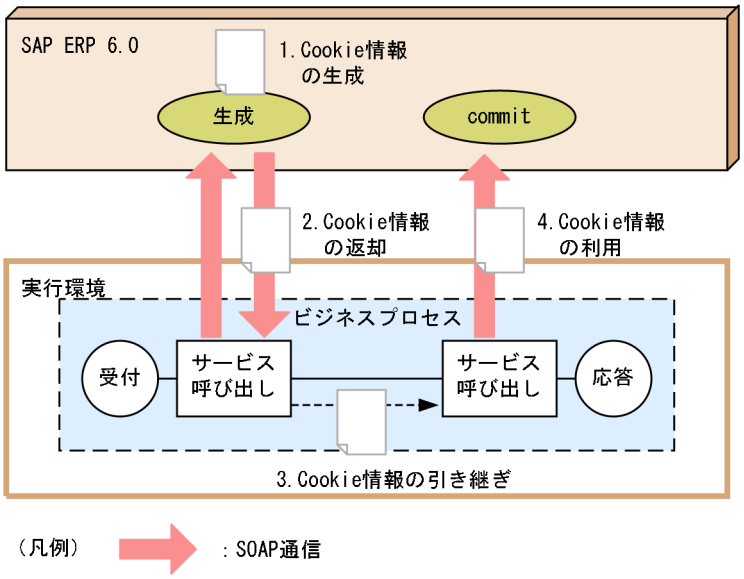
ビジネスプロセスから SAP ERP 6.0 上のサービスを呼び出す場合，更新系の処理によって返却された Cookie 情報を引き継ぐ必要があります。SAP ERP 6.0 以外のサービスを呼び出す場合，Cookie 情報の引き継ぎは不要です。

SAP ERP 6.0 上のサービスを呼び出す場合で，Cookie 情報を引き継ぐには，HCSC サーバランタイム定義ファイルの cookie-parsing プロパティの指定によって，Cookie 情報の引き継ぎを有効にする方法と，ビジネスプロセスのヘッダ割当変数に HTTP ヘッダ名を設定する方法があります。ビジネスプロセスの

ヘッダ割当変数に HTTP ヘッダ名を設定する方法の詳細については，「2.2.5　Web サービス（SOAP 通信）の場合の HTTP ヘッダ引き継ぎ」を参照してください。

Cookie 情報の引き継ぎの流れを次の図に示します。

図 2-15　SAP ERP 6.0 のサービス呼び出しでの Cookie 情報の引き継ぎ



Cookie 情報としてサービス部品が発行する Set-Cookie ヘッダの情報が引き継がれます。

次に示す Cookie 情報の引き継ぎ条件をすべて満たす場合は，Set-Cookie ヘッダの属性が Cookie ヘッダに設定されます。

- Set-Cookie ヘッダの Cookie 名とその値（NAME=VALUE）だけを設定する。
- 複数の Cookie 情報がある場合，「;」で区切って設定する。

条件を満たさない場合は，Cookie ヘッダに何も設定されません。

Cookie 情報の引き継ぎ範囲，引き継ぎ条件，およびログ出力について説明します。また，Cookie 情報の引き継ぎに関する注意事項を示します。

(1)　Cookie 情報の引き継ぎ範囲

Cookie 情報を引き継ぐビジネスプロセスの範囲を次に示します。

1. 初回のサービス呼び出しから，受付アクティビティおよび応答アクティビティを実行するまで
2. 初回のサービス呼び出しから，ビジネスプロセスインスタンスが終了するまで

1.の範囲では，次に示す場合でも Cookie 情報が引き継がれます。

- 階層的にビジネスプロセスを呼び出す場合

  呼び出し先のビジネスプロセスに受付アクティビティおよび応答アクティビティがあっても Cookie 情報は引き継がれます。

- 受付アクティビティが非同期のビジネスプロセスを呼び出す場合

  通信モデルが非同期の場合，受付アクティビティで定義したオペレーション名に対する応答アクティビティは配備できません。この場合はサービス呼出アクティビティ間に受付アクティビティがないときだけ Cookie 情報が引き継がれます。

- 応答アクティビティのあとにアクティビティが定義されているビジネスプロセスを呼び出す場合

  この場合は，サービス呼出アクティビティ間に受付アクティビティがないときだけ Cookie 情報が引き継がれます。

- 待機アクティビティが定義されているビジネスプロセスを呼び出す場合

  応答アクティビティが実行されたあとに待機アクティビティがある場合，その待機アクティビティの前後にサービス呼出アクティビティがあっても，Cookie 情報は引き継がれます。

ただし，次の場合は引き継ぎ範囲に関係なく，Cookie 情報は引き継がれません。

- 実行環境のプロセスがダウンした場合
- 複数の実行環境のプロセスでリクエストを処理する場合
- プロセスインスタンスを再実行した場合

(2) Cookie 情報の引き継ぎ条件

サービス部品を呼び出すときの情報と呼び出し先の情報に関して，次に示す条件をすべて満たす場合に Cookie 情報が引き継がれます。

- HCSC サーバランタイム定義ファイルの cookie-parsing プロパティに ON を指定している，または SOAP アダプタを呼び出しているサービス呼出アクティビティで，ヘッダ割当変数に Cookie 情報を設定している。
- 「サービス部品のホスト名:ポート番号」の文字列が完全一致する。

  ホスト名の指定形式が異なっていると，それぞれ別の URL として扱われます。例えば，同じホストを指定する場合に，ホスト名で指定しているものと IP アドレスで指定しているものが混在しているときは，それぞれ別の URL として扱われます。このとき，Cookie 情報は引き継がれません。また，ポート番号が設定されていない場合は，「:ポート番号」のない文字列で比較されます。

- path 属性が前方一致する。

  前方一致と見なされる場合および見なされない場合の例を示します。

  （例）前方一致と見なされる場合

  /sap と/sap，/sap と/sap/aaa，/sap と/sapaaa/bbb

  （例）前方一致と見なされない場合

  /sap と/aaa，/sap と/aaasap/bbb，/sap と/aaa/sap

注意事項

- Cookie 情報の引き継ぎは，SAP ERP 6.0 の Set-Cookie ヘッダ仕様の「path 属性」によって判断されます。ただし，SOAP 通信基盤（SOAP1.1 モード）を使用する場合は，「path 属性」は考慮

されないで，「ホスト名:ポート番号」だけが考慮されます。ポート番号が指定されていない場合はホスト名が考慮されます。

- Set-Cookie ヘッダに path 属性が含まれていない場合は，デフォルト値として「path=/」が設定されます。
- 次の例のように，Set-Cookie ヘッダに属性名が「path」の属性が複数設定されている場合は，path 属性に「path=/」が設定されたものとして動作します。

  （例）

  Set-Cookie: NAME=value; path=/sap/aaa; path=/sap/bbb
- 複数の異なる path 属性値を持つ Cookie 情報を HTTP ヘッダに設定する場合，Set-Cookie の読み込み順序に依存しないで，path 属性値が詳細な順に HTTP ヘッダに設定されます。

  例えば，次の Set-Cookie ヘッダを受け取った場合，HTTP ヘッダには「cookie:NAME3=value3; NAME2=value2; NAME1=value1」が設定されます。

  （例）

  Set-Cookie: NAME1=value1; path=/

  Set-Cookie: NAME2=value2; path=/sap

  Set-Cookie: NAME3=value3; path=/sap/aaa

(3) ログ出力

HCSC サーバランタイム定義ファイルの methodtrace-level プロパティに「2」以上の値を設定した場合に，メソッドトレースにログが出力されます。

(4) 注意事項

(a) サポートしていない仕様および属性について

- Netscape Communications 社の Set-Cookie ヘッダの仕様である「domain 属性」，「expires 属性」，および「secure 属性」は，サポートしていません。
- RFC2109 の Set-Cookie ヘッダの仕様である「Comment 属性」，「Domain 属性」，「Max-Age 属性」，および「Secure 属性」は，サポートしていません。
- RFC2965 でサポートされている Set-Cookie2 の仕様は，サポートしていません。
- 「JSESSIONID」は，サポートしていません。SAP ERP 6.0 では，「sap-contextid 属性」でセッション管理をしています。

(b) バージョン属性について

- Cookie バージョンは，「0」（Netscape Communication 社の Set-Cookie 仕様）または「1」（RFC2109 の Set-Cookie 仕様）だけを設定できます。なお，バージョンは次に示す順序で決定されます。

  1. Set-Cookie ヘッダに「expires 属性」が定義されている場合，バージョンは 0 になります。

2. Set-Cookie ヘッダに「Version 属性」が定義されている場合，バージョンは 1 になります。

3. Set-Cookie ヘッダに「Max-Age 属性」が定義されている場合，バージョンは 1 になります。

4. 1～3 のどれにも当てはまらない場合，バージョンは 0 になります。

- SAP ERP 6.0 が発行する Set-Cookie ヘッダには「Version 属性」は含まれません。
- 「Version 属性」が複数の Cookie 情報で異なる場合は，最初に読み込んだ Cookie 情報が持つ「Version 属性」の値が採用されます。
- RFC2109 でサポートしている Cookie 仕様で動作を行う場合，「Version 属性」の設定値を Cookie ヘッダの「cookie-version 属性」に設定する必要があります。

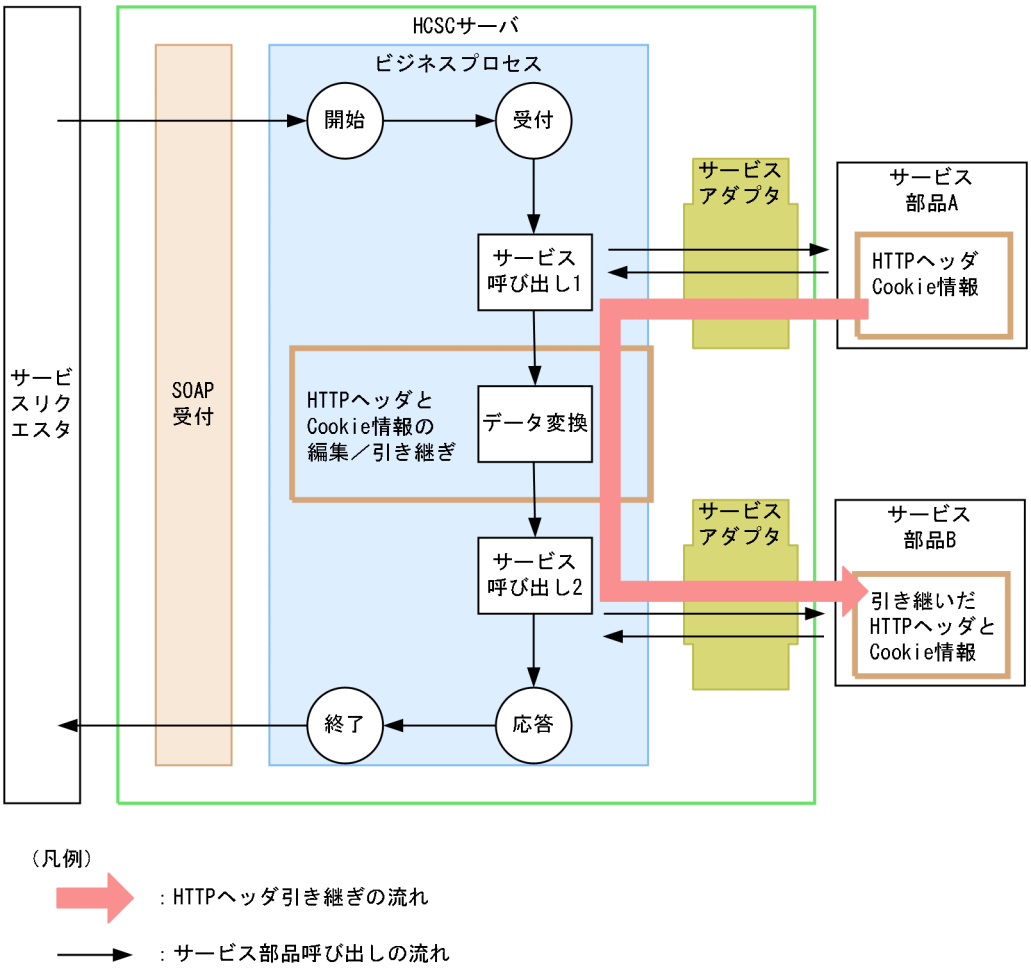
2.2.5 Web サービス（SOAP 通信）の場合の HTTP ヘッダ引き継ぎ

HTTP ヘッダ引き継ぎ機能とは，同一のビジネスプロセス内で，SAP ERP 6.0 上の異なるサービスを実行した場合に取得した HTTP ヘッダを引き継ぐ機能です。SAP ERP 6.0 以外のサービスを呼び出す場合，この機能は必要ありません。

(1) HTTP ヘッダ引き継ぎ機能の概要

HTTP ヘッダ引き継ぎ機能の概要を次の図に示します。

図 2-16　HTTP ヘッダ引き継ぎ機能の概要



(2)　HTTP ヘッダ引き継ぎ機能の引き継ぎ方法

HTTP ヘッダや Cookie 情報を，サービス呼出アクティビティのヘッダ領域に割り当てることで引き継げます。ヘッダ領域への割り当て方法の詳細については，マニュアル「サービスプラットフォーム 開発ガイド 基本開発編」の「付録 F　サービスアダプタを利用した HTTP ヘッダおよび Cookie 情報の引き継ぎ」を参照してください。

(3)　HTTP ヘッダ引き継ぎ機能の引き継ぎ情報の範囲

HCSC サーバランタイム定義ファイルの cookie-parsing プロパティの値と，サービス呼出アクティビティのヘッダ領域に情報を割り当てるかどうかによって，引き継がれる情報が異なります。

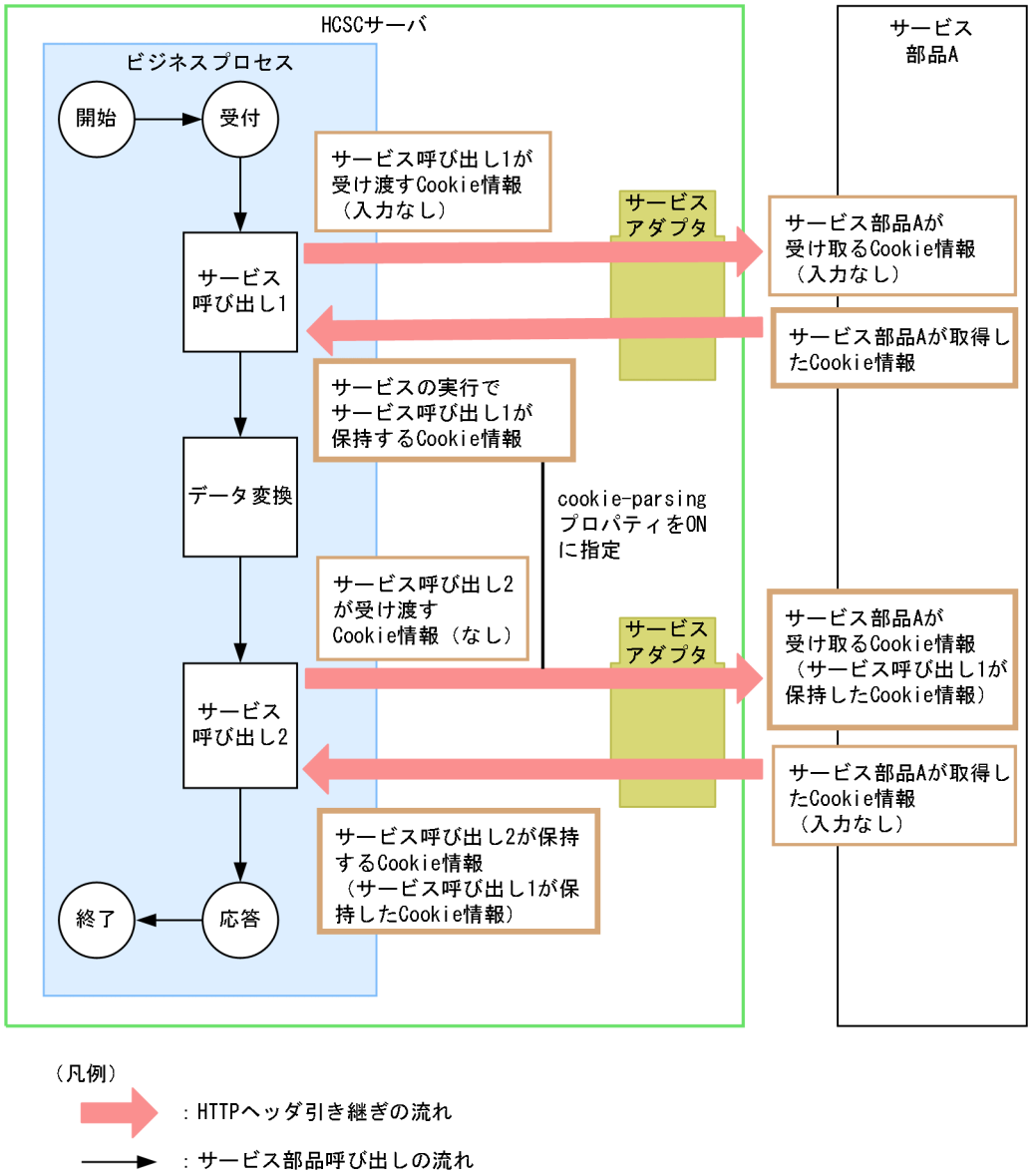
なお，Cookie 情報を送受信するサービス部品のホスト名，IP アドレス，または path 属性が一致しない場合は，Cookie 情報を送信しません。

HCSC サーバランタイム定義ファイルの cookie-parsing プロパティを ON，かつヘッダ割当変数に Cookie 情報を設定していない場合

すべての Cookie 情報が引き継がれます。サービス部品呼び出し 1 で取得した情報がサービス部品呼び出し 2 に引き継がれます。

引き継ぎの内容を次の図に示します。

図 2-17　HCSC サーバランタイム定義ファイルの cookie-parsing プロパティを ON，かつヘッダ割当変数に Cookie 情報を設定していない場合の引き継ぎ内容

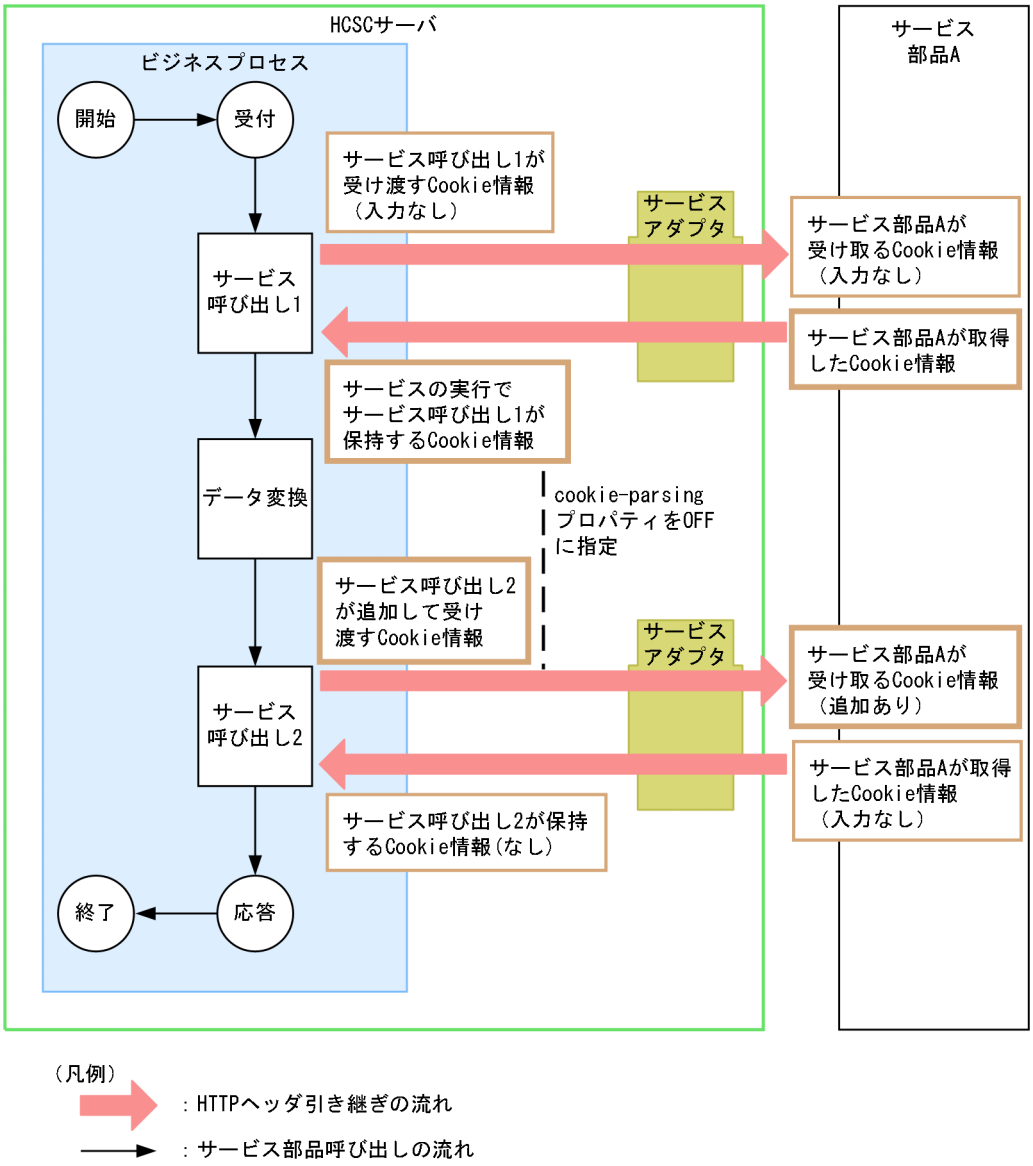


HCSC サーバランタイム定義ファイルの cookie-parsing プロパティを OFF，かつヘッダ割当変数に Cookie 情報を設定している場合

ヘッダ領域に設定された Cookie 情報がサービス部品呼び出し 2 の実行で引き継がれます。ただし，サービス部品呼び出し 2 に情報は返却されません。

引き継ぎの内容を次の図に示します。

図 2-18　HCSC サーバランタイム定義ファイルの cookie-parsing プロパティを OFF，かつヘッダ割当変数に Cookie 情報を設定している場合の引き継ぎ内容



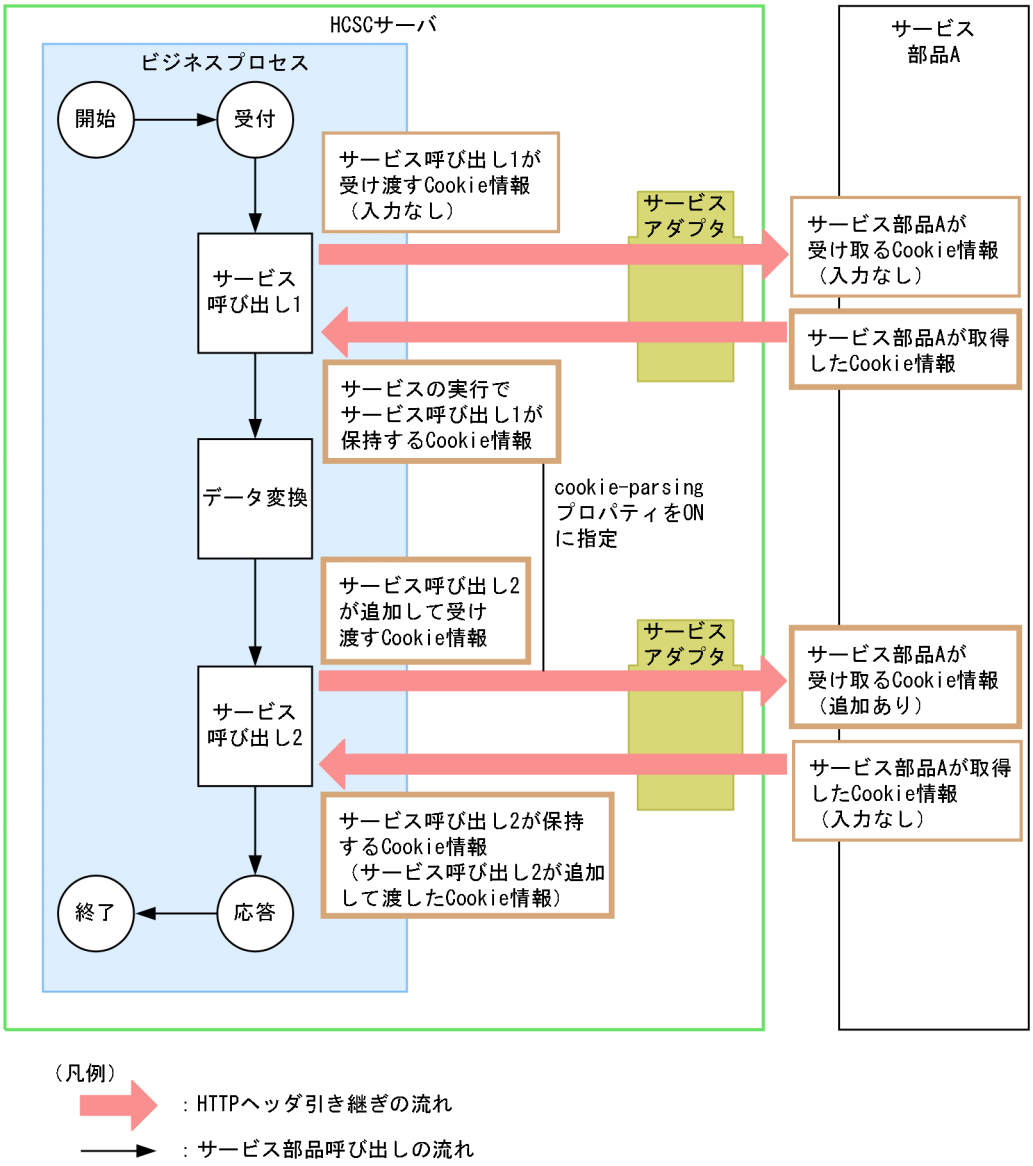
HCSC サーバランタイム定義ファイルの cookie-parsing プロパティを ON，かつヘッダ割当変数に Cookie 情報を設定している場合

name 属性，path 属性，および host 属性が同じである Cookie 情報がヘッダ領域に設定されたときは，サービス部品呼び出し 1 の実行で取得した Cookie 情報がヘッダ領域に設定された Cookie 情報に書き換えられ，サービス部品呼び出し 2 の実行で引き継がれます。

ただし，name 属性が同じでも，path 属性または host 属性が異なる場合は，サービス部品呼び出し 1 で取得した Cookie 情報とヘッダ領域に設定された Cookie 情報がサービス部品呼び出し 2 の実行で引き継がれます。

引き継ぎの内容を次の図に示します。

図 2-19　HCSC サーバランタイム定義ファイルの cookie-parsing プロパティを ON，かつヘッダ割当変数に Cookie 情報を設定している場合の引き継ぎ内容

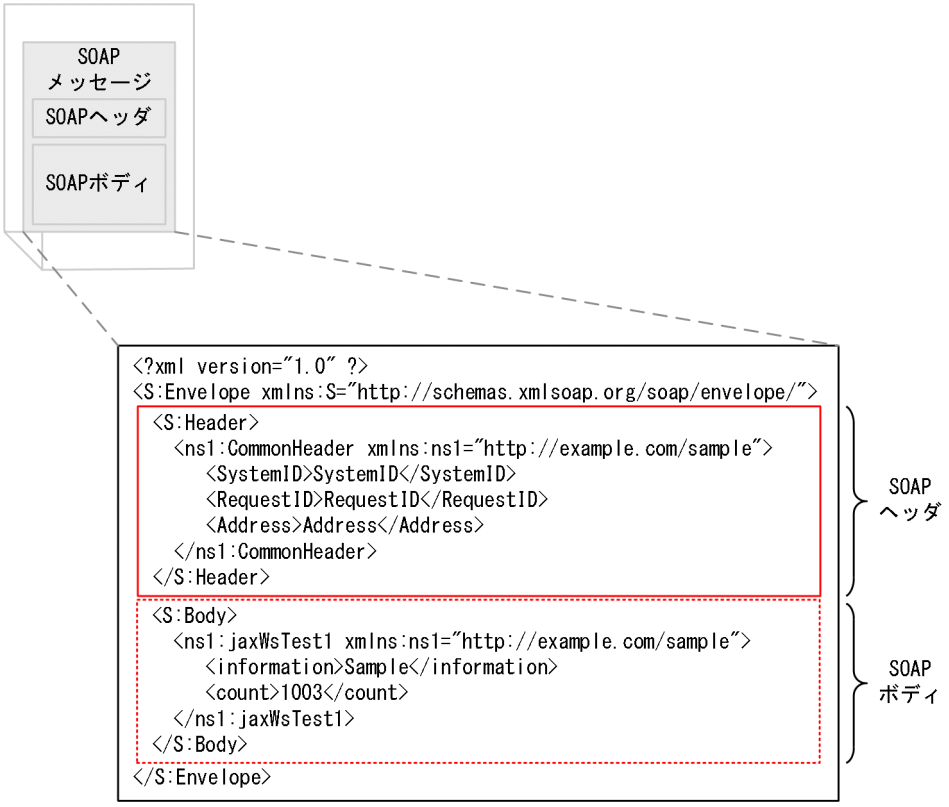


2.2.6　Web サービス（SOAP 通信）の場合の SOAP メッセージの構成

Web サービス（SOAP 通信）の場合，要求電文および応答電文としてやり取りしている SOAP メッセージは，ヘッダ（SOAP ヘッダ）とボディ（SOAP ボディ）で構成されています。SOAP ヘッダにはメッセージの付加的な情報（識別情報など）が含まれ，SOAP ボディにはメッセージ本体が含まれています。

SOAP メッセージの構成例を次の図に示します。

図 2-20　SOAP メッセージの構成例

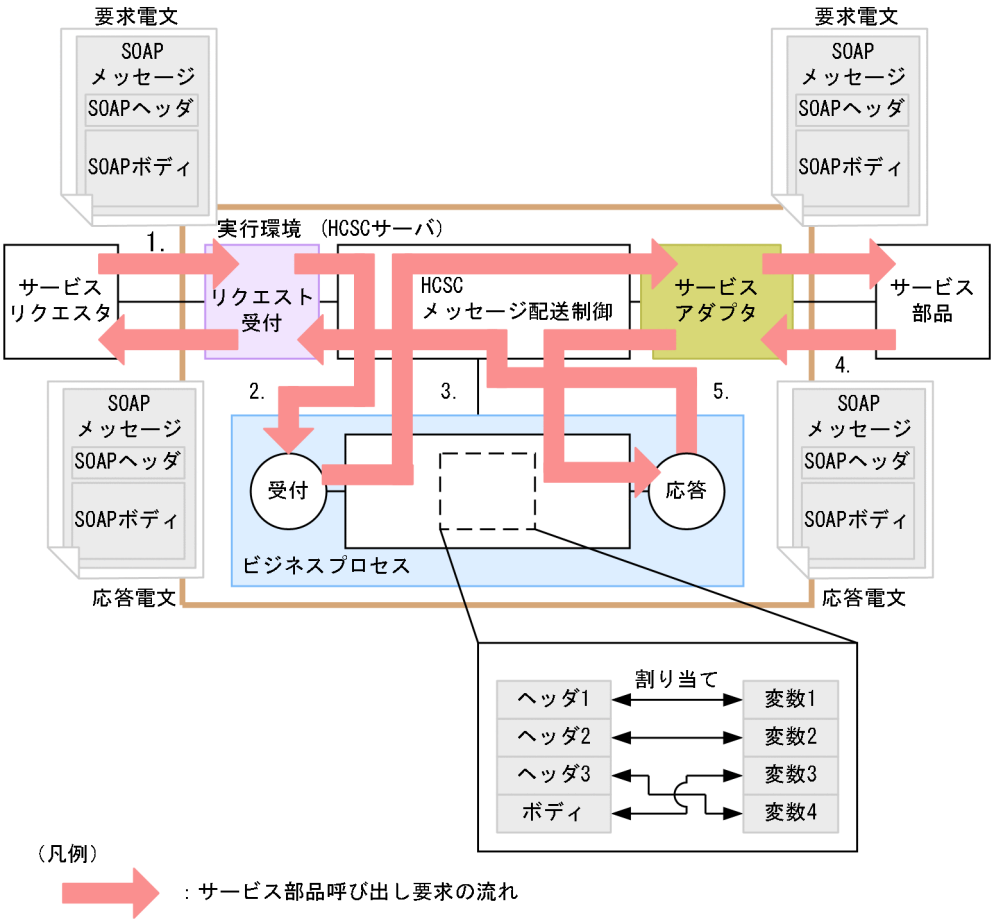


(1)　SOAP メッセージとビジネスプロセスの関係

SOAP メッセージの場合，要求電文のヘッダ情報とボディ情報は，業務処理で利用できるようにビジネスプロセスの変数に割り当てられます。また，応答電文として返されるときは，変数から再びヘッダ情報とボディ情報を生成します。

SOAP メッセージとビジネスプロセスの関係を次の図に示します。

図 2-21　SOAP メッセージとビジネスプロセスの関係



図中の 1.～5.について説明します。

説明

1. サービスリクエスタから受付に対して，業務処理に必要なパラメタを設定した要求電文（SOAP メッセージ）が渡されます。
2. 受付からビジネスプロセスが起動されます。このとき，受付が受信した SOAP メッセージに設定されているヘッダ情報およびボディ情報を，業務処理で利用できるようにビジネスプロセスの変数に割り当てます。
3. ビジネスプロセスが業務処理を実行します。サービスを呼び出すときは，変数からサービスの実行に必要なパラメタを設定した SOAP メッセージを生成し，サービスアダプタ経由でサービス部品に送信します。
4. サービス部品は処理が終わると，その結果を応答電文（SOAP メッセージ）として返します。ビジネスプロセスはサービスアダプタ経由でこの SOAP メッセージを受け取り，ヘッダやボディの情報を再び変数に割り当てます。
5. ビジネスプロセスの処理が終わると，変数から SOAP メッセージを生成し，受付を経由してサービスリクエスタに結果を返します。

(2) ユーザ電文トレースに出力される SOAP メッセージの情報

SOAP メッセージの場合，ユーザ電文トレースデータにはユーザ電文に含まれる次の情報が出力されます。

- soap:Header 要素を含む複数の子要素
- soap:Body 要素の 1 つの子要素

なお，soap:Header 要素が存在しない場合は，soap:Body 要素の子要素だけが情報として出力されます。

また，次の場合は動作が保証されませんので，要求電文を作成するときに注意が必要です。

- SOAPHeader 配下の子要素に，ユーザ定義受付に指定された名前空間と要素名を持つ子要素が入れ子になっている場合
- SOAPBody 配下の子要素に，オペレーション名が同一の子要素が複数設定された場合

soap:Header 要素および soap:Body 要素については，マニュアル「サービスプラットフォーム 開発ガイド 基本開発編」の「2.6.1　Web サービスを利用したサービス部品の適用範囲」を参照してください。

2.2.7 Web サービス（SOAP 通信）の場合の接続先動的変更

SOAP アダプタの通信先は，ビジネスプロセスで設定した接続先 URL を使って動的に変更できます。ビジネスプロセスで接続先 URL を設定しておくと，サービスアダプタを実行したときに通信先が変更されます。これにより，作成するサービスアダプタの数を削減できます。

なお，ビジネスプロセスで接続先 URL を設定しなかった場合は，開発環境で設定した Web サービスの接続先 URL で SOAP 通信を行います。

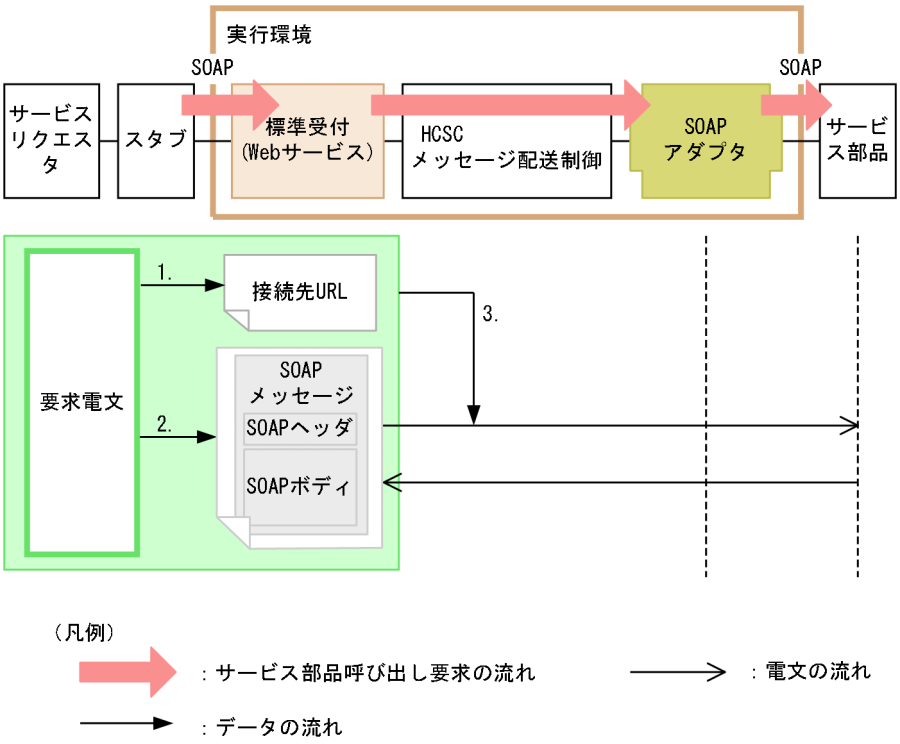
(1) 接続先動的変更の処理の流れ

通信基盤ごとの，SOAP 通信時の接続先動的変更の処理の流れを次に説明します。

(a) SOAP 通信基盤を使用するとき

通信基盤に SOAP 通信基盤を使用するときの接続先動的変更の処理の流れを次の図に示します。

図 2-22　通信基盤に SOAP 通信基盤を使用するときの接続先動的変更の処理の流れ



説明

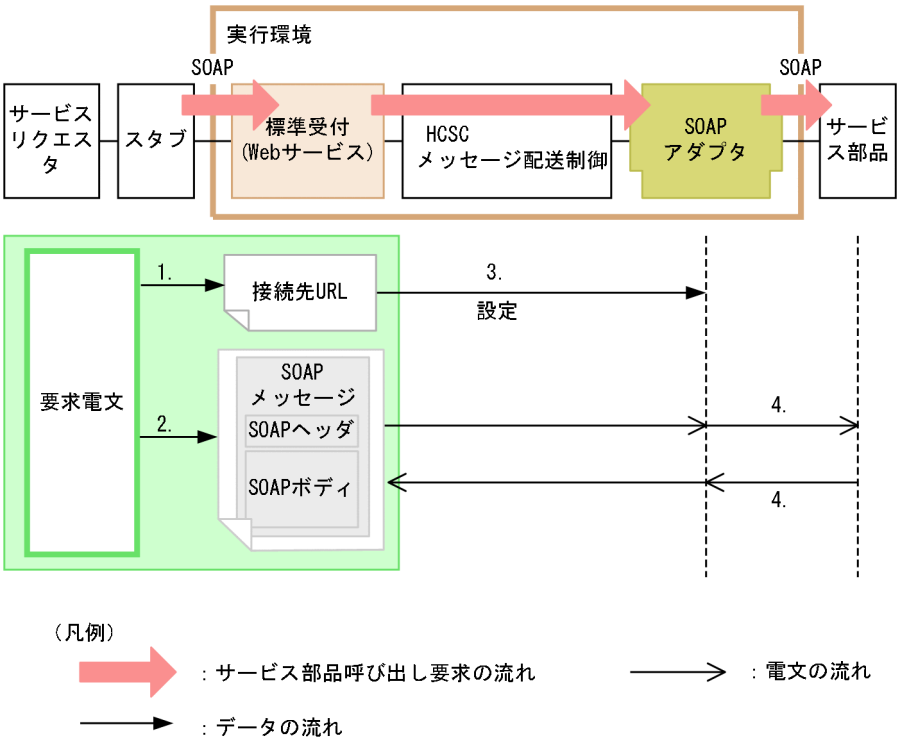
図中の番号は，次の説明の番号と対応しています。

1. 要求電文内のヘッダ電文から接続先 URL を取得します。
2. 要求電文の情報を基に SOAP メッセージを作成します。
3. 取得した接続先 URL でサービスと通信を行います。

(b)　JAX-WS エンジンを使用するとき

通信基盤に JAX-WS エンジンを使用するときの接続先動的変更の処理の流れを次の図に示します。

図 2-23　通信基盤に JAX-WS エンジンを使用するときの接続先動的変更の処理の流れ



説明

図中の番号は，次の説明の番号と対応しています。

1. 要求電文内のヘッダ電文から接続先 URL を取得します。
2. 要求電文の情報を基に SOAP メッセージを作成します。
3. 接続先 URL を設定します。
4. 設定した接続先 URL のサービスと通信を行います。

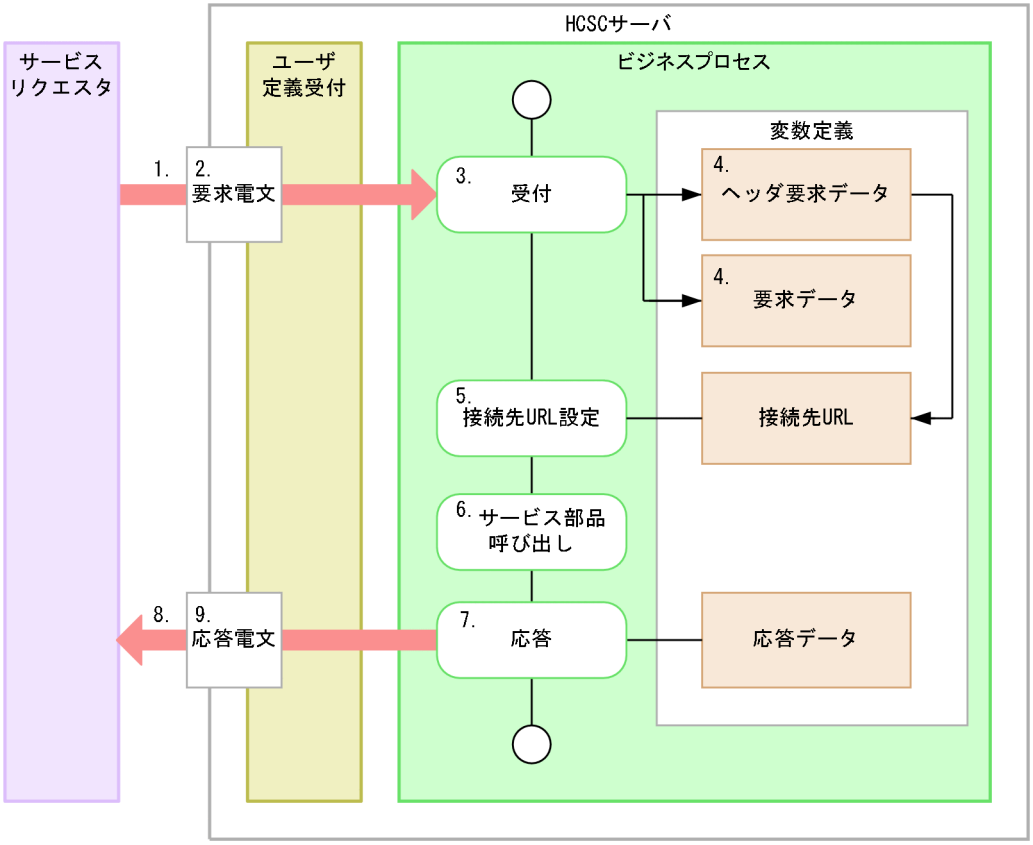
(2)　接続先情報を動的に変更する例

接続先情報ファイルを使用して，接続先情報を動的に変更する例を説明します。

(a)　ビジネスプロセスの例

接続先情報ファイルを使用するビジネスプロセスの例を次の図に示します。

図 2-24　接続先情報ファイルを使用するビジネスプロセスの例（SOAP アダプタの場合）

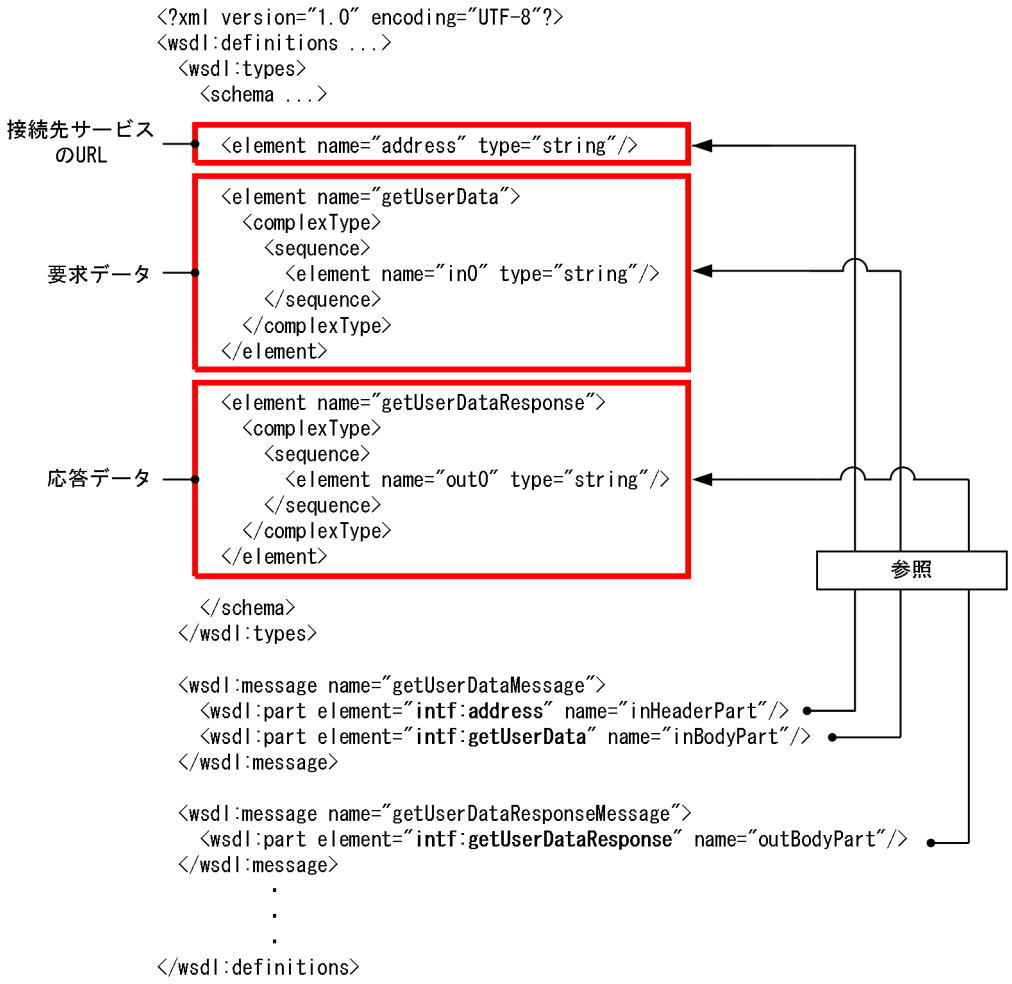


1. サービスリクエスタがビジネスプロセスにサービス部品の実行を要求します。
2. サービスリクエスタから送付する要求電文内のヘッダ電文に接続先 URL を設定します。
3. ビジネスプロセスは XML の要求電文を受付アクティビティで受け取ります。
4. ビジネスプロセスがヘッダ要求データおよび要求データを生成します。
5. ヘッダ要求データの接続先 URL を，データ変換アクティビティなどで接続先 URL 変数に設定します。
6. ビジネスプロセスは接続先 URL 変数から接続先 URL を取得して，サービス部品を呼び出します。
7. ビジネスプロセスが，サービス部品の実行結果を XML の応答電文として構成し，応答アクティビティで返します。
8. サービスリクエスタへの応答が返されます。
9. サービスリクエスタへの応答では，XML の応答電文が送付されます。

(3)　ユーザ定義受付を生成する WSDL の作成例

ユーザ定義受付を生成する WSDL の作成例を次に示します。

なお，この例では，接続先となるサービス部品の URL を SOAP ヘッダで送付しています。

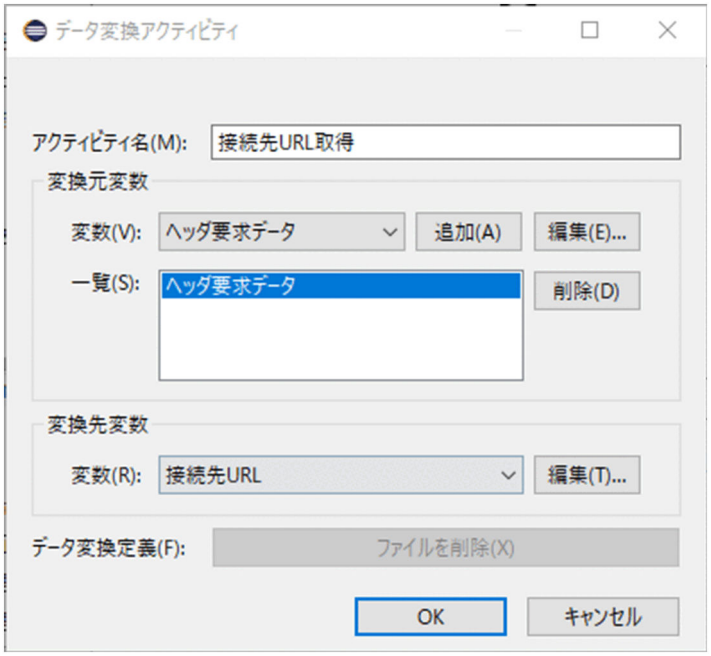


(a)　データ変換アクティビティの設定例

「図 2-24　接続先情報ファイルを使用するビジネスプロセスの例（SOAP アダプタの場合）」に示すビジネスプロセスで使用するデータ変換アクティビティの設定例を次に示します。

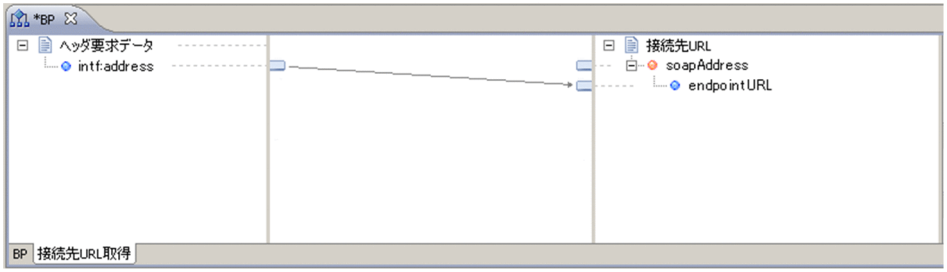
ここでは，変換元に「ヘッダ要求データ」を，変換先に「接続先 URL」を指定しています。

図 2-25　データ変換アクティビティの設定例（SOAP アダプタの場合）



また，変換のマッピングは，次の図に示すとおりに定義します。

図 2-26　変換のマッピングの設定例（SOAP アダプタの場合）



(b)　接続先情報ファイルの定義

Web サービスに接続するための URL をサービスアダプタに設定するための接続先情報ファイル（XML スキーマ）に定義します。

サービスプラットフォームでは，接続先情報ファイルを提供しています。

- 接続先情報ファイルの格納先

```
<サービスプラットフォームのインストールディレクトリ>/CSC/schema/connection/connection_soap_adapter.xsd
```

注意事項

SOAP アダプタの接続先情報を動的に変更する場合，サービスプラットフォームで提供している接続先情報ファイルを使用してください。提供している接続先情報ファイルを使用しなかった場合，動作は保証されません。

- 接続先情報ファイルの形式

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
 DO NOT EDIT THIS FILE.
 -->
<xsd:schema
    elementFormDefault="qualified"
    targetNamespace="http://www.hitachi.co.jp/soft/xml/cosminexus/csc/connection/change/soap"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="soapAddress">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="endpointURL" type="xsd:string"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

- 設定できる要素

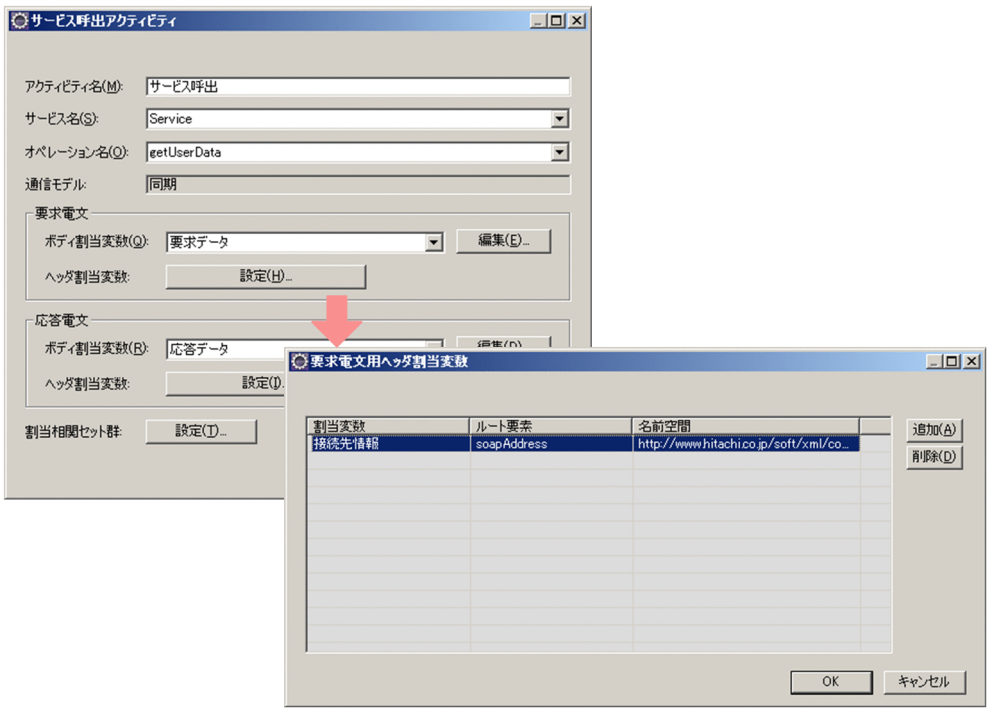
  endpointURL　～<文字列>

  サービス部品の接続先 URL を指定します。

(c)　接続先情報の設定

「図 2-24　接続先情報ファイルを使用するビジネスプロセスの例（SOAP アダプタの場合）」に示すビジネスプロセスで使用する接続先情報の設定例を次に示します。なお，接続先情報は，サービス呼出アクティビティのヘッダ割当変数に設定します。ここで設定した情報が，SOAP アダプタで参照できる情報になります。

図 2-27　接続先情報の設定例（サービス呼出アクティビティの設定）



(4) 注意事項

- ベーシック認証の利用可否，およびベーシック認証で利用するユーザ ID とパスワードについては，接続先 URL を使って動的に変更できません。ベーシック認証に関する情報は，開発環境の画面から設定した情報が有効となります。
- 次の場合，エラーメッセージが出力されて異常終了します。
  - 開発環境で複数の接続先情報がヘッダ割当変数に設定された場合
    XML データのシステム固有の名前空間，要素名が複数存在すると，接続先が特定できないためエラーになります。
  - 設定された接続先情報が不正な場合
    接続先 URL に誤りがあると，サービス呼び出し時にエラーになります。
  - 接続先情報を示す XML データの endpointURL 要素が存在しない場合
    endpointURL 要素がない場合，接続先が特定できないためエラーになります。

2.2.8 Web サービス（SOAP 通信）の場合の添付ファイル

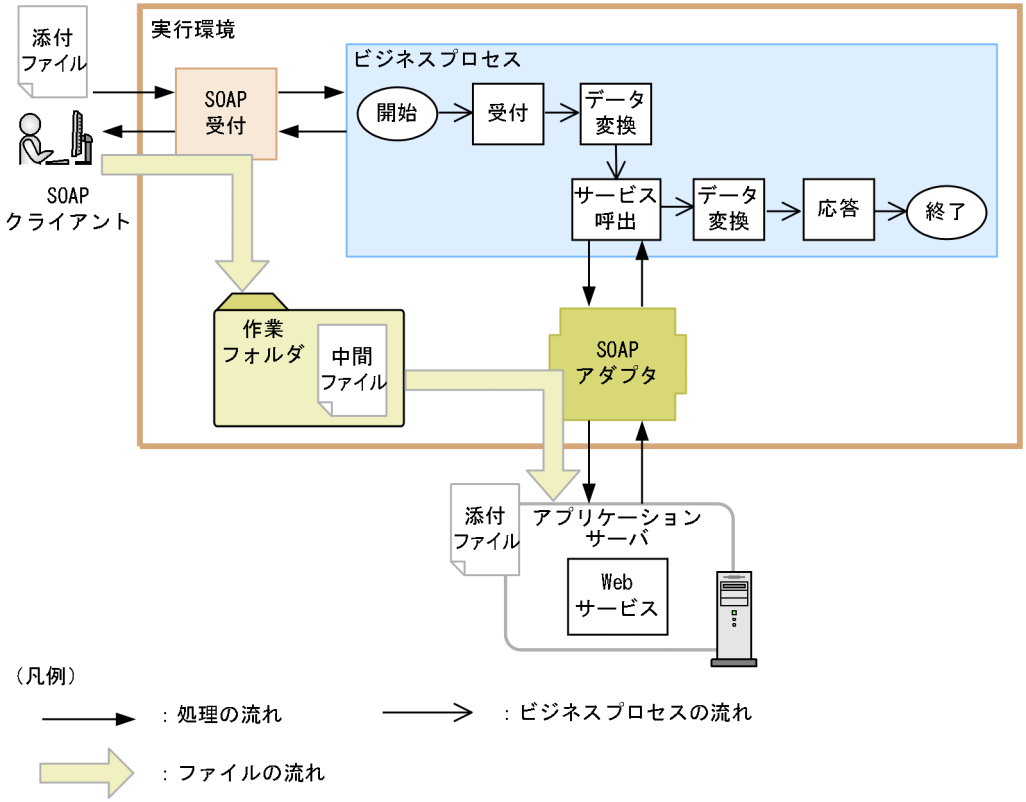
添付ファイル機能を使用すると，任意の形式のファイル（CSV ファイルなど）を Web サービスで扱えます。

(1) 添付ファイル機能の概要

添付ファイル機能では作業フォルダまたは共通フォルダを使用して，システム間のファイル転送を中継します。作業フォルダおよび共通フォルダの詳細については「10. 作業フォルダおよび共通フォルダの機能」を参照してください。

添付ファイル機能の概要を次の図に示します。

図 2-28　添付ファイル機能の概要



(a)　添付ファイルの WSDL

添付ファイル機能を使用する場合，WS-I Attachments Profile - Version 1.0 で規定された形式に基づいて記述された swaRef 形式の WSDL を使用します。swaRef 形式の WSDL の例を次に示します。

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions targetNamespace="http://localhost"
  xmlns:wsi="http://ws-i.org/profiles/basic/1.1/xsd" ...>
  <wsdl:types>
    <schema elementFormDefault="qualified" targetNamespace="http://localhost"
      xmlns="http://www.w3.org/2001/XMLSchema">
      <import namespace="http://ws-i.org/profiles/basic/1.1/xsd" schemaLocation="swaref.xsd"
/>
      <element name="getUserData">
        <complexType>
          <sequence>
            <element name="filename" type="xsd:string"/>
            <element name="attachment" type="wsi:swaRef"/>
          </sequence>
        </complexType>
      </element>
       ...
    </schema>
  </wsdl:types>
  <wsdl:message name="getUserDataRequest">
    <wsdl:part element="intf:getUserData" name="parameters"/>
  </wsdl:message>
  ...
  <wsdl:portType name="UserInfo">
```

```
    <wsdl:operation name="getUserData">
      <wsdl:input message="intf:getUserDataRequest" name="getUserDataRequest"/>
      ...
    </wsdl:operation>
  </wsdl:portType>
  ...
</wsdl:definitions>
```

(b) 添付ファイル付き SOAP メッセージの構造

添付ファイル付き SOAP メッセージは，SOAP Message with Attachments プロトコルを使用し，MIME Multiple/Related 構造でエンコードされます。メッセージの構造の詳細はマニュアル「アプリケーションサーバ Web サービス開発ガイド」の「28.4　添付ファイル付き SOAP メッセージ（wsi:swaRef 形式）」を参照してください。

(c) 添付ファイル機能で使用するスキーマ

SOAP 受付とビジネスプロセスの間，およびビジネスプロセスと SOAP アダプタの間で添付ファイルの情報を連携するために使用する電文フォーマットについて説明します。

- urecp_soap_attachment_request.xsd
  SOAP 受付で生成された添付ファイルに関する XML 電文をビジネスプロセスで扱うために使用する電文フォーマットです。スキーマは「<サービスプラットフォームのインストールディレクトリ>¥CSC¥schema¥soaprecp¥urecp_soap_attachment_request.xsd」に格納されています。
  名前空間名は「http://www.hitachi.co.jp/soft/xml/cosminexus/csc/reception/attachment/soap/request」です。

表 2-1　SOAP 受付のファイル連携用要求ヘッダフォーマット

タグ名	種別	出現回数	説明
<attachment-request>	−	1 回	−
<request-id>	string	1 回	受付で作成したリクエスト ID を示すタグです。
<files>	−	0 または 1 回	受け付けた添付ファイルを示すタグです。ファイルを受信した場合に出現します。
<file>	−	0 回以上	受け付けた 1 つの添付ファイルを示すタグです。受信したファイルの数と同じ回数出現します。
<content-id>	string	1 回	受け付けた添付ファイルの Content-Id を示すタグです。 Content-Id が設定されていない場合，または null の場合は SOAP 受付でエラーになります。
<content-type>	string	1 回	受け付けた添付ファイルの Content-Type を示すタグです。 charset 属性には，Content-Type ヘッダの charset パラメタに対応する文字コードが設定されます。
<local-file-name>	string	1 回	作業フォルダに出力した添付ファイルの中間ファイル名を示すタグです。

- urecp_soap_attachment_response.xsd

  SOAP 受付からサービスリクエスタに応答する添付ファイルの情報をビジネスプロセスから SOAP 受付に渡すために使用します。スキーマは「<サービスプラットフォームのインストールディレクトリ>¥CSC¥schema¥soaprecp¥urecp_soap_attachment_response.xsd」に格納されています。

  名前空間名は「http://www.hitachi.co.jp/soft/xml/cosminexus/csc/reception/attachment/soap/response」です。

表 2-2　SOAP 受付のファイル連携用応答ヘッダフォーマット

タグ名	種別	出現回数	説明
<attachment-response>	−	1 回	−
<files>	−	0 または 1 回	応答する添付ファイルを示すタグです。
<file>	−	0 回以上	応答する 1 つの添付ファイルを示すタグです。
<content-id>	string	1 回	応答する添付ファイルの Content-Id ヘッダを指定します。 値が設定されていない場合，または null が設定された場合はエラーになります。
<content-type>	string	0 または 1 回	応答する添付ファイルの Content-Type ヘッダを指定します。 charset 属性には，Content-Type ヘッダの charset パラメタに対応する文字コードを指定します。 省略した場合，または値が空の場合は，"application/octet-stream"になります。
<local-file-name>	string	1 回	作業フォルダ内の中間ファイル名を指定します。 次の場合はエラーになります。 • パス変更に関連する指定（"/../"など）が含まれる場合 • 指定したファイル名が不正な場合

- adpsoap_attachment_request.xsd

  ビジネスプロセスから SOAP アダプタに添付ファイルに関する情報を渡すために使用します。スキーマは「<サービスプラットフォームのインストールディレクトリ>¥CSC¥schema¥soap¥adpsoap_attachment_request.xsd」に格納されています。

  名前空間名は「http://www.hitachi.co.jp/soft/xml/cosminexus/csc/adapter/attachment/soap/request」です。

表 2-3　SOAP アダプタのファイル連携用要求ヘッダフォーマット

タグ名	種別	出現回数	説明
<attachment-request>	−	1 回	−
<request-id>	string	0 または 1 回	受付で生成されたリクエスト ID を設定します。 作業フォルダ内のファイルデータを送信する場合，または SOAP レスポンス時にダウンロードファイルを出力するファイルとして，作業フォルダを使用する場合に設定が必要です。

タグ名	種別	出現回数	説明
<files>	−	0 または 1 回	SOAP リクエストで送信するファイル情報を指定します。
<file>	−	0 回以上	送信する 1 つのファイルに関する情報を指定します。
<content-id>	string	1 回	送信する添付ファイルの Content-Id ヘッダの値を指定します。 値が設定されていない場合，または null が設定された場合はエラーになります。
<content-type>	string	0 または 1 回	送信する添付ファイルの Content-Type ヘッダの値を指定します。 charset 属性には，Content-Type ヘッダの charset パラメタに対応する文字コードを指定します。 省略した場合，または値が空の場合は，local-file-name 要素に指定したファイルの拡張子によって自動的に設定されます。なお，システムによって自動生成された中間ファイルの場合，拡張子がないため"application/octet-stream"が設定されます。設定される値の詳細についてはマニュアル「アプリケーションサーバ Web サービス開発ガイド」の「28.4.2　添付ファイルから SOAP メッセージへのマッピングの注意事項（wsi:swaRef 形式）」を参照してください。
<input-folder-name>	string	1 回	送信対象ファイルが格納されているフォルダとして，次のどちらかを指定します。 • 作業フォルダの場合 空文字を指定します。 • 共通フォルダの場合 共通フォルダ定義名を指定します。 また，common 属性には，送信対象ファイルが存在するフォルダを識別するための次の値を指定します。 • 作業フォルダの場合 「common="false"」を指定します。 • 共通フォルダの場合 「common="true"」を指定します。 なお，共通フォルダ定義名が示すフォルダが不正な場合は読み込み時にエラーになります。
<local-file-name>	string	1 回	送信対象のファイル名を指定します。シンボリックリンクのファイルは指定しないでください。 • input-folder-name 要素で作業フォルダを指定した場合 作業フォルダ内の中間ファイル名を指定します。 • input-folder-name 要素で共通フォルダを指定した場合 共通フォルダルート配下のファイル名を相対パスで指定します。 また，次の場合はエラーが発生します。

タグ名	種別	出現回数	説明
<local-file-name>	string	1 回	• 指定した名称のファイルが存在しない場合 • パス変更に関係する指定（"/../"など）が含まれる場合 • 指定したファイル名が不正な場合
<output-folder-name>	string	0 または 1 回	SOAP レスポンス時にダウンロードファイルを出力するフォルダとして，次のどちらかを指定します。 • 作業フォルダの場合 空文字を指定します。 • 共通フォルダの場合 共通フォルダ定義名を指定します。 また，common 属性には，ダウンロードファイルを出力するフォルダを識別するための次の値を指定します。 • 作業フォルダの場合 「common="false"」を指定します。 • 共通フォルダの場合 「common="true"」を指定します。 なお，共通フォルダ定義名が示すフォルダが不正な場合は書き込み時にエラーが発生します。
<output-sub-folder>	string	0 または 1 回	SOAP レスポンス時にダウンロードファイルを出力するフォルダとして，共通フォルダのサブフォルダを使用する場合に共通フォルダからの相対パスを指定します。 output-folder-name 要素で作業フォルダを指定した場合は，この要素の値は無視されます。 なお，指定されたサブフォルダが不正な場合は書き込み時にエラーが発生します。

- adpsoap_attachment_response.xsd

  SOAP アダプタで生成された添付ファイルに関する XML 電文をビジネスプロセスで扱うために使用します。スキーマは「<サービスプラットフォームのインストールディレクトリ>¥CSC¥schema¥soap¥adpsoap_attachment_response.xsd」に格納されています。

  名前空間名は「http://www.hitachi.co.jp/soft/xml/cosminexus/csc/adapter/attachment/soap/response」です。

表 2-4　SOAP アダプタのファイル連携用応答ヘッダフォーマット

タグ名	種別	出現回数	説明
<attachment-response>	−	1 回	−
<request-id>	string	0 または 1 回	adpsoap_attachment_request.xsd の request-id 要素に指定した値が設定されます。 ダウンロードファイルを作業フォルダに出力した場合に出現します。
<files>	−	0 または 1 回	SOAP レスポンスで受信したファイル情報が設定されます。

タグ名				種別	出現回数	説明
		<file>		−	0 回以上	受信した 1 つのファイルに関する情報が設定されます。
			<content-id>	string	1 回	受信した添付ファイルの Content-Id ヘッダの値が設定されます。 Content-Id が設定されていない場合，または null の場合は SOAP アダプタでエラーになります。
			<content-type>	string	1 回	受信した添付ファイルの Content-Type ヘッダの値が設定されます。 charset 属性には，Content-Type ヘッダの charset パラメタに対応する文字コードが設定されます。
			output-folder-name	string	1 回	adpsoap_attachment_request.xsd の output-folder-name 要素に指定した値が設定されます。
			<local-file-name>	string	1 回	作業フォルダまたは共通フォルダに出力されたファイル名が設定されます。 • 出力フォルダが共通フォルダの場合 一意となる名称で出力されたファイル名が設定されます。また，adpsoap_attachment_request.xsd で output-sub-folder 要素が指定されていた場合は，サブフォルダのパス＋”/”＋ファイル名が設定されます。 • 出力フォルダが作業フォルダの場合 出力された中間ファイル名が設定されます。

(2) アップロード

Web サービスの添付ファイル機能では SOAP 受付および SOAP アダプタを利用してファイルのアップロードができます。

サービスリクエスタから SOAP 受付に対して添付ファイル付き SOAP メッセージを送信すると，SOAP 受付で作業フォルダが生成され添付ファイルが格納されます。作業フォルダに格納されたファイルをどう扱うかはビジネスプロセスで定義します。

作業フォルダまたは共通フォルダに格納されているファイルは，SOAP アダプタを利用することで Web サービスに対して送信できます。SOAP アダプタから Web サービスに対してファイルを送信するための設定はビジネスプロセスで定義します。

ここではサービスリクエスタから送信された添付ファイルをビジネスプロセスで扱うための設定方法，および Web サービスに添付ファイルを送信するためのビジネスプロセスの設定方法を説明します。

(a) 受付アクティビティ

サービスリクエスタから送信され，作業フォルダに格納されたファイルをビジネスプロセスで扱うには，urecp_soap_attachment_request.xsd のスキーマを指定して定義した変数を，ビジネスプロセスの受付アクティビティでヘッダ割当変数に設定します。

SOAP 受付に添付ファイル付き SOAP メッセージを送信した場合に SOAP 受付で作成される電文の例を次に示します。

```
<attachment-request xmlns="http://www.hitachi.co.jp/soft/xml/cosminexus/csc/reception/attachment/soap/request">
  <request-id>RCP01---010200030004201101021403050120000/HCSC</request-id>
  <files>
    <file>
      <content-id>d1cd6fed-368c-4966-9a20-375a120b3c05@jaxws.cosminexus.com</content-id>
      <content-type charset="UTF-8">text/plain</content-type>
      <local-file-name>csc_8216969970665213344</local-file-name>
    </file>
  </files>
</attachment-request>
```

(b) サービス呼出アクティビティ

SOAP アダプタを使って作業フォルダまたは共通フォルダに格納されているファイルをアップロードするには，adpsoap_attachment_request.xsd のスキーマを指定して定義した変数を，ビジネスプロセスのサービス呼出アクティビティで要求電文のヘッダ割当変数に設定します。

SOAP アダプタから添付ファイル付き SOAP メッセージを送信する場合に，ビジネスプロセスのデータ変換アクティビティで作成する電文の例を次に示します。

作業フォルダを使用する場合

```
<attachment-request xmlns="http://www.hitachi.co.jp/soft/xml/cosminexus/csc/adapter/attachment/soap/request">
  <request-id>RCP01---010200030004201101021403050120000/HCSC</request-id>
  <files>
    <file>
      <content-id>459d752f-5694-493b-9671-4d49af688cd8@jaxws.cosminexus.com</content-id>
      <content-type charset="UTF-8">text/plain</content-type>
      <input-folder-name common="false"/>
      <local-file-name>csc_8216969970665213344</local-file-name>
    </file>
  </files>
</attachment-request>
```

共通フォルダを使用する場合

```
<attachment-request xmlns="http://www.hitachi.co.jp/soft/xml/cosminexus/csc/adapter/attachment/soap/request">
  <files>
    <file>
      <content-id>459d752f-5694-493b-9671-4d49af688cd8@jaxws.cosminexus.com</content-id>
      <content-type charset="UTF-8">text/plain</content-type>
      <input-folder-name common="true">com001</input-folder-name>
      <local-file-name>sample.txt</local-file-name>
    </file>
  </files>
</attachment-request>
```

(3) ダウンロード

Web サービスの添付ファイル機能では SOAP 受付および SOAP アダプタを利用してファイルのダウンロードができます。

SOAP 受付定義ファイルの urecp-soap.switchover.file-transfer.mode で作業フォルダの生成の有無を設定できます。SOAP 受付定義ファイルの詳細についてはマニュアル「サービスプラットフォーム リファレンス」の「3.6.1　SOAP 受付定義ファイル」を参照してください。

ダウンロードする場合は次のケースを除き urecp-soap.switchover.file-transfer.mode に true を設定してください。true を設定しないと作業フォルダが生成されず，エラーになります。

- 共通フォルダを使用する
- 1 つのリクエストでサービスリクエスタからのアップロードも実施する

Web サービスから SOAP アダプタに対して添付ファイル付き SOAP メッセージを送信すると，作業フォルダまたは共通フォルダに添付ファイルが格納されます。添付ファイルを作業フォルダに格納するか共通フォルダに格納するか，また格納されたファイルをどう扱うかはビジネスプロセスで定義します。

作業フォルダに格納されているファイルは，SOAP 受付を利用することでサービスリクエスタに対して送信できます。SOAP 受付からサービスリクエスタに対してファイルを送信するための設定はビジネスプロセスで定義します。

ここでは Web サービスから受信した添付ファイルをビジネスプロセスで扱うための設定方法，およびサービスリクエスタに添付ファイルを送信するためのビジネスプロセスの設定方法を説明します。

(a) 受付アクティビティ

作業フォルダを使用する場合，SOAP 受付で生成されたリクエスト ID をビジネスプロセスで扱うには，urecp_soap_attachment_request.xsd のスキーマを指定して定義した変数を，ビジネスプロセスの受付アクティビティでヘッダ割当変数に設定します。

SOAP 受付で作業フォルダおよびリクエスト ID を生成した場合に SOAP 受付で作成される電文の例を次に示します。

```
<attachment-request xmlns="http://www.hitachi.co.jp/soft/xml/cosminexus/csc/reception/attachment/soap/request">
  <request-id>RCP01---010200030004201101021403050120000/HCSC</request-id>
</attachment-request>
```

(b) サービス呼出アクティビティ

Web サービスから送信された添付ファイルをビジネスプロセスで扱うには，adpsoap_attachment_request.xsd のスキーマを指定して定義した変数を，ビジネスプロセスのサービス呼出アクティビティで要求電文のヘッダ割当変数に設定して，adpsoap_attachment_response.xsd のス

キーマを指定して定義した変数を，ビジネスプロセスのサービス呼出アクティビティで応答電文のヘッダ割当変数に設定します。

SOAP アダプタで添付ファイル付き SOAP メッセージを受信する場合に，ビジネスプロセスのデータ変換アクティビティで作成する電文の例を次に示します。

作業フォルダを使用する場合

```
<attachment-request xmlns="http://www.hitachi.co.jp/soft/xml/cosminexus/csc/adapter/attac
hment/soap/request">
  <request-id>RCP01---0102000300042011010214030501200 0/HCSC</request-id>
  <output-folder-name common="false"/>
</attachment-request>
```

共通フォルダを使用する場合

```
<attachment-request xmlns="http://www.hitachi.co.jp/soft/xml/cosminexus/csc/adapter/attac
hment/soap/request">
  <output-folder-name common="true">com002</output-folder-name>
  <output-sub-folder>test01</output-sub-folder>
</attachment-request>
```

SOAP アダプタが添付ファイル付き SOAP メッセージを受信した場合に，SOAP アダプタで作成される電文の例を次に示します。

作業フォルダを使用する場合

```
<attachment-response xmlns="http://www.hitachi.co.jp/soft/xml/cosminexus/csc/adapter/atta
chment/soap/response">
  <request-id>RCP01---0102000300042011010214030501200 0/HCSC</request-id>
  <files>
    <file>
      <content-id>459d752f-5694-493b-9671-4d49af688cd8@jaxws.cosminexus.com</content-id>
      <content-type charset="UTF-8">text/plain</content-type>
      <output-folder-name common="false"/>
      <local-file-name>csc_8216969970665213344</local-file-name>
    </file>
  </files>
</attachment-response>
```

共通フォルダを使用する場合

```
<attachment-response xmlns="http://www.hitachi.co.jp/soft/xml/cosminexus/csc/adapter/atta
chment/soap/response">
  <files>
    <file>
      <content-id>154765f0-9850-4c1f-8f6c-3f1f9306b4c0@jaxws.cosminexus.com</content-id>
      <content-type charset="UTF-8">text/plain</content-type>
      <output-folder-name common="true">com002</output-folder-name>
      <local-file-name>test01/HCSCCLS1----HCSC1---1------------d6j93dx74g---1---</local-f
ile-name>
    </file>
  </files>
</attachment-response>
```

(c) 応答アクティビティ

作業フォルダに格納されている中間ファイルをサービスリクエスタに応答するには，urecp_soap_attachment_response.xsd のスキーマを指定して定義した変数を，ビジネスプロセスの応答アクティビティでヘッダ割当変数に設定します。

SOAP 受付から添付ファイル付き SOAP メッセージを送信する場合に，ビジネスプロセスのデータ変換アクティビティで作成する電文の例を次に示します。

```
<attachment-response xmlns="http://www.hitachi.co.jp/soft/xml/cosminexus/csc/reception/attachment/soap/response">
  <files>
    <file>
      <content-id>d1cd6fed-368c-4966-9a20-375a120b3c05@jaxws.cosminexus.com</content-id>
      <content-type charset="UTF-8">text/plain</content-type>
      <local-file-name>csc_8216969970665213344</local-file-name>
    </file>
  </files>
</attachment-response>
```

(4) 注意事項

添付ファイル機能の注意事項について説明します。また，マニュアル「アプリケーションサーバ Web サービス開発ガイド」も合わせてご確認ください。

(a) 作業フォルダ使用時の注意事項

作業フォルダ内のファイルは cscfswrm コマンドによって定期的に削除する必要があります。なお，リクエスト処理完了直後は削除できない場合があるため，削除できなかった場合は少し時間を空けてから再度削除を実施してください。

(b) 添付ファイルの MIME ヘッダ

添付ファイルの MIME ヘッダに設定されている Content-Id,，Content-Type 以外のヘッダは無視されます。添付ファイルに関する情報（ファイル名など）を送受信する際は，SOAP ボディなどに設定してください。

(c) 添付ファイル使用時のメモリ使用量

添付ファイル機能を使用する場合，添付ファイルのサイズに応じてメモリ使用量が増大します。詳細はマニュアル「アプリケーションサーバ Web サービス開発ガイド」の「付録 C.3　添付ファイル使用時の 1 リクエスト当たりのメモリ使用量」を参照してください。

2.3　SessionBean を使用したサービスとの接続

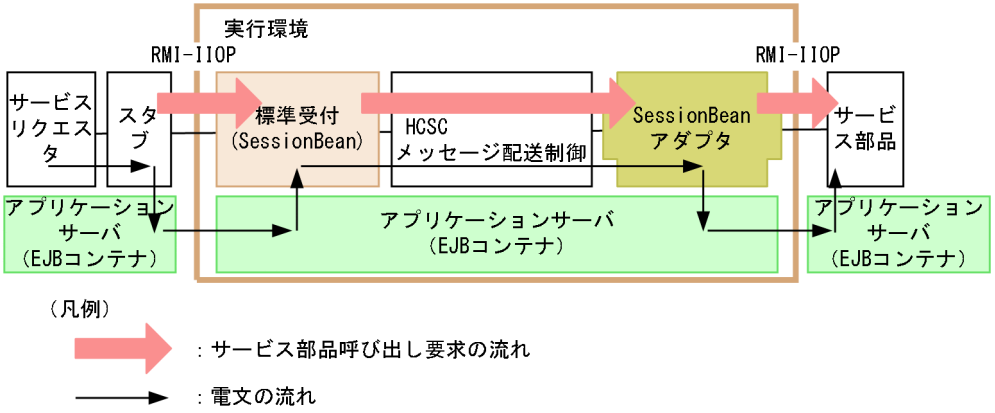
2.3.1　アプリケーションサーバ（EJB コンテナ）との関係

実行環境の HCSC サーバは，アプリケーションサーバの EJB コンテナ上で動作します。

HCSC サーバのリクエスト受付は，EJB（Stateless Session Bean）のリモートインターフェースとして提供され，RMI-IIOP で接続します。

HCSC サーバのサービスアダプタからは，EJB（Stateless Session Bean または Stateful Session Bean）で作成されたサービス部品が実行されます。アプリケーションサーバ（EJB コンテナ）との関係を次の図に示します。

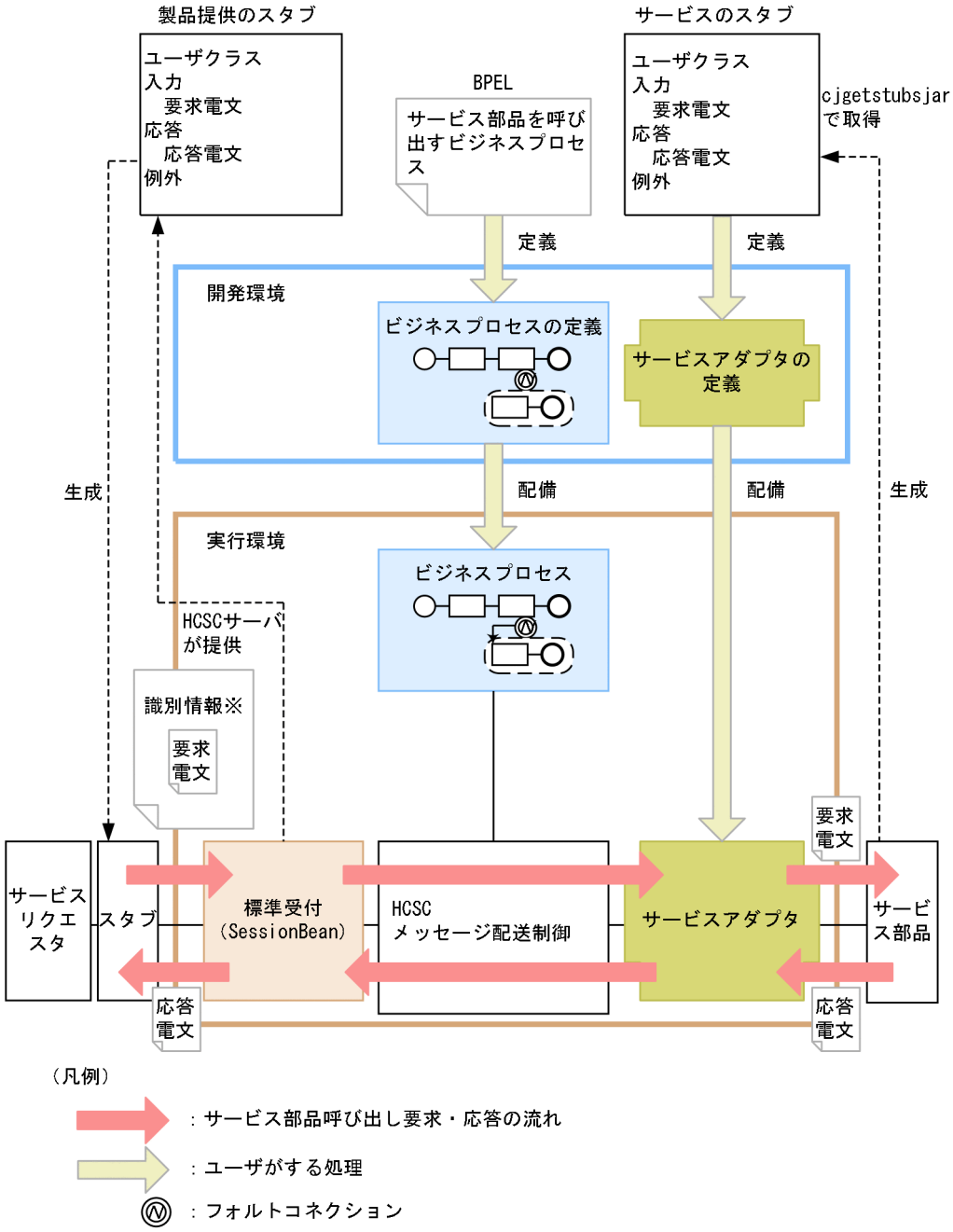
図 2-29　アプリケーションサーバ（EJB コンテナ）との関係



2.3.2　サービス部品呼び出し時のユーザ電文とサービス部品のスタブとの関係

SessionBean 使用時のユーザ電文とスタブの関係を次の図に示します。

図 2-30　SessionBean 使用時のユーザ電文とスタブの関係



注※　識別情報とは，サービス名，クライアント相関IDなどです。

サービスアダプタを定義するときは，サービス部品のスタブを定義します（サービス部品のスタブには，作成したサービス部品のユーザクラスが含まれています。また，要求電文，応答電文および例外の形式が定義されています）。

サービス部品側の J2EE サーバにアプリケーションサーバを使用している場合，サービス部品のスタブを取得するには，J2EE サーバの cjgetstubsjar コマンドを使用します。

サービスリクエスタからサービス部品を呼び出すときは，製品提供のスタブを使用します。製品提供のスタブは，標準受付（SessionBean）のインターフェースになっています。サービスリクエスタから標準受

付（SessionBean）を呼び出すときには，サービス部品側の電文フォーマットに合わせた要求電文を事前に作成し，標準受付（SessionBean）のパラメタ（ユーザ電文のパラメタ）に設定して実行します。

サービスリクエスタからビジネスプロセスを呼び出すときは，要求電文はビジネスプロセスの受付アクティビティに定義した電文フォーマットになります。サービスリクエスタ内で，ビジネスプロセスの受付アクティビティで定義されている要求電文の電文フォーマットに合わせた電文を作成し，作成した電文を標準受付（SessionBean）のパラメタ（ユーザ電文のパラメタ）に設定して実行します。

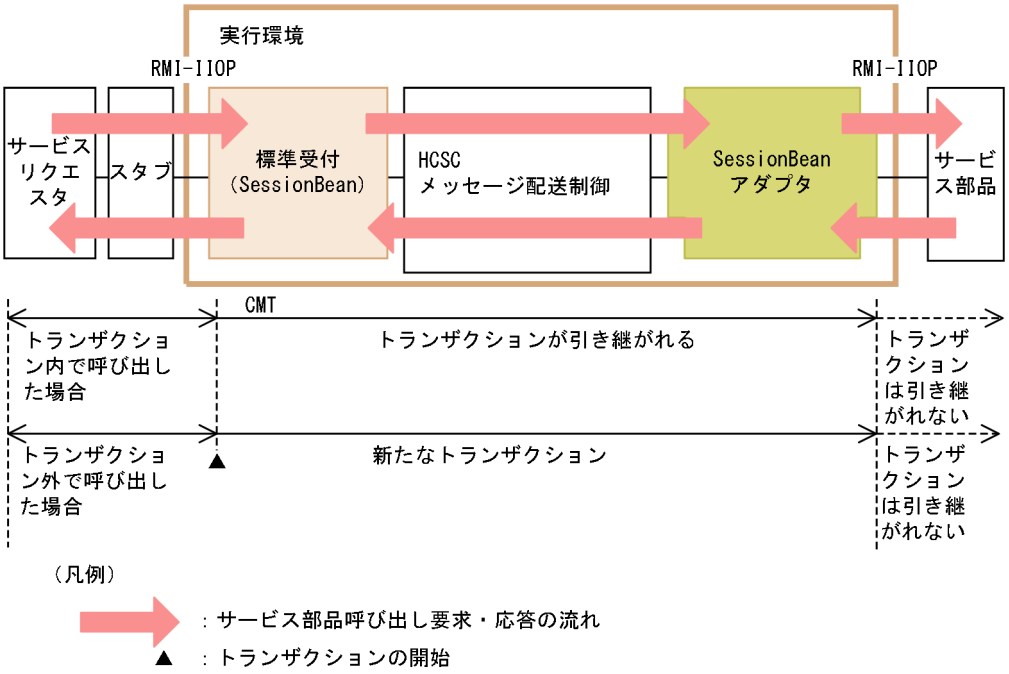
サービスリクエスタへの応答は，サービスアダプタで定義した応答電文の電文フォーマットを使用します。

cjgetstubsjar コマンドの詳細については，マニュアル「アプリケーションサーバ リファレンス コマンド編」の「cjgetstubsjar（アプリケーションの RMI-IIOP スタブおよびインタフェースの取得）」を参照してください。

2.3.3 SessionBean 使用時のトランザクション

SessionBean 使用時のトランザクションを次の図に示します。

図 2-31 SessionBean 使用時のトランザクション

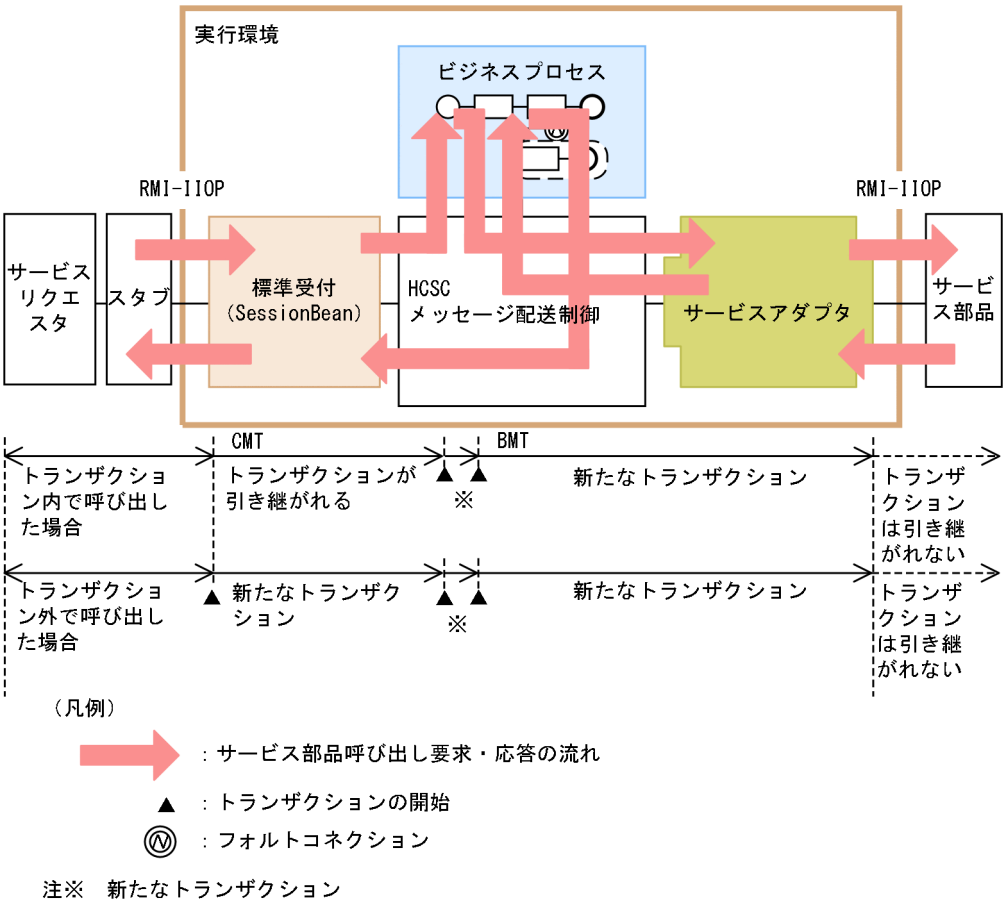


SessionBean 使用時，HCSC サーバ内は CMT（Container Managed Transaction）で動作します。また，標準受付（SessionBean）のトランザクション属性は Required 属性です。そのため，サービスリクエスタがトランザクションスコープ内で HCSC サーバの標準受付（SessionBean）を呼び出したとき，トランザクションコンテキストが HCSC サーバ側に引き継がれ，HCSC サーバ内の処理はサービスリクエスタ側のトランザクションスコープ内に入ります。また，サービスリクエスタがトランザクションスコープ外で HCSC サーバの標準受付（SessionBean）を呼び出したとき，HCSC サーバ側で新しいトランザクションが開始されます。

なお，サービス部品のトランザクション属性は RequiresNew または NotSupported でなければなりません。そのため，トランザクションは，サービス部品側には引き継がれません。SessionBean を利用したサービス部品のトランザクション属性を指定できるかどうかについては，マニュアル「サービスプラットフォーム 開発ガイド 基本開発編」の「2.6.2　SessionBean を利用したサービス部品の適用範囲」を参照してください。

ビジネスプロセスは BMT（Bean Managed Transaction）で動作します。ビジネスプロセス自体は新たなトランザクションで制御します。ビジネスプロセスを呼び出す場合は，ビジネスプロセスのサービス呼出アクティビティに定義したサービス部品呼び出し処理ごとに，新たなトランザクションで制御します。そのため，サービスリクエスタがトランザクションスコープ内で HCSC サーバの標準受付（SessionBean）を呼び出したときでも，トランザクションはサービスアダプタまで引き継がれません。ビジネスプロセスを介した場合のトランザクションを次の図に示します。

図 2-32　ビジネスプロセスを介した場合のトランザクション



永続化するビジネスプロセスの場合，ビジネスプロセスを実行する過程でのトランザクションについては，「3.4　ビジネスプロセスのトランザクション」を参照してください。

2.3.4　SessionBean アダプタ使用時の注意事項

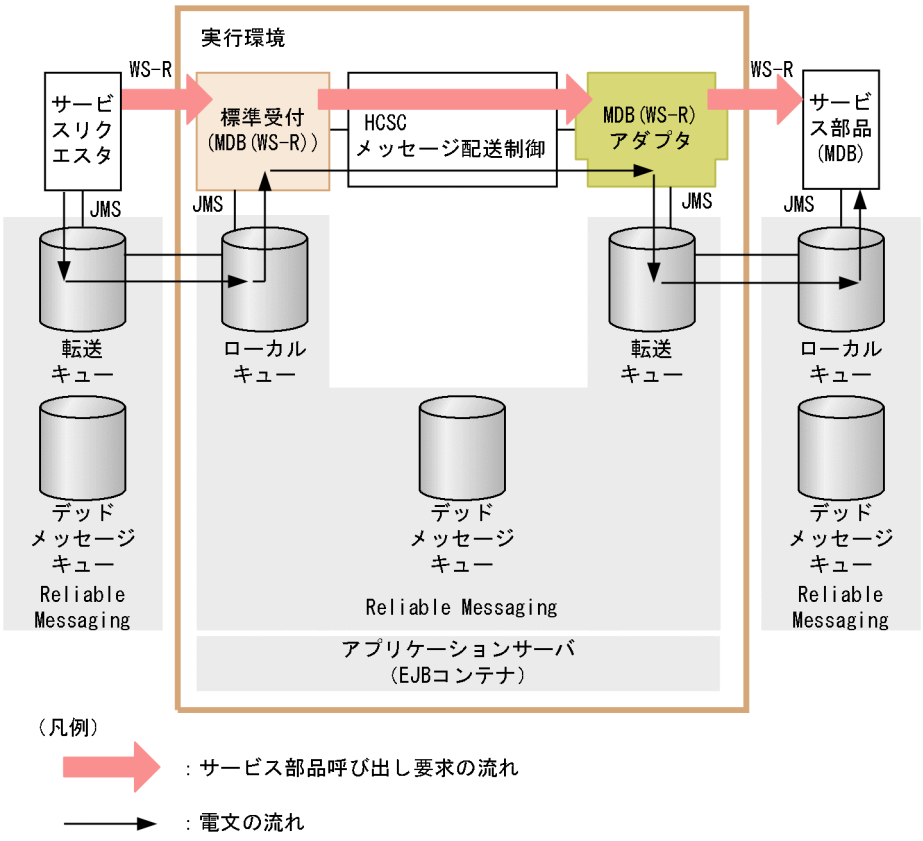
ネーミングのキャッシングを有効（ON）にした環境で SessionBean サービスを再起動すると，再起動後の初回の SessionBean サービス呼び出し時に KDEC03002-E のメッセージを出力してエラーとなる場合があります。このエラーを回避するには，次の対処をしてください。

- 参照系システムの場合：EJB ホームオブジェクトへの再接続機能を有効にしてください。
- 参照系以外のシステムの場合：HCSC サーバを再起動してください。

2.4.1　Reliable Messaging との関係

非同期のサービス部品を呼び出すには，Reliable Messaging を使用します。また，サービスリクエスタ側の転送キューと HCSC サーバ側のローカルキューの間，そして HCSC サーバ側の転送キューとサービス部品側のローカルキューの間は WS-R（WS-Reliability）を使用します。Reliable Messaging は，アプリケーションサーバの EJB コンテナ上で動作します。Reliable Messaging との関係を次の図に示します。

図 2-33　Reliable Messaging との関係



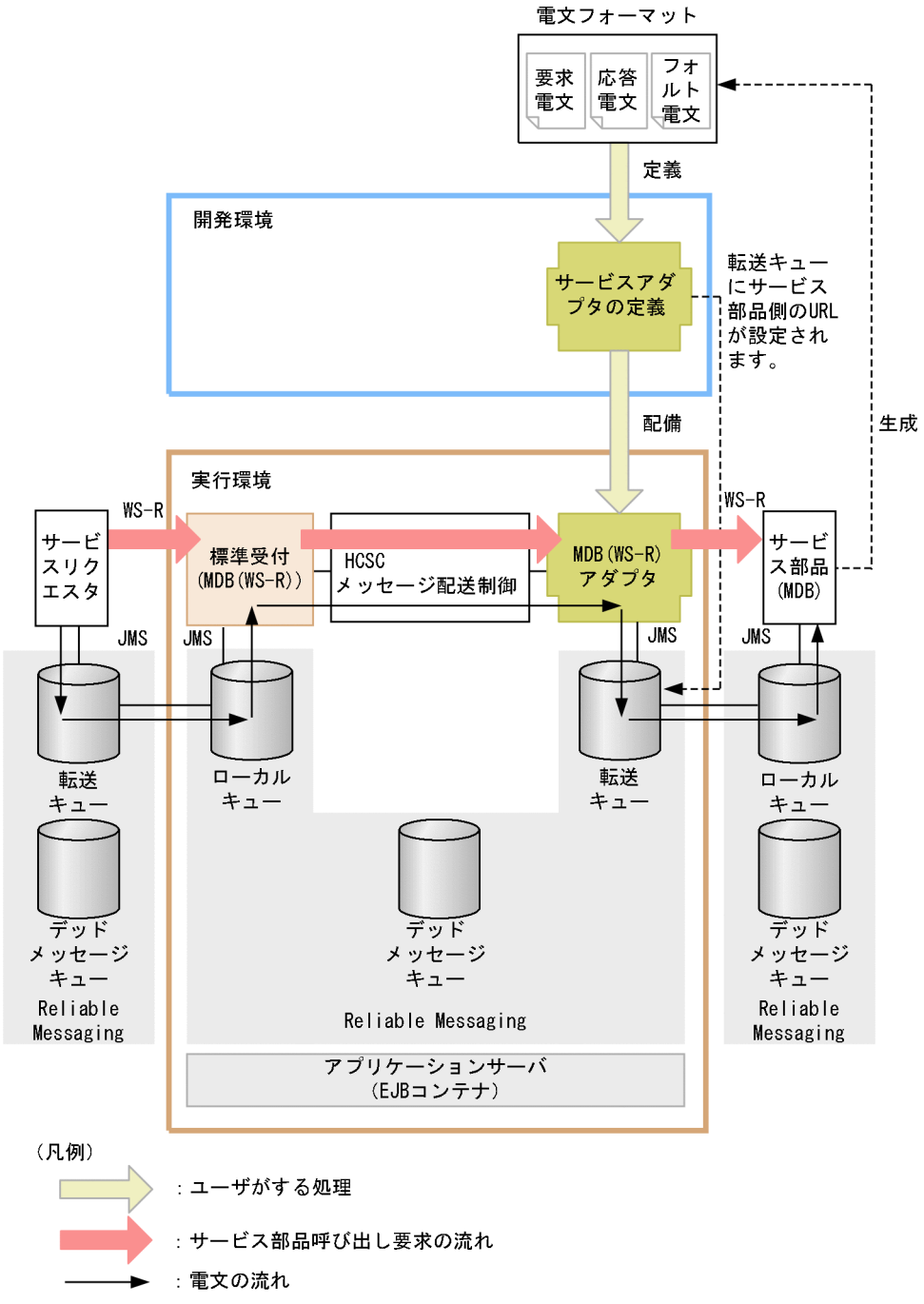
2.4.2　サービス部品呼び出し時のメッセージの流れ

HCSC サーバの標準受付（MDB（WS-R））は，MDB（Message Driven Bean）で動作し，HCSC サーバ側のローカルキューを監視しています。HCSC サーバ側のローカルキューにメッセージが送信されると，標準受付（MDB（WS-R））はメッセージを取り出してサービス部品を呼び出します。

サービスリクエスタ側の転送キューには，HCSC サーバの宛先アドレスおよび転送先のローカルキューのキュー名を設定します。サービスリクエスタからは，サービスリクエスタ側の転送キューに対して，JMS メッセージを送信して，HCSC サーバの標準受付（MDB（WS-R））を呼び出します。

サービス部品を呼び出すときには，サービスアダプタ側の転送キューに送信します。そして，転送キューは，開発環境のサービスアダプタ定義画面（基本画面のアドレスに指定した転送先のキュー名および詳細画面の宛先 URL）で定義したサービス部品側のローカルキューへ送信します。サービス部品を MDB で作成することで，サービス部品側のローカルキューに送信されたメッセージを取り出して処理できます。非同期（MDB（WS-R））の場合の宛先の指定を次の図に示します。

図 2-34　非同期（MDB（WS-R））の場合の宛先の指定

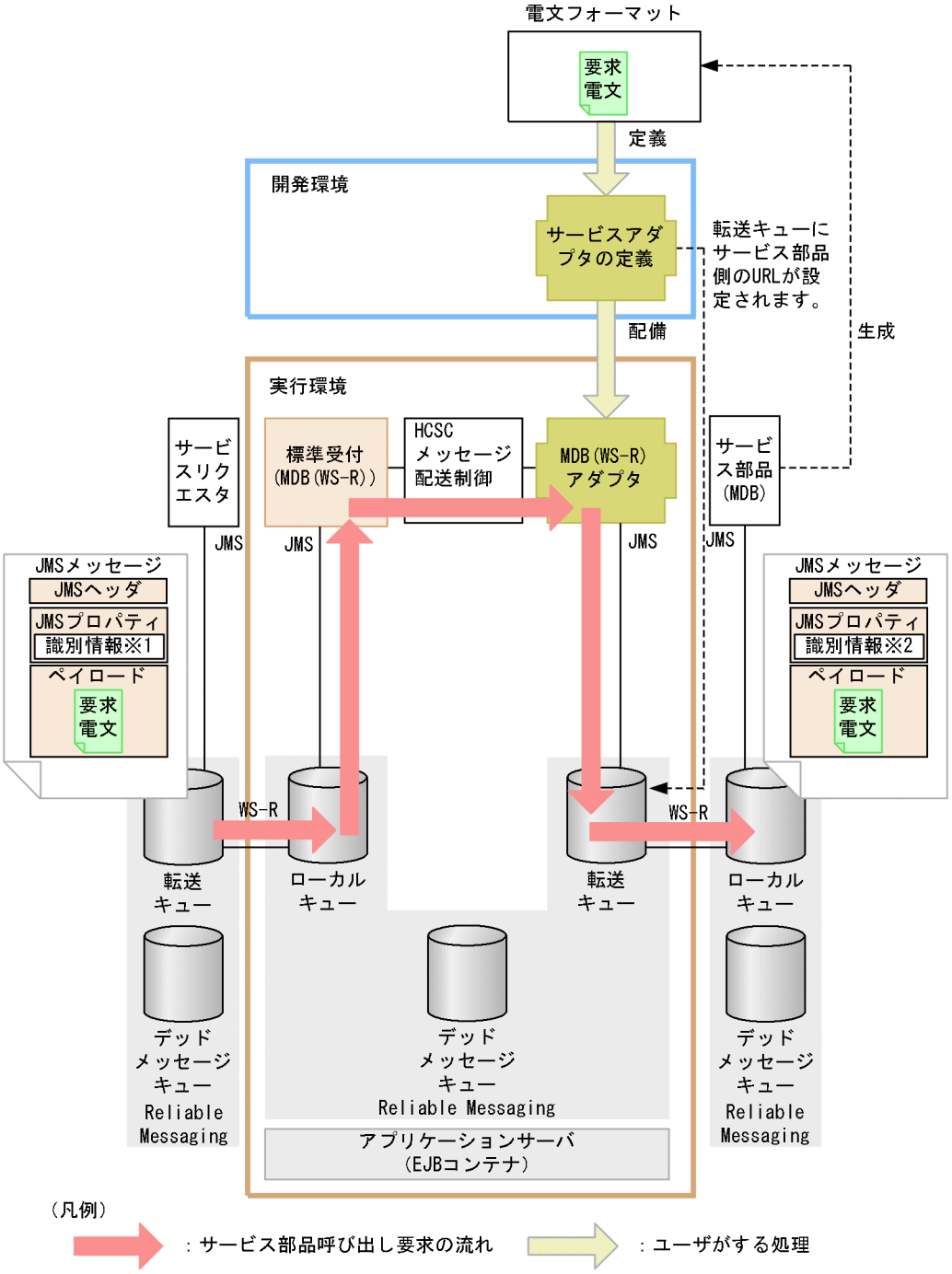


転送キューを作成する手順については，マニュアル「サービスプラットフォーム 開発ガイド 基本開発編」の「9.5.2　転送キューを作成する」を参照してください。JMS メッセージを生成する手順については，マニュアル「サービスプラットフォーム 開発ガイド 基本開発編」の「9.5.3　JMS メッセージを生成する」を参照してください。

サービスリクエスタからサービス部品を呼び出すときに転送キューに送信する JMS メッセージの形式は，標準受付（MDB（WS-R））のインターフェースになります。

JMS メッセージのペイロード（ヘッダを除いたデータ本体）に設定した内容が，ユーザ電文としてサービス部品に伝わります。そのため，サービスリクエスタから標準受付（MDB（WS-R））を呼び出すときに，サービス部品側の電文フォーマットに合わせた要求電文を事前に作成し，ペイロードに設定してメッセージを送信します。非同期サービス部品呼び出し時のユーザ電文と JMS メッセージの関係を次の図に示します。

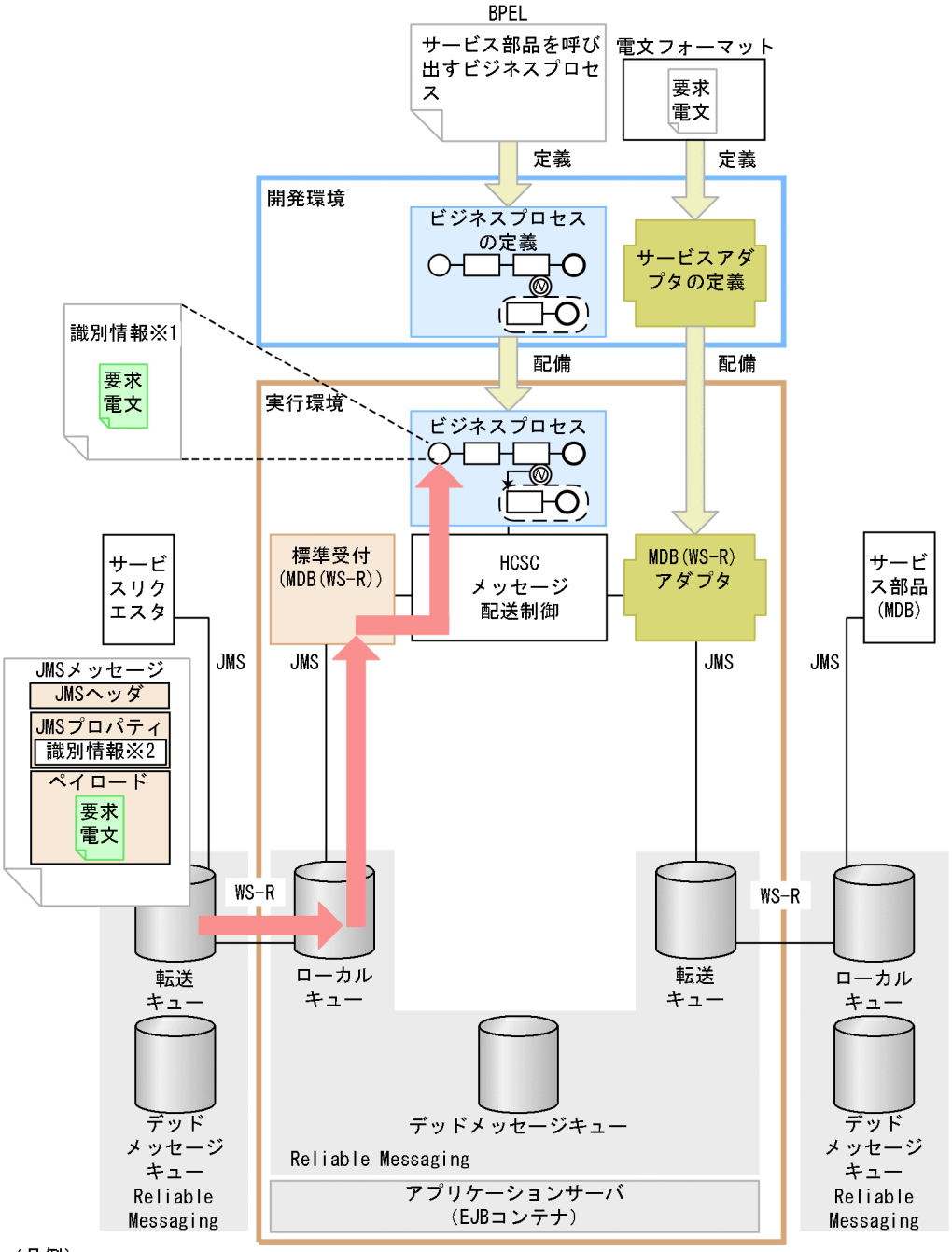
図 2-35　非同期サービス部品呼び出し時のユーザ電文と JMS メッセージの関係



注※1　識別情報とは，サービス名，クライアント相関ID，およびユーザ定義プロパティです。
注※2　識別情報とは，ユーザ定義プロパティです。

サービスリクエスタからビジネスプロセスを呼び出すときは，要求電文はビジネスプロセスの受付アクティビティに定義した電文フォーマットになります。サービスリクエスタ内でビジネスプロセスの受付アクティビティで定義されている要求電文の電文フォーマットに合わせた電文を作成し，ペイロードに設定してメッセージを送信します。ビジネスプロセスを呼び出す場合のユーザ電文と JMS メッセージの関係を次の図に示します。

図 2-36　ビジネスプロセスを呼び出す場合のユーザ電文と JMS メッセージの関係



注※1　識別情報とは，サービス名，クライアント相関IDなどです。
注※2　識別情報とは，サービス名，クライアント相関ID，およびユーザ定義プロパティです。

2.4.3　同期のサービス部品を呼び出す場合の応答電文の扱い

標準受付（MDB（WS-R））は，非同期の要求電文用の標準受付で，基本的にサービス部品からの応答がない場合に使用します。しかし，非同期（MDB（WS-R））の標準受付から同期のサービス部品（Web サービスまたは SessionBean など）やビジネスプロセスを要求した場合，同期のサービス部品から応答が

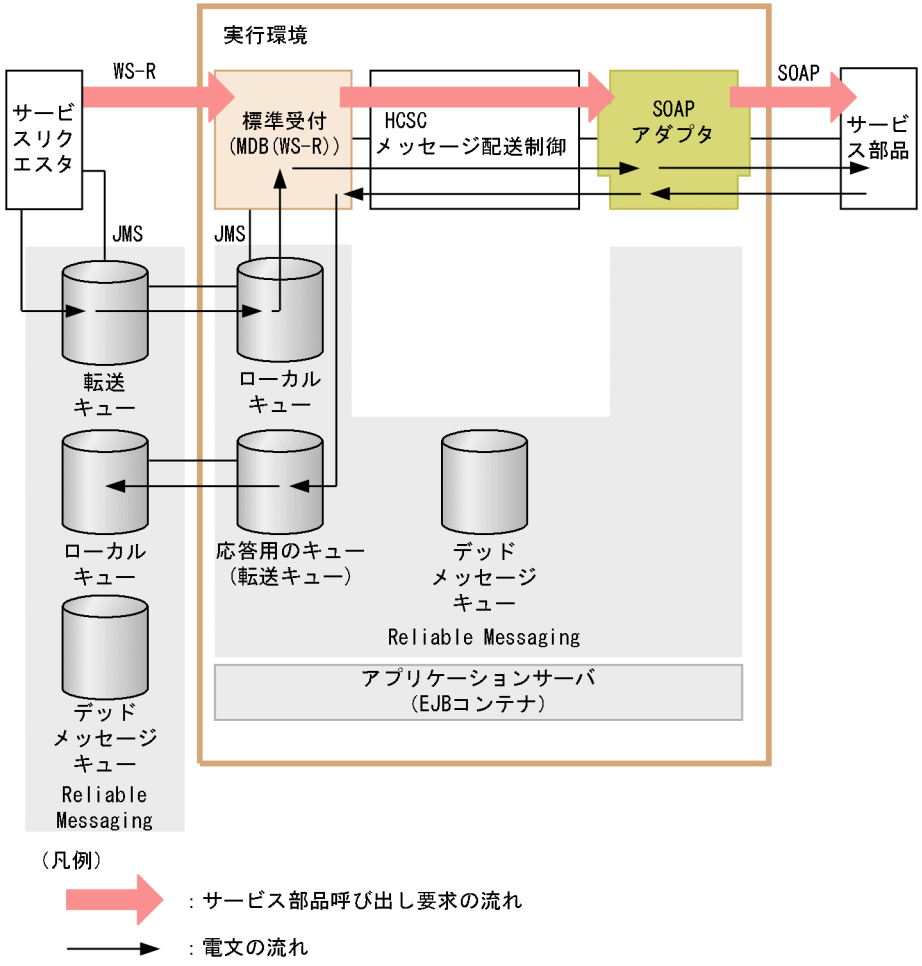
返ることがあります。そのため，HCSC サーバでは応答を応答用のキューに送信する手段を提供しています。

応答用のキューを利用するには，HCSC サーバの環境構築時に HCSC サーバ側に転送キュー（応答用のキュー）を作成し，転送キューにサービスリクエスタ側マシンのキュー間転送用 Web アプリケーションの宛先アドレスおよび転送先キュー名を指定しておきます。

サービスリクエスタから標準受付（MDB（WS-R））に対してサービス部品の呼び出しを要求するとき，標準受付（MDB（WS-R））のパラメタに作成した転送キュー（応答用のキュー）のキュー名を指定すると，サービス部品からの応答が返ってきたときに，指定されたキューに JMS メッセージが送信されます。

転送キュー（応答用のキュー）に送信された JMS メッセージは，サービスリクエスタ側のローカルキューに転送されるため，ローカルキューからメッセージを取り出すことで，サービスリクエスタは，サービス部品からの応答電文を取得できます。同期サービス部品を呼び出した場合の応答の扱いを次の図に示します。

図 2-37　同期サービス部品を呼び出した場合の応答の扱い



応答を取り出す方法については，マニュアル「サービスプラットフォーム 開発ガイド 基本開発編」の「9.5.8　応答を取り出す」を参照してください。

同期サービス部品でも応答があるかどうかによって，次の表に示すように応答用のキューへメッセージを送信するかどうかが異なります。

なお，サービスリクエスタから応答用のキューのキュー名を指定していても，HCSC サーバ側に指定したキュー名のキューが作られていない場合は，応答用のキューにメッセージは送信できないため，サービス部品の呼び出し処理がエラーになります。

表 2-5　応答用のキューへメッセージを出力する場合

項番	応答用のキューへ出力する場合	応答電文の有無			
		応答電文がある場合	応答電文がない場合	サービス部品からのユーザ定義例外の場合	ユーザ定義例外以外のエラーの場合
1	呼び出し先のサービス部品が同期サービス部品の場合 （WebService，SessionBean など）	ペイロードに応答電文を設定して送信	ペイロードなしのメッセージを送信	ペイロードに応答電文を設定して送信	×※
2	呼び出し先のサービス部品が非同期サービス部品の場合 （MDB（WS-R）および MDB（DB キュー）など）	×	×	－	×※
3	呼び出し先のサービス部品がビジネスプロセスの場合	ペイロードに応答電文を設定して送信	ペイロードなしのメッセージを送信	ペイロードに応答電文を設定して送信	×※

（凡例）

×：メッセージを送信しません。

－：非同期のサービス部品からユーザ定義例外は存在しないため該当しません。

注※

エラーの情報はログおよびトレースに出力します。

サービスリクエスタからの要求で応答用のキューのキュー名を指定していない場合は，応答電文の内容は破棄されます（応答電文が破棄されると，KDEC00031-W メッセージが出力されます）。

また，サービスリクエスタ側で相関関係を対応づけるために，サービスリクエスタが指定した JMS プロパティが引き継がれます。引き継がれるヘッダおよびプロパティの詳細については，マニュアル「サービスプラットフォーム 開発ガイド 基本開発編」の「9.5.8　応答を取り出す」を参照してください。

2.4.4　MDB（WS-R）使用時のトランザクション

MDB（WS-R）の場合，サービスリクエスタから HCSC サーバの間，そしてサービスアダプタからサービス部品の間は，異なるトランザクションになります。

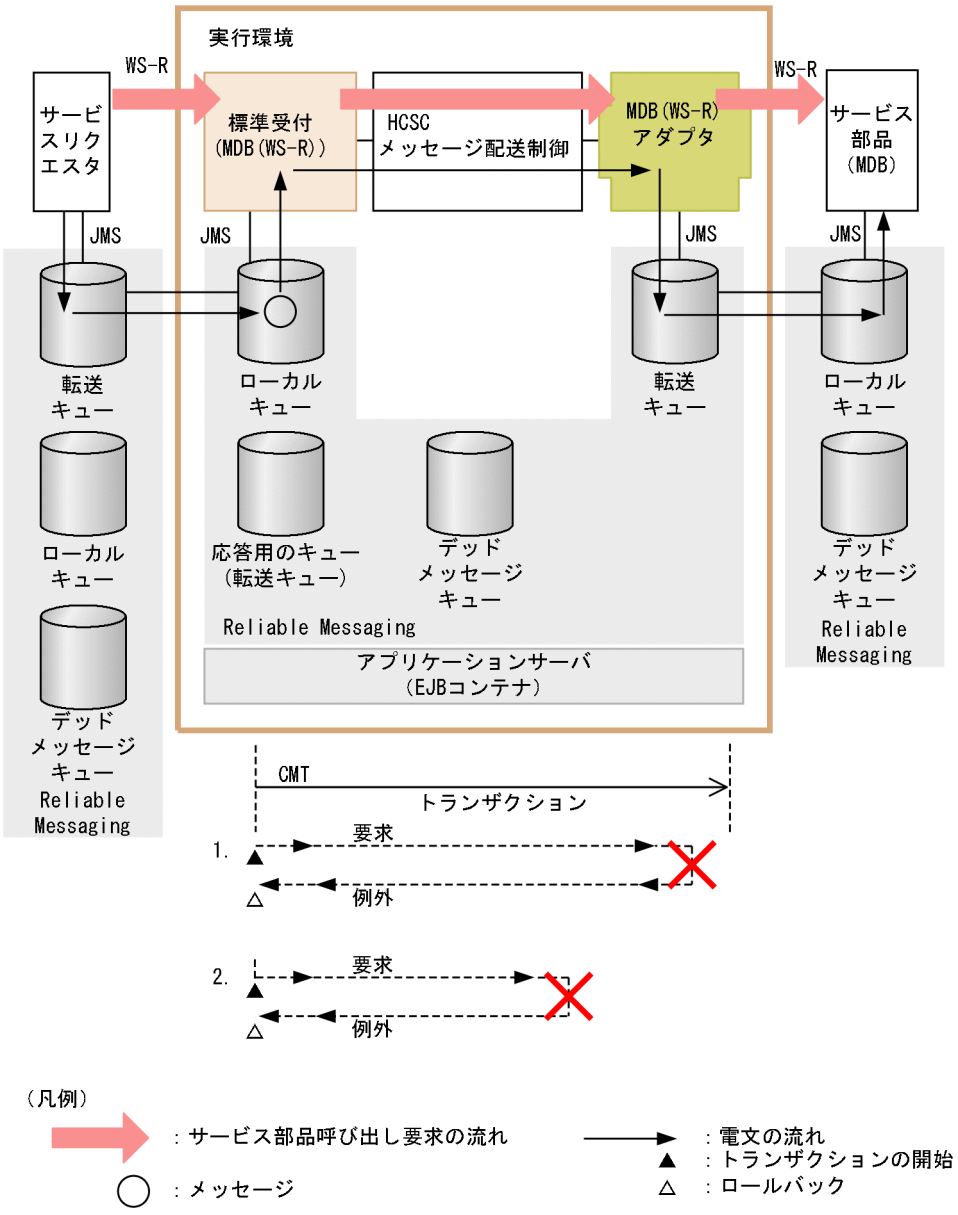
(1)　非同期のサービスアダプタへメッセージを送信する場合

非同期の標準受付（MDB（WS-R））でサービス部品呼び出し要求を受け付け，非同期の MDB（WS-R）アダプタへメッセージを送信する場合，標準受付（MDB（WS-R））でのローカルキューからのメッセージの取り出しとサービスアダプタでの転送キューへの送信は，1 つのトランザクションで処理します。

転送キューへのメッセージ送信が失敗した場合や，指定したサービスアダプタが起動していない場合など，何らかのエラーが発生した場合は，トランザクションをロールバックします。

ロールバックすると，標準受付（MDB（WS-R））で取り出したメッセージが復元され，再度ローカルキューからメッセージを取り出して，サービス部品を呼び出します。受付からサービス部品呼び出しまでが非同期の場合のトランザクションについて次の図に示します。

図 2-38　受付からサービス部品呼び出しまでが非同期の場合のトランザクション



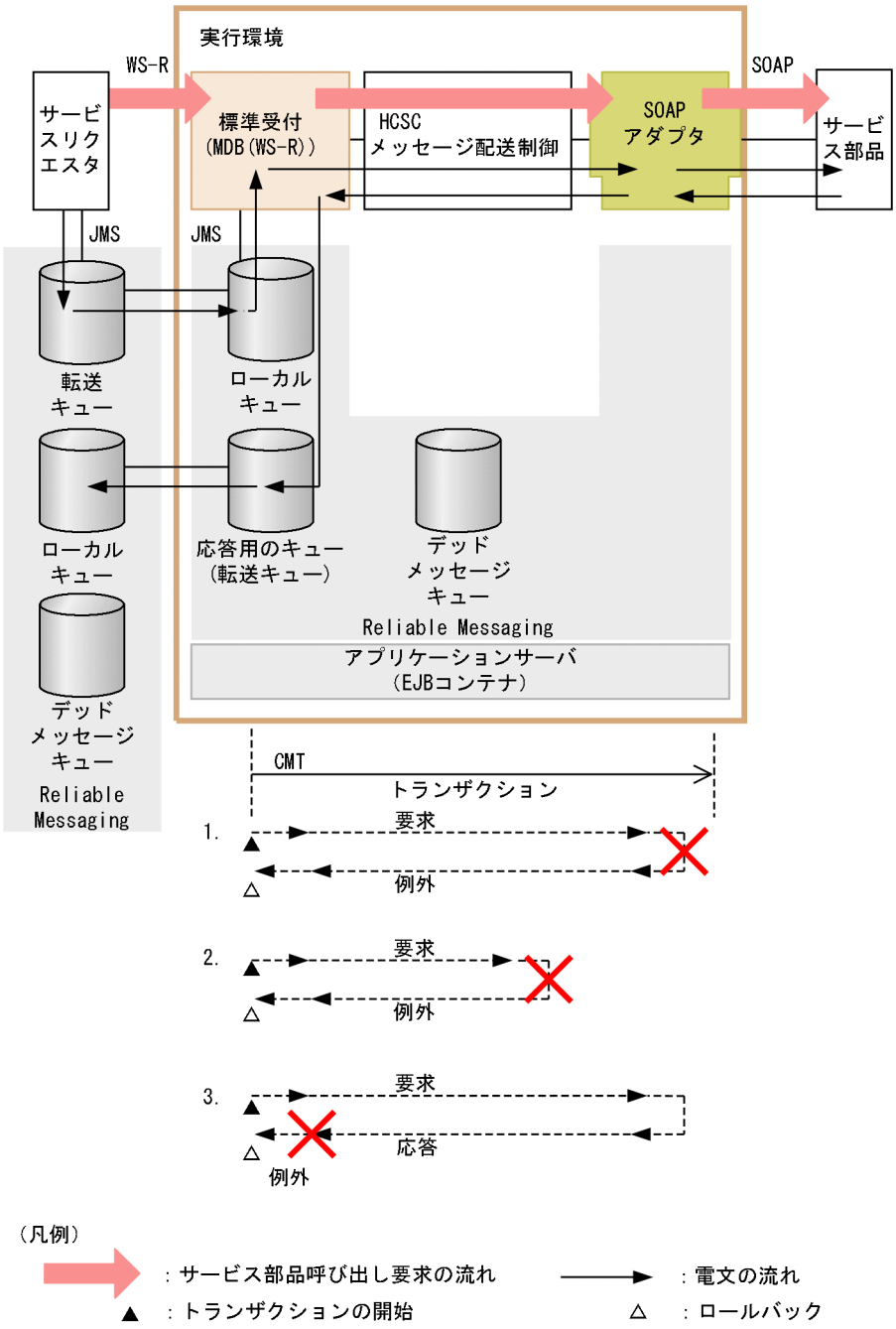
エラーが発生したときの対処方法については，マニュアル「サービスプラットフォーム システム構築・運用ガイド」の「7.7.3　MDB（WS-R）実行時の障害対策」を参照してください。

(2)　同期のサービスアダプタへメッセージを送信する場合

非同期（MDB（WS-R））の標準受付でサービス部品呼び出し要求を受け付け，同期のサービスアダプタへメッセージを送信する場合，標準受付（MDB（WS-R））でのローカルキューからのメッセージの取り出しからサービスアダプタでのサービス部品呼び出し，および応答が返ってきて応答用のキューへメッセージを送信するまでを，1 つのトランザクションで処理します。

転送キューへのメッセージの送信が失敗した場合や，指定したサービスアダプタが起動していない場合，応答用のキューへのメッセージ送信が失敗した場合など，何らかのエラーが発生した場合は，トランザクションをロールバックします。ロールバックすると，標準受付（MDB（WS-R））で取り出したメッセージが復元され，再度ローカルキューからメッセージを取り出して，サービス部品を呼び出します。受付が非同期であるがサービス部品が同期の場合のトランザクションについて次の図に示します。

図 2-39　受付が非同期であるがサービス部品が同期の場合のトランザクション



トランザクションが失敗すると，次のようになります。番号は，図中の番号と対応しています。

1. 転送キューへのメッセージの送信が失敗した場合

   サービスアダプタから SOAP でのサービス部品呼び出しが失敗すると，ローカルキューにメッセージが戻り，再度標準受付（MDB（WS-R））でメッセージを取り出します。

2. 指定したサービスアダプタが起動していない場合

   指定したサービスアダプタが起動していないと，ローカルキューにメッセージが戻り，再度標準受付（MDB（WS-R））でメッセージを取り出します。

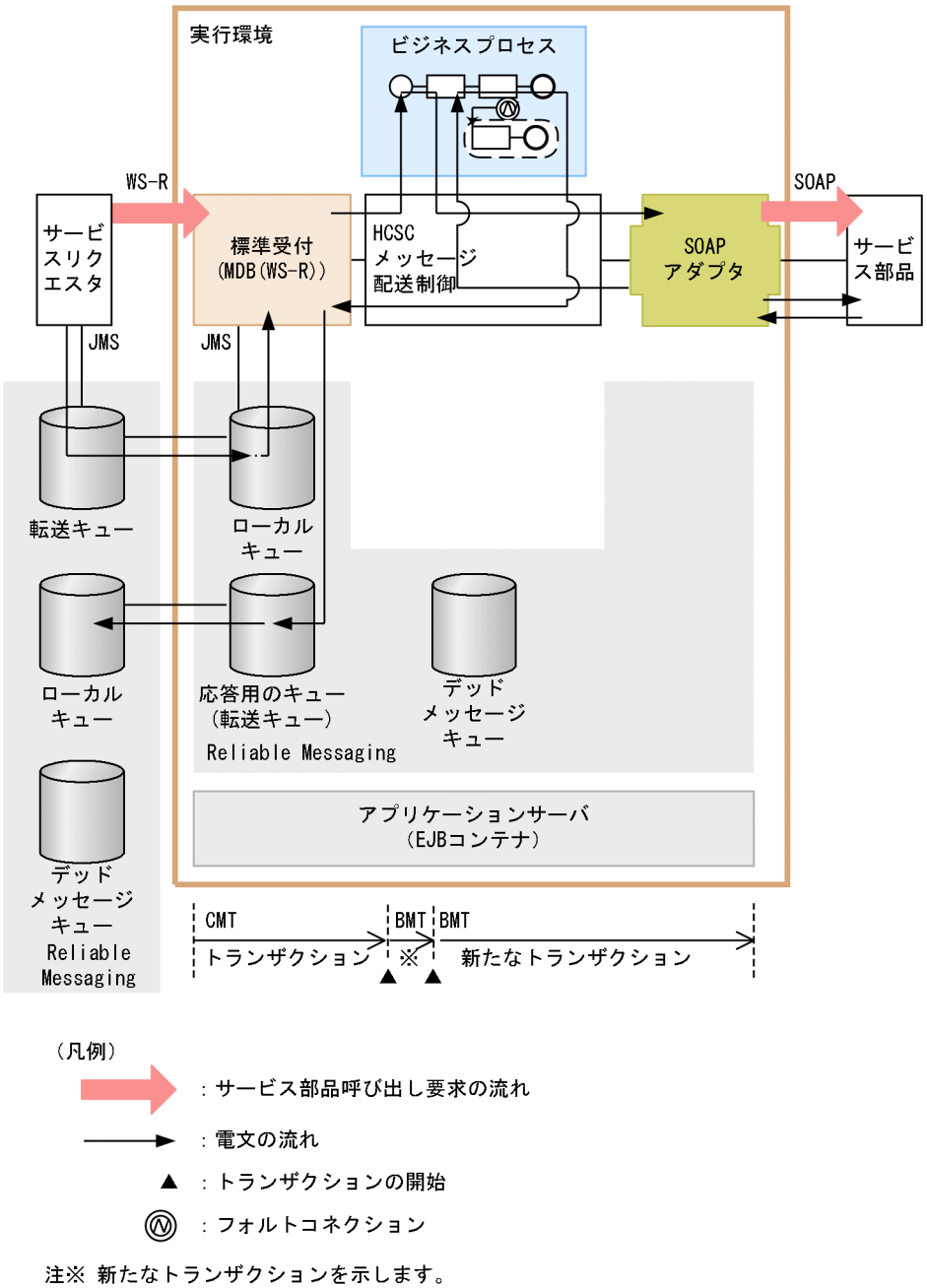
3. 応答用のキューへのメッセージ送信が失敗した場合

応答が返ってきて応答用のキューへのメッセージ送信が失敗すると，ローカルキューにメッセージが戻り，再度標準受付（MDB（WS-R））でメッセージを取り出します。

エラーが発生したときの対処方法については，マニュアル「サービスプラットフォーム システム構築・運用ガイド」の「7.7.3　MDB（WS-R）実行時の障害対策」を参照してください。

ビジネスプロセスは BMT（Bean Managed Transaction）で動作します。ビジネスプロセス自体は新たなトランザクションで制御します。ビジネスプロセスを呼び出す場合は，ビジネスプロセスのサービス呼出アクティビティに定義したサービス部品呼び出し処理ごとに，新たなトランザクションで制御します。ビジネスプロセスを介した場合のトランザクションについて次の図に示します。

図 2-40　ビジネスプロセスを介した場合のトランザクション



（凡例）

：サービス部品呼び出し要求の流れ

：電文の流れ

▲：トランザクションの開始

Ⓝ：フォルトコネクション

注※ 新たなトランザクションを示します。

永続化するビジネスプロセスの場合，ビジネスプロセスを実行する過程でのトランザクションについては，「3.4　ビジネスプロセスのトランザクション」を参照してください。

2.4.5　デッドメッセージキュー

HCSC サーバ稼働マシンでのサービス部品の呼び出しがロールバックして，ロールバックの回数（メッセージの配送回数）が最大値に達した場合や，配送回数に無制限の設定をしている場合，そのメッセージはデッドメッセージキューに遷移します。

エラーが発生したときにデッドメッセージキューに遷移する仕組みについては，マニュアル「サービスプラットフォーム システム構築・運用ガイド」の「7.7.3　MDB（WS-R）実行時の障害対策」を参照してください。

2.4.6　MDB（WS-R）で使用するキューの名称と個数

HCSC サーバの実行環境上では，次の表に示すキューを使用します。

表 2-6　MDB（WS-R）で使用するキューの名称と個数

項番	用途	説明	キューの種類	キューの名称	個数	備考
1	非同期（MDB（WS-R））の標準受付用	サービスリクエスタ側の転送キューの宛先となるキュー。	ローカルキュー	• 単独構成およびロードバランスクラスタの場合 CSC<HCSC サーバ名>ACPT_RCVQ • HA クラスタの場合 CSC<クラスタ名>ACPT_RCVQ	1 個	－
2	MDB（WS-R）アダプタ用	サービス部品側のローカルキューへ転送するキュー。	転送キュー	CSC_<クラスタ名><アダプタ名>	非同期の MDB（WS-R）アダプタ数	－
3	デッドメッセージキュー	MDB（DB キュー）用のデッドメッセージキューと共用。	ローカルキューまたは転送キュー	任意の名称	システムに 1 個	DB キューでのデッドメッセージキューと同一。
4	応答用のキュー	サービス部品呼び出しの応答を受け取る場合に使用。	ローカルキューまたは転送キュー	任意の名称	任意の個数	－

（凡例）

－：該当しません。

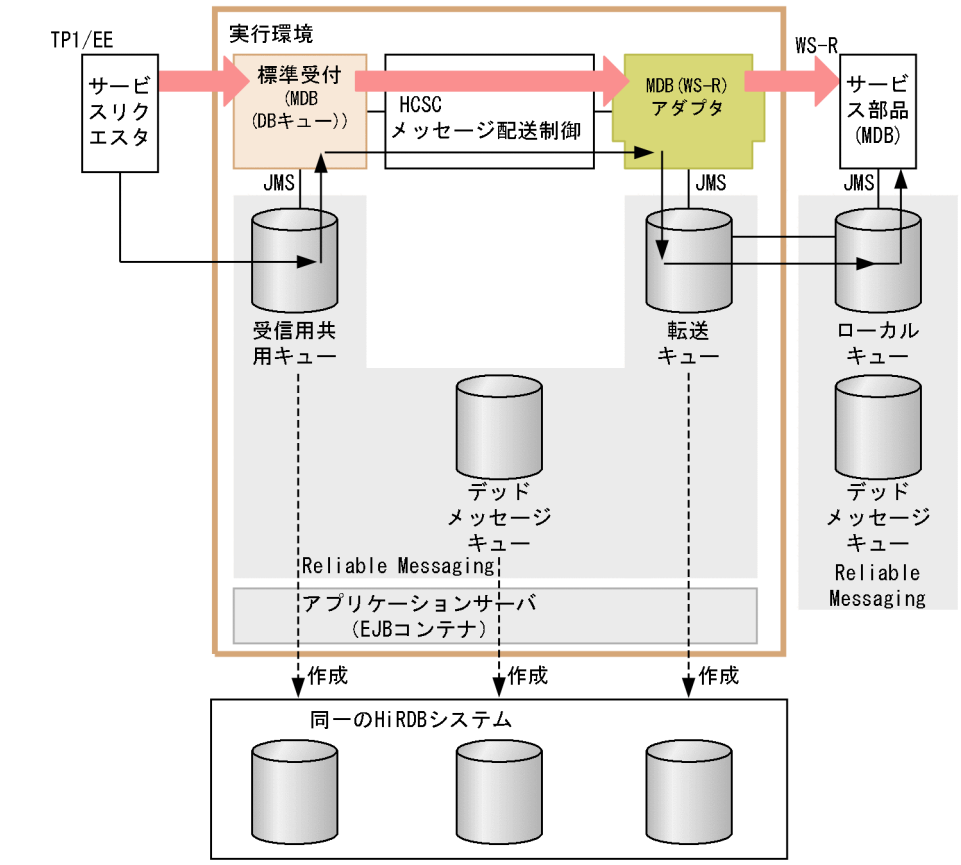
2.5.1　Reliable Messaging との関係

DB キューを使った非同期のサービス部品呼び出しでも，Reliable Messaging を使用します（通常は，TP1/EE と連携するために使用します。TP1/EE 側では DB キューという名前で呼びます）。

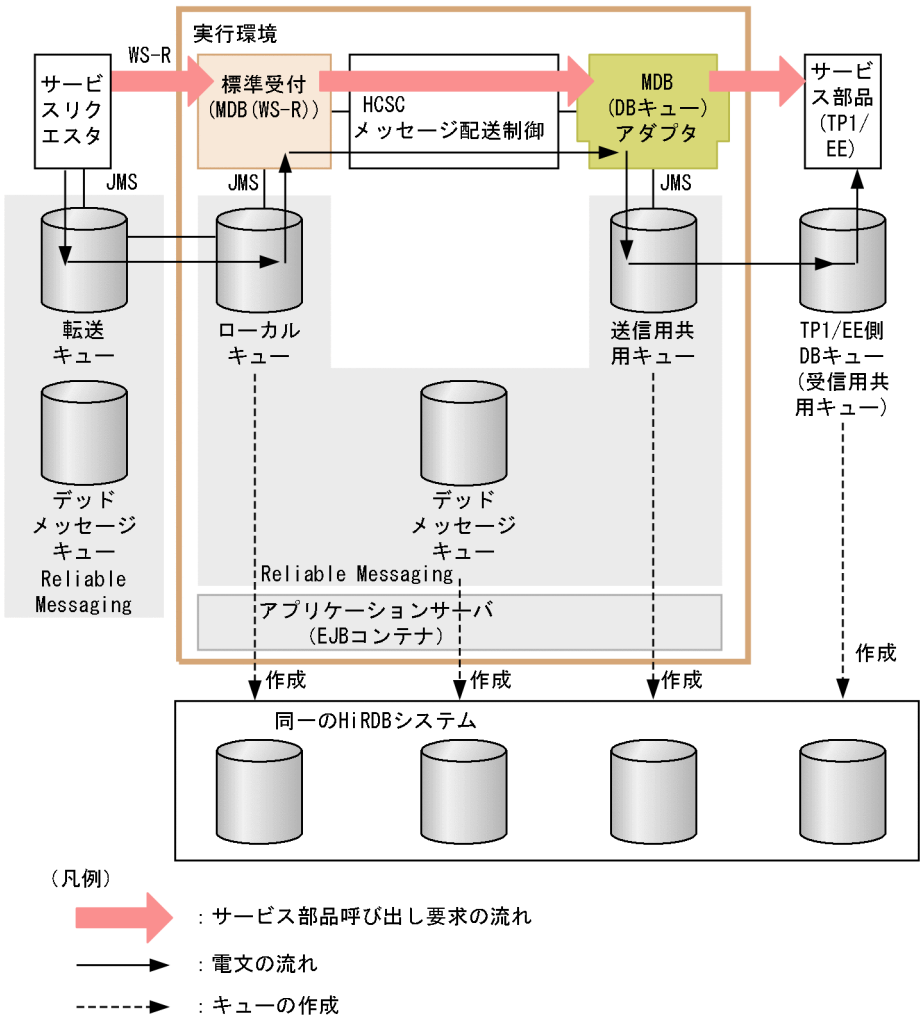
サービスリクエスタ側の TP1/EE とのインターフェースには受信用共用キューを，サービス部品側の TP1/EE とのインターフェースには送信用共用キューを使用します。

なお，非同期（MDB（DB キュー））の標準受付から，非同期の MDB（DB キュー）アダプタを呼び出すことはできません。必ず異なるプロトコルのリクエスト受付またはサービスアダプタと組み合わせて使用する必要があります。Reliable Messaging との関係を次の図に示します。

図 2-41　Reliable Messaging との関係



●サービス部品側がTP1/EEの場合



注意事項

実行環境内と HiRDB に別々のキューがそれぞれ作成されるわけではありません。データベース上のキューは同一のシステム上に作成されるため，実行環境内のキューと HiRDB のキューは同じものになります。TP1/EE 側の DB キューも同様です。

2.5.2　サービス部品呼び出し時のメッセージの流れ

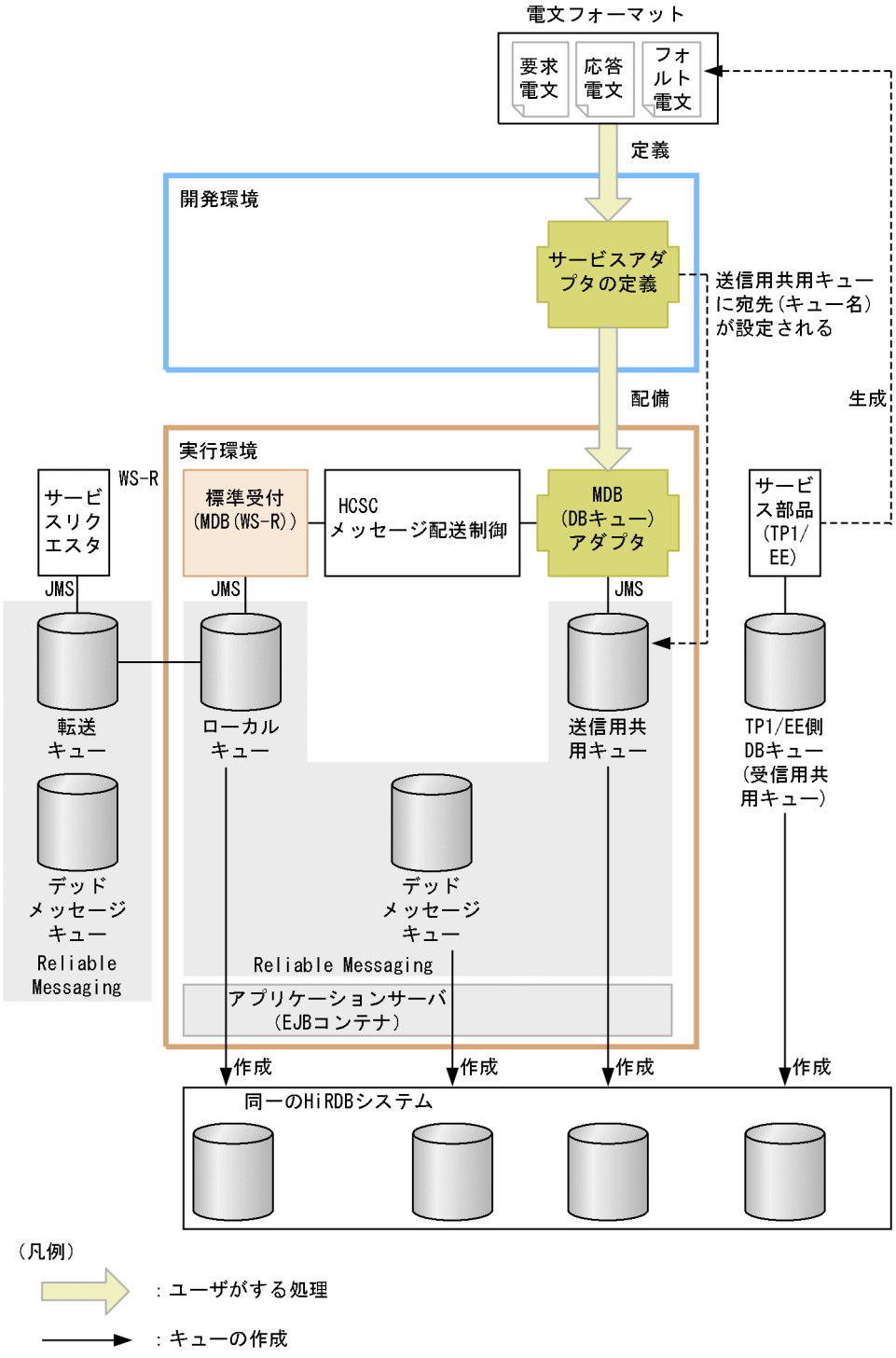
HCSC サーバのリクエスト受付は，MDB（Message Driven Bean）で動作し，HCSC サーバ側の受信用共用キューを監視しています。HCSC サーバ側の受信用共用キューにメッセージが送信されると，メッセージを取り出してサービス部品を呼び出します。

TP1/EE 側のサービスリクエスタからは，受信用共用キューがあるデータベースに対して，直接アクセスしてメッセージを送信します。

サービスリクエスタ側の送信用共用キューには，宛先となるキュー名を設定します（開発環境のサービスアダプタ定義画面で設定します）。

宛先となる TP1/EE 側の DB キュー（受信用共用キュー）は，TP1/EE のコマンドを使用して作成します。なお，TP1/EE 側の DB キューは，送信用共用キューがある同一データベース内（同一の HiRDB システム内）に作成する必要があります。非同期（MDB（DB キュー））の場合の宛先の指定について次の図に示します。

図 2-42　非同期（MDB（DB キュー））の場合の宛先の指定



注意事項

実行環境内と HiRDB に別々のキューがそれぞれ作成されるわけではありません。データベース上のキューは同一のシステム上に作成されるため，実行環境内のキューと HiRDB のキューは同じものになります。TP1/EE 側の DB キューも同様です。

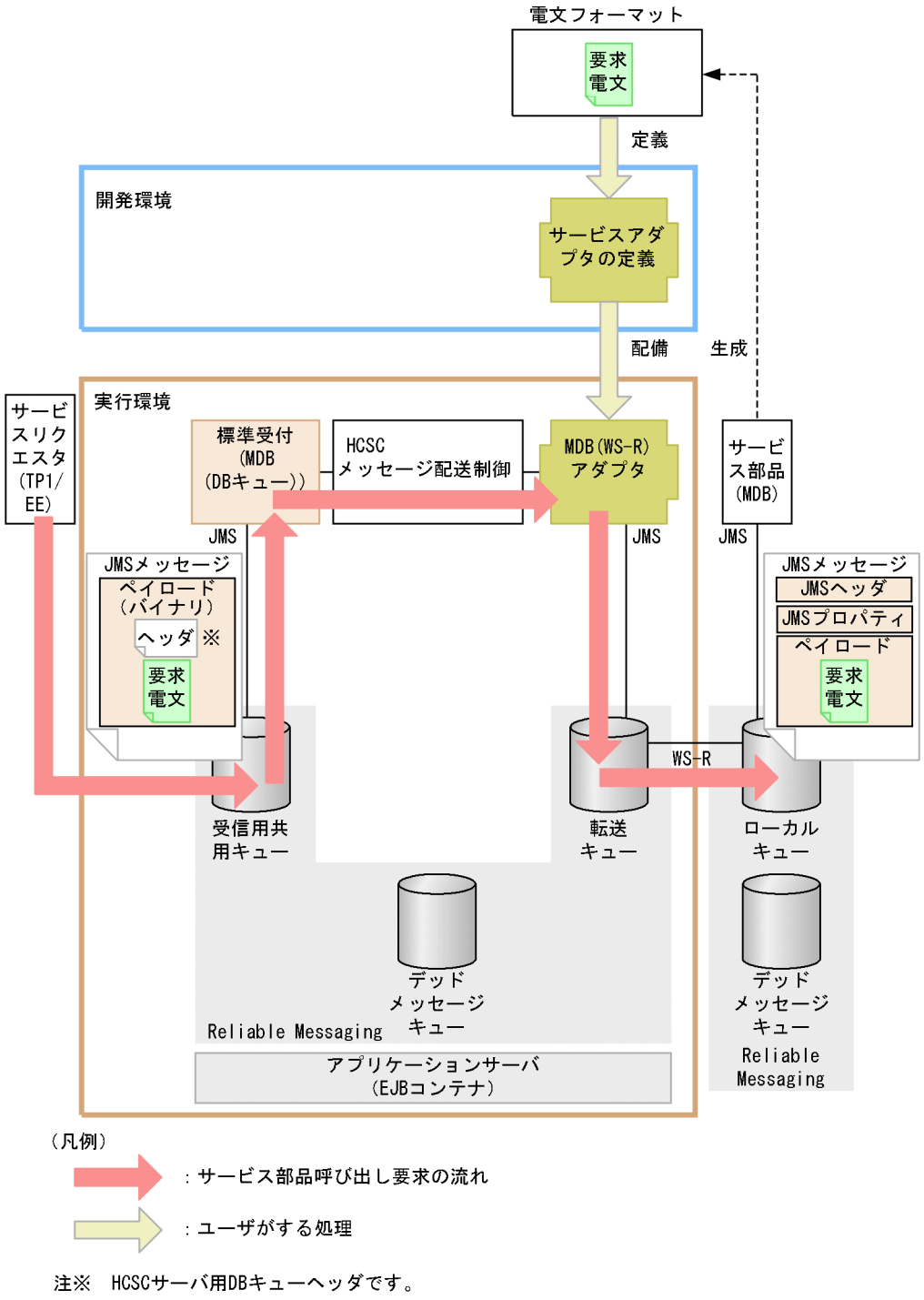
DB キューの作成の詳細については，マニュアル「TP1/Server Base Enterprise Option　使用の手引」の DB キューに関する内容を参照してください。

(1)　TP1/EE からサービス部品を呼び出す場合

サービスリクエスタからサービス部品を呼び出すときに，受信用共用キューに送信するメッセージの形式は，BytesMessage です。

呼び出すサービス名やクライアント相関 ID などのパラメタを設定した HCSC サーバ用 DB キューヘッダをバイナリ形式で作成し，そのあとに要求電文（バイナリ形式）を組み立てます。これによって，要求電文（バイナリ形式）の内容が，ユーザ電文としてサービス部品に伝わります。サービスリクエスタから標準受付（MDB（DB キュー））を呼び出すとき，サービス部品側の電文フォーマットに合わせた要求電文を事前に作成し，ペイロードに設定してメッセージを送信します。TP1/EE からサービス部品を呼び出すときのユーザ電文と電文フォーマットの関係を次の図に示します。

図 2-43　TP1/EE からサービス部品を呼び出すときのユーザ電文と電文フォーマットの関係



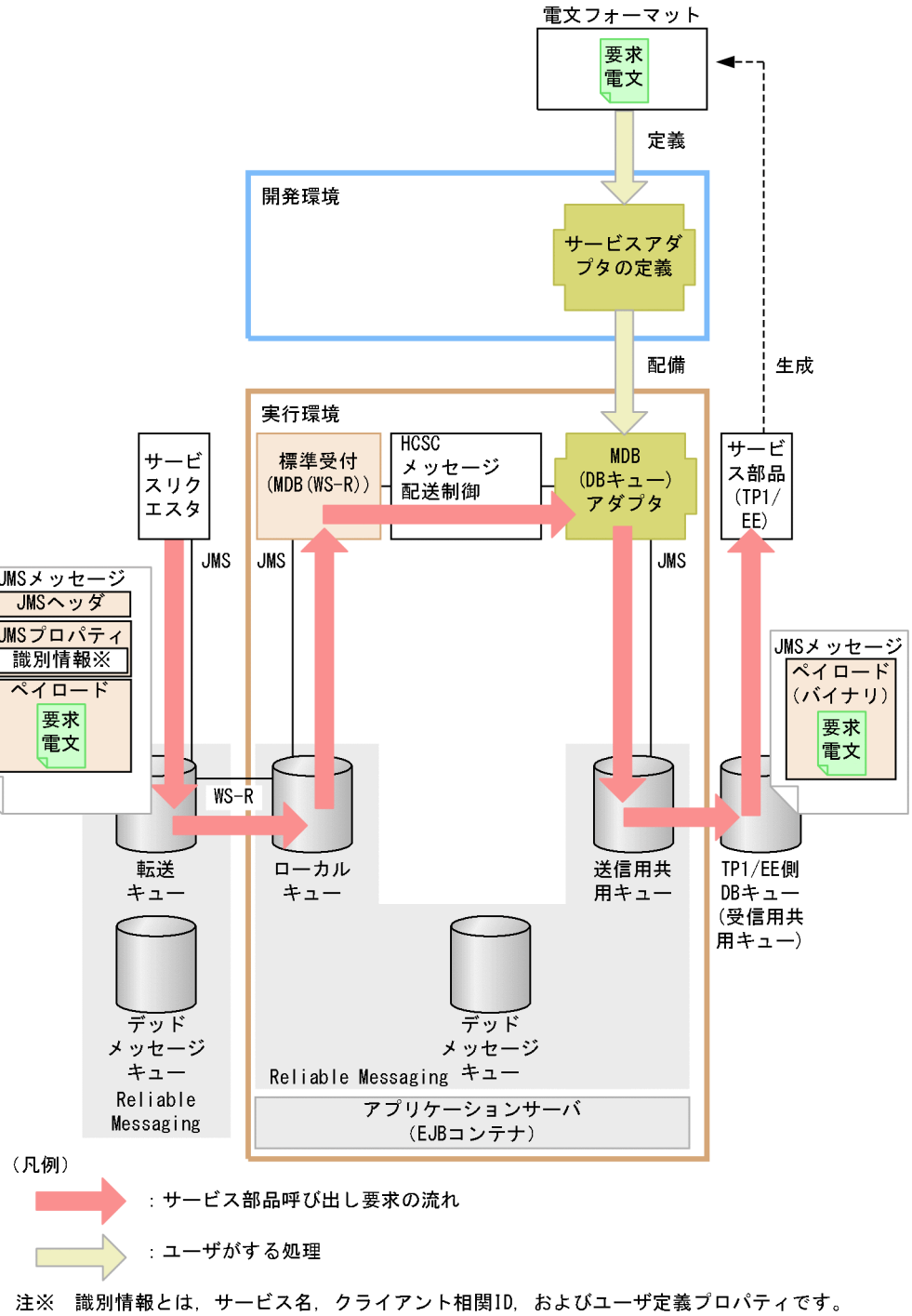
バイナリデータを作成する方法については，マニュアル「サービスプラットフォーム 開発ガイド 基本開発編」の「9.6.4　バイナリデータを作成する（TP1/EE，JMS）」を参照してください。

(2)　TP1/EE のサービス部品を呼び出す場合

サービスアダプタからサービス部品を呼び出すときに送信用共用キューに送信するメッセージの形式は，BytesMessage です。

サービスアダプタに定義した電文フォーマットに合わせた要求電文をバイナリ電文としてメッセージに設定します。TP1/EE のサービス部品を呼び出すときのユーザ電文と電文フォーマットの関係を次の図に示します。

図 2-44　TP1/EE のサービス部品を呼び出すときのユーザ電文と電文フォーマットの関係



注※　識別情報とは，サービス名，クライアント相関ID，およびユーザ定義プロパティです。

2.5.3　同期のサービス部品を呼び出す場合の応答電文の扱い

非同期（MDB（DB キュー））の標準受付は，非同期の要求電文の標準受付（MDB（DB キュー））で，基本的にサービス部品からの応答がない場合に使用します。しかし，非同期（MDB（DB キュー））の標準受付から同期のサービス部品（Web サービスまたは SessionBean など）やビジネスプロセスを要求した場合，同期のサービス部品から応答が返るときがあります。

そのため，HCSC サーバでは応答を応答用のキューに送信する手段を提供しています。

応答用のキューを利用するには，HCSC サーバの環境構築時に HCSC サーバ側に送信用共用キューを作成し，送信用共用キューに送信先のキュー名を指定しておきます。

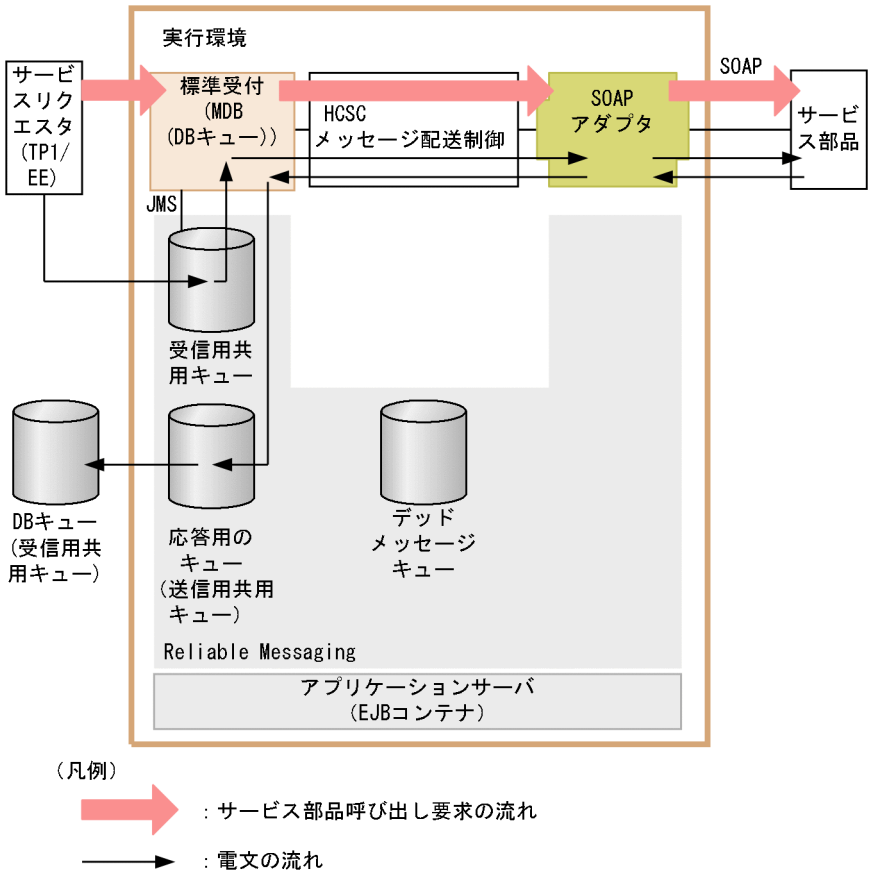
サービスリクエスタから標準受付（MDB（DB キュー））に対してサービス部品の呼び出しを要求する場合，標準受付（MDB（DB キュー））のパラメタに作成した送信用共用キューのキュー名を指定すると，サービス部品からの応答が返ってきた際に指定されたキューに JMS メッセージが送信されます。

送信用共用キューからサービスリクエスタ側の DB キュー（受信用共用キュー）にメッセージが転送されるため，DB キュー（受信用共用キュー）からメッセージを取り出すことで，サービス部品からの応答電文を取得できます。

応答を取り出す方法については，マニュアル「サービスプラットフォーム 開発ガイド 基本開発編」の「9.6.10　応答を取り出す（TP1/EE，JMS）」を参照してください。

同期サービス部品を呼び出した場合の応答の扱いを次の図に示します。

図 2-45　同期サービス部品を呼び出した場合の応答の扱い



応答用のキューへメッセージを出力するのは，次に示す場合です。

同期サービス部品でも応答があるかどうかによって，次のような形式で応答用のキューへメッセージを送信します。

なお，サービスリクエスタから応答用のキューのキュー名を指定していても，HCSC サーバ側に指定したキュー名のキューが作成されていない場合は，応答用のキューにメッセージは送信できないで，サービス部品呼び出し処理がエラーになります。

表 2-7　応答用のキューへメッセージを出力するケース

項番	応答用のキューへ出力するケース	応答電文の有無			
		応答電文がある場合	応答電文がない場合	サービス部品からのユーザ定義例外の場合	ユーザ定義例外以外のエラーの場合
1	呼び出し先のサービス部品が同期サービス部品の場合（WebService，SessionBean および汎用カスタムアダプタなど）	ペイロードに応答電文を設定	ペイロードなしのメッセージ	ペイロードに応答電文を設定	×※
2	呼び出し先のサービス部品が非同期サービス部品の場合	×	×	−	×※

項番	応答用のキューへ出力するケース	応答電文の有無			
		応答電文がある場合	応答電文がない場合	サービス部品からのユーザ定義例外の場合	ユーザ定義例外以外のエラーの場合
2	（MDB（WS-R）および MDB（DB キュー）など）	×	×	－	×※
3	呼び出し先のサービス部品がビジネスプロセスの場合	ペイロードに応答電文を設定	ペイロードなしのメッセージ	ペイロードに応答電文を設定	×※

（凡例）

×：メッセージを送信しません。

－：非同期のサービス部品からユーザ定義例外は存在しないため該当しません。

注※

エラーの情報はログおよびトレースに出力します。

サービスリクエスタからの要求で応答用のキューのキュー名を指定していない場合は，応答電文の内容は破棄されます（応答電文を破棄したことを示す KDEC00031-W メッセージが出力されます）。

なお，MDB（WS-R）の場合のように，JMS プロパティを引き継ぐことはできません。サービス部品呼び出しの相関関係を対応づける場合は，ユーザ電文内に相関関係を示す情報を埋め込みます。

2.5.4　MDB（DB キュー）使用時のトランザクション

MDB（DB キュー）の場合，サービスリクエスタから HCSC サーバの間，そしてサービスアダプタからサービス部品の間は，異なるトランザクションになります。

標準受付（MDB（DB キュー））でのローカルキューからのメッセージの取り出しとサービスアダプタでの転送キューへの送信は，1 つのトランザクションで処理します。MDB（WS-R）の場合のトランザクションと同一です。

永続化するビジネスプロセスの場合，ビジネスプロセスを実行する過程でのトランザクションについては，「3.4　ビジネスプロセスのトランザクション」を参照してください。

2.5.5　デッドメッセージキュー

HCSC サーバ稼働マシンでのサービス部品の呼び出しをロールバックして，ロールバックの回数（メッセージの配送回数）が最大値に達した場合や，配送回数に無制限の設定をしている場合，そのメッセージはデッドメッセージキューに遷移します。

エラーが発生したときにデッドメッセージキューに遷移する仕組みについては，マニュアル「サービスプラットフォーム システム構築・運用ガイド」の「7.7.4　MDB（DB キュー）実行時の障害対策」を参照してください。

2.5.6 MDB（DB キュー）で使用するキューの名称と個数

HCSC サーバの実行環境上では，次の表に示すキューの名称と個数を使用します。データベース（DB キュー）で使用するキューの名称と個数を次の表に示します。

表 2-8 データベース（DB キュー）で使用するキューの名称と個数

項番	用途	説明	キューの種類	キューの名称	個数	備考
1	非同期（MDB（DB キュー））の標準受付用	サービスリクエスタ側の TP1/EE システムから要求を受信するキュー。	受信用共用キュー	• 単独構成およびロードバランスクラスタの場合 CSC<HCSC サーバ名>ACPT_DBQ • HA クラスタの場合 CSC<クラスタ名>ACPT_DBQ	1 個	－
2	MDB（DB キュー）アダプタ用	サービス部品側の DB キューへ転送するキュー。	送信用共用キュー	CSC_<クラスタ名><アダプタ名>	非同期の MDB（DB キュー）アダプタ数	－
3	デッドメッセージキュー	MDB（WS-R）用のデッドメッセージキューと共用。	ローカルキューまたは転送キュー	任意の名称	システムに 1 個	WS-R でのデッドメッセージキューと同一
4	応答用のキュー	サービス部品呼び出しの応答を受け取る場合に使用。	送信用共用キュー	任意の名称	任意の個数	－

（凡例）

－：該当しません。

2.6　データベースとの接続

サービス部品としてデータベースを呼び出す場合には，DB アダプタを使用します。DB アダプタを使用する場合のデータベースのアクセスと設定について説明します。

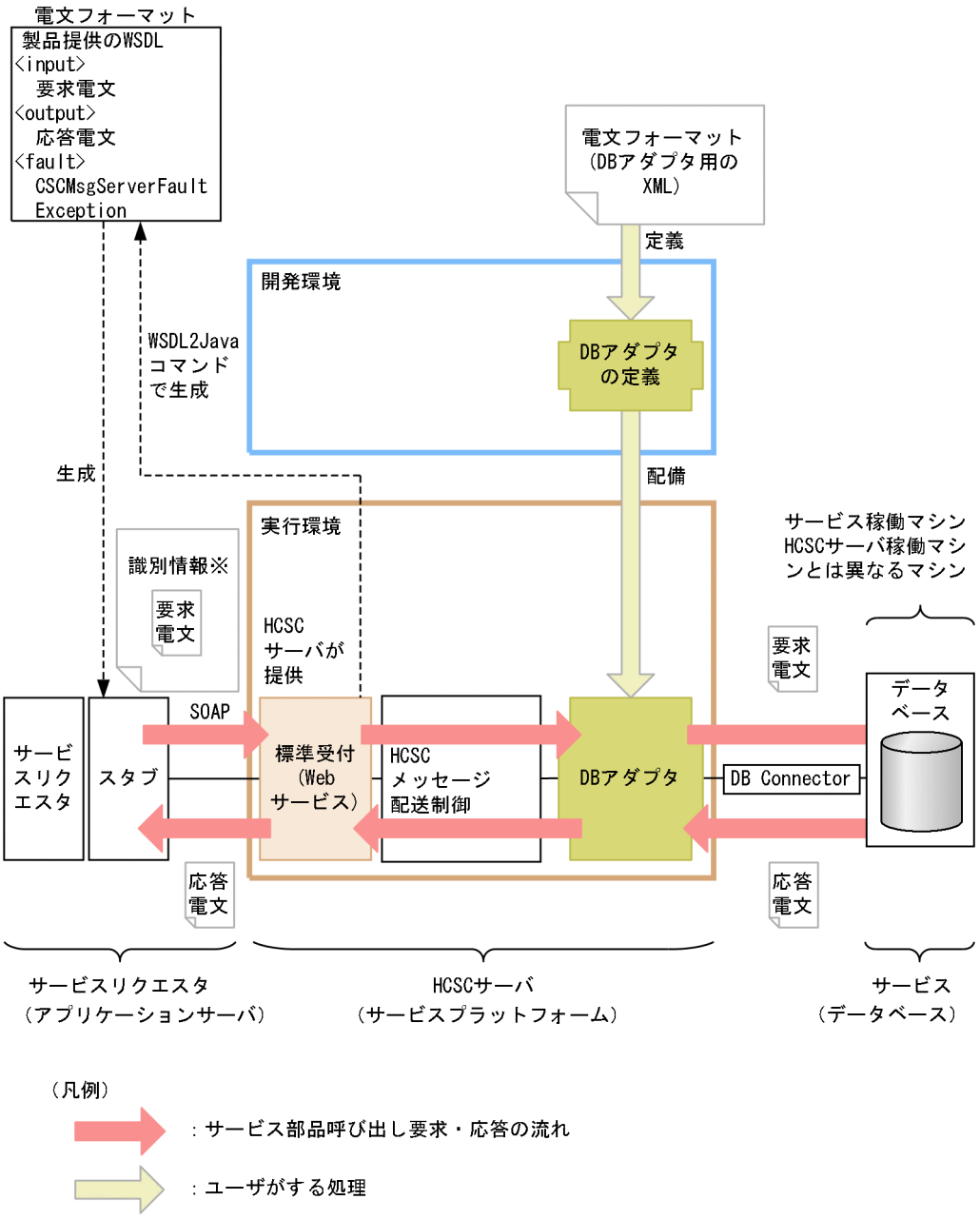
2.6.1　DB アダプタを使ったデータベースへのアクセス

DB アダプタは，サービスリクエスタやビジネスプロセスからの要求を受けて，データベースに対して直接アクセスできます。データベースに対するアクセスは，DB アダプタ内で定義されている SQL に従います。

(1)　DB アダプタによる SQL の実行

DB アダプタは，開発環境で定義します。DB アダプタに対して要求するサービスリクエスタからは，実行する SQL 文を記述した XML 文書を作成して，作成した XML 電文を要求電文に設定して実行します。DB アダプタによる SQL の実行について次の図に示します。

図 2-46　DB アダプタによる SQL の実行



注※　識別情報とは，サービス名，クライアント相関IDなどです。

DB アダプタを定義する手順，および DB アダプタに要求を出すサービスリクエスタの作成については，マニュアル「サービスプラットフォーム 開発ガイド 受付・アダプタ定義編」の「3.3.5　DB アダプタを定義する」を参照してください。

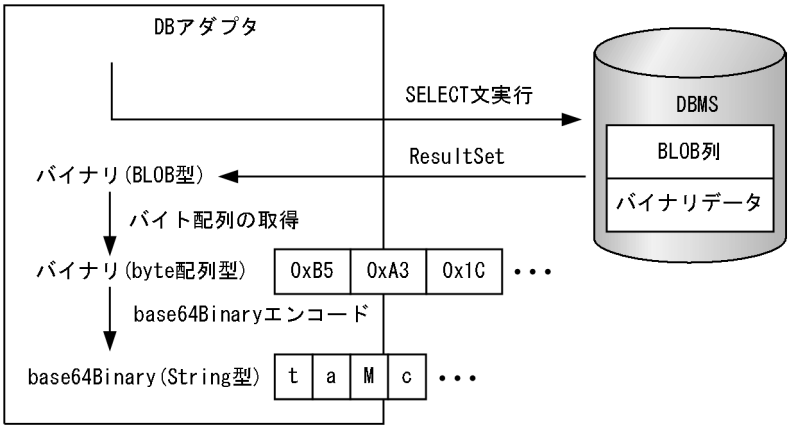
(2)　バイナリデータ操作時の処理

DB アダプタ内部で，バイナリデータは byte 配列型，base64Binary 型および hexBinary 型は String 型として扱います。SELECT 文実行時と INSERT 文実行時に BLOB 型を操作する場合の例を次に説明します。

(a) SELECT 文実行時の処理

BLOB 列に対して SELECT 文を実行すると，DB アダプタでは byte 配列から XML Schema の base64Binary 型または hexBinary 型に変換（エンコード）して，要求元に BLOB 列の値を返却します。

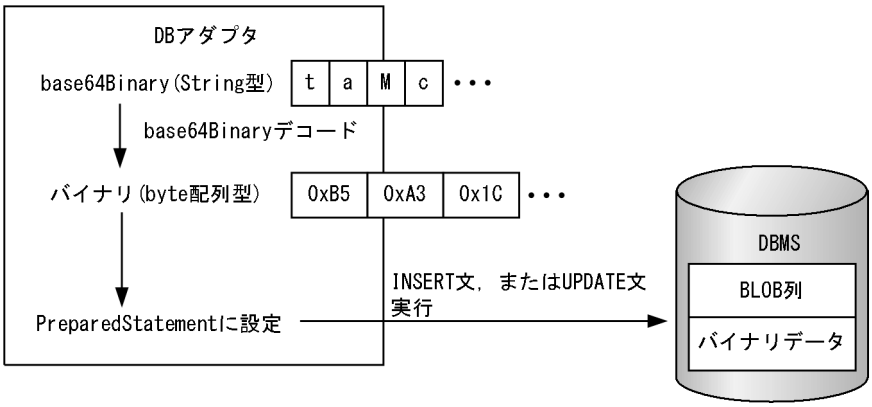
図 2-47 SELECT 文実行時の処理（base64Binary 型に変換する場合）



(b) INSERT 文，UPDATE 文実行時の処理

BLOB 列に対して INSERT 文，または UPDATE 文を実行する際は，要求元が XML Schema の base64Binary 型または hexBinary 型のデータを DB アダプタに渡します。DB アダプタでは base64Binary 型または hexBinary 型から byte 配列に変換（デコード）して，SQL を実行します。

図 2-48 INSERT 文，または UPDATE 文実行時の処理（base64Binary 型から変換する場合）



xsd:hexBinary 型または xsd:base64Binary 型のデータの変換（デコード）でエラーが発生した場合，KDEC63001-E（xsd:hexBinary 型の場合は原因コード=22，xsd:base64Binary 型の場合は原因コード=23）を出力します。また，DB アダプタの定義時に，「システム例外をフォルトに変換する」を有効にしている場合は DB アダプタの呼び出し元にはフォルト電文が送出され，無効にしている場合はシステム例外が送出されます。

(3) SQL 命令の INSERT 文，UPDATE 文および DELETE 文を使用する場合の注意事項

DB アダプタでは定義された SQL を実行することでデータベースが更新されます。

そのため，HCSC サーバの DB アダプタと他システムで同一のデータベースを利用する場合，トランザクションの終了タイミングやデータの整合性を考慮してデータベースにアクセスする必要があります。

DB アダプタ経由でデータの INSERT 文，UPDATE 文および DELETE 文を実行する場合は，他システムに影響を与えないことを確認して利用してください。

2.6.2 DB アダプタが使用する DB Connector の設定

データベースにアクセスするには，HCSC サーバをセットアップするときに定義した DB Connector とは別に，サービス部品稼働マシン（データベースが稼働しているマシン）に対してアクセスする DB Connector を準備する必要があります。

(1) 使用できる DB Connector

HCSC サーバをセットアップするときに定義した DB Connector のトランザクション属性によって，使用できる DB Connector に制約があります。

DB アダプタで使用できる DB Connector は，HCSC サーバセットアップ定義ファイルの dbcon-xadisplayname プロパティに指定した XATransaction または LocalTransaction の値に依存します。そのため，HCSC サーバのセットアップ時には，dbcon-xadisplayname プロパティに指定した XATransaction または LocalTransaction の値を考慮する必要があります。

表 2-9 使用可能な DB Connector

使用している DB Connector	DB アダプタ		使用可否	否の理由
HCSC サーバ本体側(dbcon-xadisplayname)	使用可能な DB Connector	DB アダプタの接続先		
LocalTransaction	LocalTransaction	HCSC サーバが使用するデータベースと同じデータベース	可	−
	LocalTransaction	HCSC サーバが使用するデータベースと異なるデータベース	否	Local トランザクション内で複数の DB Connector を混在させることができないため
	XATransaction	HCSC サーバが使用するデータベースと異なるデータベース	否	Local トランザクション内で XA（グローバルトランザクション）を混在させることができないため

使用している DB Connector	DB アダプタ		使用可否	否の理由
HCSC サーバ本体側(dbcon-xadisplayname)	使用可能な DB Connector	DB アダプタの接続先		
XATransaction	LocalTransaction	HCSC サーバが使用するデータベースと異なるデータベース	否	XA トランザクション内で複数の DB Connector を混在させることができないため
	XATransaction	HCSC サーバが使用するデータベースと同じデータベース	可	－
	XATransaction	HCSC サーバが使用するデータベースと異なるデータベース	可	－

（凡例）

－：該当しません。

(2) DB Connector のインポートおよび配備

DB アダプタが利用する DB Connector を HCSC サーバ（J2EE サーバ）にインポートし，配備します。

DB アダプタを利用する場合にインポートする RAR ファイルの選択方法については，マニュアル「サービスプラットフォーム システム構築・運用ガイド」の「3.1.2　実行環境に必要なソフトウェアを設定する」を参照してください。

1. DB Connector のインポート

次に示すコマンドを使用して DB Connector をインポートします。

```
<サービスプラットフォームのインストールディレクトリ>¥CC¥admin¥bin¥cjimportres <J2EEサーバ名> -nameserver corbaname::localhost:<インプロセスのネーミングサービス用のポート番号> -type rar -f <サービスプラットフォームのインストールディレクトリ>¥CC¥DBConnector¥<インポートするRARファイル>
```

2. DB Connector の配備

次に示すコマンドを使用して DB Connector を配備します。

```
<サービスプラットフォームのインストールディレクトリ>¥CC¥admin¥bin¥cjdeployrar <J2EEサーバ名> -nameserver corbaname::localhost:<インプロセスのネーミングサービス用のポート番号> -resname <インポートしたDB Connectorの名称>
```

(3) DB Connector のプロパティ定義

DB Connector をインポートしたあと，プロパティを定義します。プロパティ定義ファイルの詳細については，マニュアル「アプリケーションサーバ リファレンス 定義編(アプリケーション／リソース定義)」の「4.1　Connector 属性ファイル」を参照してください。リソースに別名を設定する場合の付与規則につい

ては，マニュアル「アプリケーションサーバ 機能解説 基本・開発編(コンテナ共通機能)」の「2.6.2　別名の付与規則」を参照してください。

参考

配備した DB Connector に対してプロパティを設定するときに，新規にプロパティ定義ファイルの XML を作成するのではなく，配備した DB Connector のプロパティを取得し，取得した XML に対して編集をすると効率良く作成できます。

1. プロパティ定義用のファイルの取得

次に示すコマンドを使用して DB Connector のプロパティ定義用のファイルを取得します。

```
<サービスプラットフォームのインストールディレクトリ>¥CC¥admin¥bin¥cjgetrarprop <J2EEサーバ名> -nameserver corbaname::localhost:<インプロセスのネーミングサービス用のポート番号> -resname <インポートしたDB Connectorの名称> -c property.xml
```

2. プロパティ定義用のファイルの編集

取得したプロパティ定義用の XML ファイルについて，各属性の値を変更します。

プロパティ定義用の XML ファイル（Connector 属性ファイル）の設定内容については，マニュアル「サービスプラットフォーム システム構築・運用ガイド」の「3.1.2　実行環境に必要なソフトウェアを設定する」を参照してください。

Connector 属性ファイルの詳細については，マニュアル「アプリケーションサーバ アプリケーション設定操作ガイド」の「4.2.2　DB Connector のプロパティ定義」，およびマニュアル「アプリケーションサーバ リファレンス 定義編(アプリケーション／リソース定義)」の「4.1　Connector 属性ファイル」を参照してください。

3. 編集したプロパティ定義用のファイルの設定

編集したプロパティ定義用の XML ファイルの内容を配備している DB Connector に設定します。

次に示すコマンドを使用して DB Connector に定義内容を設定します。

```
<サービスプラットフォームのインストールディレクトリ>¥CC¥admin¥bin¥cjsetrarprop <J2EEサーバ名> -nameserver corbaname::localhost:<インプロセスのネーミングサービス用のポート番号> -resname <インポートしたDB Connectorの名称> -c property.xml
```

注

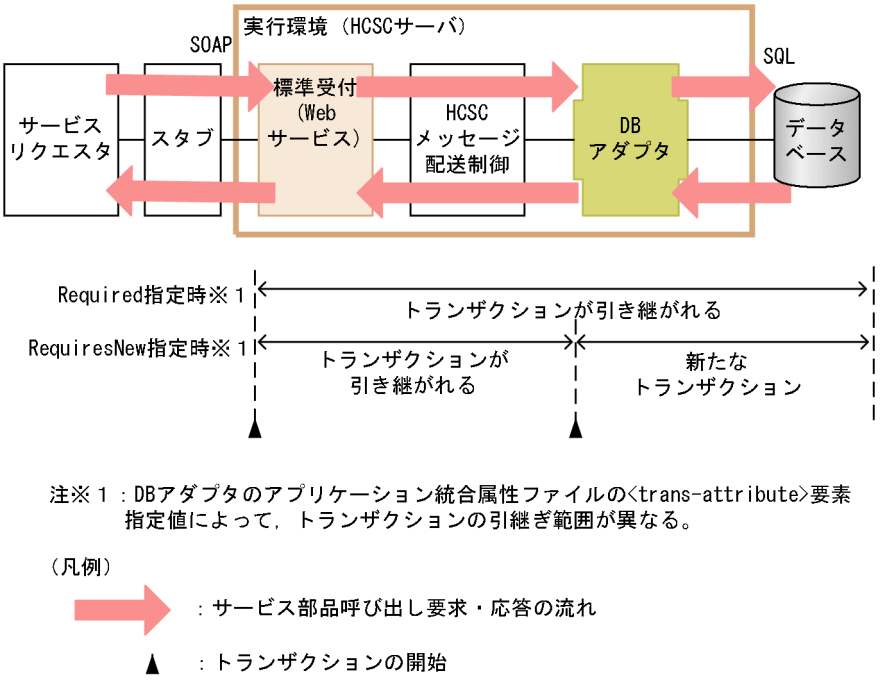
プロパティ定義用のファイル（property.xml）の設定後，DB Connector のプロパティの各属性の値は上書きされます。

2.6.3　DB アダプタ使用時のトランザクション

DB アダプタの場合，HCSC サーバ内で管理しているトランザクションをデータベース側に引き継ぎます。サービスリクエスタから HCSC サーバの間のトランザクションについては，それぞれの標準受付のトランザクション制御についての説明を参照してください。

SOAP 通信を利用したサービスリクエスタを例に，DB アダプタの場合のトランザクションを次の図に示します。

図 2-49　DB アダプタのトランザクション



注※１：DBアダプタのアプリケーション統合属性ファイルの<trans-attribute>要素
指定値によって，トランザクションの引継ぎ範囲が異なる。

（凡例）

：サービス部品呼び出し要求・応答の流れ

：トランザクションの開始

2.6.4　DB アダプタの接続先情報の動的変更

DB アダプタの接続先は，接続先の DB Connector のリソースの別名を DB アダプタに設定することで動的に変更できます。ビジネスプロセスで接続先の DB Connector のリソースの別名を設定しておくと，DB アダプタを実行したときに接続先が変更されます。これによって，作成する DB アダプタの数を削減できます。

(1)　接続先動的変更の処理の流れ

DB アダプタの接続先動的変更の処理の流れを次に示します。

1. 要求電文内のヘッダ電文から接続先情報を取得します。
2. 取得した接続情報を基にデータベースとのコネクションを確立します。

なお，接続情報がヘッダ電文に設定されていないなど，接続先情報の取得に失敗した場合は，リクエスト処理がエラーになります。

(2)　接続先動的変更の前提条件

DB アダプタの接続先を動的に変更するための前提条件を次に示します。

(a)　リソース名の別名付与

接続先の DB Connector にリソースの別名を付けます。

別名の付与規則については，マニュアル「アプリケーションサーバ 機能解説 基本・開発編(コンテナ共通機能)」の「2.6.2　別名の付与規則」を参照してください。

(b)　SQL オペレーション定義ファイルの設定

SQL オペレーション定義ファイルの dynamic 属性に「Y」を設定します。

(c)　DB Connector 種別の選択

接続先の DB Connector の種別は，SQL オペレーション定義ファイルに設定したデータベース種別と同一にします。

(d)　トランザクションサポートレベルの設定

接続先の DB Connector のトランザクションサポートレベルは，「LocalTransaction」または「XATransaction」を指定できます。

指定できるトランザクションサポートレベルと指定できる条件を次の表に示します。

表 2-10　指定できるトランザクションサポートレベルと指定できる条件

指定できるトランザクションサポートレベル	指定できる条件
LocalTransaction，または XATransaction	• ビジネスプロセスでステータスの永続化をしない場合 • DB アダプタのトランザクションをビジネスプロセスとは別に開始する場合 • ビジネスプロセスでステータスを永続化し，かつ DB アダプタのトランザクションをビジネスプロセスとは別に開始する場合
XATransaction	上記以外の場合

(e)　ビジネスプロセスの設定

ビジネスプロセスで接続先情報を設定します。

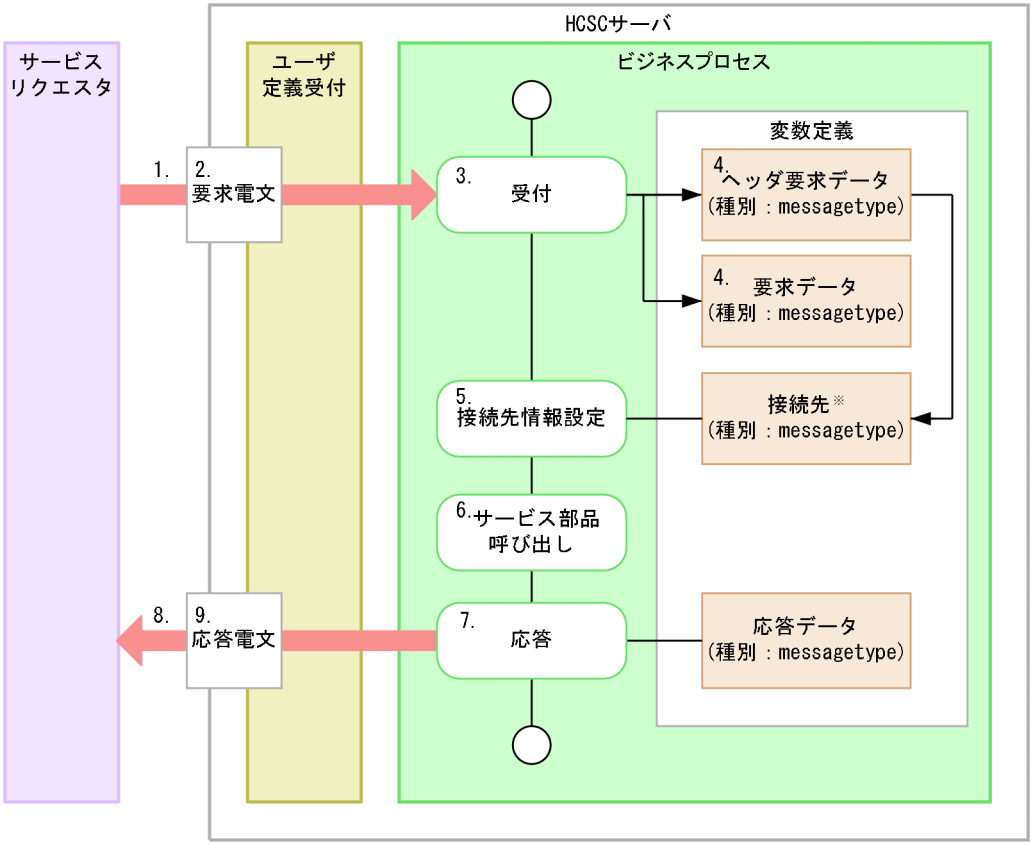
(3)　接続先情報を動的に変更する例

接続先情報ファイルを使用して，接続先情報を動的に変更する例を説明します。

(a)　ビジネスプロセスの例

接続先情報ファイルを使用するビジネスプロセスの例を次の図に示します。

図 2-50　接続先情報ファイルを使用するビジネスプロセスの例（DB アダプタの場合）



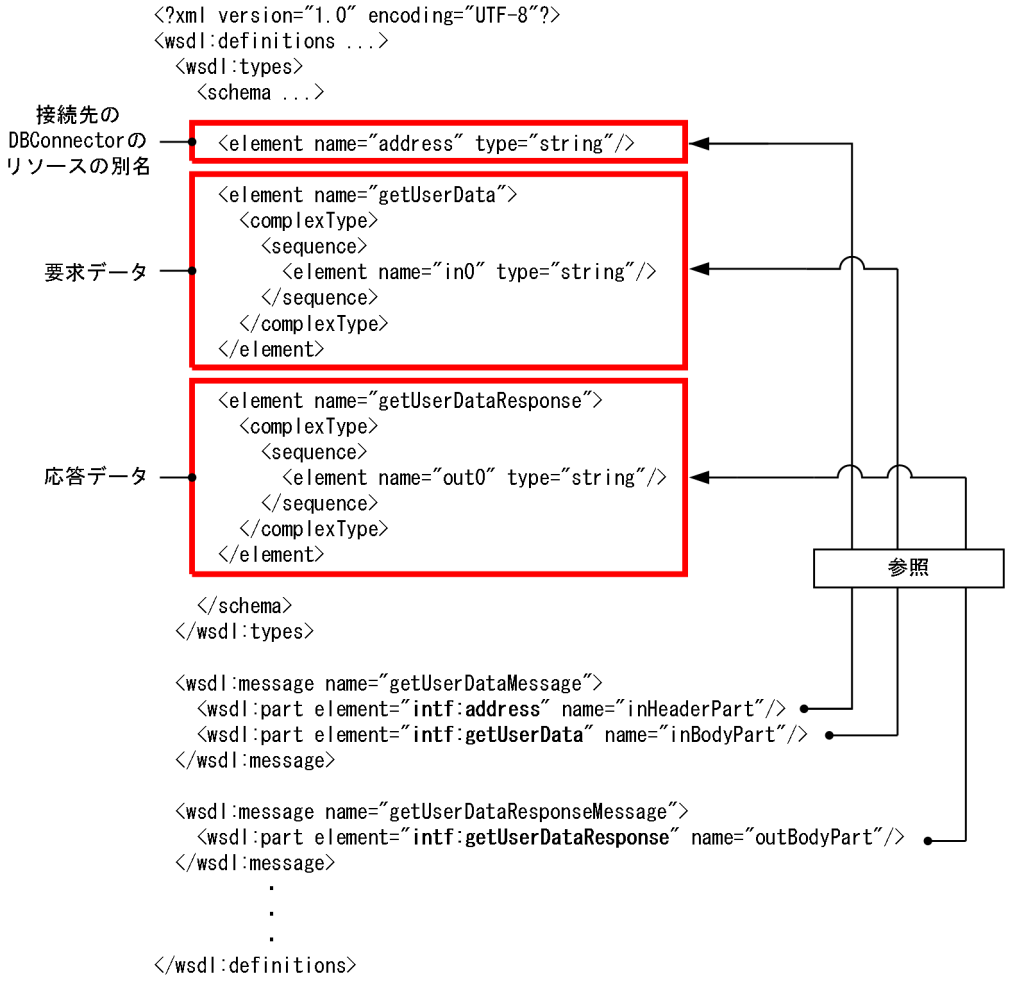
注※　接続先情報ファイルを使用して定義した変数

1. サービスリクエスタがビジネスプロセスにサービス部品の実行を要求します。
2. サービスリクエスタから送付する要求電文内のヘッダ電文に接続先情報（DB Connector のリソースの別名）を設定します。
3. ビジネスプロセスは XML の要求電文を受付アクティビティで受け取ります。
4. ビジネスプロセスがヘッダ要求データおよび要求データを生成します。
5. ヘッダ要求データの接続先情報を，データ変換アクティビティなどで接続先情報変数に設定します。
6. ビジネスプロセスは接続先情報変数に含まれる接続先情報を DB アダプタに設定して，サービス部品を呼び出します。
7. ビジネスプロセスが，サービス部品の実行結果を XML の応答電文として構成し，応答アクティビティで返します。
8. サービスリクエスタへの応答が返されます。
9. サービスリクエスタへの応答では，XML の応答電文が送付されます。

(4)　ユーザ定義受付を生成する WSDL の作成例

ユーザ定義受付を生成する WSDL の作成例を次に示します。

なお，この例では，接続先となる DB Connector のリソースの別名を SOAP ヘッダで送付しています。

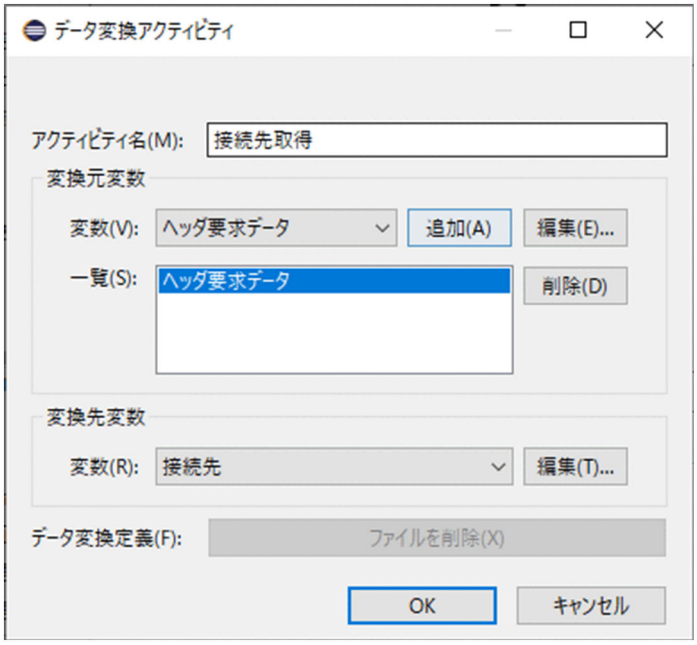


(a)　データ変換アクティビティの設定例

「図 2-50　接続先情報ファイルを使用するビジネスプロセスの例（DB アダプタの場合）」に示すビジネスプロセスで使用するデータ変換アクティビティの設定例を次に示します。

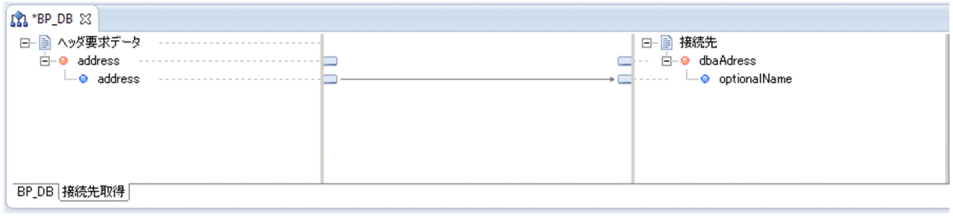
ここでは，変換元に「ヘッダ要求データ」を，変換先に「接続先」を指定しています。

図 2-51　データ変換アクティビティの設定例（DB アダプタの場合）



また，変換のマッピングは，次の図に示すとおりに定義します。

図 2-52　変換のマッピングの設定例（DB アダプタの場合）



(b)　接続先情報ファイルの定義

DB Connector のリソースの別名を DB アダプタに設定するための接続先情報ファイル（XML スキーマ）に定義します。

サービスプラットフォームでは，接続先情報ファイルを提供しています。

- 接続先情報ファイルの格納先

```
<サービスプラットフォームのインストールディレクトリ>/CSC/schema/connection/connection_db_adapter.xsd
```

> **注意事項**
>
> DB アダプタの接続先情報を動的に変更する場合，サービスプラットフォームで提供している接続先情報ファイルを使用してください。提供している接続先情報ファイルを使用しなかった場合，動作は保証されません。

- 接続先情報ファイルの形式

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
 DO NOT EDIT THIS FILE.
 -->
<xsd:schema
    elementFormDefault="qualified"
    targetNamespace="http://www.hitachi.co.jp/soft/xml/cosminexus/csc/connection/change/dba"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="dbaAddress">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="optionalName" type="xsd:string"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

- 設定できる要素

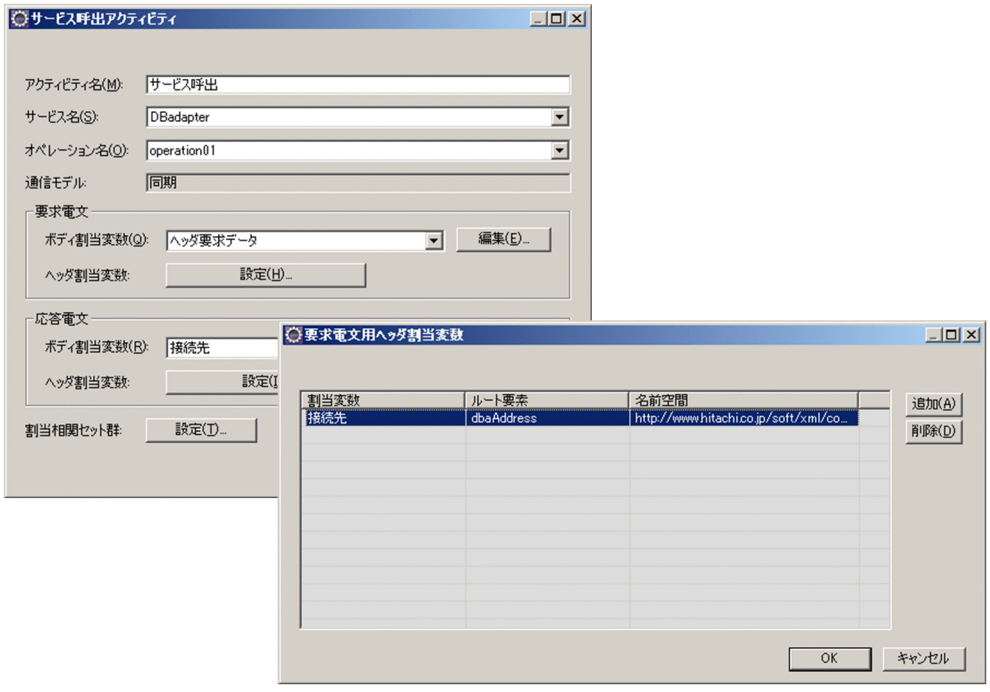
  optionalName

  DB Connector に設定したリソースの別名を指定します。

(c) 接続先情報の設定

「図 2-50 接続先情報ファイルを使用するビジネスプロセスの例（DB アダプタの場合）」に示すビジネスプロセスで使用する接続先情報の設定例を次に示します。なお，接続先情報は，サービス呼出アクティビティのヘッダ割当変数に設定します。ここで設定した情報が，DB アダプタで参照できる情報になります。

図 2-53 接続先情報の設定例（サービス呼出アクティビティの設定）



2.7　OpenTP1 との接続

OpenTP1 などの既存システムのサービス部品を呼び出したり，OpenTP1 などの既存システムからサービス部品を呼び出したりできます。呼び出し方で接続のしかたが異なります。

2.7.1　TP1 から非同期の標準受付を使った HCSC サーバのサービス部品の呼び出し

同期応答型の要求のほかに，非同期で OpenTP1 などの既存システムから，HCSC サーバを経由してサービス部品を呼び出せます。

HCSC サーバの標準受付（MDB（DB キュー））に対して要求する非同期応答型の送信ができます。

詳細は，「2.5　DB キューを使用したサービスとの接続」を参照してください。

2.7.2　TP1 から TP1/RPC 受付を使った HCSC サーバのサービス部品の呼び出し

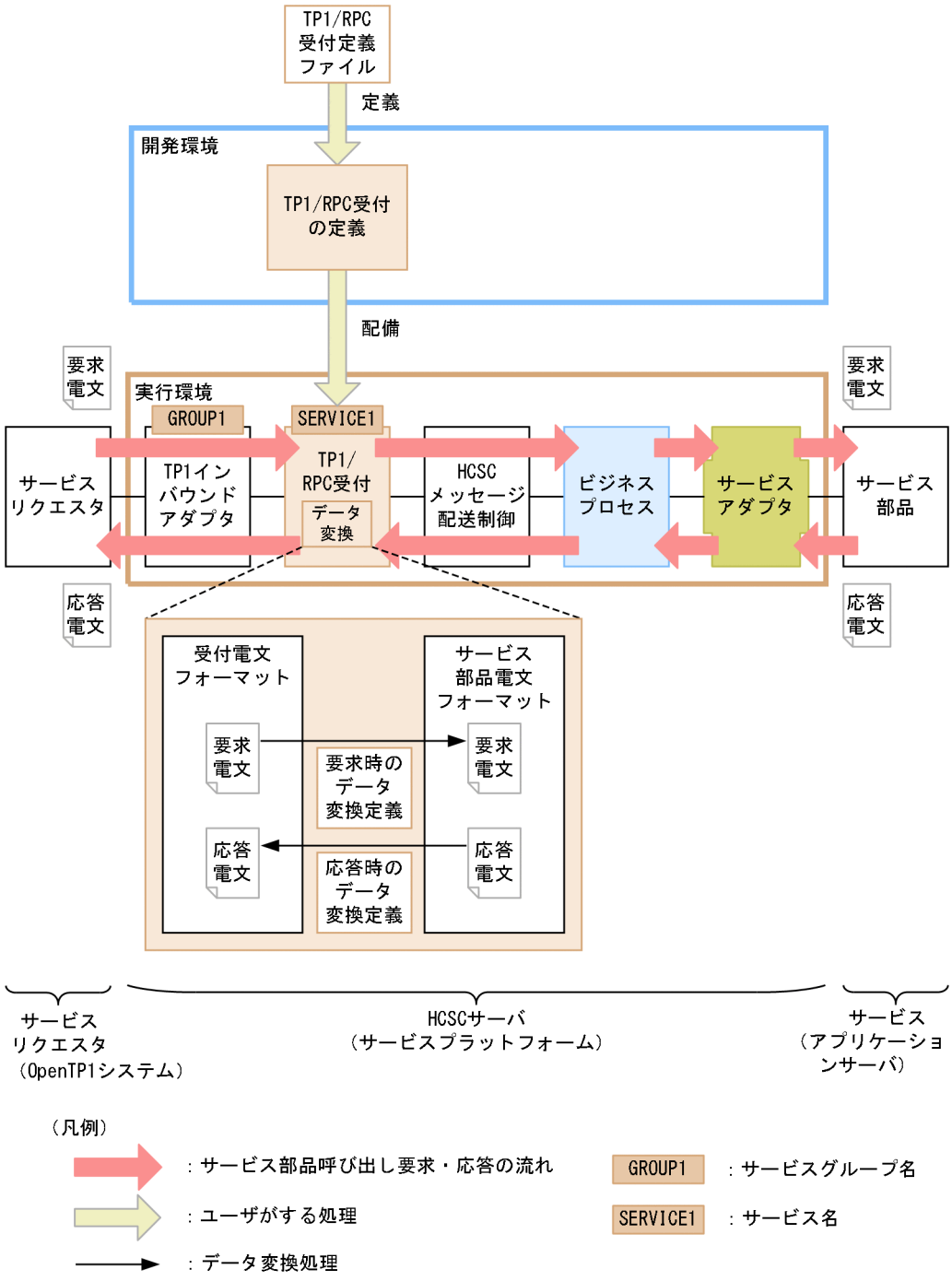
HCSC サーバの標準受付（SessionBean，MDB（DB キュー））のほかに，ユーザ定義受付（TP1/RPC 受付）を使用することで，既存の OpenTP1 システムから，HCSC サーバを経由してサービス部品を呼び出せます。

TP1/RPC 受付は，開発環境で TP1/RPC 受付定義ファイルを作成し，実行環境へ配備することで利用できます。なお，サービスリクエスタからのサービス部品呼び出し要求を受けた場合，TP1/RPC 受付は，リソースアダプタである TP1 インバウンドアダプタから電文を受け取って処理をします。そのため，TP1/RPC 受付を使用する場合は，TP1 インバウンドアダプタのセットアップが必要になります。

TP1/RPC 受付を使用することで，サービスリクエスタはビジネスプロセスを，リモートプロシジャコール通信（同期応答型 RPC）で呼び出せます。サービスリクエスタが呼び出すビジネスプロセスは，dc_rpc_call 関数に指定するサービスグループ名とサービス名で特定されます。このとき指定したサービスグループ名とサービス名が，それぞれ TP1 インバウンドアダプタと TP1/RPC 受付に結び付けられます。

OpenTP1 システムからのサービス部品の呼び出しを次の図に示します。

図 2-54　TP1/RPC 受付を使用した OpenTP1 システムからのサービス部品の呼び出し



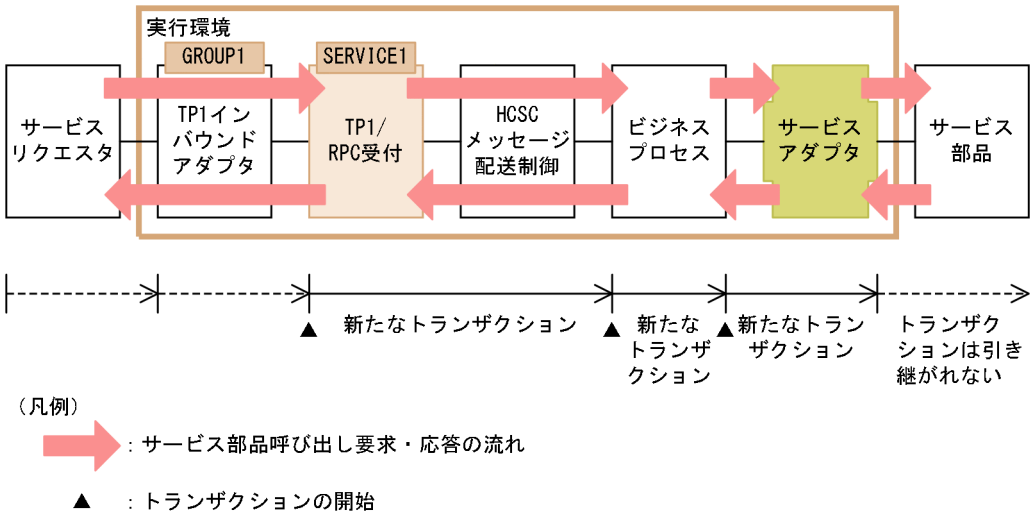
(1)　TP1/RPC 受付使用時のトランザクション

TP1/RPC 受付では，サービスリクエスタおよび TP1 インバウンドアダプタのトランザクションとは異なる新たなトランザクションを開始します。

また，ビジネスプロセス内ではさらに TP1/RPC 受付と異なる新たなトランザクションが開始されます。

ビジネスプロセスを呼び出すときの TP1/RPC 受付のトランザクションを次に示します。

図 2-55　TP1/RPC 受付のトランザクション



(2)　TP1/RPC 受付のデータ変換

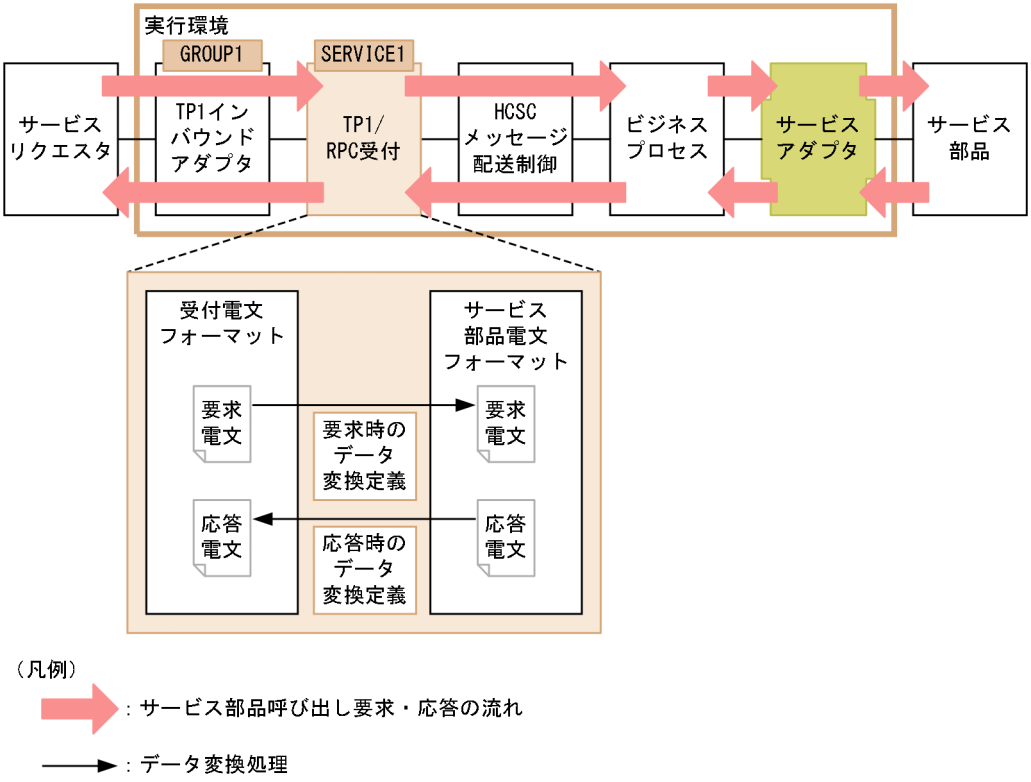
TP1/RPC 受付でのデータの流れとデータ変換の関係，およびデータ変換時の構造変換スキップ機能の指定について説明します。

(a)　TP1/RPC 受付でのデータの流れとデータ変換の関係

サービスリクエスタから受け取った要求電文（バイナリ形式）は，TP1/RPC 受付によってデータ変換（バイナリ形式から XML 形式への変換）が実行されます。データ変換によって変換された要求電文（XML 形式）は，HCSC メッセージ配送制御を経由してビジネスプロセスに渡されます。ビジネスプロセスから返ってきた応答電文（XML 形式）は，TP1/RPC 受付によってデータ変換（XML 形式からバイナリ形式への変換）が実行されます。データ変換によって変換された応答電文（バイナリ形式）は，TP1 インバウンドアダプタを経由してサービスリクエスタに返されます。

TP1/RPC 受付でのデータの流れとデータ変換の関係を次に示します。

図 2-56　TP1/RPC 受付でのデータの流れとデータ変換の関係



(b)　データ変換時の構造変換スキップ機能の指定

構造変換スキップ機能とは，データ変換時に電文の構造変換をスキップする機能のことです。

要求電文のデータ変換では，サービスリクエスタから受け取ったバイナリ形式の要求電文から生成した DOM ツリーを，そのままビジネスプロセスに渡したいときに，構造変換スキップ機能を有効にできます。

応答電文のデータ変換では，ビジネスプロセスから返ってきた DOM ツリーから，直接，バイナリ形式の応答電文を生成し，サービスリクエスタに返したいときに，構造変換スキップ機能を有効にできます。

●構造変換スキップ機能の設定方法

構造変換スキップ機能の設定を有効にするには，開発環境のユーザ定義受付定義画面で，［独自定義ファイル］に TP1/RPC 受付定義ファイルを設定します。

ユーザ定義受付定義画面に表示される項目については，マニュアル「サービスプラットフォーム リファレンス」の「1.2.5　ユーザ定義受付定義画面」を参照してください。

TP1/RPC 受付定義ファイルの詳細は，マニュアル「サービスプラットフォーム リファレンス」の「3.4.1　TP1/RPC 受付定義ファイル」を参照してください。

●構造変換スキップ機能を使用するときの注意事項

開発環境のユーザ定義受付定義画面で，要求電文および応答電文の受付電文フォーマットにバイナリフォーマット定義ファイルを，要求電文および応答電文のサービス部品電文フォーマットに XML フォーマット定義ファイルを指定します。

開発環境のユーザ定義受付定義画面で，要求電文および応答電文のサービス部品電文フォーマットには，受付電文フォーマット（バイナリフォーマット定義ファイル）から cscfdx2xsd コマンドを使用して生成した XML フォーマット定義ファイルを指定してください。

(3) TP1/RPC 受付のチューニング

ここでは，TP1/RPC 受付の同時実行数，タイムアウト，トランザクションタイムアウト，およびサービス名の設定変更について説明します。

(a) 同時実行数の設定方法

同時実行数の設定を変更するときの TP1 インバウンドアダプタでの設定，および TP1/RPC 受付での設定について説明します。

●TP1 インバウンドアダプタでの設定

TP1 インバウンドアダプタでの設定については，マニュアル「アプリケーションサーバ 機能解説 基本・開発編(コンテナ共通機能)」の「4.12.2　リソースアダプタの設定」を参照してください。

●TP1/RPC 受付での設定

TP1/RPC 受付では，次の環境でチューニングパラメタを変更できます。

- 開発環境（ユーザ定義受付定義画面で TP1/RPC 受付定義ファイルを設定）
- 運用環境（コマンドで TP1/RPC 受付定義ファイルを設定，またはコマンドでアプリケーション統合属性ファイルを設定）

TP1/RPC 受付の同時実行数の設定を変更するとき，TP1/RPC 受付で変更できる内容を次に示します。

●TP1/RPC 受付定義ファイルを設定する場合

次に示す TP1/RPC 受付定義ファイルのプロパティに値を設定します。

- urecp-tp1rpc.activation-config.queue_max_length プロパティ
- urecp-tp1rpc.pooled-instance.minimum プロパティ
- urecp-tp1rpc.pooled-instance.maximum プロパティ

●アプリケーション統合属性ファイルを設定する場合

アプリケーション統合属性ファイルの要素に値を設定します。

設定する要素については，次に示す XPath 式を参照してください。

- //hitachi-application-all-property/ejb-jar/hitachi-ejb-jar-property/display-name[.='cscmsg_urecp_tp1rpc']/../../hitachi-message-bean-property/display-name[.='CSCMsgTp1RpcServiceDelivery']/../activation-config/activation-config-property/activation-config-property-name[.='queue_max_length']/../activation-config-property-value
- //hitachi-application-all-property/ejb-jar/hitachi-ejb-jar-property/display-name[.='cscmsg_urecp_custom']/../../hitachi-session-bean-property/display-

name[.='CSCMsgCustomServiceDelivery']/../session-runtime/stateless/pooled-instance/minimum

- //hitachi-application-all-property/ejb-jar/hitachi-ejb-jar-property/display-name[.='cscmsg_urecp_custom']/../../hitachi-session-bean-property/display-name[.='CSCMsgCustomServiceDelivery']/../session-runtime/stateless/pooled-instance/maximum
- //hitachi-application-all-property/ejb-jar/hitachi-ejb-jar-property/display-name[.='cscmsg_urecp_tp1rpc']/../../hitachi-message-bean-property/display-name[.='CSCMsgTp1RpcServiceDelivery']/../message-runtime/pooled-instance/maximum

(b) タイムアウト時間の設定方法

タイムアウトの設定を変更するときの TP1 インバウンドアダプタでの設定，および TP1/RPC 受付での設定について説明します。

●TP1 インバウンドアダプタでの設定

TP1 インバウンドアダプタでは，パフォーマンスチューニングのための機能を定義する場合，Connector 属性ファイルを使用します。

TP1 インバウンドアダプタでの設定については，マニュアル「アプリケーションサーバ 機能解説 基本・開発編(コンテナ共通機能)」の「4.12.2　リソースアダプタの設定」を参照してください。ただし，サービス実行タイムアウトの設定は行わないでください。

●TP1/RPC 受付での設定

TP1/RPC 受付で設定の変更はありません。

(c) トランザクションタイムアウト時間の設定方法

トランザクションタイムアウトの設定を変更するときの TP1 インバウンドアダプタでの設定，および TP1/RPC 受付での設定について説明します。

●TP1 インバウンドアダプタでの設定

TP1 インバウンドアダプタで設定の変更はありません。

●TP1/RPC 受付での設定

TP1/RPC 受付では，次の環境でチューニングパラメタを変更できます。

- 開発環境（ユーザ定義受付定義画面で TP1/RPC 受付定義ファイルを設定）
- 運用環境（コマンドで TP1/RPC 受付定義ファイルを設定，またはコマンドでアプリケーション統合属性ファイルを設定）

TP1/RPC 受付のトランザクションタイムアウトの設定を変更するとき，TP1/RPC 受付で変更できる内容を次に示します。

●TP1/RPC 受付定義ファイルを設定する場合

次に示す TP1/RPC 受付定義ファイルのプロパティに値を設定します。

- urecp-tp1rpc.ejb-transaction-timeout プロパティ

●アプリケーション統合属性ファイルを設定する場合

アプリケーション統合属性ファイルの要素に値を設定します。

設定する要素については，次に示す XPath 式を参照してください。

- //hitachi-application-all-property/ejb-jar/hitachi-ejb-jar-property/display-name[.='cscmsg_urecp_custom']/../../hitachi-session-bean-property/display-name[.='CSCMsgCustomServiceDelivery']/../ejb-transaction-timeout/method/method-name[.='*']/../../transaction-timeout

(d) サービス名の設定方法

サービス名の設定を変更するときの TP1/RPC 受付での設定について説明します。

●TP1/RPC 受付での設定

TP1/RPC 受付では，次の環境でチューニングパラメタを変更できます。

- 開発環境（ユーザ定義受付定義画面で TP1/RPC 受付定義ファイルを設定）
- 運用環境（コマンドで TP1/RPC 受付定義ファイルを設定，またはコマンドでアプリケーション統合属性ファイルを設定）

TP1/RPC 受付のサービス名の設定を変更するとき，TP1/RPC 受付で変更できる内容を次に示します。

●TP1/RPC 受付定義ファイルを設定する場合

次に示す TP1/RPC 受付定義ファイルのプロパティに値を設定します。

- urecp-tp1rpc.activation-config.service プロパティ

●アプリケーション統合属性ファイルを設定する場合

アプリケーション統合属性ファイルの要素に値を設定します。

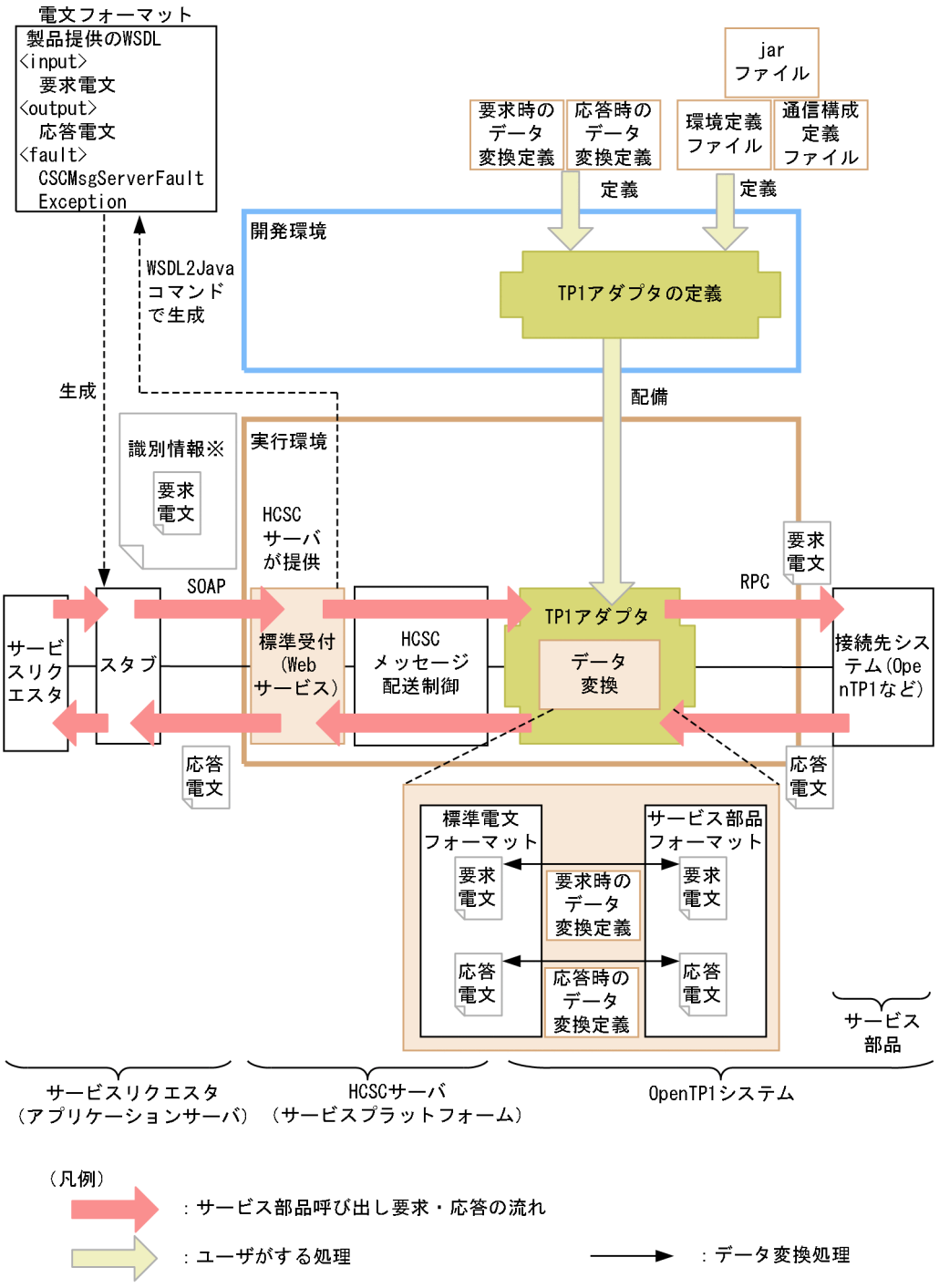
設定する要素については，次に示す XPath 式を参照してください。

- //hitachi-application-all-property/ejb-jar/hitachi-ejb-jar-property/display-name[.='cscmsg_urecp_tp1rpc']/../../hitachi-message-bean-property/display-name[.='CSCMsgTp1RpcServiceDelivery']/../activation-config/activation-config-property/activation-config-property-name[.='service']/../activation-config-property-value

2.7.3 HCSC サーバから TP1 アダプタを使った TP1 サービス部品の呼び出し

TP1 アダプタを使用することで，OpenTP1 を使用したシステムのサービス部品を呼び出せます。HCSC サーバからの OpenTP1 システムのサービス部品の呼び出しを次の図に示します。

図 2-57　HCSC サーバからの OpenTP1 システムのサービス部品の呼び出し



注※　識別情報とは，サービス名，クライアント相関IDなどです。

開発環境で TP1 アダプタを作成する手順については，マニュアル「サービスプラットフォーム 開発ガイド 受付・アダプタ定義編」の「3.　サービスアダプタの定義」を参照してください。

開発環境での操作については，マニュアル「サービスプラットフォーム 開発ガイド 基本開発編」を参照してください。

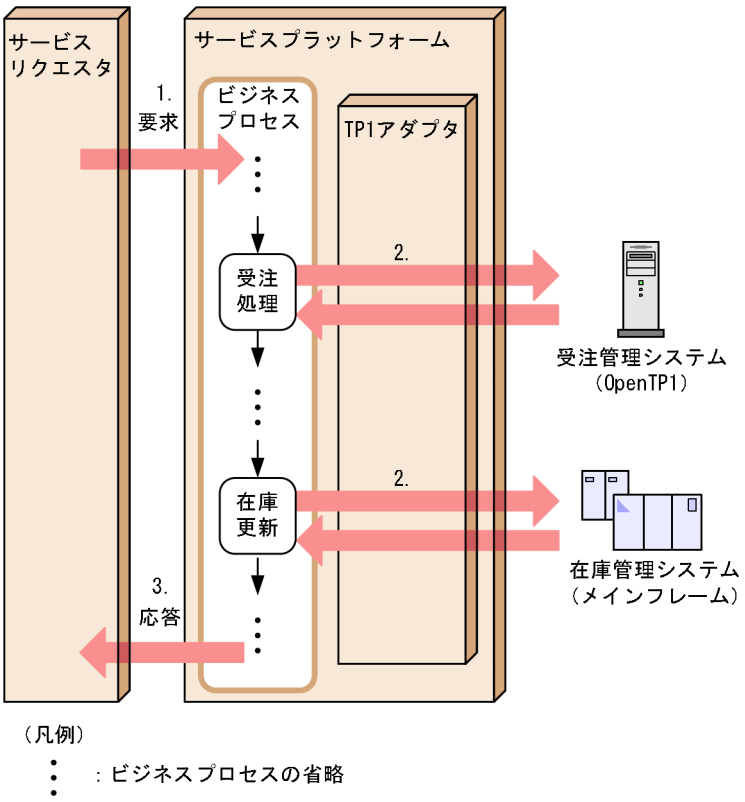
実行環境への配備方法については，マニュアル「サービスプラットフォーム システム構築・運用ガイド」の「3.1.8　サービスアダプタを配備する」を参照してください。

(1)　TP1 アダプタを使用したシステムの運用例

TP1 アダプタは，OpenTP1 上で動作する既存のシステムをサービスプラットフォーム上のサービスとして呼び出す以外に，メインフレーム（TMS-4V，DCCM3 など）上で動作する既存のシステムをサービスプラットフォーム上のサービスとして呼び出すこともできます。

TP1 アダプタを使用したシステムの運用例を次の図に示します。

図 2-58　TP1 アダプタの運用例



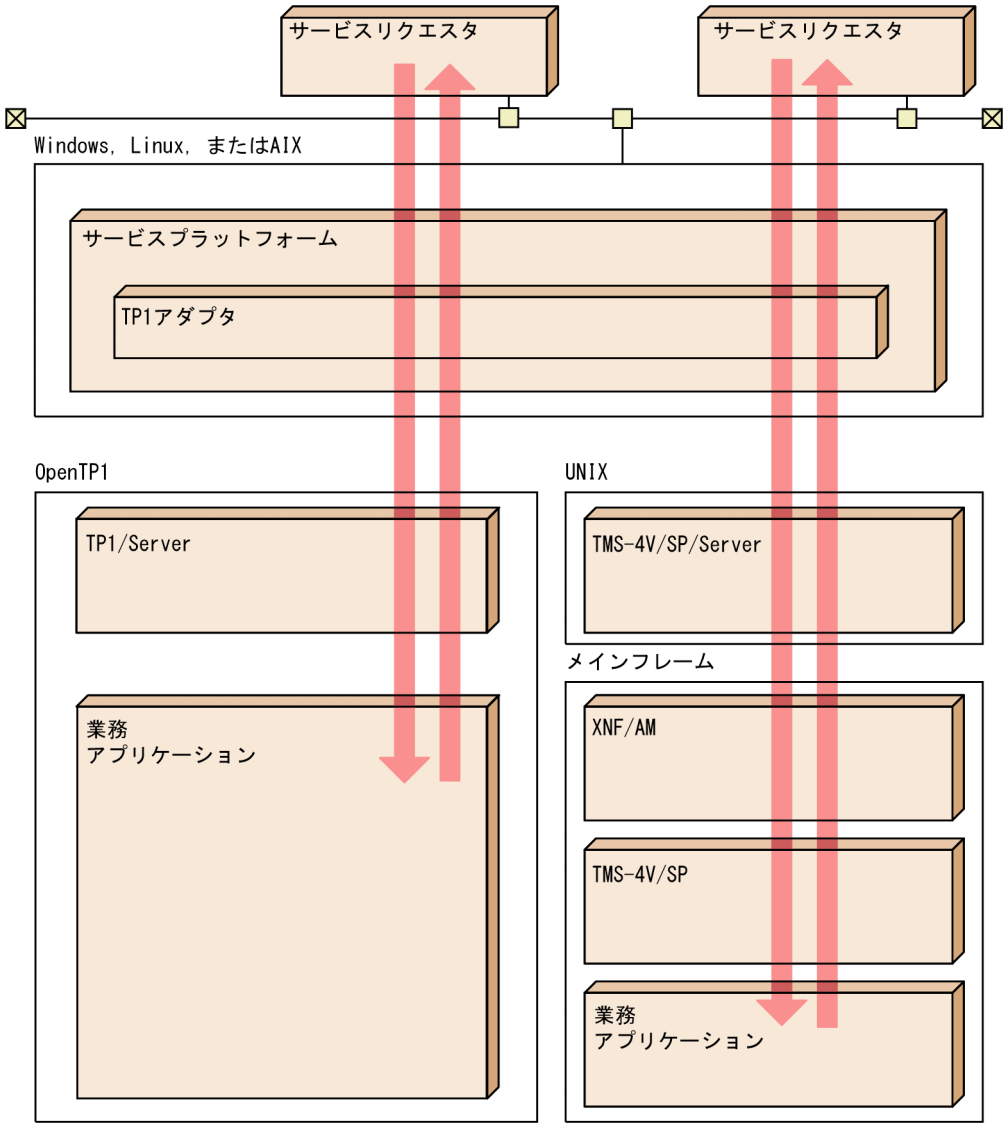
図内の各処理について説明します。

1. サービスリクエスタからの要求によって，ビジネスプロセスが起動されます。
2. ビジネスプロセスから TP1 アダプタを介して，OpenTP1 上で動作する受注管理システムおよびメインフレーム上で動作する在庫管理システムがサービスとして呼び出されます。
3. サービスの呼び出し結果がサービスリクエスタに送信されます。

これによって，OpenTP1，メインフレームなどの既存システムのアプリケーションをサービスプラットフォームのサービスとして実行できます。

サービスプラットフォーム環境で，TP1 アダプタを使用する場合のシステムの構成例を次の図に示します。

図 2-59　TP1 アダプタを使用したシステムの構成例



OpenTP1 の概要については，マニュアル「OpenTP1 Version 7 分散トランザクション処理機能 OpenTP1　解説」を参照してください。また，TMS-4V/SP/Server については，マニュアル「TMS-4V/SP/Server」を参照してください。

(2)　TP1 アダプタの RPC 通信

TP1 アダプタは，RPC 通信を使用して，サービスプラットフォームから OpenTP1 のシステムを利用できます。

(a)　RPC のサービス要求方式

TP1 アダプタの RPC 通信では，接続先システムによって，サポートするサービス要求方式が次のとおり異なります。

- 接続先システムが TP1/Server Base の場合※1

- リモート API 機能
- ネームサービス機能
- スケジューラダイレクト機能
- 通信先を指定した接続

- 接続先システムが OpenTP1 と RPC 通信できるメインフレームシステム（TMS-4V，DCCM3 など）の場合※2
  - リモート API 機能

注※1

TP1/Server Base との接続では，TP1/Client/J 環境定義でネームサービス機能およびスケジューラダイレクト機能を選択します。

注※2

メインフレームシステム（TMS-4V，DCCM3 など）との接続では，端末識別子情報を付加した通信はできません。

注意事項

これらの機能は，Service Architect の機能を使用しています。ただし，TP1 アダプタでは，次に示す機能をサポートしていません。

- RPC 通信の連鎖型 RPC 機能
- トランザクション制御機能
- XA リソースサービス機能
- TP1/Web 接続機能
- 一方通知受付機能
- 動的定義変更機能

また，TP1 アダプタに同梱されている Service Architect のモジュールを使用して，ユーザプログラムを作成しないでください。

●リモート API 機能

TP1 アダプタと接続先システムの間にコネクションを確立します。そのあと，TP1 アダプタが発行した API を接続先システムに転送して，接続先システム側のプロセスで実行できます。ただし，接続先システムがリモート API 接続に対応している必要があります。

TP1 アダプタでは，TP1 アダプタの通信構成定義によって，次の設定ができます。

●コネクションの確立方式の指定

確立するコネクションを次に示すどちらかから指定できます。

- 常設コネクション

- 非常設コネクション

●コネクションの管理方式の指定

コネクションの管理方式を次に示すどちらかから指定できます。

- オートコネクトモード
- 非オートコネクトモード

確立するコネクションによって，指定できる管理方式は次のとおり異なります。

- **常設コネクションを確立する場合**

  コネクションの確立を自動化するオートコネクトモード，または明示的に確立要求を行う非オートコネクトモードのどちらかから指定できます。

  オートコネクトモードを指定した場合の処理の概要を「図 2-60　常設コネクションを確立した場合の処理概要（オートコネクトモード）」，非オートコネクトモードを指定した場合の処理の概要を「図 2-61　常設コネクションを確立した場合の処理概要（非オートコネクトモード）」に示します。

  また，接続先動的変更を使用した場合，常設コネクションの確立はサービス要求時に行われます。ただし，確立した常設コネクションはコネクションプールがいっぱいになるまで保持されるため，2 回目以降の同一接続先へのサービス要求については，確立済みの常設コネクションが使用されます。接続先動的変更については，「2.7.6　TP1 アダプタの接続先動的変更」を参照してください。

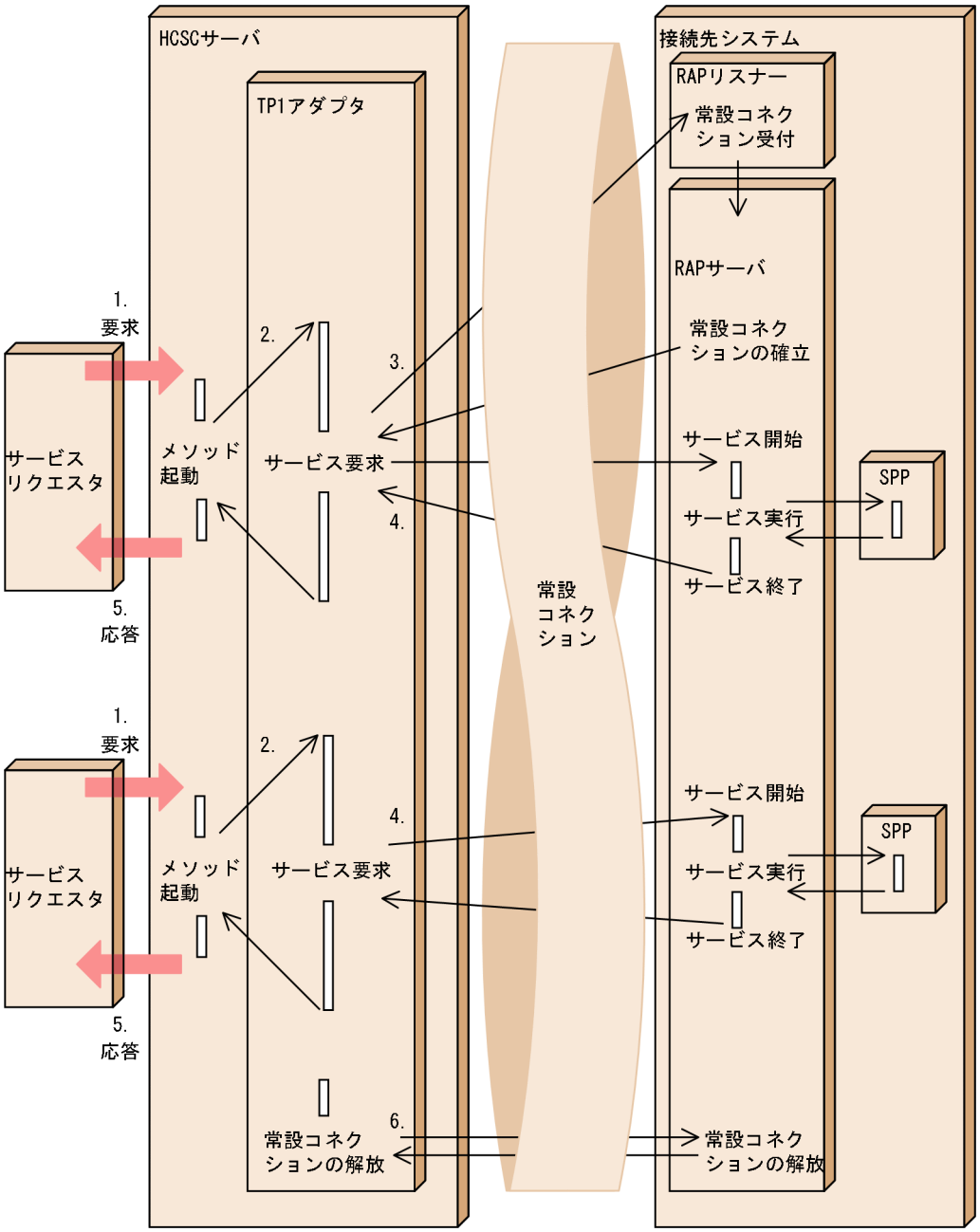
- **非常設コネクションを確立する場合**

  コネクションの管理方式は指定できません。コネクションは，非オートコネクトモードで確立されます。

  非常設コネクションを確立した場合の処理の概要を「図 2-62　非常設コネクションを確立した場合の処理の概要（非オートコネクトモード）」に示します。

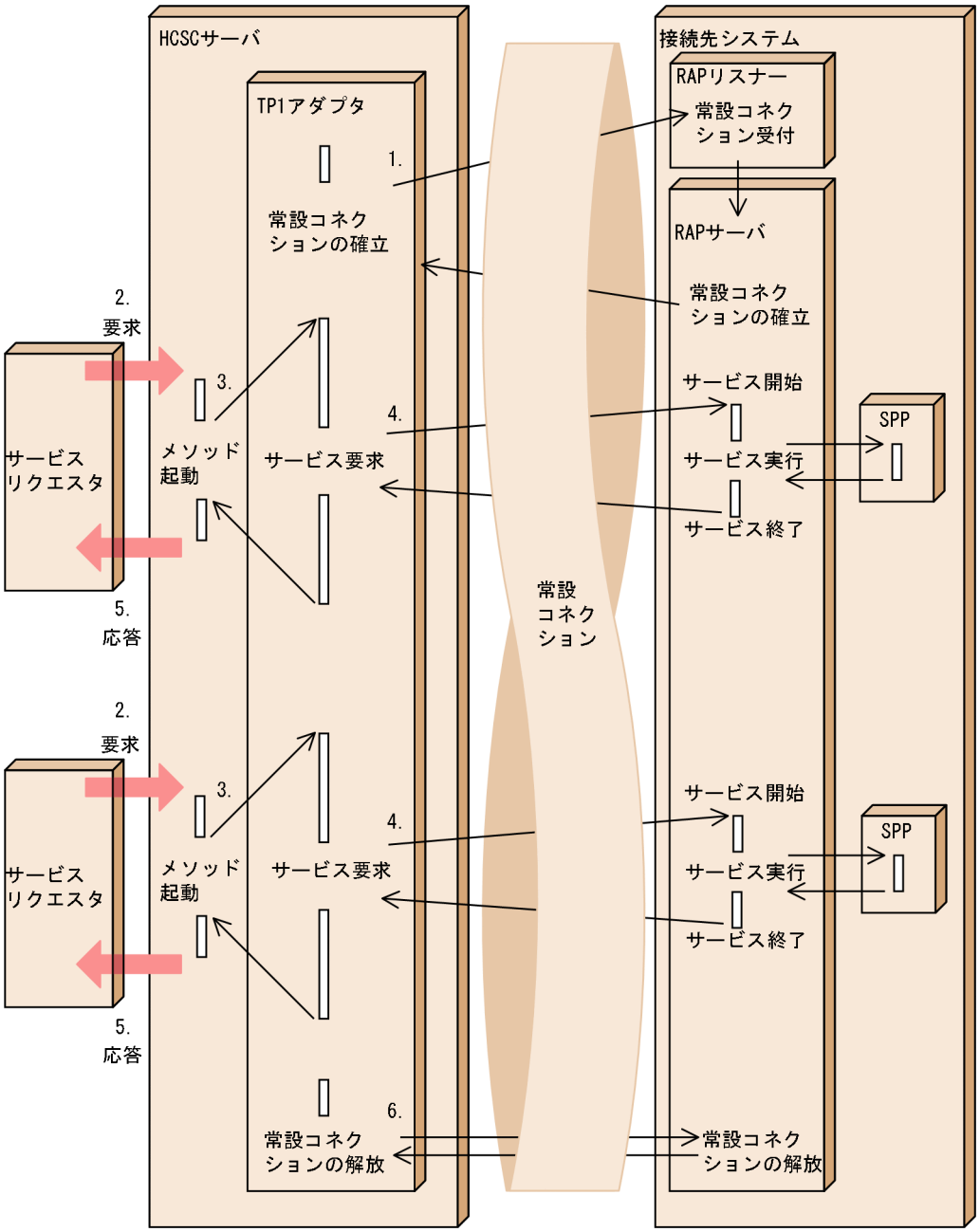
なお，TP1 アダプタの通信構成定義の詳細は，マニュアル「サービスプラットフォーム 開発ガイド 受付・アダプタ定義編」の「3.3.6　TP1 アダプタを定義する」を参照してください。

図 2-60　常設コネクションを確立した場合の処理概要（オートコネクトモード）



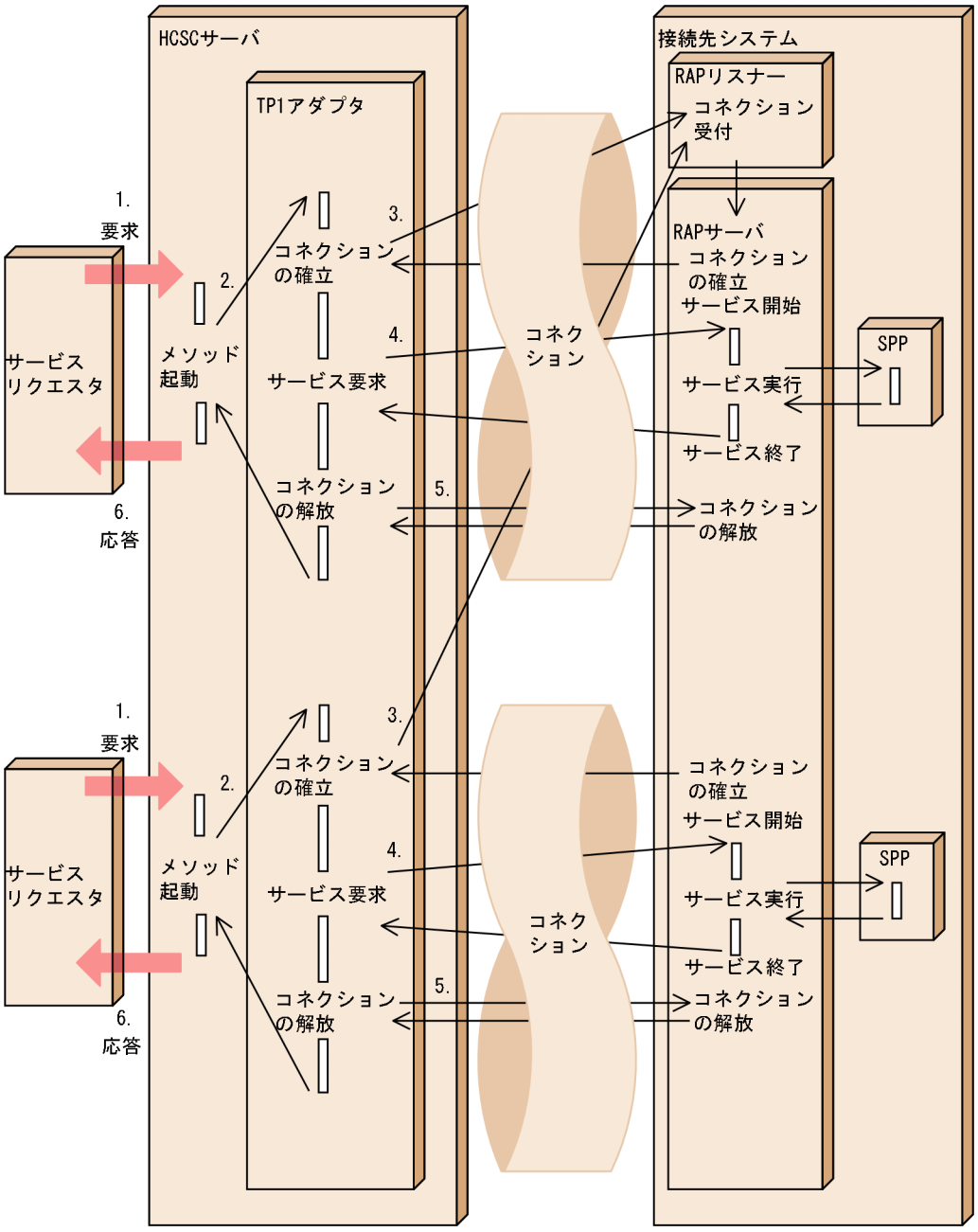
1. サービスリクエスタからの要求を HCSC サーバで受け付けます。
2. HCSC サーバで TP1 アダプタのメソッドを起動します。
3. TP1 アダプタで最初のサービス実行時に常設コネクションを確立します。
4. RPC 通信を行い，サービスを実行します。
5. RPC 通信の結果（応答データ）をサービスリクエスタに応答します。
6. TP1 アダプタの停止時に常設コネクションを解放します。

図 2-61　常設コネクションを確立した場合の処理概要（非オートコネクトモード）



1. TP1 アダプタを開始したときに，常設コネクションを確立します。
2. サービスリクエスタからの要求を HCSC サーバで受け付けます。
3. HCSC サーバで TP1 アダプタのメソッドを起動します。
4. RPC 通信を行い，サービスを実行します。
   接続先動的変更を使用した場合は，必要に応じて RPC 通信に先立ち常設コネクションを確立します。
5. RPC 通信の結果（応答データ）をサービスリクエスタに応答します。
6. TP1 アダプタの停止時に常設コネクションを解放します。

図 2-62　非常設コネクションを確立した場合の処理の概要（非オートコネクトモード）



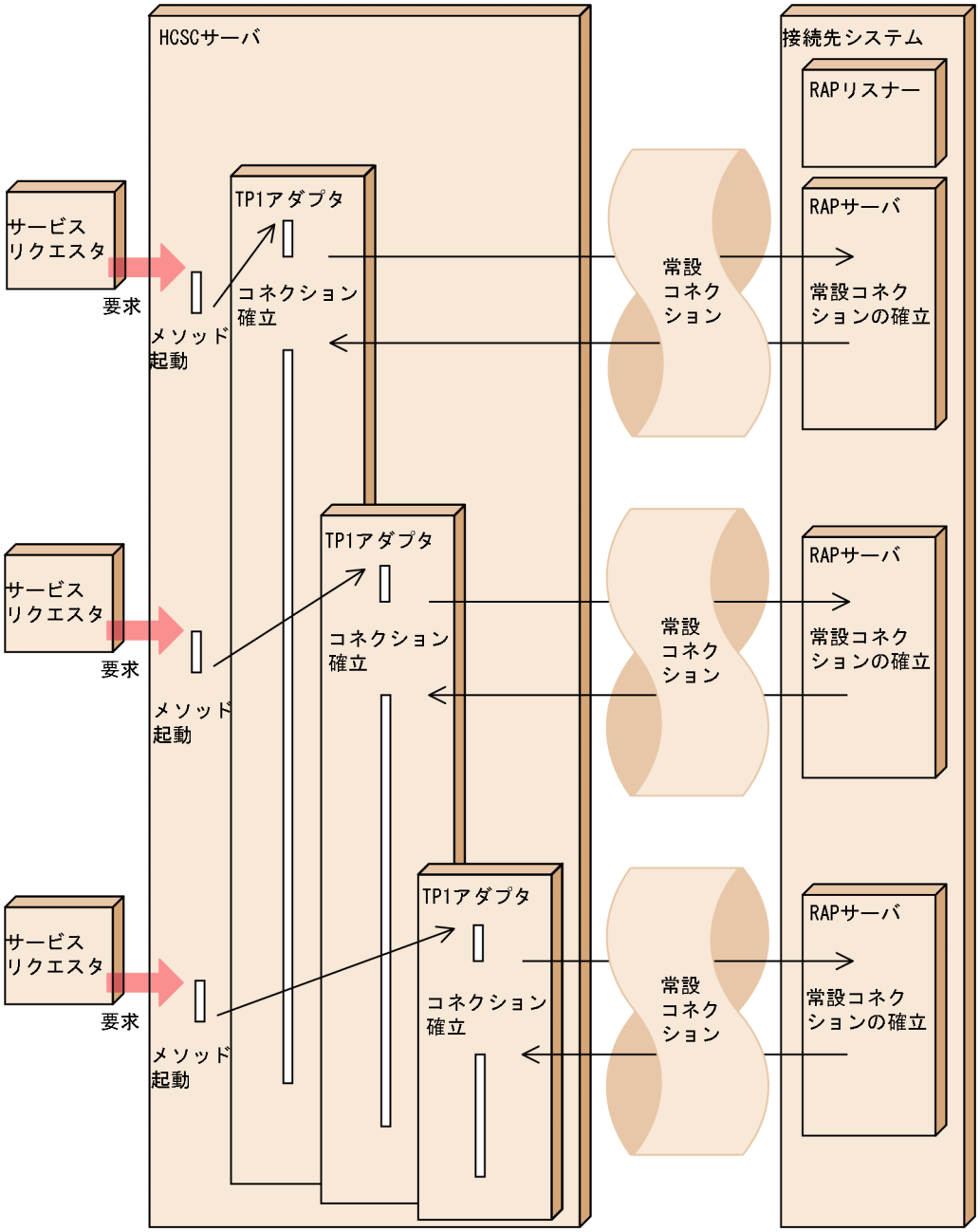
1. サービスリクエスタからの要求を HCSC サーバで受け付けます。
2. HCSC サーバで TP1 アダプタのメソッドを起動します。
3. 接続先システム（RAP リスナー）にコネクションの確立要求を行います。
4. RPC 通信を行い，サービスを実行します。
5. 確立したコネクションの解放要求を行います。
6. RPC 通信の結果（応答データ）をサービスリクエスタに応答します。

コネクションの管理方式にオートコネクトモードを指定した場合，リモート API 接続時のコネクション確立は，サービス要求を受け付けたときの RPC 通信に先だって行われます。したがって，オートコ

ネクトモードを指定した通信環境では，TP1 アダプタ運用中に多数のサービス要求を一時期に受け付けた場合，サービス要求を受け付けた数だけコネクション確立処理を行います。

複数のコネクションが確立された場合の処理の概要を次の図に示します。

図 2-63　複数のコネクションが確立された場合の処理の概要（オートコネクトモード）

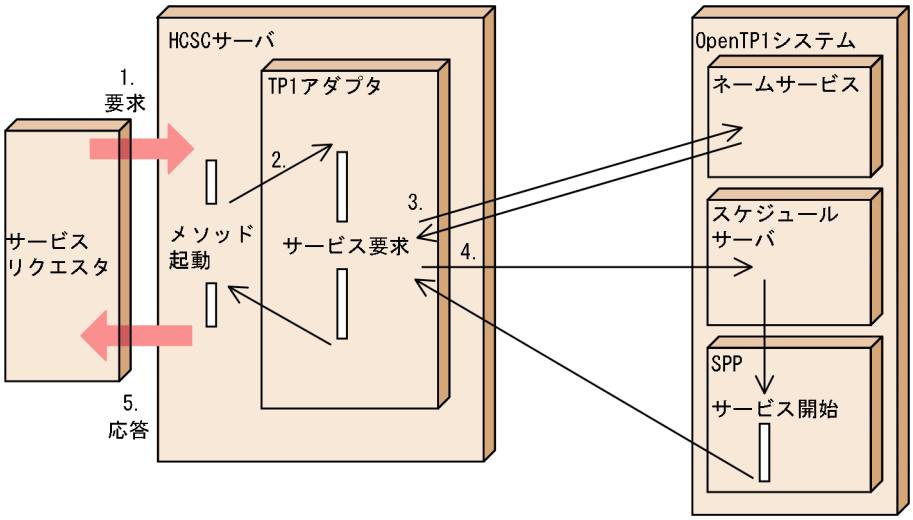


●ネームサービス機能

TP1 アダプタは，接続先の OpenTP1 システムに対して，サービス要求時にサービスグループ名およびサービス名を指定した上でサーバ UAP にサービスを要求します。ネームサービスでサーバ UAP があるノードのネットワークアドレスとサービス名を管理しているため，ネットワーク上のどのノードにサーバ UAP があるかを TP1 アダプタで意識する必要がありません。

ネームサービス機能を使用した場合の処理の概要を次の図に示します。

図 2-64　ネームサービス機能を使用した場合の処理の概要



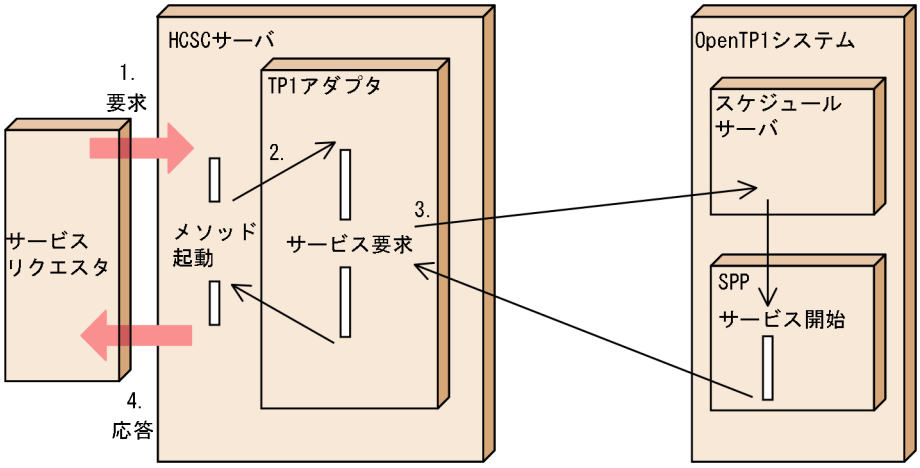
1. サービスリクエスタからの要求を HCSC サーバで受け付けます。
2. HCSC サーバで TP1 アダプタのメソッドを起動します。
3. ネームサービスにサービスの所在を問い合わせます。
4. RPC 通信を行い，サービスを実行します。
5. RPC 通信の結果（応答データ）をサービスリクエスタに応答します。

●スケジューラダイレクト機能

接続先 OpenTP1 システムのスケジューラデーモンを直接指定してサービスを要求します。

スケジューラダイレクト機能を使用した場合の処理の概要を次の図に示します。

図 2-65　スケジューラダイレクト機能を使用した場合の処理の概要



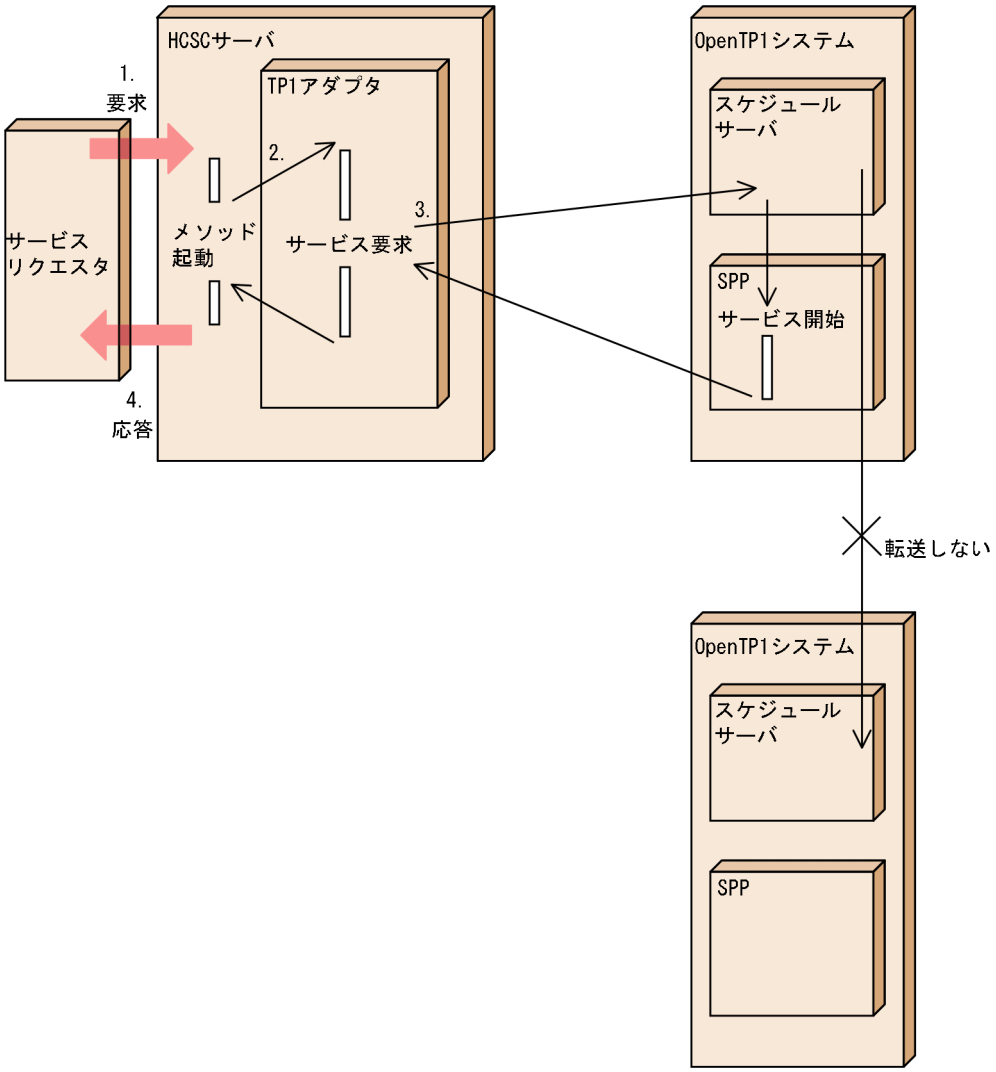
1. サービスリクエスタからの要求を HCSC サーバで受け付けます。
2. HCSC サーバで TP1 アダプタのメソッドを起動します。
3. RPC 通信を行い，サービスを実行します。
4. RPC 通信の結果（応答データ）をサービスリクエスタに応答します。

●通信先を指定した接続

接続先 OpenTP1 システムのスケジューラデーモンを直接指定してサービスを要求します。スケジューラダイレクト機能との違いは，通信先のスケジューラデーモンが持つノードの SPP に異常が発生した場合も，ほかのスケジューラデーモンに転送しないという点です。

接続先 OpenTP1 システムのスケジューラデーモンを指定した場合の接続処理の概要を次の図に示します。

図 2-66　通信先を指定した場合の接続処理の概要



1. サービスリクエスタからの要求を HCSC サーバで受け付けます。
2. HCSC サーバで TP1 アダプタのメソッドを起動します。
3. 通信先を指定した RPC 通信を行い，サービスを実行します。
4. RPC 通信の結果（応答データ）をサービスリクエスタに応答します。

(b)　RPC の通信形態

TP1 アダプタでは，RPC の通信形態として次の通信形態をサポートします。

- 同期応答型 RPC 通信

- 非応答型 RPC 通信

RPC の通信形態は，TP1 アダプタの設定でオペレーション情報の通信モデルで選択します。詳細は，マニュアル「サービスプラットフォーム 開発ガイド 受付・アダプタ定義編」の「3.3.6　TP1 アダプタを定義する」のサービスアダプタ定義画面での操作について説明している個所を参照してください。

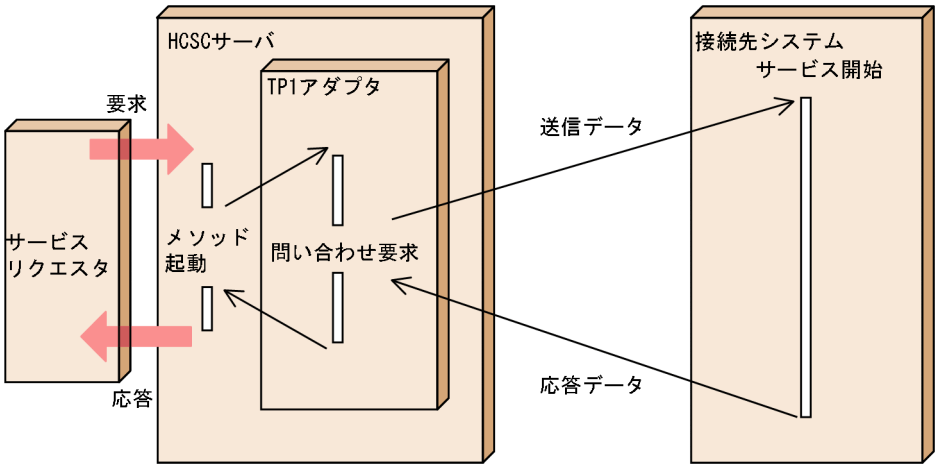
実行環境への配備方法については，マニュアル「サービスプラットフォーム システム構築・運用ガイド」の「3.1.8　サービスアダプタを配備する」を参照してください。

●同期応答型 RPC 通信

TP1 アダプタから接続先システムに問い合わせデータを送信して，問い合わせた応答データを受け取る通信形態です。TP1 アダプタ側では，接続先システムからの応答を受け取るまでメソッドを終了しません。

同期応答型 RPC 通信をする場合の処理の概要を次の図に示します。

図 2-67　同期応答型 RPC 通信をする場合の処理の概要

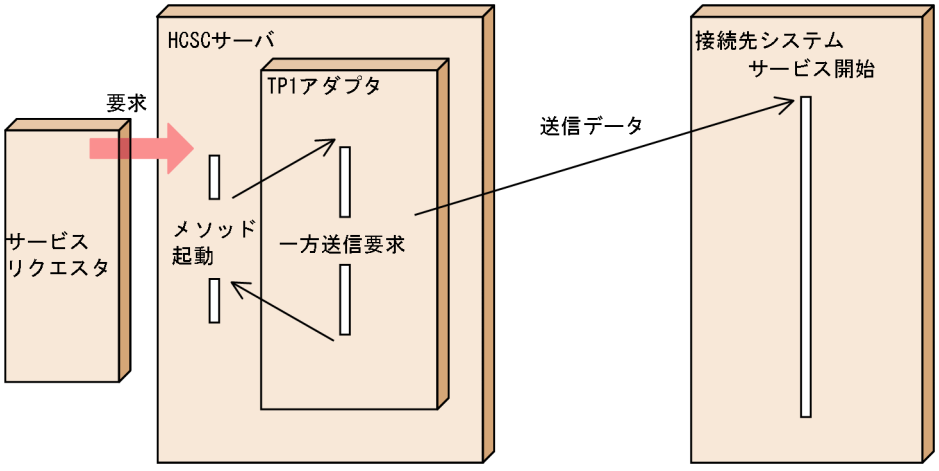


●非応答型 RPC 通信

TP1 アダプタから接続先システムに一方送信データを送信して，応答は受け取らない通信形態です。TP1 アダプタ側では，データの送信が完了するとメソッドは終了します。

非応答型 RPC 通信をする場合の処理の概要を次の図に示します。

図 2-68　非応答型 RPC 通信をする場合の処理の概要



(c)　RPC の送受信バッファの生成

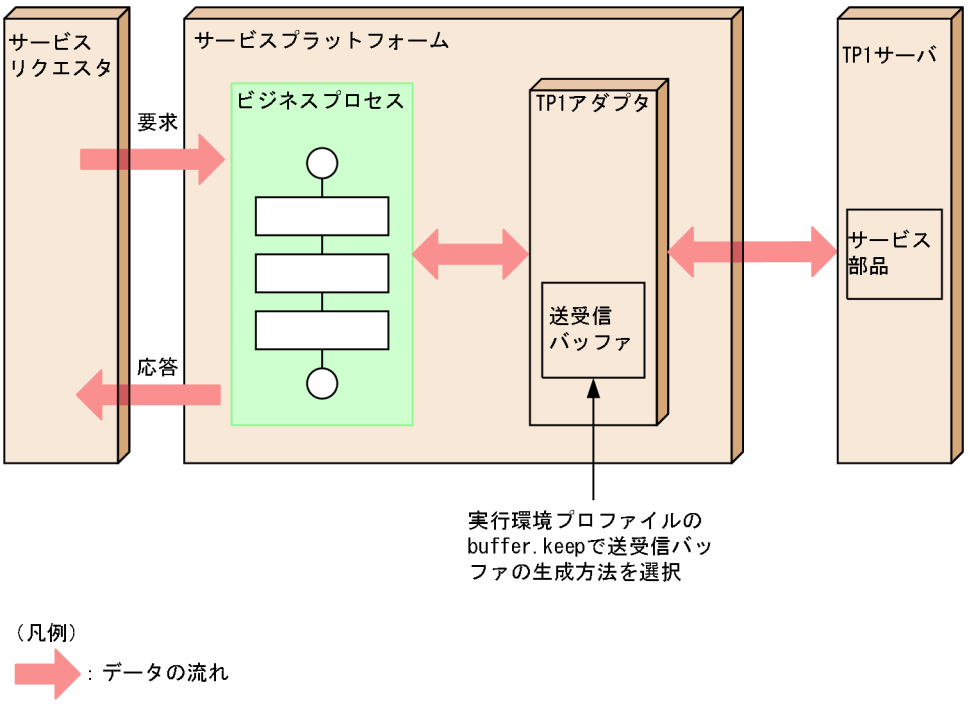
TP1 サービスプログラムとの RPC 通信で使用する送信バッファと受信バッファは，TP1 アダプタ起動時のインスタンス初期化処理で生成・保持することによって，運用中のバッファのメモリ確保失敗のリスクを排除しています。しかし，この送受信バッファはインスタンスごとに生成・保持されるため，TP1 アダプタ起動時にバッファサイズ分のメモリ量が必要となります。

このため，多くの TP1 アダプタを起動したり，バッファサイズの大きい TP1 アダプタのインスタンスを複数生成したりすると，メモリ不足が原因で TP1 アダプタを起動できないことがあります。

この場合，送受信バッファの生成方法を TP1 アダプタ起動時のインスタンス初期化処理での生成・保持から，RPC 通信処理時に動的に生成・解放する処理に変更するオプションを使用することで，TP1 アダプタ起動時のメモリ不足を解消できます。

RPC 通信処理時に動的に生成・解放するオプションの概要を次の図に示します。

図 2-69　RPC 通信処理時に動的に生成・解放するオプションの概要



RPC 通信時に動的に送受信バッファを生成・解放するには，TP1 アダプタ実行環境プロパティファイルの buffer.keep プロパティで送受信バッファの生成オプションの定義を追加する必要があります。定義方法については，マニュアル「サービスプラットフォーム 開発ガイド 受付・アダプタ定義編」の「3.3.6　TP1 アダプタを定義する」の TP1 アダプタ実行環境プロパティファイルの作成について説明している個所を参照してください。

注意事項

RPC 通信時に送受信バッファを生成する設定にした場合は，動的に生成される送受信バッファ分のメモリ量を考慮してメモリのリソース設計をするようにしてください。送受信バッファに必要なメモリ量の目安を次に示します。

送受信バッファサイズ（単位：バイト）＝同時に処理できるRPC要求の最大数※1×（送信バッファサイズ※2＋受信バッファサイズ※3）

注※1

通信構成定義ファイルの<con_pool_num>タグの値を示します。

注※2

通信構成定義ファイルの<send_buff_size>タグの値を示します。

注※3

通信構成定義ファイルの<recv_buff_size>タグの値を示します。

RPC 通信時に送受信バッファの生成をする設定の有無によって，TP1 アダプタの最大メモリ所要量が変わることはありません。また，複数の TP1 アダプタを使用する場合は，各送受信バッファサイズを総和したメモリ量が，全体のバッファサイズとして必要になります。

(3) MHP を接続先とした TCP/IP 通信

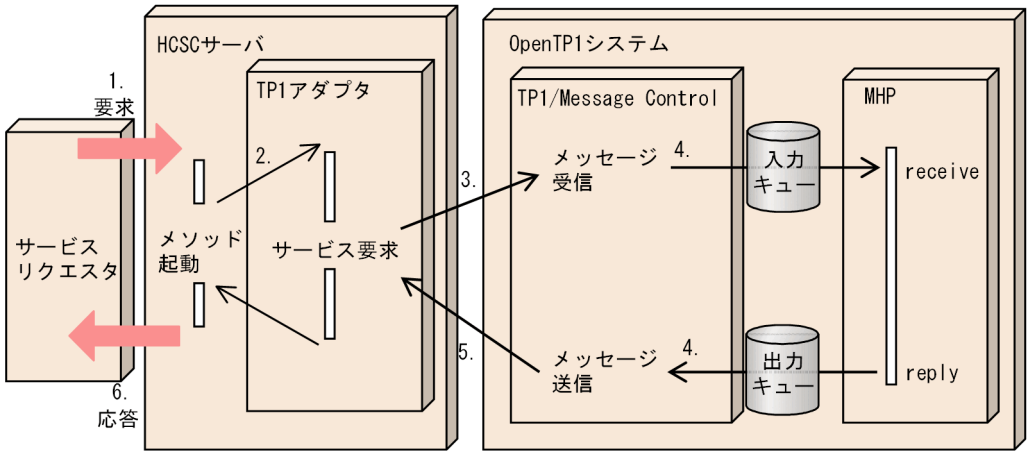
TP1 アダプタは，TCP/IP 通信を使用して，サービスプラットフォームから OpenTP1 のシステムを利用できます。TP1 アダプタから通信先の OpenTP1 システムの TP1/Message Control に対して，メッセージの送受信を行い，MHP（Message Handling Program）を呼び出すことができます。

MHP を接続先とした TCP/IP 通信は，通信先が TP1/NET/TCP/IP の受信メッセージの組み立て機能を使用している場合に使用できます。

受信メッセージの組み立て機能については，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。

MHP を接続先とした TCP/IP 通信処理の概要を次の図に示します。

図 2-70　MHP を接続先とした TCP/IP 通信処理の概要



1. サービスリクエスタからの要求を HCSC サーバで受け付けます。
2. HCSC サーバで TP1 アダプタのメソッドを起動します。
3. MHP を呼び出す TCP/IP 通信（cltAssemSend メソッドを使用）を行います。
4. 受信したメッセージをキューに格納し，MHP を起動してメッセージを送信します。
5. MHP で受信したメッセージを基にサービスを実行します。
6. TCP/IP 通信の結果（応答データ）をサービスリクエスタに応答します。

(a)　MHP を接続先とした TCP/IP 通信の通信形態

TP1 アダプタでは，MHP を接続先とした TCP/IP 通信として次の通信形態をサポートします。

- 応答型 MHP サービス（ans 型）
- 非応答型 MHP サービス（noans 型）

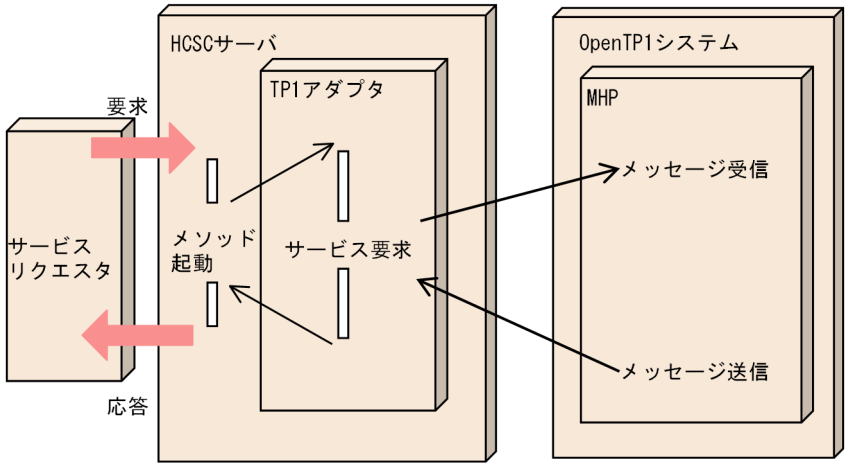
どちらの通信形態を使用するかは，サービスリクエスタから TP1 アダプタへの要求で応答領域の設定によって判断します。応答領域が設定されている場合は，応答型 MHP サービス（ans 型）を呼び出します。応答領域に NULL（設定なし）が設定されている場合は，非応答型 MHP サービス（noans 型）を呼び出します。

なお，MHP サービスには，通信先システムとのメッセージの受送信を連続させる継続問い合わせ応答型 MHP サービス（cont 型）がありますが，サービスプラットフォームからのサービス呼び出しでは使用できません。

●応答型 MHP サービス（ans 型）との通信

TP1 アダプタから受信したメッセージを処理し，処理したメッセージを応答するサービスと通信します。

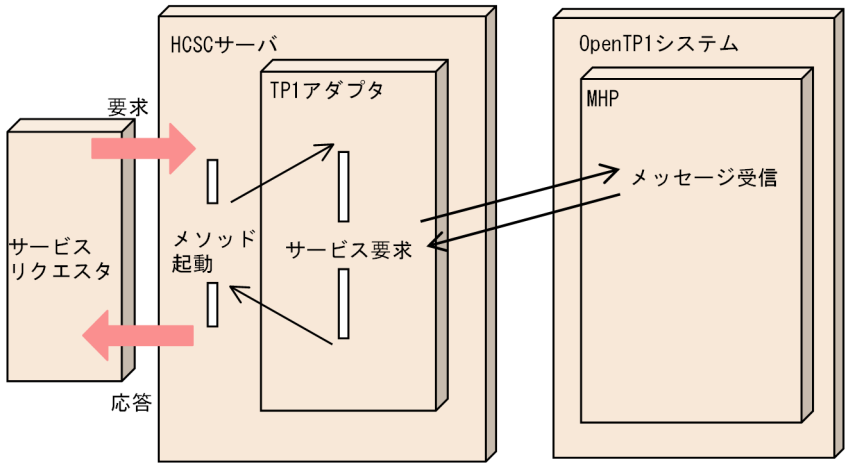
図 2-71　応答型 MHP サービス（ans 型）との通信



●非応答型 MHP サービス（noans 型）との通信

TP1 アダプタから受信したメッセージを処理し，応答を返さないサービスと通信します。

図 2-72　非応答型 MHP サービス（noans 型）との通信



(4)　時間監視

TP1 アダプタから接続先システムへ問い合わせを送信してから，応答を受信するまでの経過時間を監視します。経過時間を監視することで，接続先システム上のアプリケーションがループ，停止，または回線障害になった場合に，サービスリクエスタが無限待ちになることを防止します。

TP1 アダプタで適用する監視時間の値は，通信処理の状況や接続形態によって異なります。通信処理が開始中または停止中の場合と運用中の場合に分けて説明します。

指定した監視時間内に接続先システムからの応答がない場合，TP1 アダプタから HCSC サーバへエラー情報または例外を通知します。

(a) 通信処理が開始中および停止中の場合

TP1/Client/J 環境定義ファイルで指定する最大応答待ち時間の値（dcwatchtim オペランドの指定値）を監視時間の値として適用します。通信処理の状況によって，監視時間の値は，次の表に示すとおりに適用されます。

表 2-11 通信処理の開始中および停止中に監視時間として適用される値

通信処理		接続形態			
通信状況	コネクション状況	リモート API 接続			通常 RPC 接続
		常設コネクション		非常設コネクション	
		オートコネクト	非オートコネクト		
開始中	コネクション確立	－	－	－	－
	コネクション解放	－	○※	－	－
停止中	コネクション解放	○	○	－	－

（凡例）

○：時間監視できます。

－：時間監視できません。

注※

通信障害が発生した場合に適用されます。

(b) 通信処理が運用中の場合

TP1 アダプタの通信構成定義ファイルで指定したサービス要求のオペレーションに対する時間監視値（watch_time 要素で指定）を監視時間の値として適用します。通信処理の状況によって，監視時間の値は，次の表に示すとおりに適用されます。

表 2-12 通信処理の運用中に監視時間に適用される値

通信処理		接続形態			
通信状況	コネクション状況	リモート API 接続			通常 RPC 接続
		常設コネクション		非常設コネクション	
		オートコネクト	非オートコネクト		
運用中	コネクション確立	－	－	－	－
	RPC 通信	○	○	○	○
	コネクション解放	○※	○※	○	－

（凡例）

○：時間監視できます。

－：時間監視できません。

注※

通信障害が発生した場合に適用されます。

(5) TP1 アダプタ使用時の注意事項

- リモート API 接続で TP1 アダプタのコネクション解放が失敗した場合，TP1 アダプタではコネクション解放状態となりますが，接続先のシステムではコネクションの解放を検知していない場合があります。そのため，コネクション状態の不整合が発生し，接続先システムでの rap サーバリソース不足などで，次の通信処理が失敗するおそれがあります。その場合は接続先システムでコネクション確立状態を一度初期化してください。
- リモート API 接続で TP1 アダプタ運用中に障害が発生して，正常に停止処理がされなかった場合，接続先システムでコネクションの解放を検知していない場合があります。その場合は接続先システムでコネクション確立状態を一度初期化してください。
- J2EE サーバ用ユーザプロパティファイル（usrconf.properties）の ejbserver.container.passivate.scan.interval プロパティを，デフォルトの 0 から変更しないでください。

2.7.4 HCSC サーバからの MDB（DB キュー）サービスアダプタを使った TP1 サービス部品の呼び出し

同期応答型のサービス部品呼び出しのほかに，非同期で HCSC サーバから，OpenTP1 システムのサービス部品を呼び出せます。非同期で HCSC サーバから，OpenTP1 システムのサービス部品を呼び出すには，HCSC サーバの非同期 MDB（DB キュー）アダプタを実行します。

詳細は，「2.5 DB キューを使用したサービスとの接続」を参照してください。

2.7.5 TP1 アダプタのオペレーション動的変更

TP1 アダプタの実行オペレーションをビジネスプロセス内で動的に変更できます。

TP1 アダプタのオペレーション動的変更の詳細については，「2.22 TP1 アダプタおよび Message Queue アダプタのオペレーション動的変更機能」を参照してください。

2.7.6 TP1 アダプタの接続先動的変更

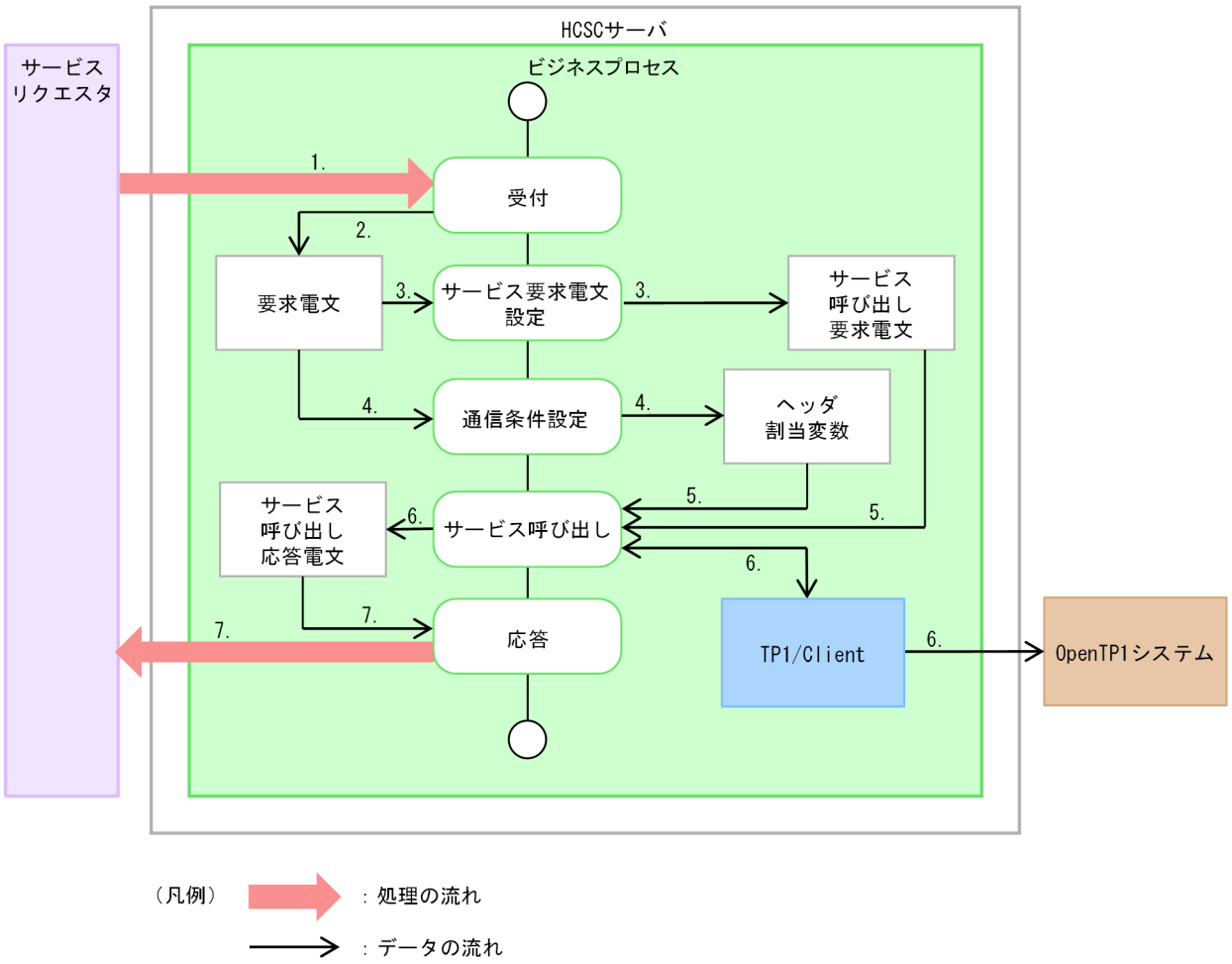
TP1/Client が使用する OpenTP1 システムとの RPC 通信条件を，ヘッダ割当変数に設定することで動的に変更できます。

なお，接続先動的変更では，通信接続方式は変更できません。

(1) TP1 アダプタの接続先動的変更の処理の流れ

TP1 アダプタの接続先動的変更の処理の流れを次の図に示します。

図 2-73 TP1 アダプタの接続先動的変更の処理の流れ



1. サービスリクエスタがビジネスプロセスにサービス部品の実行を要求します。
2. 受付アクティビティは，受け取ったサービスリクエスタからの要求電文を要求電文の変数に格納します。
3. 要求電文変数から，TP1 アダプタのサービス呼び出しの実行に使用するサービス呼び出し要求電文（ボディ割当変数）を作成します。
4. 要求電文変数から，各種接続用パラメタを決定し，データ変換アクティビティなどで，ヘッダ割当変数に格納します。
5. TP1 アダプタを呼び出します。
   サービス呼び出しでは，要求電文のヘッダ割当変数として，4.で定義したヘッダ割当変数を指定します。
6. TP1 アダプタは，指定された接続用パラメタを TP1/Client に設定し，TP1/Client から OpenTP1 システムとの RPC 通信を行います。また，応答をサービス呼び出し応答電文の変数に格納します。
7. 応答アクティビティは，サービス呼び出し応答電文をサービスリクエスタに返します。

(2) ヘッダ割当変数の定義

TP1 アダプタ接続先動的変更ヘッダ変数スキーマを使用して，ヘッダ割当変数を定義します。TP1 アダプタ接続先動的変更ヘッダ変数スキーマは「<サービスプラットフォームのインストールディレクトリ>¥CSC¥custom-adapter¥TP1¥schema¥templates¥adptp1_dynamic_header.xsd」に格納されています。

TP1 アダプタ接続先動的変更ヘッダ変数スキーマの定義例を次に示します。

TP1 アダプタ接続先動的変更ヘッダ変数スキーマの定義例

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- All Rights Reserved. Copyright (C) 2024. Hitachi, Ltd. -->
<xsd:schema elementFormDefault="qualified" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:tpdh="http://www.hitachi.co.jp/soft/xml/cosminexus/csc/tp1dynamics/header"
  targetNamespace="http://www.hitachi.co.jp/soft/xml/cosminexus/csc/tp1dynamics/header">
  <xsd:element name="TP1DynamicConnection">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="watch_time" minOccurs="0" maxOccurs="1" type="xsd:unsignedShort"/
>
        <xsd:element name="static_host" minOccurs="0" maxOccurs="1" type="xsd:string"/>
        <xsd:element name="host" minOccurs="0" maxOccurs="1" type="xsd:string"/>
        <xsd:element name="static_port" minOccurs="0" maxOccurs="1" type="tpdh:portNo"/>
        <xsd:element name="port" minOccurs="0" maxOccurs="1" type="tpdh:portNo"/>
        <xsd:element name="tcpip_wait_time" minOccurs="0" maxOccurs="1" type="tpdh:tcpipWait
Time"/>
        <xsd:element name="dccltinquiretime" minOccurs="0" maxOccurs="1" type="tpdh:cltInqui
reTime"/>
        <xsd:element name="dccltdelay" minOccurs="0" maxOccurs="1" type="xsd:unsignedShort"/
>
        <xsd:element name="dcwatchtim" minOccurs="0" maxOccurs="1" type="xsd:unsignedShort"/
>
        <xsd:element name="dccltextend" minOccurs="0" maxOccurs="1" type="tpdh:cltExtend"/>
        <xsd:element name="dcscdloadpriority" minOccurs="0" maxOccurs="1" type="xsd:boolean"
/>
        <xsd:element name="dcwatchtiminherit" minOccurs="0" maxOccurs="1" type="xsd:boolean"
/>
        <xsd:element name="dcwatchtimrpcinherit" minOccurs="0" maxOccurs="1" type="xsd:boole
an"/>
        <xsd:element name="dchost" minOccurs="0" maxOccurs="1" type="xsd:string"/>
        <xsd:element name="dcnamport" minOccurs="0" maxOccurs="1" type="tpdh:portNo"/>
        <xsd:element name="dcscdport" minOccurs="0" maxOccurs="1" type="tpdh:portNo"/>
        <xsd:element name="dcrapport" minOccurs="0" maxOccurs="1" type="tpdh:portNo"/>
        <xsd:element name="dccltconnectinf" minOccurs="0" maxOccurs="1" type="tpdh:cltConnec
tInfo"/>
        <xsd:element name="dcsndhost" minOccurs="0" maxOccurs="1" type="xsd:string"/>
        <xsd:element name="dcsndport" minOccurs="0" maxOccurs="1" type="tpdh:scdPort"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>

  <xsd:simpleType name="tcpipWaitTime">
    <xsd:restriction base="xsd:int">
      <xsd:minInclusive value="-1"/>
```

```
      <xsd:maxInclusive value="65535"/>
    </xsd:restriction>
  </xsd:simpleType>
  <xsd:simpleType name="cltInquireTime">
    <xsd:restriction base="xsd:unsignedInt">
      <xsd:minInclusive value="0"/>
      <xsd:maxInclusive value="1048575"/>
    </xsd:restriction>
  </xsd:simpleType>
  <xsd:simpleType name="cltExtend">
    <xsd:restriction base="xsd:unsignedByte">
      <xsd:minInclusive value="0"/>
      <xsd:maxInclusive value="1"/>
    </xsd:restriction>
  </xsd:simpleType>
  <xsd:simpleType name="portNo">
    <xsd:restriction base="xsd:unsignedShort">
      <xsd:minInclusive value="5001"/>
    </xsd:restriction>
  </xsd:simpleType>
  <xsd:simpleType name="cltConnectInfo">
    <xsd:restriction base="xsd:hexBinary">
      <xsd:maxLength value="64"/>
    </xsd:restriction>
  </xsd:simpleType>
  <xsd:simpleType name="scdPort">
    <xsd:restriction base="xsd:unsignedShort">
      <xsd:minInclusive value="1"/>
    </xsd:restriction>
  </xsd:simpleType>

</xsd:schema>
```

(3) 動的変更できる通信条件のパラメタ

RPC 通信条件は，TP1 アダプタ通信構成定義ファイル，または TP1/Client/J 環境定義ファイルに定義されています。

TP1 アダプタ通信構成定義ファイル，および TP1/Client/J 環境定義ファイルの，RPC 通信条件を変更できる項目を次に示します。

表 2-13 TP1 アダプタ通信構成定義ファイルで変更できるパラメタ

パラメタ	説明
watch_time	応答監視時間（単位：秒）
static_host	接続先ホスト名
static_port	接続先ポート番号
host	接続先ホスト名（通信先を指定した RPC の接続先動的変更機能との互換性を確保する） static_host の別名のため，該当するパラメタを変更した場合は static_host が変更されたことがログに出力されます。

パラメタ	説明
port	接続先ポート番号（通信先を指定した RPC の接続先動的変更機能との互換性を確保する） static_port の別名のため，該当するパラメタを変更した場合は static_port が変更されたことがログに出力されます。
tcpip_wait_time	メッセージ受信時の最大待ち時間（単位：秒）

表 2-14　TP1/Client/J 環境定義ファイルで変更できるオペランド

オペランド	説明
dccltinquiretime	常設コネクション問い合わせ間隔最大時間
dccltdelay	最大通信遅延時間
dcwatchtim	最大応答待ち時間
dccltextend	TP1/Client/J の機能拡張レベルの設定を指定
dcscdloadpriority	サービス要求を受け付けた窓口となる TP1/Server を優先して負荷分散するかどうかを指定
dcwatchtiminherit	リモート API 機能を使用した RPC を行う場合に，CUP の最大応答待ち時間を rap サーバ側に引き継ぐかどうかを指定
dcwatchtimrpcinherit	CUP の最大応答待ち時間を TP1/Server 側に引き継ぐかどうかを指定
dchost	窓口となる TP1/Server
dcnamport	ネームサービスのポート番号
dcscdport	スケジュールサービスのポート番号
dcrapport	rap リスナーのポート番号
dccltconnectinf	端末識別情報として，DCCM3 論理端末の論理端末名称を EBCDIK コードで指定
dcsndhost	接続する MHP が存在するノードのホスト名
dcsndport	接続する MHP のポート番号

TP1 アダプタ通信構成定義ファイル，または TP1/Client/J 環境定義ファイルについては，マニュアル「サービスプラットフォーム リファレンス」の「3.4.3　TP1 アダプタ通信構成定義ファイル」，および「付録 A.2　TP1/Client/J の定義」を参照してください。

2.7.7　TP1 アダプタの接続先情報の動的変更

TP1 アダプタの接続先は，接続先の OpenTP1 のホスト名とポート番号を TP1 アダプタに設定することで動的に変更できます。ビジネスプロセスで接続先の OpenTP1 のホスト名とポート番号を設定しておくと，TP1 アダプタを実行したときに接続先が変更されます。これによって，作成する TP1 アダプタの数を削減できます。

TP1 アダプタの接続先は，動的に変更できます。ビジネスプロセスで接続先を設定しておくと，TP1 アダプタを実行したときに接続先が変更されます。これにより，作成する TP1 アダプタの数を削減できます。

(1)　接続先動的変更の処理の流れ

TP1 アダプタの接続先動的変更の処理の流れを次に示します。

1. 要求電文内のヘッダ電文から接続先情報を取得します。
2. 取得した接続情報を基にサービスと通信を行います。

なお，接続情報がヘッダ電文に設定されていないなど，接続先情報の取得に失敗した場合は，リクエスト処理がエラーになります。

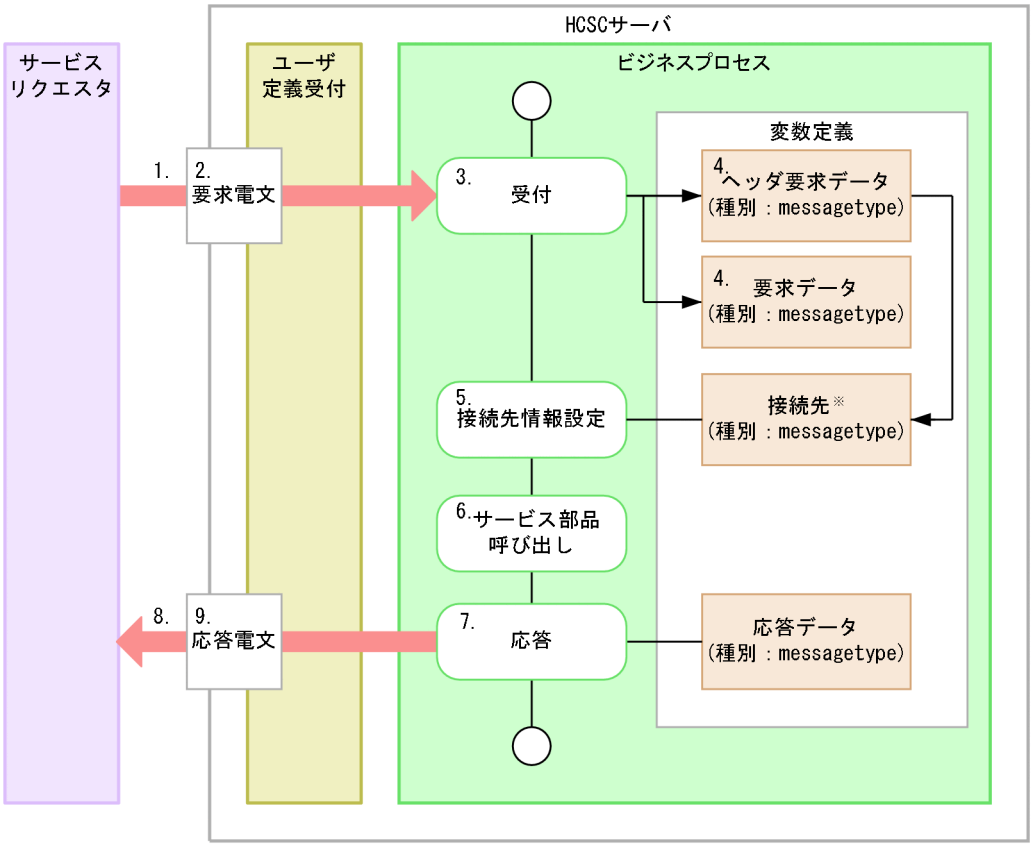
(2)　接続先情報を動的に変更する例

接続先情報ファイルを使用して，接続先情報を動的に変更する例を説明します。

(a)　ビジネスプロセスの例

接続先情報ファイルを使用するビジネスプロセスの例を次の図に示します。

図 2-74　接続先情報ファイルを使用するビジネスプロセスの例（TP1 アダプタの場合）



注※　接続先情報ファイルを使用して定義した変数

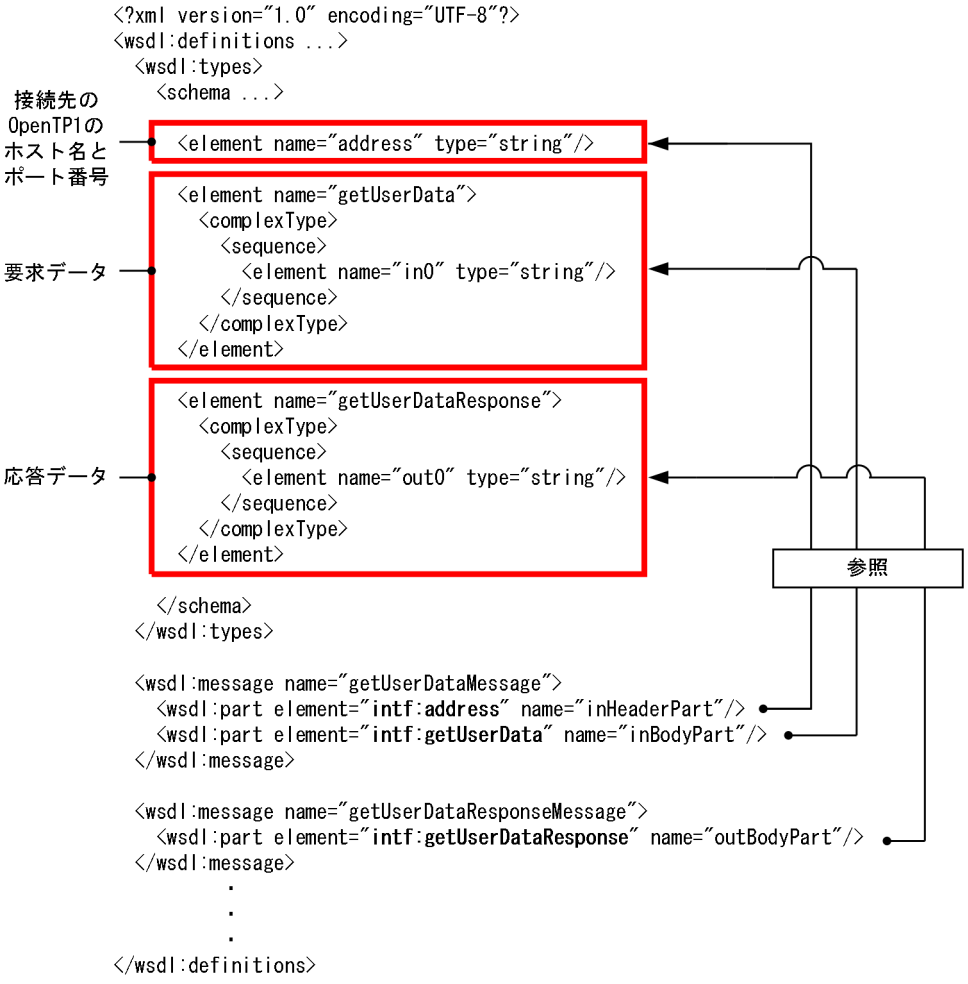
1. サービスリクエスタがビジネスプロセスにサービス部品の実行を要求します。

2. サービスリクエスタから送付する要求電文内のヘッダ電文に接続先情報（OpenTP1 のホスト名とポート番号）を設定します。
3. ビジネスプロセスは XML の要求電文を受付アクティビティで受け取ります。
4. ビジネスプロセスがヘッダ要求データおよび要求データを生成します。
5. ヘッダ要求データの接続先情報を，データ変換アクティビティなどで接続先情報変数に設定します。
6. ビジネスプロセスは接続先情報変数に含まれる接続先情報を TP1 アダプタに設定して，サービス部品を呼び出します。
7. ビジネスプロセスが，サービス部品の実行結果を XML の応答電文として構成し，応答アクティビティで返します。
8. サービスリクエスタへの応答が返されます。
9. サービスリクエスタへの応答では，XML の応答電文が送付されます。

(3) ユーザ定義受付を生成する WSDL の作成例

ユーザ定義受付を生成する WSDL の作成例を次に示します。

なお，この例では，接続先となる OpenTP1 のホスト名とポート番号を SOAP ヘッダで送付しています。




```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions ...>
  <wsdl:types>
    <schema ...>
      <element name="address" type="string"/>
      <element name="getUserData">
        <complexType>
          <sequence>
            <element name="in0" type="string"/>
          </sequence>
        </complexType>
      </element>
      <element name="getUserDataResponse">
        <complexType>
          <sequence>
            <element name="out0" type="string"/>
          </sequence>
        </complexType>
      </element>
    </schema>
  </wsdl:types>

  <wsdl:message name="getUserDataMessage">
    <wsdl:part element="intf:address" name="inHeaderPart"/>
    <wsdl:part element="intf:getUserData" name="inBodyPart"/>
  </wsdl:message>

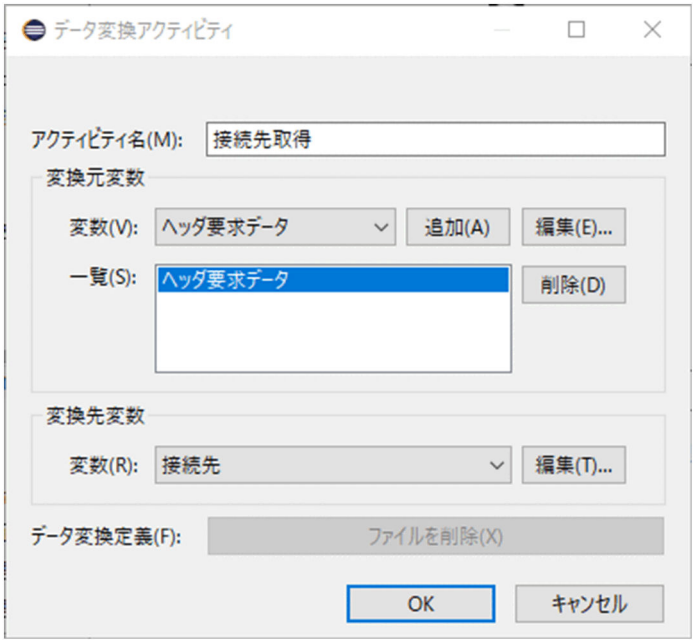
  <wsdl:message name="getUserDataResponseMessage">
    <wsdl:part element="intf:getUserDataResponse" name="outBodyPart"/>
  </wsdl:message>
          .
          .
          .
</wsdl:definitions>
```

(a)　データ変換アクティビティの設定例

「図 2-74　接続先情報ファイルを使用するビジネスプロセスの例（TP1 アダプタの場合）」に示すビジネスプロセスで使用するデータ変換アクティビティの設定例を次に示します。

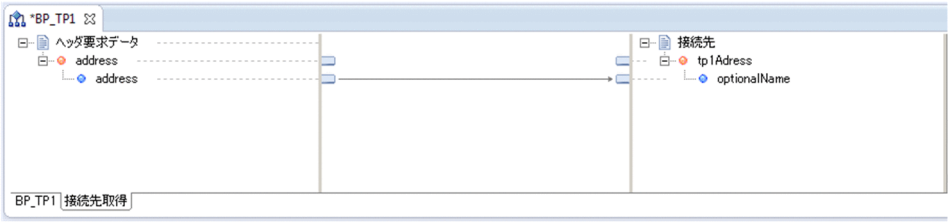
ここでは，変換元に「ヘッダ要求データ」を，変換先に「接続先」を指定しています。

図 2-75　データ変換アクティビティの設定例（TP1 アダプタの場合）



また，変換のマッピングは，次の図に示すとおりに定義します。

図 2-76　変換のマッピングの設定例（TP1 アダプタの場合）



(b)　接続先情報ファイルの定義

OpenTP1 に接続するためのホスト名とポート番号を TP1 アダプタに設定するための接続先情報ファイル（XML スキーマ）に定義します。

サービスプラットフォームでは，接続先情報ファイルを提供しています。

- 接続先情報ファイルの格納先

```
<サービスプラットフォームのインストールディレクトリ>/CSC/schema/connection/connection_tp1_adapter.xsd
```

注意事項

TP1 アダプタの接続先情報を動的に変更する場合，サービスプラットフォームで提供している接続先情報ファイルを使用してください。提供している接続先情報ファイルを使用しなかった場合，動作は保証されません。

- 接続先情報ファイルの形式

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
 DO NOT EDIT THIS FILE.
 -->
<xsd:schema
    elementFormDefault="qualified"
    targetNamespace="http://www.hitachi.co.jp/soft/xml/cosminexus/csc/connection/change/tp1"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="tp1Address">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="host" type="xsd:string"/>
        <xsd:element name="port" type="xsd:int"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

- 設定できる要素

host　～<文字列>((1－255 文字))

接続先のホスト名を指定します。この要素は，「通信先を指定した RPC」を利用する場合に有効になります。そのため，次の場合はこの要素は無効になります。

- その他の機能（ネームサービス，スケジューラダイレクト機能，またはリモート API 機能）を使用した RPC によって接続する場合
- MHP を接続先とした TCP/IP 接続をする場合

port　～<符号なし整数>((0－65535))

接続先のポート番号を指定します。この要素は，「通信先を指定した RPC」を利用する場合に有効になります。そのため，次の場合はこの要素は無効になります。

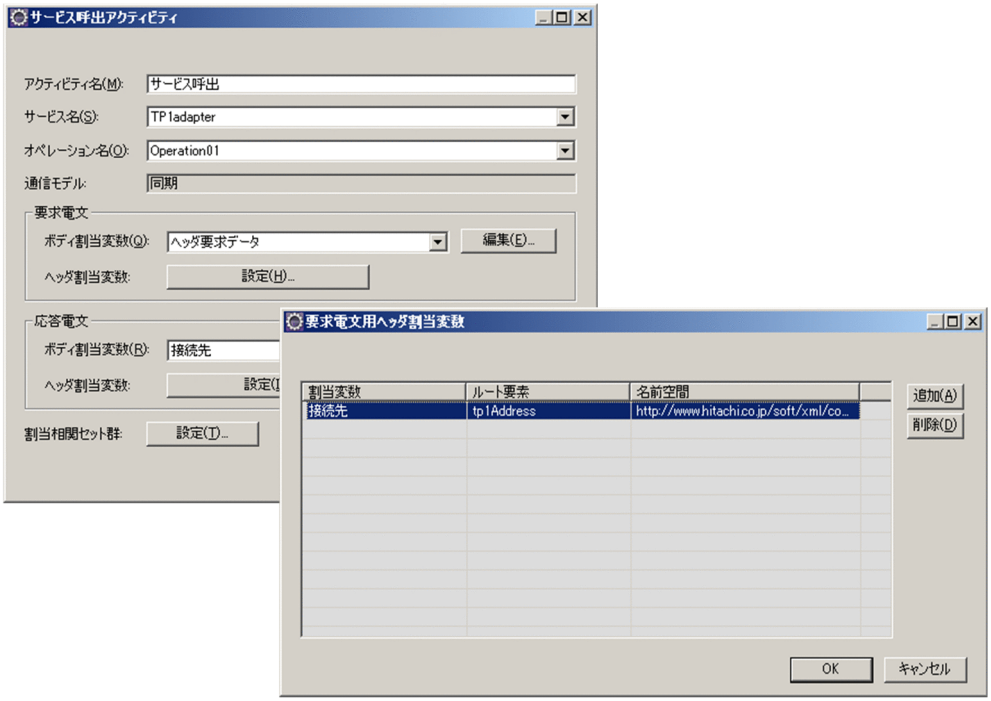
- その他の機能（ネームサービス，スケジューラダイレクト機能，またはリモート API 機能）を使用した RPC によって接続する場合
- MHP を接続先とした TCP/IP 接続をする場合

(c)　接続先情報の設定

「図 2-74　接続先情報ファイルを使用するビジネスプロセスの例（TP1 アダプタの場合）」に示すビジネスプロセスで使用する接続先情報の設定例を次に示します。なお，接続先情報は，サービス呼出アクティ

ビティのヘッダ割当変数に設定します。ここで設定した情報が，TP1 アダプタで参照できる情報になります。

図 2-77　接続先情報の設定例（サービス呼出アクティビティの設定）



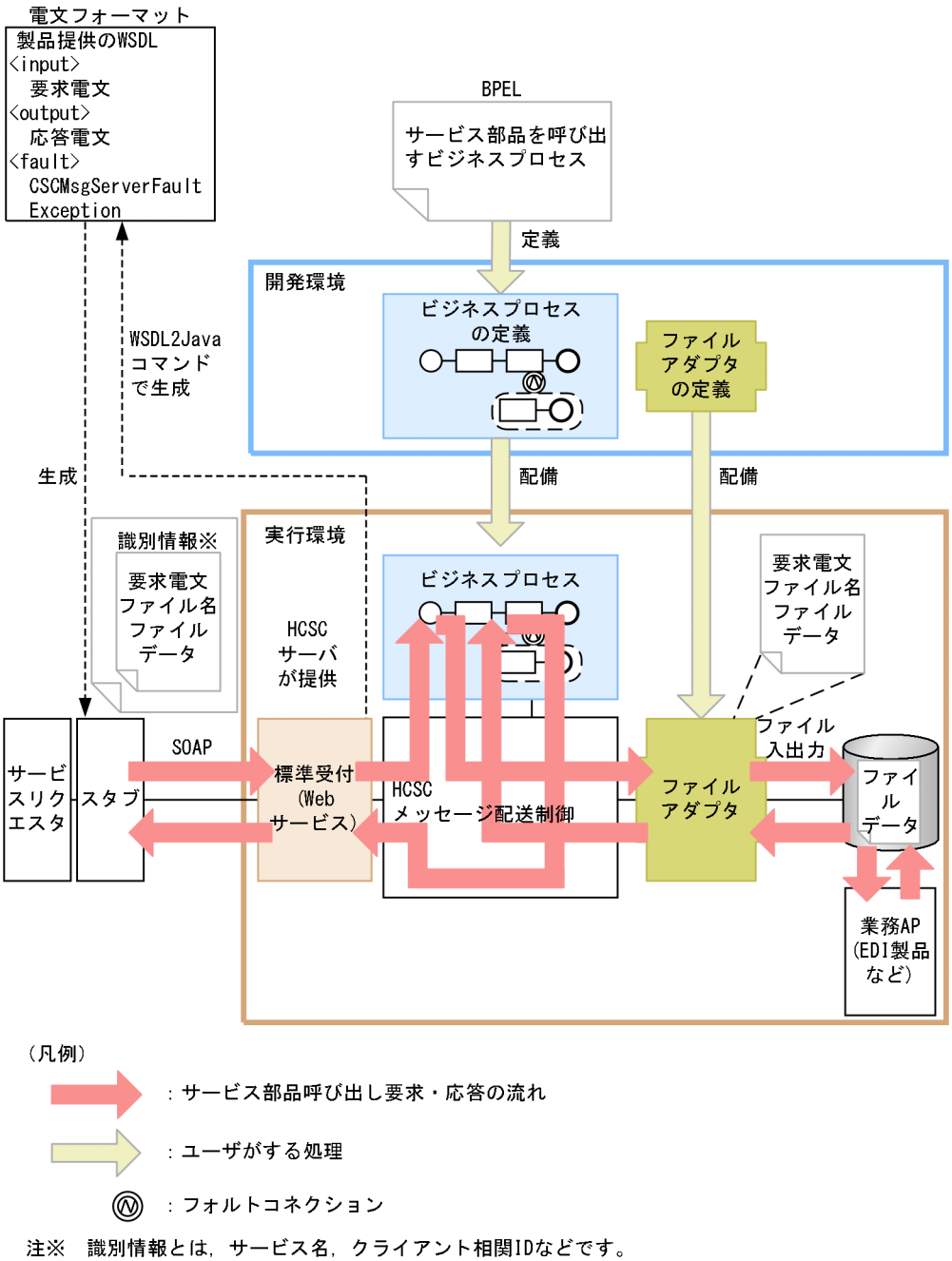
2.8　ファイルを扱うシステムとの接続

サービスリクエスタやビジネスプロセスからの要求を受けて，HCSC サーバ上のローカルディスクに対して直接ファイルの入出力をする場合は，ファイルアダプタを使用します。

2.8.1　ファイルアダプタを使ったファイルへのアクセス

ファイルアダプタによるファイルアクセスの実行について，次の図に示します。

図 2-78　ファイルアダプタによるファイルアクセスの実行



（凡例）

：サービス部品呼び出し要求・応答の流れ

：ユーザがする処理

Ⓝ：フォルトコネクション

注※　識別情報とは，サービス名，クライアント相関IDなどです。

ファイルアダプタは，開発環境で定義します。ファイルアダプタを定義する手順の詳細については，マニュアル「サービスプラットフォーム 開発ガイド 受付・アダプタ定義編」の「3. サービスアダプタの定義」を参照してください。

2.8.2 ファイルアダプタを使ったファイルの読み込みと書き込み

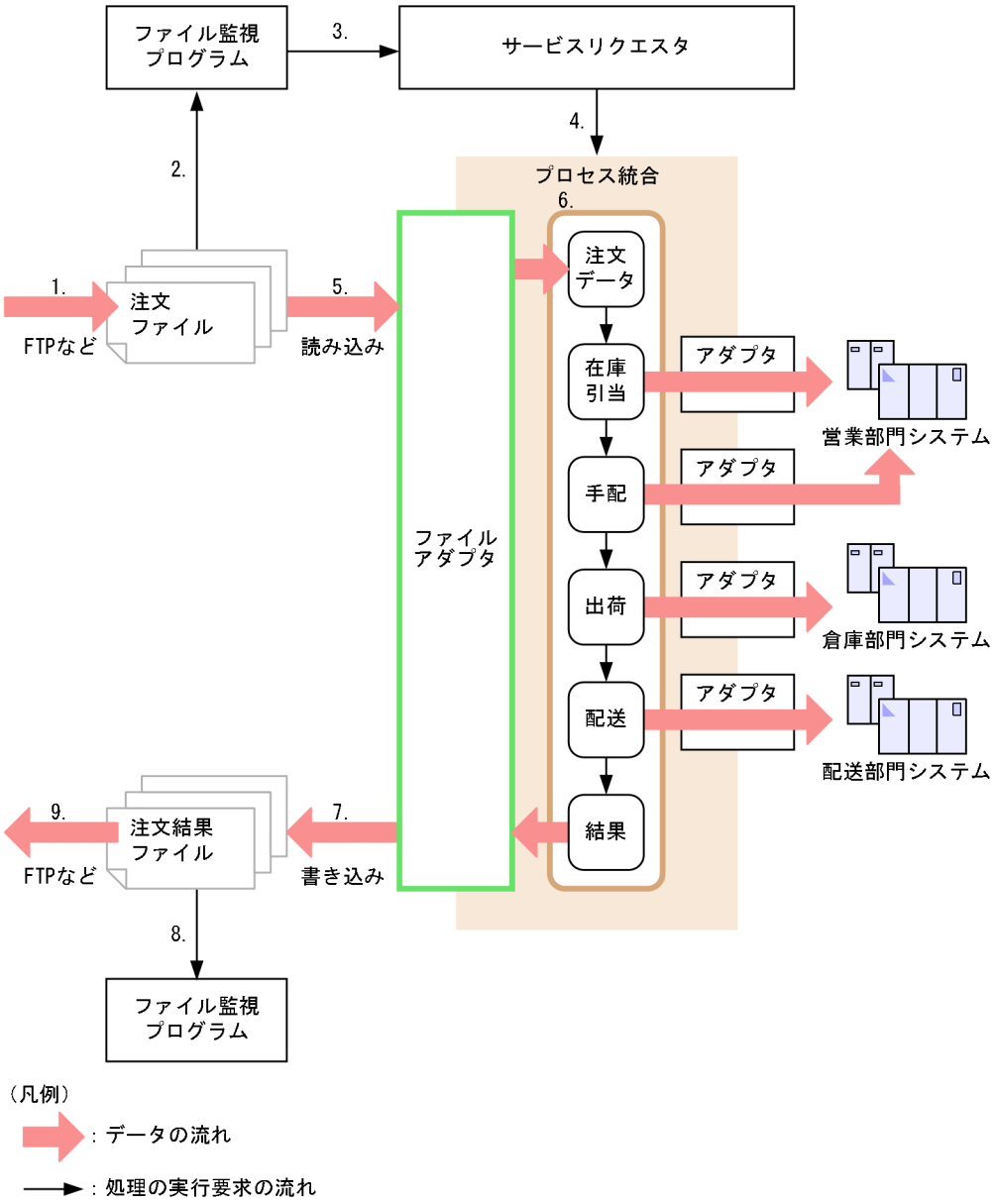
ファイルアダプタは，サービスプラットフォームで，ファイルからデータを読み込んだり，ファイルへデータを書き込んだりするサービスアダプタです。

ファイルアダプタの目的は，ファイルを介した業務処理システムのプログラム資産をシステムで活用することです。ファイルアダプタをビジネスプロセスのアクティビティとして用いることで，ファイルの入出力を伴う既存システムとの連携など，ファイル内容に基づいた業務処理を SOA で実現できます。

ファイルアダプタは，プロセス統合での使用を目的としており，サービスリクエスタからの直接呼び出しではなく，ビジネスプロセスからの呼び出しを前提としています。

ファイルアダプタを使用したシステムの運用例を次の図に示します。

図 2-79　ファイルアダプタの運用例



図内の各処理について説明します。

1. 他社からの注文ファイルを FTP などで受け取ります。
2. ファイルの受け取りは，ファイル監視プログラムによって検知されます。
3. ファイル監視プログラムからサービスリクエスタが起動されます。
4. サービスリクエスタから SOAP などを介して，ビジネスプロセスが起動されます。
5. ビジネスプロセスから呼び出されたファイルアダプタによって，注文ファイルのデータが読み込まれます。
6. 各業務のサービスが呼び出されます。
7. ファイルアダプタによって，サービスの呼び出し結果が注文結果ファイルに書き込まれます。
8. 注文結果ファイルへの書き込みが，ファイル監視プログラムによって検知されます。

9. 注文結果ファイルが，FTP などで転送されます。

このようなシステムによって，SOA に対応していない他社システムと連携できるようになります。

2.8.3　ファイルアダプタで使用できるファイル形式

ファイルアダプタで使用できるファイル形式を次に示します。

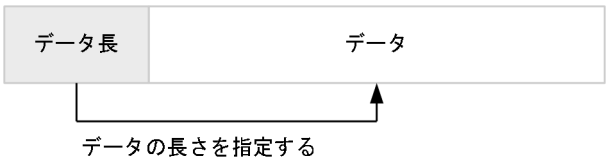
(1)　ファイルアダプタで XML データとして扱われるファイル形式

- XML 形式
  W3C が制定したマークアップ言語（Extensible Markup Language）で記述されたファイル形式です。HCSC サーバで使用できる XML スキーマの適用範囲で，XML 形式のファイルをサポートします。詳細は，マニュアル「サービスプラットフォーム 開発ガイド 基本開発編」の「2.6.5　XML スキーマの適用範囲」を参照してください。

(2)　ファイルアダプタでバイナリデータとして扱われるファイル形式

- 固定長形式
  レコードの各データ要素が決められた長さを持つファイル形式です。
- レングスタグ形式
  レコードの各データ要素に対して，その長さを表す数値情報を付けることで可変長データを扱うファイル形式です。

  図 2-80　レングスタグ形式



- セパレータ形式
  レコードの各データ要素の区切りに，セパレータ（デリミタ）を使用して可変長データを扱うファイル形式です。一般的な形式として CSV ファイルがあります。

2.8.4　ファイル操作が行われるときのモード

ファイルアダプタで，ファイル操作が行われるときのモードについて説明します。

(1)　読み込みモード

ファイルアダプタで，ファイルの読み込みが行われるときのモードを次に示します。読み込みができるファイルの最大サイズは 500MB です。ファイルアダプタ実行環境プロパティファイルで設定を変更できます。

また，0 バイトのファイルは読み込みできません。ファイルアダプタでのファイルの読み込みは，HCSC サーバの管理者の権限で行われます。

- 一括
  読み込み対象のファイル全体が読み込まれます。

(2) 書き込みモード

ファイルアダプタで，ファイルの書き込みが行われるときのモードを次に示します。ファイルアダプタでのファイルの書き込みは，HCSC サーバの管理者の権限で行われます。

- 新規
  書き込み対象のファイルが新規に作成され，データが書き込まれます。書き込み対象のファイルがすでに存在する場合は，書き込み対象のデータで上書きされます。書き込み処理中にエラーが発生した場合，エラーが発生する直前までのデータが書き込まれます。エラーが発生するタイミングによっては，空ファイルが作成されることがあります。
- 追加
  書き込み対象のデータが，既存のファイルに追加で書き込まれます。書き込み対象のファイルが存在しない場合は，ファイルが新規に作成され，書き込み対象のデータが書き込まれます。XML 形式のファイルでは，書き込みモードに追加を指定できません。書き込み処理中にエラーが発生した場合，エラーが発生する直前までのデータが書き込まれます。ファイルが新規に作成される場合，エラーが発生するタイミングによっては，空ファイルが作成されることがあります。

2.8.5 ファイルの読み込みおよび書き込み時の排他制御

ファイルアダプタでは，ファイル操作時に排他制御を行うことで，処理が競合してデータの不整合が起こることを防止します。

排他制御は，1 つのファイルアダプタ内で処理対象となるファイル単位に行われます。したがって，ファイルアダプタで処理中のファイルを外部アプリケーションから操作した場合，排他制御は行われません。また，単一の HCSC サーバで複数のファイルアダプタを使用した場合，ファイルアダプタ間の排他制御は行われません。同一形式のファイルに対しては 1 つのファイルアダプタを使用することを推奨します。

ファイルの読み込み時および書き込み時に行われる排他制御を次の表に示します。

表 2-15 ファイルの読み込みおよび書き込み時の排他制御

排他制御	読み込み	書き込み
参照排他（読み込み中）	○	×
更新排他（書き込み中）	×	×

（凡例）
○：処理できます。

×：処理できません（待機状態）。

(1) 参照排他

ファイルアダプタでファイルの読み込みを実行すると，参照排他が行われます。参照排他中のファイルに対しては，読み込みはできますが書き込みはできません。参照排他中のファイルに対して書き込みを行った場合，実行中の読み込み処理が終了するまで待機状態になります。

(2) 更新排他

ファイルアダプタでファイルの書き込みを実行すると，更新排他が行われます。更新排他中のファイルに対しては，読み込みおよび書き込みができません。更新排他中のファイルに対して読み込みまたは書き込みを行った場合，実行中の書き込み処理が終了するまで待機状態になります。

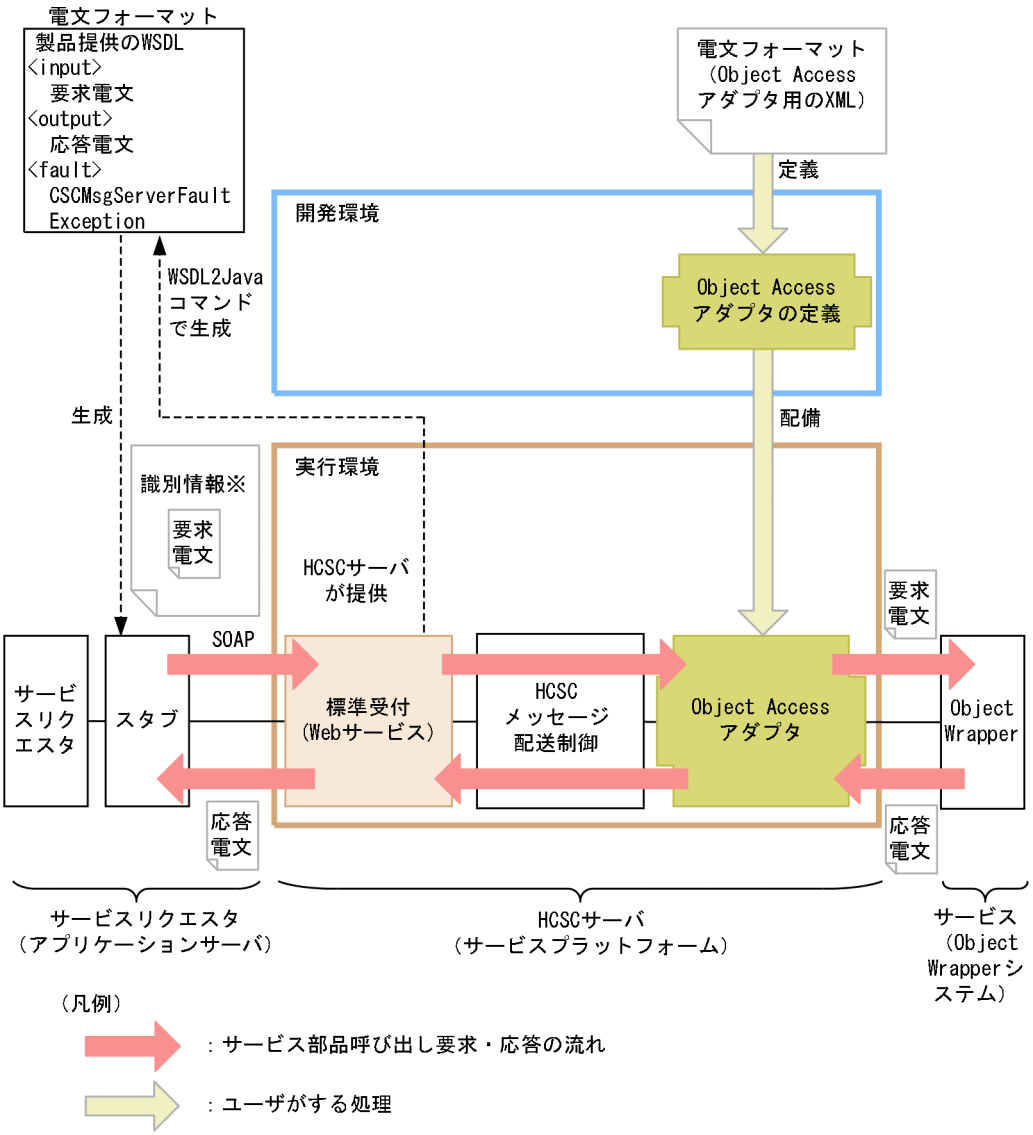
2.9　Object Wrapper システムとの接続

既存の TPBroker システム（Object Wrapper システム）のサービス部品を呼び出す場合は，Object Access アダプタを使用します。

2.9.1　Object Access アダプタを使った Object Wrapper システムへのアクセス

HCSC サーバからの Object Wrapper システムのサービス部品の呼び出しを次の図に示します。

図 2-81　HCSC サーバからの Object Wrapper システムのサービス部品呼び出し



注※　識別情報とは，サービス名，クライアント相関IDなどです。

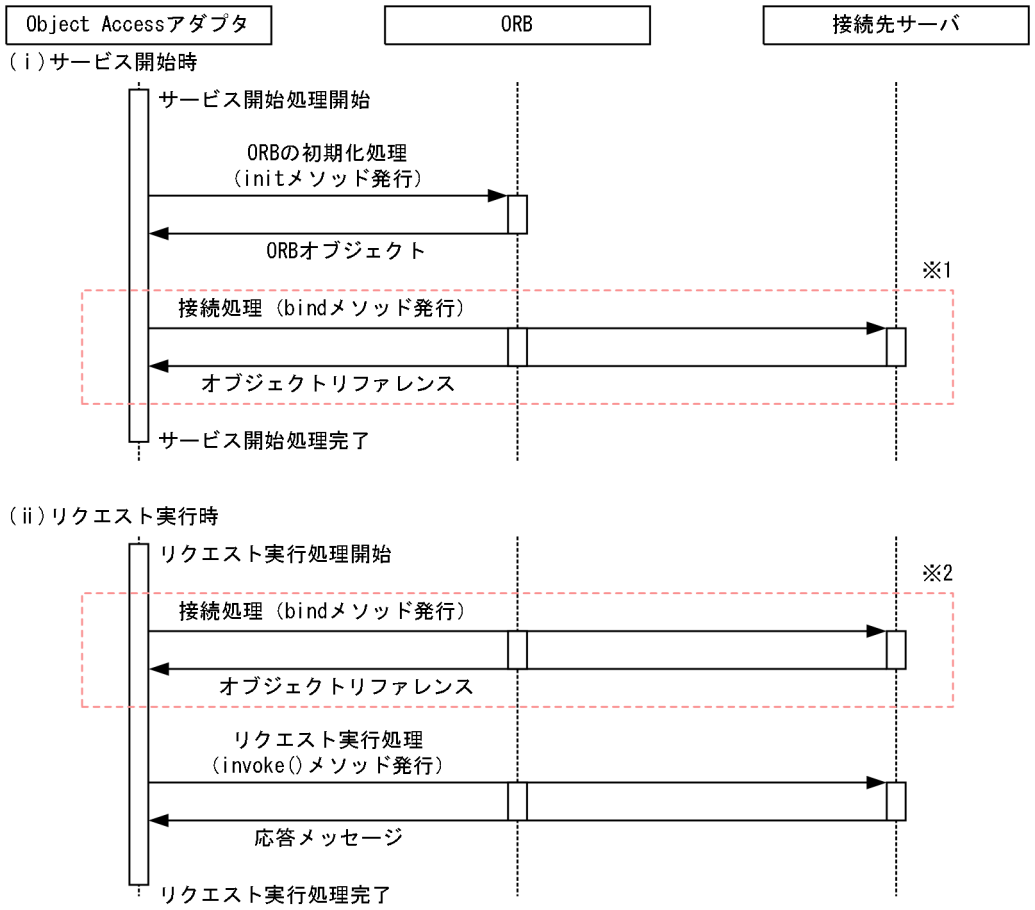
(1) Object Wrapper システムとの接続のタイミング

Object Access アダプタでは，次に示すタイミングで TPBroker と接続します。

- サービス開始時
- リクエスト実行時

接続方式は，Object Access アダプタ通信構成定義ファイルの bind_type に指定します。

図 2-82 IIOP 通信接続が発行するメソッドの処理タイミング



注※1

Object Accessアダプタ通信構成定義ファイルの＜bind_type＞要素にType2を指定した場合となります。これによって，サーバが単体の場合，ORBの初期化時に接続することで，リクエスト実行の際に処理時間を短縮できます。また，Type2では，スタート時にサーバが起動していないと接続処理を中断し，アダプタの開始処理を続行します。

注※2

Object Accessアダプタ通信構成定義ファイルの＜bind_type＞要素にType1を指定した場合となります。これによって，サーバが複数ある場合，リクエスト実行の際に負荷を分散できます。

(2) 時間監視

Object Access アダプタは通信先システムとの次の通信に対して時間を監視します。

- コネクションの確立

- リクエスト要求に対する応答

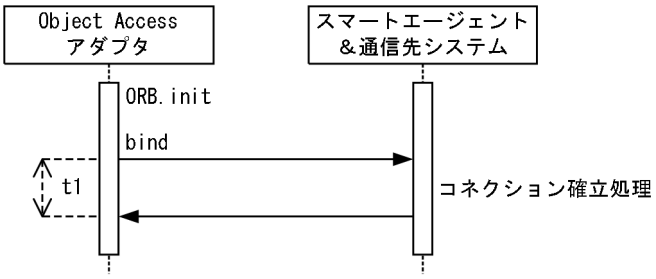
(a) コネクションの確立

時間監視は，通信システムとのコネクション確立が完了するまでの時間を監視する場合に，設定します。指定値は，サービス単位に設定します。指定した時間を超えるとタイムアウトとなり，例外を通知します。

コネクションの確立時間監視は，Object Access アダプタ通信構成定義ファイルの connect_watch_time に指定します。

時間監視の範囲を次に示します。図中の t1 が時間監視の範囲です。

図 2-83 コネクション確立までの時間監視



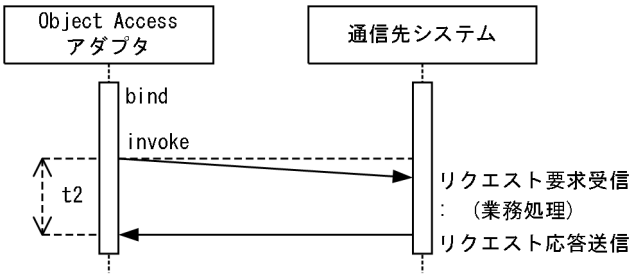
(b) リクエスト要求に対する応答

時間監視は，リクエスト要求を送信してから応答が返ってくるまでの時間を監視する場合に，設定します。指定値は，サービス要求のオペレーション単位に設定します。指定した時間を超えるとタイムアウトとなり，例外を通知します。

コネクションの確立時間監視は，Object Access アダプタ通信構成定義ファイルの return_watch_time に指定します。

時間監視の範囲を次に示します。図中の t2 が時間監視の範囲です。

図 2-84 リクエスト送信から応答メッセージが到達するまでの時間監視

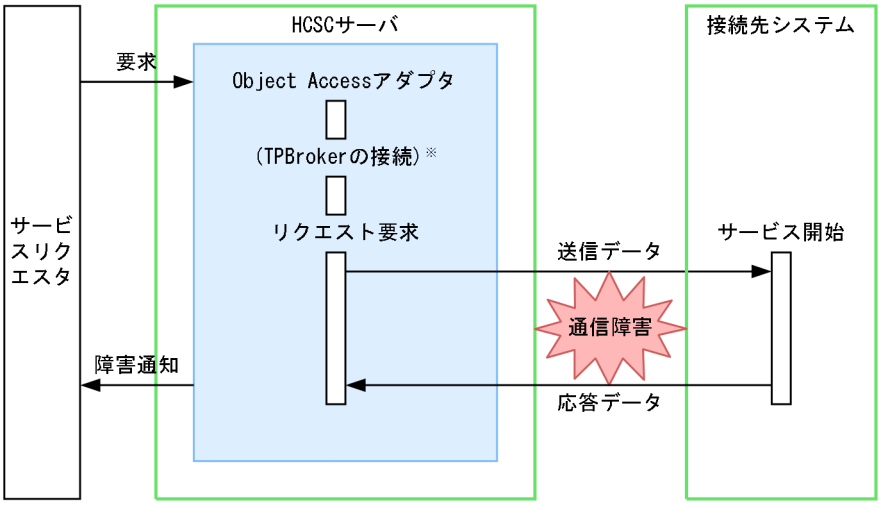


(3) 障害検知

Object Access アダプタでは，TPBroker の通信時に接続が切断された場合，再接続を実行しません。接続が切断された場合は，処理を中断して例外を通知します。

TPBroker との接続をサービス開始時に実行している場合は，再接続を障害発生以降のリクエスト要求時に実行します。

図 2-85　接続の切断時の処理概略



注※　TPBrokerとの接続をサービス開始時に実行している場合，リクエスト要求時に接続が切断されていると再接続処理を実行します。また，TPBrokerとの接続をリクエスト実行時に実行する場合は，リクエスト要求時に接続処理を実行します。

Object Access アダプタで障害を検知した場合，サービスを実行したサービスリクエスタ，またはビジネスプロセスに，例外情報および Fault 情報を通知します。

Object Access アダプタでは例外情報と Fault 情報の障害通知を次の条件によって決定します。

例外を通知する場合

- 開始処理で発生した障害
- IIOP 通信処理で発生した IDL で定義されたユーザ例外以外

Fault 情報を通知する場合

IIOP 通信処理で発生した IDL で定義されたユーザ例外

2.9.2　接続時の電文の形式変換

Object Access アダプタのサービスリクエスタとのインターフェースで使用する電文には，要求電文と応答電文（Fault 情報を含む）があります。Object Wrapper で使用する IDL および Object Wrapper の IDL と電文の形式変換について次に示します。

(1)　IDL のサポート範囲

Object Access アダプタがサポートしている IDL を次に示します。

表 2-16　IDL 定義のサポート範囲

IDL	備考
module 定義	−
interface 定義	継承定義は使用不可
operation 定義	−
typedef 定義	sequence 型だけ
struct 定義	−
exception 定義	−

表 2-17　operation 定義のサポート範囲

IDL	備考
in 引数	−
out 引数	−
inout 引数	−
リターン値	−
raises 定義	ユーザ例外は 1 つだけ

表 2-18　IDL データ型のサポート範囲

IDL	備考
long	−
float	−
string	文字列長は指定不可
any	「(4) any 型の扱い」を参照してください。
struct	−
sequence	配列サイズは指定不可

(2)　型変換

Object Wrapper の IDL の定義型と電文で扱う型情報で変換する内容を次に示します。

表 2-19　型変換が行われる IDL 型

Object Wrapper の IDL	電文
typedef sequence<octet> OctSeq	xsd:string
struct Binary	xsd:base64Binary

(3) 文字コード変換

Object Wrapper の IDL に定義した string 型は，SJIS コードとして扱います。文字コード変換の対象には，型変換で string 型に変換される「typedef sequence<octet> OctSeq」を含みます。文字コード変換対象の，Object Wrapper の IDL を次に示します。

表 2-20 文字コード変換が行われる IDL 型

Object Wrapper の IDL	電文
method_name	xsd:string
typedef sequence<octet> OctSeq	xsd:string

(4) any 型の扱い

Object Access アダプタで IDL の CORBA.Any は，単純型（primitive 型）と複合型（complex 型）の sequence<octet>の値が扱えます。

CORBA.Any の電文での表現，および電文の各要素に指定できる値を次に示します。

- CORBA.Any の電文フォーマット

```
<xsd:complexType name="CORBA.Any">
<xsd:sequence>
<xsd:elementname="type" type="xsd:string" max0ccurs="1" min0ccurs="1"/>
<xsd:elementname="value" type="xsd:string" max0ccurs="1" min0ccurs="1" />
</xsd:sequence>
</xsd:complexType>
```

表 2-21 type 要素と value 要素の指定値

Object Wrapper の IDL	type 要素の値	value 要素の値
long	long	-2147483648 から 2147483647 の数字
float	float	単精度の浮動小数点文字列
typedef sequence<octet>OctSeq	EComm2.OctSeq	単数・複数の文字
struct Binary	EComm2.Binary	base64Binary 表記文字列

2.10　メッセージキューとの接続

JMS プロバイダのメッセージキュー（IBM MQ システム）から MQ リソースアダプタ経由でメッセージを送受信する場合は，Message Queue 受付を使用します。

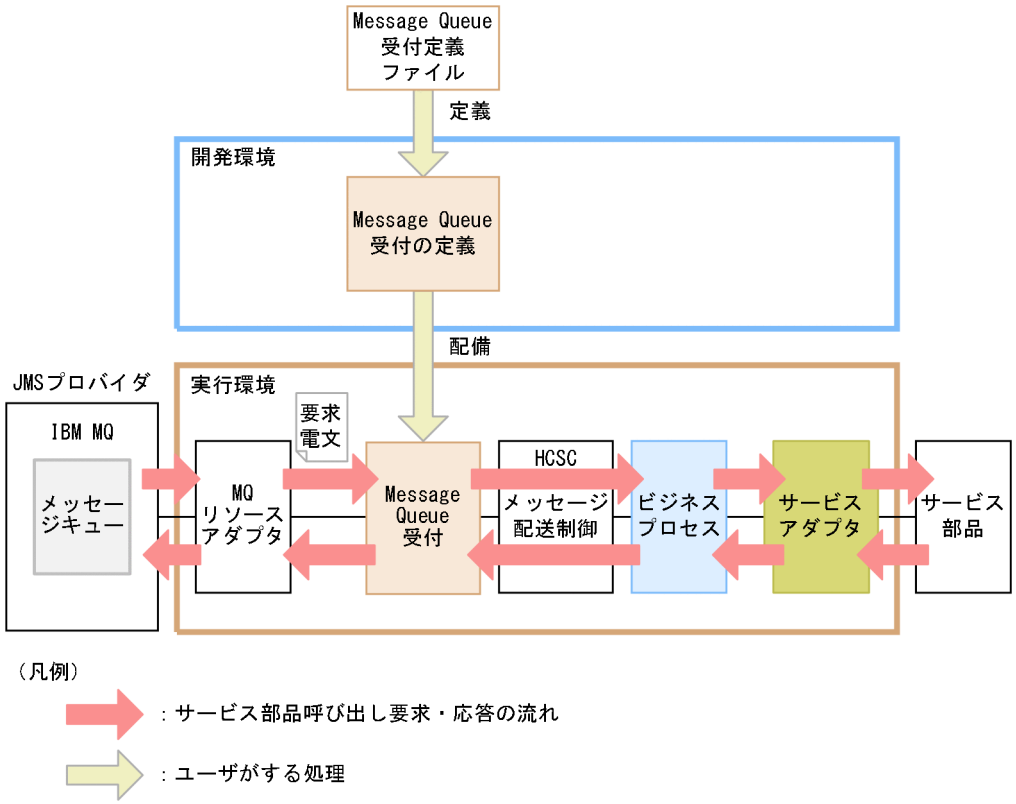
また，既存のメッセージキュー（IBM MQ システム）に対してメッセージを送受信する場合は，Message Queue アダプタを使用します。

2.10.1　Message Queue 受付を使用したサービス部品の呼び出し

Message Queue 受付を利用することで，JMS プロバイダのメッセージキュー（IBM MQ）から MQ リソースアダプタを介してビジネスプロセスを呼び出すことができます。

Message Queue 受付を使用したサービス部品呼び出しの流れを次の図に示します。

図 2-86　Message Queue 受付を使用した JMS プロバイダ（メッセージキュー）からのサービス部品の呼び出し



Message Queue 受付は，MQ リソースアダプタを経由して要求電文を受け付けます。サービス部品の実行要求を受け付けた Message Queue 受付は，ビジネスプロセスを呼び出し，サービスアダプタを介して各種サービス部品を実行します。

要求電文の受け取りは，MDB（Message-Driven Bean）を使用します。

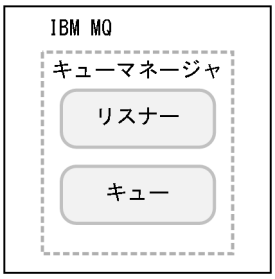
Message Queue 受付からは，応答電文を返しません。

(1) IBM MQ の概要

Message Queue 受付では，JMS プロバイダとして IBM MQ を利用します。IBM MQ は，異機種システムのプログラム間でデータ通信をするためのメッセージキューイング型の非同期通信基盤です。

IBM MQ の構成を次に示します。

図 2-87 IBM MQ の構成



（凡例）

キューマネージャ：キューやリスナーをまとめる管理プログラム

キュー：JMS オブジェクトをまとめるキューオブジェクト

リスナー：キューマネージャがメッセージを受信または送信するためのリスナー

注意事項

IBM MQ のキューマネージャは，初期状態ではチャネル認証によるサーバへのアクセスができません。そのため，チャネル認証を使用しないように設定を変更するか，デフォルトで設定されているチャネル認証ルールを変更する必要があります。IBM MQ の設定方法については，IBM 社のマニュアルを参照してください。

(2) Message Queue 受付の送信オプション

メッセージを送信する場合，送信オプション（JMS ヘッダおよびプロパティ）を指定できます。Message Queue 受付の送信オプションの詳細を次に示します。

(a) JMS ヘッダの詳細

送信オプション（JMS ヘッダ）の設定値は，すべて String 型に変換され，ヘッダ電文の中に設定されます。このとき，値が null，または空の場合は，空タグになります。

JMS ヘッダの詳細を次に示します。

表 2-22　JMS ヘッダの詳細

ヘッダ名	設定内容	設定タイミング	クライアントでの設定可否
JMSDestination	メッセージが登録されるキュー名	QueueSession.createSender(Queue queue)メソッド発行時に設定されます。	×
JMSDeliveryMode	メッセージの永続性を示す値	メッセージ送信時に設定されます。	×
JMSMessageID	プロバイダによって送信される各メッセージを一意に識別するメッセージ識別子	メッセージ送信時に設定されます。	×
JMSTimestamp	メッセージの送信時刻を表す値（単位：ミリ秒）	メッセージ送信時に設定されます。	×
JMSExpiration	メッセージの有効期間となる時刻を表す値（単位：ミリ秒）	メッセージ送信時に設定されます。	×
JMSRedelivered	メッセージが再配送中であるかどうかを示す値	アプリケーションに配送済みのメッセージがリカバリされたなどで再度配送する必要があるときに設定されます。	×
JMSPriority	メッセージのプライオリティ	QueueSender.setPriority()メソッド，またはQueueSender.send()メソッドで指定できます。	○
JMSReplyTo	応答先のあて先	Message.setJMSReplyTo()メソッド発行時に設定されます。	○
JMSCorrelationID	あるメッセージをほかのメッセージと関連づけるための相関識別子	Message.setJMSCorrelationID()メソッド発行時に設定されます。	○
JMSType	メッセージが送信されるときに，アプリケーションによって提供されるメッセージタイプ	Message.setJMSType()メソッド発行時に設定されます。	○

（凡例）

○：指定できます。

×：指定できません。

(b)　プロパティの詳細

メッセージのプロパティは，JMS プロバイダによって設定されるプロパティと，クライアントでユーザが設定できるプロパティで構成されます。

●JMS プロバイダによって設定されるプロパティ

JMS プロバイダによって設定されるプロパティには，JMS 標準プロパティと IBM MQ で設定されるプロパティがあります。どちらのプロパティも設定されている項目については，すべてヘッダ電文に設定されます。

- JMS 標準プロパティ

  JMS 標準のプロパティは，「JMSX」が項目名のプレフィックスに付いているプロパティです。

  JMS 標準プロパティの一覧を次の表に示します。

表 2-23　JMS 標準プロパティの一覧

項目名	説明
JMSXUserID	メッセージの送信者のユーザ ID
JMSXAppID	メッセージの送信アプリケーションの ID
JMSXDeliveryCount	メッセージの再受信回数
JMSXGroupID	メッセージが属するグループの ID
JMSXGroupSeq	グループ内でのメッセージ順序番号
JMSXProducerTXID	トランザクション識別子
JMSXConsumerTXID	トランザクション識別子
JMSXRcvTimestamp	メッセージ受信時間
JMSXState	プロバイダ内部で使用する項目

- IBM MQ で設定されるプロパティ

  IBM MQ で設定されるプロパティは，プロバイダ用プロパティの中に含まれます。プロバイダ用プロパティには，「JMS_」が項目名のプレフィックスに付いています。その中で，「JMS_IBM_」が項目名のプレフィックスに付いているものが IBM MQ のプロパティです。

●ユーザが設定するプロパティ

ユーザが任意に設定できるプロパティは，「set<型名>Property(String name,<型名> value)」メソッドを使用して設定します。

プロパティに設定された値はすべて String 型に変換し，ヘッダ電文の中に設定されます。

プロパティに設定できるデータ型の一覧を次の表に示します。

表 2-24　プロパティに設定できるデータ型の一覧

データ型	設定するメソッド
Boolean	setBooleanProperty(String name, boolean value)
Byte※1	setByteProperty(String name, byte value)
Short	setShortProperty(String name, short value)
Integer	setIntProperty(String name, int value)
Long	setLongProperty(String name, long value)
Float	setFloatProperty(String name, float value)
Double	setDoubleProperty(String name, double value)
String	setStringProperty(String name, String value)

データ型	設定するメソッド
Object※2	setObjectProperty(String name, Object value)

注※1

Byte 型の値を設定すると，Integer 型の数値に変換されたものを文字列としてヘッダ電文に設定します。設定される数値の範囲は-127～127 です。

注※2

データ型に Object を設定すると，すべてのデータ型（Boolean，Byte，Short，Integer，Long，Float，Double および String）が使用できます。

(3) Message Queue 受付と MQ リソースアダプタとの関係

MQ リソースアダプタは，JMS プロバイダ（IBM MQ システム）からのメッセージを Message Queue 受付に渡すためのリソースアダプタです。

MQ リソースアダプタと Message Queue 受付の接続関係は，1 つのリソースアダプタに対して複数の受付を関連づけることができますが，1 つの受付に対して複数のリソースアダプタを関連づけることができません。

MQ リソースアダプタと Message Queue 受付の関連づけは，Message Queue 受付定義ファイルで指定します。指定した内容は，Message Queue 受付を起動したときに有効になります。このとき，Message Queue 受付と関連づけをする MQ リソースアダプタを事前に起動しておく必要があります。関連づけをする MQ リソースアダプタが起動されていないと，Message Queue 受付は起動できないため，Message Queue 受付定義ファイルの設定内容も有効になりません。

Message Queue 受付と MQ リソースアダプタの接続関係を次に示します。

(a) 構築できる接続関係（MQ リソースアダプタの場合）

構築できる接続関係は次のどちらかです。

- リソースアダプタと受付が 1：1 の接続関係の場合
- リソースアダプタと受付が 1：n の接続関係の場合

例を次に示します。

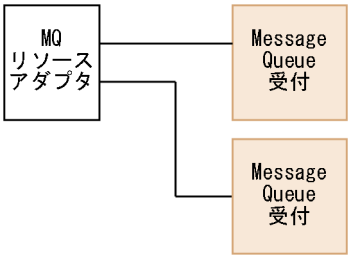
（例 1） リソースアダプタと受付が 1：1 の接続関係の場合



説明

- 1 つの MQ リソースアダプタに関連づいている Message Queue 受付は 1 つ
- 1 つの Message Queue 受付に関連づいている MQ リソースアダプタは 1 つ

(例 2) リソースアダプタと受付が 1：n の接続関係の場合



説明

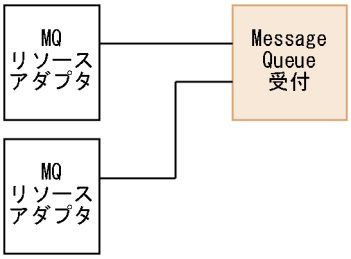
- 1 つの MQ リソースアダプタに関連づいている Message Queue 受付は複数
- 1 つの Message Queue 受付に関連づいている MQ リソースアダプタは 1 つ

(b) 構築できない接続関係（MQ リソースアダプタの場合）

1 つの受付に対して複数のリソースアダプタを指定できないため，リソースアダプタと受付の接続関係が n：1 の構成は構築できません。

例を次に示します。

(例)



説明

- 1 つの MQ リソースアダプタに関連づいている Message Queue 受付は 1 つ
- 1 つの Message Queue 受付に関連づいている MQ リソースアダプタは複数

(4) Message Queue 受付とキューとの関係

キューは，JMS オブジェクトをまとめるキューオブジェクトで，キューマネージャによって管理されます。

キューと Message Queue 受付の接続関係は，1 つのキューに対して複数の受付を関連づけることができますが，1 つの受付に対して複数のキューを関連づけることができません。

Message Queue 受付とキューの接続関係を次に示します。

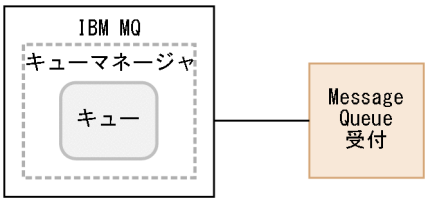
(a) 構築できる接続関係（キューの場合）

構築できる接続関係は次のどちらかです。

- キューと受付の接続関係が 1：1 の場合
- キューと受付の接続関係が 1：n の場合

例を次に示します。

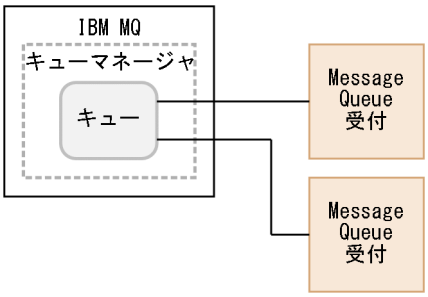
（例 1）キューと受付の接続関係が 1：1 の場合



説明

- 1 つのキューマネージャの中に存在するキューは 1 つ
- 1 つのキューに対して関連づけられた Message Queue 受付は 1 つ

（例 2）キューと受付の接続関係が 1：n の場合



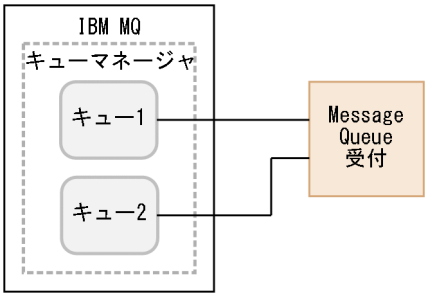
説明

- 1 つのキューマネージャの中に存在するキューは 1 つ
- 1 つのキューに対して関連づけられた Message Queue 受付は 2 つ

(b)　構築できない接続関係（キューの場合）

1 つの受付に対して複数のキューを指定できないため，キューと受付の接続関係が n：1 の構成は構築できません。

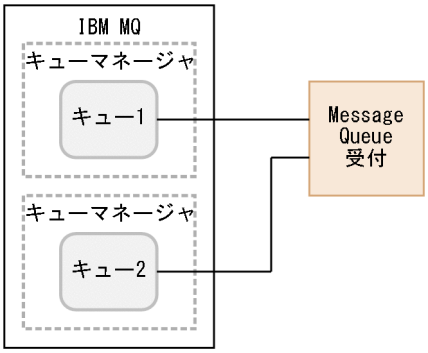
（例 1）キューマネージャが 1 つの場合



説明

- 1 つのキューマネージャの中に存在するキューは 2 つ
- キュー 1 およびキュー 2 に対して関連づけられた Message Queue 受付は 1 つ

(例 2) キューマネージャが複数の場合



説明

- 2 つのキューマネージャが存在
- それぞれのキューマネージャの中に存在するキューは 1 つずつ
- キュー 1 およびキュー 2 に対して関連づけられた Message Queue 受付は 1 つ

(5) Message Queue 受付とビジネスプロセスとの関係

ビジネスプロセスを作成するときは，受付アクティビティの通信モデルを Message Queue 受付の通信モデルと同一にする必要があります。

ビジネスプロセスの永続化については，永続化でも非永続化でも設定できます。

ただし，非永続化を設定すると，障害発生時にメッセージの二重配信が発生するおそれがあるので注意が必要です。

ビジネスプロセスと Message Queue 受付の接続関係は，1 つのビジネスプロセスに対して複数の受付を関連づけることができますが，1 つの受付に対して複数のビジネスプロセスを関連づけることができません。

Message Queue 受付とビジネスプロセスの接続関係を次に示します。

(a)　構築できる接続関係（ビジネスプロセスの場合）

構築できる接続関係は次のどちらかです。

- ビジネスプロセスと受付の接続関係が 1：1 の場合
- ビジネスプロセスと受付の接続関係が 1：n の場合

例を次に示します。

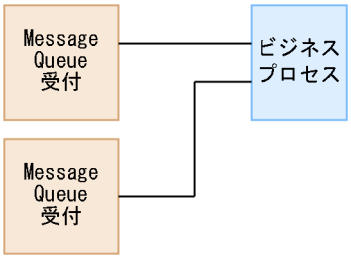
（例 1）ビジネスプロセスと受付の接続関係が 1：1 の場合



説明

- 1 つの Message Queue 受付に関連づけるビジネスプロセスは 1 つ
- 1 つのビジネスプロセスに関連づける Message Queue 受付は 1 つ

（例 2）ビジネスプロセスと受付の接続関係が 1：n の場合



説明

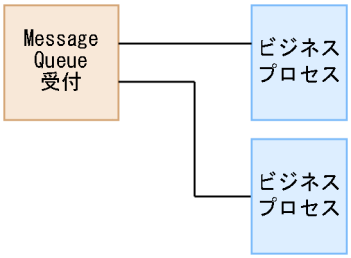
- 1 つの Message Queue 受付に関連づけるビジネスプロセスは 1 つ
- 1 つのビジネスプロセスに関連づける Message Queue 受付は複数

(b)　構築できない接続関係（ビジネスプロセスの場合）

1 つの受付に対して複数のビジネスプロセスを定義できないため，ビジネスプロセスと受付の接続関係が n：1 の構成は構築できません。

例を次に示します。

（例）



説明

- 1 つの Message Queue 受付に関連づけるビジネスプロセスは複数
- 1 つのビジネスプロセスに関連づける Message Queue 受付は 1 つ

(6) Message Queue 受付の構成例

Message Queue 受付では，使用する通信モデルに応じてさまざまなシステム構成を実現できます。

ここでは，次の場合の構成例を次に示します。

- Message Queue 受付を非同期の通信モデルで使用する場合
- Message Queue 受付を同期の通信モデルで使用する場合

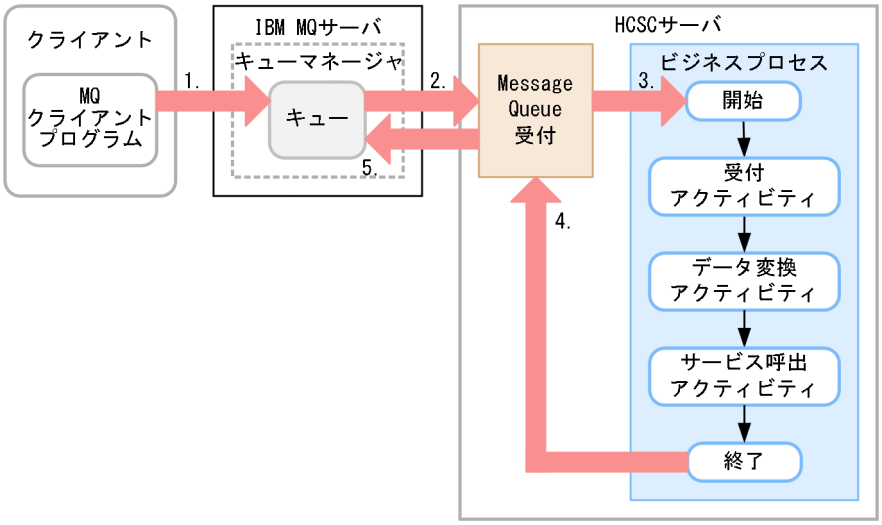
(a) Message Queue 受付を非同期の通信モデルで使用する場合

前提条件

1. Message Queue 受付を定義したビジネスプロセスを作成します。
2. Message Queue 受付の通信モデルを「非同期」で定義します。

構成例

Message Queue 受付とビジネスプロセスの構成の例を次の図に示します。



説明

1. MQ クライアントのプログラムから，IBM MQ サーバのキューにメッセージを登録します。
2. IBM MQ サーバから Message Queue 受付（HCSC サーバ）に対し，メッセージが送信されます。
3. Message Queue 受付は，ビジネスプロセスを呼び出します。
4. すべてのアクティビティの処理が終了したあと，Message Queue 受付も処理を終了します。
5. IBM MQ サーバがトランザクションをコミットし，キューからメッセージを削除します。

> **参考**
>
> この構成例の場合，リクエスト処理が完了するまで，メッセージが保持されます。このため，ビジネスプロセスが処理している間は，キューからメッセージは削除されません。ただし，二重配信防止の処理が実行された場合，またはロールバック制限でメッセージが移動した場合は，キューからメッセージが削除されます。メッセージの二重配信防止については，「(8) Message Queue 受付使用時のメッセージの二重配信防止」を参照してください。

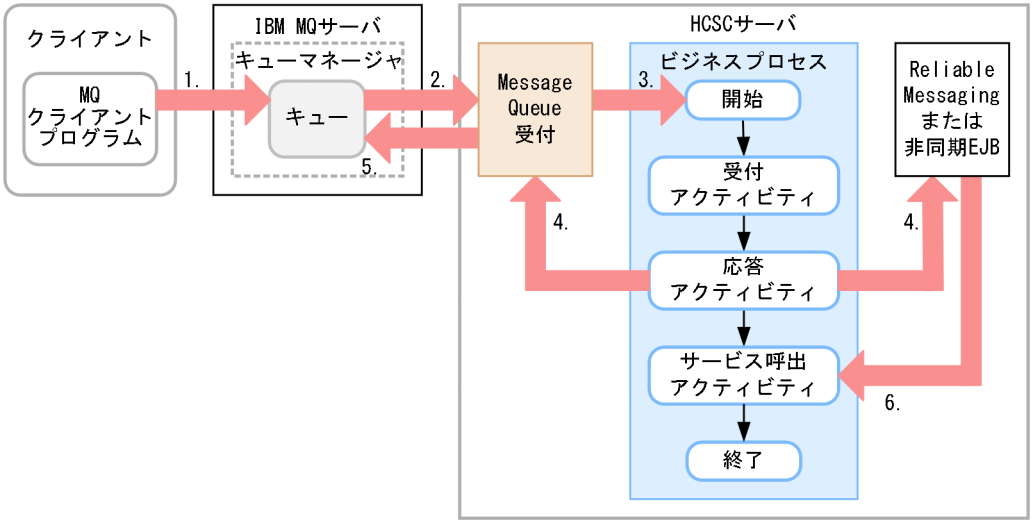
(b)　Message Queue 受付を同期の通信モデルで使用する場合

前提条件

1. Message Queue 受付を定義したビジネスプロセスを作成します。
2. Message Queue 受付の通信モデルを「同期」で定義します。

構成例

Message Queue 受付とビジネスプロセスの構成の例を次の図に示します。



説明

1. MQ クライアントのプログラムから，IBM MQ サーバのキューにメッセージを登録します。
2. IBM MQ サーバから Message Queue 受付（HCSC サーバ）に対し，メッセージが送信されます。
3. Message Queue 受付は，ビジネスプロセスを呼び出します。

4. 応答アクティビティの処理が終了したあと，応答アクティビティのあとの実行に必要な情報を Reliable Messaging または非同期 EJB へ格納し，Message Queue 受付も処理を終了します。
5. IBM MQ サーバがトランザクションをコミットし，キューからメッセージを削除します。
6. Reliable Messaging または非同期 EJB に格納した情報に従って，応答アクティビティのあとに設定されたアクティビティが実行されます。

参考

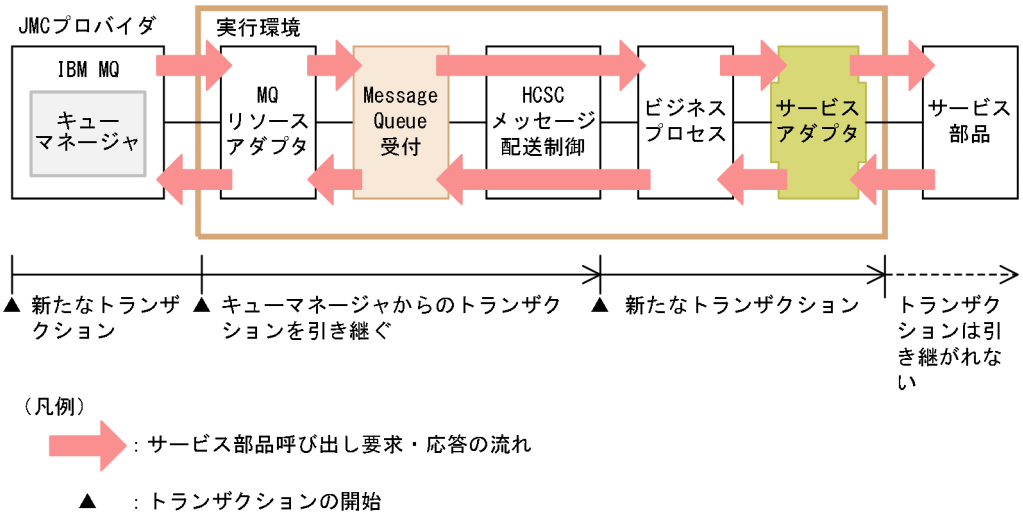
この構成例の場合，ビジネスプロセスの処理が終了する前に，キューからメッセージを削除できます。ただし，応答アクティビティのあとのビジネスプロセスの処理で例外が発生した場合に，キューからメッセージが削除されているので，リクエストを再送信する必要があります。

(7) Message Queue 受付使用時のトランザクション

Message Queue 受付では，例外発生時はキューをロールバックするため，IBM MQ のキューマネージャと同一のトランザクションを使用する必要があります。そのため，Message Queue 受付のトランザクションはコンテナ管理のトランザクションを使用します。それによって，トランザクションのコミットは IBM MQ のキューがコミットされるタイミングと同時となります。

Message Queue 受付を使用した場合のトランザクションの範囲を次の図に示します。

図 2-88 Message Queue 受付を使用した場合のトランザクションの範囲



Message Queue 受付は，IBM MQ のキューマネージャと同一のトランザクションを使用して処理しますが，ビジネスプロセスは，IBM MQ のキューマネージャとは異なるトランザクションで処理します。そのため，Message Queue 受付で処理が異常終了し，ロールバックを実行するためにメッセージがキューの先頭に戻った場合でも，ビジネスプロセスのトランザクションが正常終了していることがあります。このような状態のときにロールバックによって再びメッセージが送信されると，一度処理を実行したビジネスプロセスで再び同じ処理が実行されてしまうことになります。このような処理を防ぐには，メッセージの

二重配信防止の処理をしてください。メッセージの二重配信防止については，「(8) Message Queue 受付使用時のメッセージの二重配信防止」を参照してください。

(8) Message Queue 受付使用時のメッセージの二重配信防止

リクエスト処理が異常終了した場合，ロールバックを実行するためにメッセージはキューの先頭に戻り，再びメッセージが送信されることで，一度処理を実行したビジネスプロセスが再び処理されてしまうことがあります。

Message Queue 受付では，メッセージの二重配信防止の設定をすることで，リクエスト処理が異常終了した場合に，メッセージの再送信による実行済みビジネスプロセスの再実行を防止します。

(a) 二重配信防止の前提条件

次に示す条件をすべて満たす場合に，二重配信防止のチェックをします。

- データベースを使用している。
- 実行履歴が有効になっている（ビジネスプロセスのプロパティで，ステータスの永続化が「yes」になっている）。

(b) 二重配信防止使用時の注意点

二重配信防止を設定している場合の注意点を次に示します。

- 実行履歴の管理が必要なため，データベース容量の見積もりが必要です。
- システム例外が発生したビジネスプロセスは，再実行する必要があります。

(c) ビジネスプロセスで障害が発生したときの対策

ビジネスプロセスで発生する主な障害の例を挙げ，二重配信防止を設定している場合の動作および対処方法を説明します。

（例 1）メッセージ関連づけ情報テーブルへ登録前に障害が発生した場合

この場合の実行履歴の表示，リクエストメッセージの状態および対処方法を次に示します。

- **実行履歴の表示**

  未実行

- **リクエストメッセージの状態**

  ロールバックされ，再度リクエストメッセージが送信されます。

  電文不正（null など）が原因だった場合は，指定したロールバックの実行回数分だけロールバックを続けます。

- **対処方法**

  ロールバックの制限回数を設定してください。

  障害の発生するメッセージを除去してください。

(例 2) ビジネスプロセスの処理実行中に障害が発生した場合

この場合の実行履歴の表示，リクエストメッセージの状態および対処方法を次に示します。

- 実行履歴の表示

  実行中またはエラー

- リクエストメッセージの状態

  障害発生時にロールバックされ，2 回目のリクエストメッセージ送信時に二重配信防止のチェックによって正常終了としてコミットされます。

- 対処方法

  再実行コマンドまたは運用環境の画面を使用し，ビジネスプロセスを再実行してください。

(例 3) ビジネスプロセスの処理終了後に障害が発生した場合

この場合の実行履歴の表示，リクエストメッセージの状態および対処方法を次に示します。

- 実行履歴の表示

  正常終了

- リクエストメッセージの状態

  障害発生時にロールバックされ，2 回目のリクエストメッセージ送信時に二重配信防止のチェックによって正常終了としてコミットされます。

- 対処方法

  必要ありません。

(例 4) ビジネスプロセスの応答アクティビティ処理終了後に障害が発生した場合

この場合の実行履歴の表示，リクエストメッセージの状態および対処方法を次に示します。

- 実行履歴の表示

  実行中

- リクエストメッセージの状態

  応答アクティビティの処理終了後に，キューはコミットされるため，リクエストメッセージが送信されることはありません。

- 対処方法

  再実行コマンドまたは運用環境の画面を使用し，応答アクティビティのあとのアクティビティを再実行してください。

(9) メッセージのロールバック回数の設定

Message Queue 受付では，リクエスト処理時に障害が発生した場合，必ずロールバックします。これは，キューマネージャのデッドレターキューまたはキューのバックアウトリキューに指定されたキューへメッセージを移動させることで，無限ループを防ぐためです。

ロールバック回数は，IBM MQ のキューマネージャおよびキューで設定します。

なお，設定方法については，IBM MQ のマニュアルを参照してください。

●キューマネージャで設定する項目

キューマネージャで設定するロールバック関連の設定項目を次に示します。

表 2-25　キューマネージャで設定するロールバック関連の設定項目

設定項目	設定内容	設定可能な値	デフォルト値
DEADQ	デッドレターキュー名	キュー名	SYSTEM.DEAD.LETTER.QUEUE

●キューで設定する項目

キューで設定するロールバック関連の設定項目を次に示します。

表 2-26　キューで設定するロールバック関連の設定項目

設定項目	設定内容	設定可能な値	デフォルト値
BOTHRESH	バックアウトの閾値	0～999999999	0
BOQNAME	バックアウトリキュー名	キュー名	なし

注意事項

クライアントで設定する QueueSender#send パラメタに値（メッセージを保持する時間）が指定されている場合は，注意が必要です。メッセージをデッドレターキューに移動しても，QueueSender#send パラメタに指定されている時間を過ぎるとメッセージは削除されます。

(10)　同時実行数の設定

Message Queue 受付では，同時実行数を設定することで，同時に複数のリクエストを処理できます。

同時実行数は，Message Queue 受付および MQ リソースアダプタで設定します。

(a)　Message Queue 受付の設定

次に示す Message Queue 受付定義ファイルのプロパティに設定します。

- pooled-instance.maximum プロパティ
- activation-config.maxPoolDepth プロパティ

(b)　MQ リソースアダプタの設定

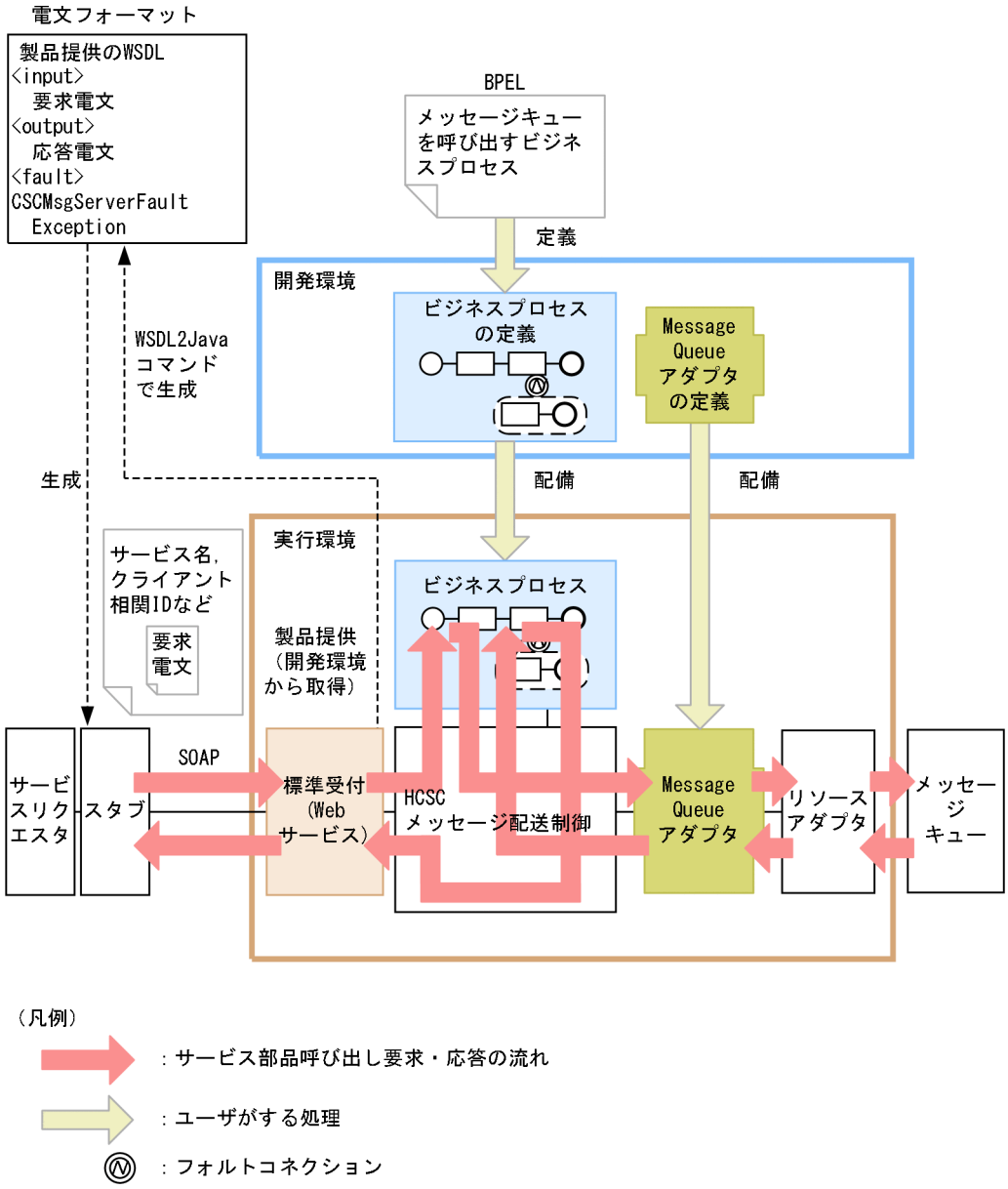
次に示す MQ リソースアダプタのアプリケーション統合属性ファイルの要素に値を設定します。

- //hitachi-connector-property/resourceadapter/config-property[./config-property-name='maxConnections'] ../config-property-value
- //hitachi-connector-property/resourceadapter/property[./property-name='MaxTPoolSize'] ../property-value

2.10.2　Message Queue アダプタを使ったメッセージキューへのアクセス

Message Queue アダプタによるメッセージ送受信について，次の図に示します。

図 2-89　Message Queue アダプタによるメッセージ送受信の実行



注意事項

- Message Queue アダプタはビジネスプロセスからの要求ごとにコネクション解放処理を実行しますが，実際には，リソースアダプタのコネクションプール機能によって一定時間（デフォルト 5 分）リソースアダプタと IBM MQ 間のコネクションは切断されません。

  Connector 属性ファイルの指定（デフォルトは切断しない）に依存します。

- コネクション解放処理でコネクションの解放に失敗すると，Message Queue アダプタは，コネクション解放状態となりますが，接続先の JMS プロバイダシステムではコネクションの解放

を検知していない場合があります。そのため，コネクション状態の不整合が発生し，次の通信処理が失敗する可能性があります。その場合は接続先 JMS プロバイダシステムでコネクション確立状態を 1 度初期化する対応が必要です。

(1) Message Queue アダプタの送信オプション

Message Queue アダプタのメッセージ送信機能は，ビジネスプロセスからのメッセージ送信要求に従い，リソースアダプタを経由し，JMS プロバイダとコネクション接続およびセッション設定を行い，JMS プロバイダへメッセージを送信する機能です。

メッセージの種類は，XML 形式メッセージおよびバイナリ形式メッセージを送信できます。

メッセージを送信する場合，送信オプション（JMS プロパティ）を指定できます。Message Queue アダプタ送信オプション一覧を次に示します。

表 2-27 Message Queue アダプタ送信オプション一覧

項目	送信オプションの概要	指定方法	
		電文	定義
メッセージ送信先 (JMSDestination)	メッセージの送信キュー。	○	○
メッセージ保持時間 (JMSExpiration)	メッセージの保持時間。メッセージ保持時間とは，JMS プロバイダの送信キューに保持し続ける時間のことです。	×	○
メッセージ永続性 (JMSDeliveryMode)	メッセージの永続性。メッセージ永続性とは，システム障害後，または JMS プロバイダを再開した場合など，メッセージをキューに保持するかどうかの指定です。	×	○
メッセージ識別 ID (JMSMessageID)	メッセージを識別する ID。 IBM MQ Resource Adapter を使用する場合，任意の情報が設定されます。	−	−
タイムスタンプ (JMSTimestamp)	メッセージのタイムスタンプ。	−	−
報告メッセージ送信先 (JMSReplyTo)	報告メッセージの送信キュー。	×	×
メッセージタイプ (JMSType)	メッセージタイプ。	×	○
相関識別子 (JMSCorrelationID)	メッセージの相関識別子。	×	△
優先度 (JMSPriority)	メッセージの優先度。IBM MQ Resource Adapter の場合，4 が設定されます。	−	−

（凡例） 指定方法：

電文：ビジネスプロセスからの要求電文による指定の有無。

定義：Message Queue アダプタ通信構成定義ファイルによる指定の有無。

○ ：指定できます。

△ ：メッセージ相関識別子のプレフィックスだけ指定できます。
メッセージ相関識別子を送信する場合，Message Queue アダプタが自動生成します。

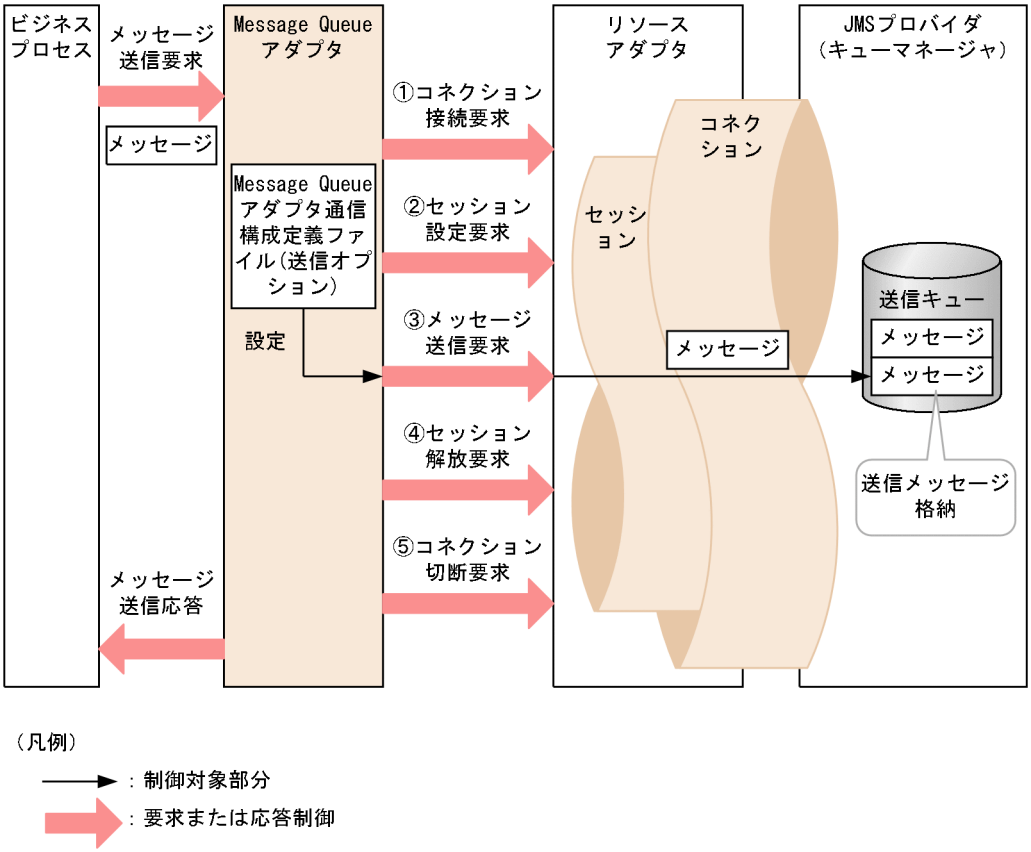
× ：指定できません。

－ ：Message Queue アダプタでは設定しない。リソースアダプタの仕様に依存します。

メッセージ相関識別子を設定したメッセージ送信は，「2.10.2(2) メッセージ相関識別子の生成方法」を参照してください。

メッセージ送信機能の概要を次に示します。

図 2-90 メッセージ送信機能



1. Message Queue アダプタは，ビジネスプロセスからのメッセージ送信要求を受けてコネクションを接続します。
2. コネクション接続後，送信キューとのセッションを設定します。
3. メッセージ送信要求を実行し，メッセージを送信キューに格納します。Message Queue アダプタ通信構成定義ファイルに送信オプションの指定がある場合，送信オプションを設定してメッセージを送信します。
4. メッセージ送信完了後，セッションを解放します。

5. JMS プロバイダとのコネクションを切断します。

コネクション切断後，メッセージ送信応答をビジネスプロセスに返します。

(2) メッセージ相関識別子の生成方法

Message Queue アダプタは，Message Queue アダプタ通信構成定義ファイルの<id_type>タグで Type2 が指定されている場合，メッセージ相関識別子を自動的に生成し送信メッセージに付与します。

メッセージ相関識別子は，プレフィックスと Message Queue アダプタ固有情報から構成されています。メッセージ相関識別子の生成方法は，プレフィックスを付与する場合としない場合の 2 つの方法があります。

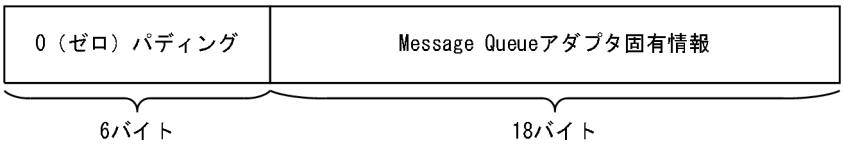
なお，Message Queue アダプタ固有情報は，Message Queue アダプタが設定する情報です。

(a) プレフィックスを付与しない場合

メッセージ相関識別子にプレフィックスを付与しない場合は，<JMSCorrelID_PFX>タグの指定がない場合です。

この場合，先頭 6 バイトを 0（ゼロ）パディングしたものと，Message Queue アダプタ固有情報を合わせてメッセージ相関識別子とします。

図 2-91 メッセージ相関識別子の生成方法（プレフィックスなし）

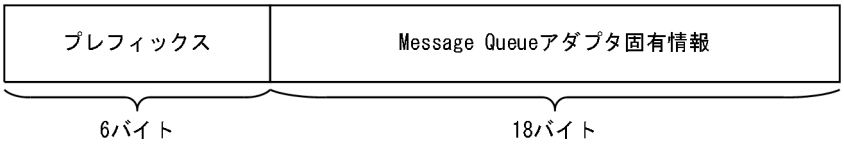


(b) プレフィックスを付与する場合

メッセージ相関識別子にプレフィックスを付与する場合は，<JMSCorrelID_PFX>タグが指定されている場合に，<JMSCorrelID_PFX>タグ値をメッセージ相関識別子の一部とします。プレフィックスは最大で 6 バイト指定できますが，6 バイト未満の場合は，先頭から 0（ゼロ）パディングして 6 バイトとします。例えば，<JMSCorrelID_PFX>タグの指定値が”ABC”の場合,”000ABC”をプレフィックス部分に設定します。

このプレフィックスと Message Queue アダプタ固有情報を合わせてメッセージ相関識別子とします。

図 2-92 メッセージ相関識別子の生成方法（プレフィックスあり）



参考

メッセージ相関識別子を設定したメッセージ送信が失敗した場合，データトレースから送信メッセージおよびメッセージ相関識別子を特定できます。

(3) Message Queue アダプタ使用時のトランザクション

リソースアダプタで提供されるトランザクションサポートレベルは「LocalTransaction」または「XATransaction」を指定してください。ただし，「LocalTransaction」はビジネスプロセスでステータスの永続化をしない場合だけ指定できます。「NoTransaction」は指定できません。

「LocalTransaction」または「XATransaction」を指定すると，JTA（Java Transaction API）と関連を持ちます。例えば，JTA のトランザクションタイムアウト指定値が Message Queue アダプタ通信構成定義ファイルのメッセージ受信応答監視時間（receive_timeout）より短い場合，JTA のトランザクションタイムアウト指定値でタイムアウトします。

ビジネスプロセスからメッセージ受信要求またはメッセージ送受信要求を受けると，Message Queue アダプタはメッセージ受信要求を発行します。このメッセージ受信要求に対してリソースアダプタが一定時間内にメッセージを受信できない場合，リソースアダプタはメッセージ受信応答監視時間のタイムアウトとなります。リソースアダプタからの通知に従い，Message Queue アダプタは，ビジネスプロセスへ Fault 応答情報を通知します。また，ビジネスプロセスからブラウズ・メッセージ受信要求を受け，ブラウズ・メッセージ受信要求を行いますが，受信キューにメッセージが存在しない場合も，ビジネスプロセスへ Fault 応答情報を通知します。

フォルトが発生したときの処理の定義については，マニュアル「サービスプラットフォーム 開発ガイド 基本開発編」の「5.4.4　フォルト処理を定義する」を参照してください。

(4) メッセージ送信や受信で使用する JMS API

(a) メッセージ送信

javax.jms.BytesMessage と javax.jms.TextMessage の送信をサポートしています。

Message Queue アダプタは次のメソッドを使用して，メッセージインスタンスにデータを書き込みます。

- javax.jms.TextMessage の場合：javax.jms.TextMessage#setText()
- javax.jms.BytesMessage の場合：javax.jms.BytesMessage#writeBytes()

(b) メッセージ受信，またはブラウズ・メッセージ受信

javax.jms.BytesMessage と javax.jms.TextMessage の受信をサポートしています。

Message Queue アダプタは次のメソッドを使用して，メッセージインスタンスからデータを読み取ります。

- javax.jms.TextMessage の場合：javax.jms.TextMessage#getText()

- javax.jms.BytesMessage の場合：javax.jms.BytesMessage#readBytes()

(5)　Message Queue アダプタのオペレーション動的変更

Message Queue アダプタの実行オペレーションをビジネスプロセス内で動的に変更できます。

Message Queue アダプタのオペレーション動的変更の詳細については，「2.22　TP1 アダプタおよび Message Queue アダプタのオペレーション動的変更機能」を参照してください。

2.11　FTP クライアントおよび FTP サーバとの接続

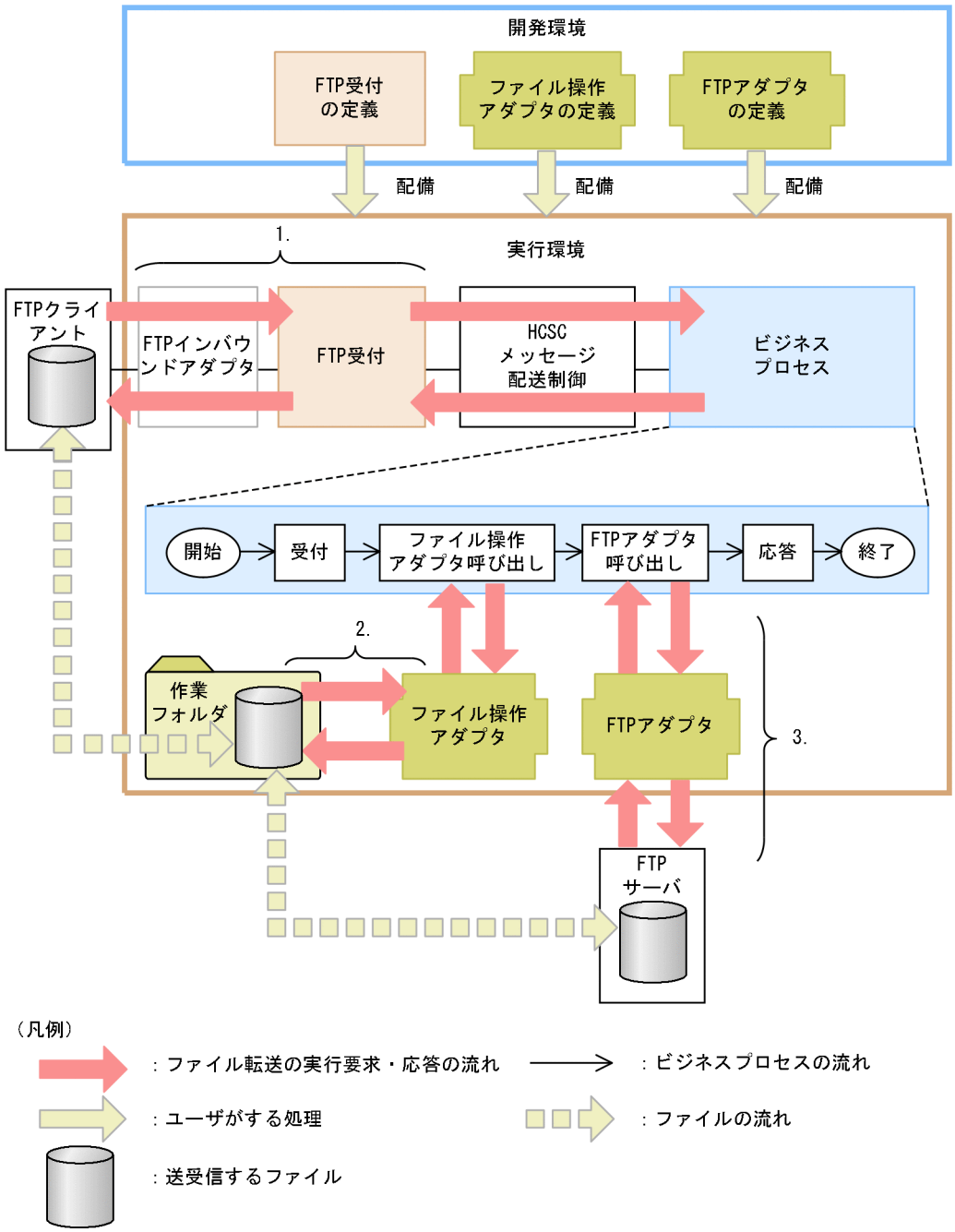
FTP 受付，FTP アダプタ，およびファイル操作アダプタを使用することで，クライアント・サーバ間の FTP によるファイルの送受信をサービスプラットフォーム上で中継できます。

FTP 受付，FTP アダプタ，およびファイル操作アダプタの機能については，「8.　FTP と連携してファイルを転送する機能（FTP 連携）」を参照してください。また，FTP 受付の定義については，マニュアル「サービスプラットフォーム 開発ガイド 受付・アダプタ定義編」の「2.4　FTP 受付の定義」を，FTP アダプタ，およびファイル操作アダプタの定義については，マニュアル「サービスプラットフォーム 開発ガイド 受付・アダプタ定義編」の「3.　サービスアダプタの定義」をそれぞれ参照してください。

2.11.1　FTP と連携したファイルの転送

FTP によるファイルの送受信の流れを次の図に示します。

図 2-93　FTP によるファイルの送受信の流れ



図中の 1.～3.について説明します。

1. FTP クライアントからのリクエスト受付

   FTP インバウンドアダプタおよび FTP 受付を使用して，FTP クライアントからのファイル転送要求を受け付けます。FTP インバウンドアダプタでは，FTP クライアントの認証，ファイル転送処理などの処理を行います。FTP クライアントからのファイル転送要求は，FTP インバウンドアダプタを経由して FTP 受付に送られます。そのあと，FTP 受付からビジネスプロセスが呼び出されます。

   ファイルの送受信は，FTP 受付が作成する HCSC サーバ上の作業フォルダを利用して行われます。

2. ファイルのデータ変換，複製，および削除

送信側のファイル形式と受信側のファイル形式が異なる場合，ファイル操作アダプタを使用してファイルのレイアウト変換，および文字コード変換を行います。

また，ファイル操作アダプタは，HCSC サーバ上の作業フォルダや任意のフォルダにアクセスし，フォルダ間のファイルの複製や削除，作業フォルダの削除などの操作を行います。

ファイル操作アダプタはビジネスプロセスから呼び出されます。

3. FTP サーバとの接続

FTP アダプタを使用して，FTP サーバとの接続や作業フォルダと FTP サーバとの間のファイル送受信を行います。

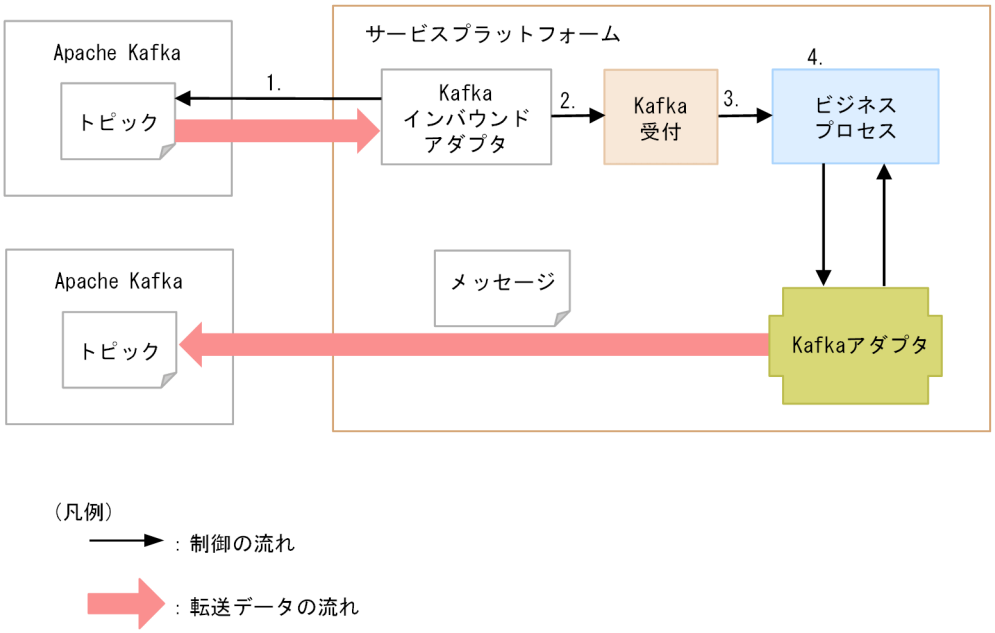
FTP アダプタはビジネスプロセスから呼び出されます。

2.12　Kafka との接続

サービスプラットフォームでは，Apache Kafka と連携してメッセージの送受信をする機能を提供しています。メッセージの受信には，Kafka 受付および Kafka インバウンドアダプタを使用します。メッセージの送信には，Kafka アダプタを使用します。

Apache Kafka と連携したメッセージの送受信の流れを次の図に示します。

図 2-94　Apache Kafka と連携したメッセージの送受信の流れ



メッセージの受信

図中の 1.～4.について説明します。

1. Kafka インバウンドアダプタが，Apache Kafka からメッセージを取得します。
2. Kafka インバウンドアダプタが，取得したメッセージを Kafka 受付に渡します。
3. Kafka 受付が，ビジネスプロセスにメッセージを渡します。
4. ビジネスプロセス内で，データ変換アクティビティなどを使用して，メッセージを編集します。

メッセージの送信

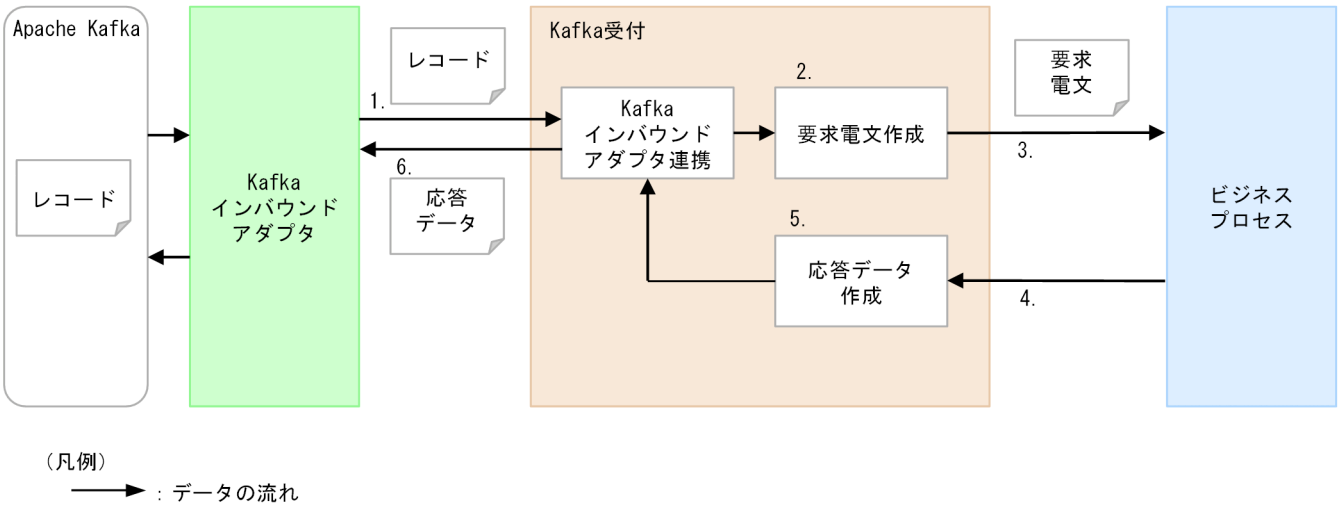
Kafka アダプタは，ビジネスプロセスからのメッセージ送信要求に従って，Apache Kafka にメッセージを送信します。

2.12.1　Kafka 受付の機能

Kafka 受付を利用することで，Kafka から Kafka インバウンドアダプタを経由してメッセージを取得し，ビジネスプロセスに渡すことができます。また，ビジネスプロセス内でデータ変換アクティビティなどを使用してメッセージのデータを編集することもできます。

Kafka 受付を利用したデータの流れを次に示します。ここでは，Kafka から取得したメッセージを「レコード」と表記します。

図 2-95　Kafka 受付を利用したデータの流れ



図中の 1.～6.について説明します。

1. Kafka インバウンドアダプタから，インバウンドアダプタ連携を使用してレコードを受け取ります。
   Kafka インバウンドアダプタの詳細については，「2.12.3　Kafka インバウンドアダプタの機能」を参照してください。
2. 受け取ったレコードから要求電文を作成します。
   要求電文の作成については，「(3)　要求電文の作成」を参照してください。
3. Kafka 受付からビジネスプロセスに作成した要求電文を渡します。
4. ビジネスプロセスから応答が返ります。
5. Kafka インバウンドアダプタに返却する応答データを作成します。
   Kafka 受付やビジネスプロセスで例外が発生した場合は，メッセージ処理でエラーとなったレコードの情報（トピック・パーティション・オフセット）を応答データに設定して，Kafka インバウンドアダプタに返却します。すべてのレコードの処理が正常終了した場合，応答データには何も設定されません。
6. Kafka インバウンドアダプタに応答データを返却します。

(1)　Kafka 受付の同時実行数

Kafka 受付の同時実行数は 1 つです。同時実行数は変更できません。

(2)　Kafka 受付で使用できる通信モデル

Kafka 受付の通信モデルは，非同期通信です。通信モデルに「同期」を設定した場合の動作は保証されません。

(3) 要求電文の作成

Kafka 受付は，Kafka インバウンドアダプタから渡されたレコードから，要求電文を作成します。

Kafka 受付の要求電文の形式については，マニュアル「サービスプラットフォーム 開発ガイド 受付・アダプタ定義編」の「2.8.2 Kafka 受付の電文フォーマットを作成する」を参照してください。

(a) レコードのキーおよび値の設定

Kafka インバウンドアダプタから渡されたレコードのキーおよび値には，Kafka 受付定義ファイルで指定したデシリアライザ（urecp-kafka.record-binary.encoding）によってデシリアライズされた値が設定されます。要求電文（ヘッダ）に設定するレコードのキー，および要求電文（ボディ）に設定するレコードの値には，レコードから取得した値を設定します。

レコードのキーがバイナリ形式のデータの場合は，バイナリ形式のデータのエンコーディングを行い，エンコーディング後の値を要求電文に設定します。

デシリアライザの設定については，マニュアル「サービスプラットフォーム リファレンス」の「3.14.1 Kafka 受付定義ファイル」を参照してください。

(b) バイト配列形式のデータのエンコーディング

レコードのキーがバイト配列形式の場合は，Kafka 受付定義ファイルの urecp-kafka.record-binary.encoding の設定値に応じてエンコードされます。エンコードされたレコードのキーの値は，要求電文の対応する要素値に設定されます。エンコードは，バイト配列の長さが 1 以上のときに行います。

また，レコードのヘッダの値は，文字セット「UTF-8」を使用してデコードされます。デコードされたレコードのヘッダの値は，ヘッダ電文の対応する要素値に設定されます。

(c) レコードのデータ型とデシリアライザの対応

レコード送信時のシリアライズする前のレコードのキーおよびレコードの型と，Kafka 受付定義ファイルで設定するデシリアライザの対応を次に示します。レコードのキーおよびレコードの型と，デシリアライザの設定が一致しない場合は，値の取得時にエラーが発生する，または意図しない値になることがあります。

表 2-28 レコードのキーおよびレコードの型とデシリアライザの対応

レコードのキーおよびレコードの型	デシリアライザの設定
java.lang.String	StringDeserializer
byte[]	ByteArrayDeserializer

(d) レコードの値とデシリアライザの対応

電文形式およびレコードの値に応じて，次のデシリアライザを設定してください。

表 2-29　レコードの値とデシリアライザの対応

電文形式	レコードの値	デシリアライザの設定
XML	JSON 形式	StringDeserializer
	XML 形式	
バイナリ	CSV 形式	ByteArrayDeserializer
	文字列	
	バイナリデータ	
any	文字列	ByteArrayDeserializer
	バイナリデータ	

(4)　ビジネスプロセスで処理できるレコード数

Kafka インバウンドアダプタから渡されるレコードには，1 つのレコードの情報が格納されています。そのため，ビジネスプロセスで処理できるレコード数は 1 つです。

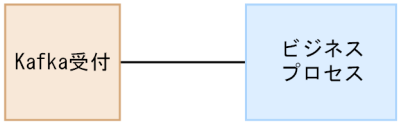
(5)　ビジネスプロセスとの関係

ビジネスプロセスと Kafka 受付が構築できる接続関係は次のどちらかです。

- ビジネスプロセスと Kafka 受付の接続関係が 1：1 の場合
- ビジネスプロセスと Kafka 受付の接続関係が 1：n の場合

例を次に示します。

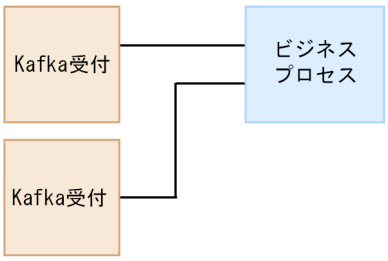
(例 1) ビジネスプロセスと Kafka 受付の接続関係が 1：1 の場合



説明

- 1 つの Kafka 受付に関連づけるビジネスプロセスは 1 つ
- 1 つのビジネスプロセスに関連づける Kafka 受付は 1 つ

(例 2) ビジネスプロセスと Kafka 受付の接続関係が 1：n の場合



説明

- 1 つの Kafka 受付に関連づけるビジネスプロセスは 1 つ
- 1 つのビジネスプロセスに関連づける Kafka 受付は複数

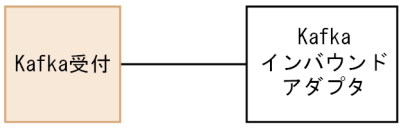
(6) Kafka インバウンドアダプタとの関係

Kafka インバウンドアダプタと Kafka 受付が構築できる接続関係は次のどちらかです。

- Kafka インバウンドアダプタと Kafka 受付が 1：1 の接続関係の場合
- Kafka インバウンドアダプタと Kafka 受付が 1：n の接続関係の場合

例を次に示します。

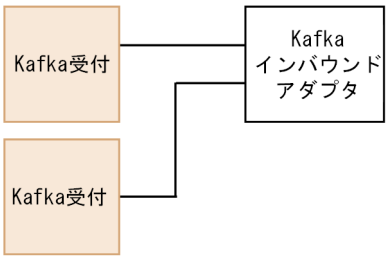
(例 1) Kafka インバウンドアダプタと Kafka 受付が 1：1 の接続関係の場合



説明

- 1 つの Kafka インバウンドアダプタに関連づいている Kafka 受付は 1 つ
- 1 つの Kafka 受付に関連づいている Kafka インバウンドアダプタは 1 つ

(例 2) Kafka インバウンドアダプタと Kafka 受付が 1：n の接続関係の場合



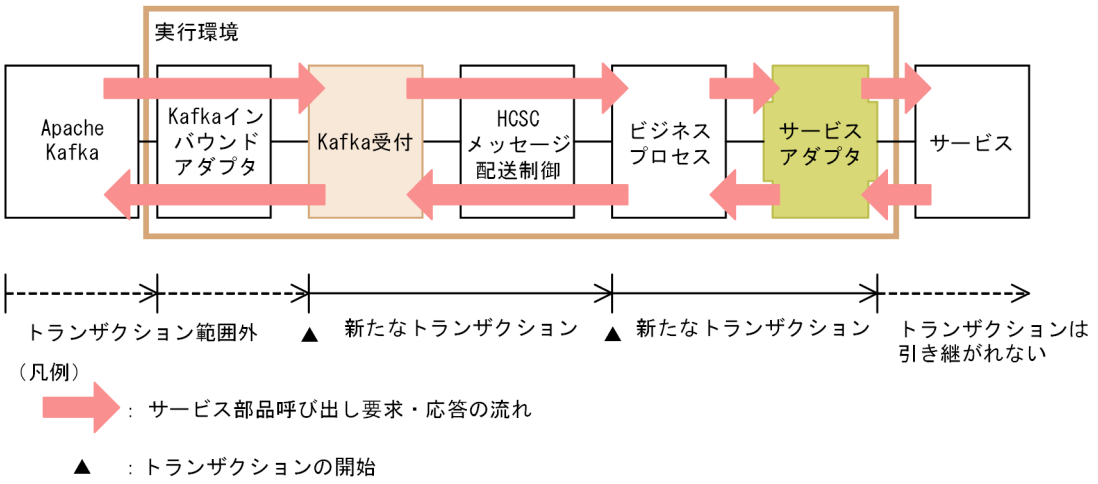
説明

- 1 つの Kafka インバウンドアダプタに関連づいている Kafka 受付は複数
- 1 つの Kafka 受付に関連づいている Kafka インバウンドアダプタは 1 つ

(7)　Kafka 受付使用時のトランザクション

ビジネスプロセスを呼び出すときの Kafka 受付のトランザクションを次に示します。

図 2-96　Kafka 受付のトランザクション

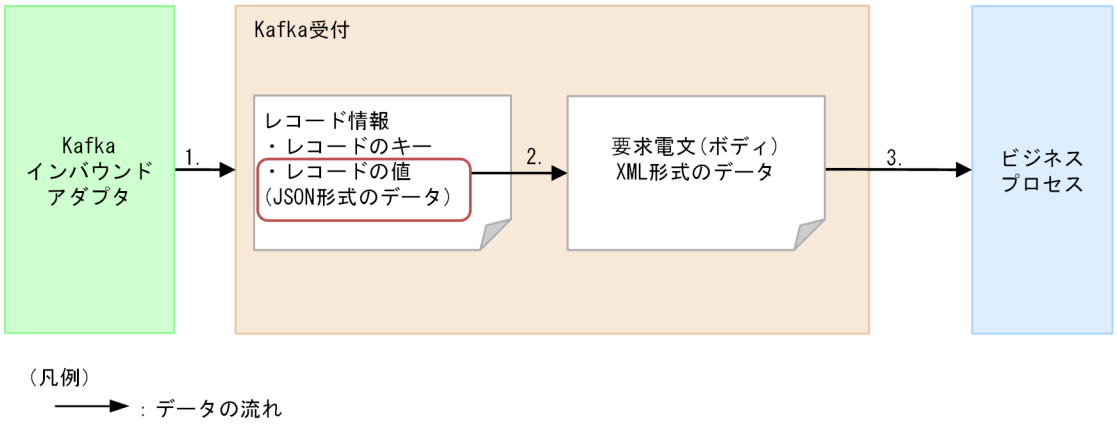


2.12.2　Kafka 受付の JSON-XML 変換機能

Kafka 受付の JSON-XML 変換機能では，JSON 形式のレコードの値をビジネスプロセスで処理できる XML 形式に変換します。Kafka 受付では，レコードの値が JSON 形式かどうかはチェックしないで，レコードの値が JSON 形式とみなして変換します。

レコードの値を JSON-XML 形式に変換する処理の流れを次に示します。

図 2-97　レコードの値を JSON-XML 形式に変換する流れ



図中の 1.～3.について説明します。

1. Kafka インバウンドアダプタから Kafka 受付が呼び出され，Kafka インバウンドアダプタからレコード情報が渡されます。

2. Kafka 受付の JSON-XML 変換機能を使用して，レコードの値（JSON 形式のデータ）を XML 形式のデータに変換します。
3. Kafka 受付がビジネスプロセスを呼び出します。
   ビジネスプロセスに渡す要求電文（ボディ）は，2.で作成した XML 形式のデータです。

Kafka 受付で JSON-XML 変換を行う場合は，Kafka 受付定義ファイルの urecp-kafka.json2xml で指定します。Kafka 受付定義ファイルの urecp-kafka.json2xml については，マニュアル「サービスプラットフォーム リファレンス」の「3.14.1　Kafka 受付定義ファイル」を参照してください。

JSON-XML 変換機能では，変換処理の動作定義を設定できます。Kafka 受付定義ファイルに JSON-XML 変換定義ファイルが指定されていた場合，JSON-XML 変換定義ファイルの設定が JSON-XML 変換機能に反映されます。なお，JSON-XML 変換定義ファイルが指定されていない場合は，デフォルト値が設定されます。JSON-XML 変換定義ファイルは，Kafka 受付の起動時に読み込まれます。

JSON-XML 変換定義ファイルの詳細については，マニュアル「サービスプラットフォーム リファレンス」の「3.18.1　JSON-XML 変換定義ファイル」を参照してください。

2.12.3　Kafka インバウンドアダプタの機能

Kafka インバウンドアダプタは Kafka からメッセージを受信するアダプタです。サービスプラットフォームから Kafka に接続し，Broker のトピックからメッセージとしてレコードを取得して，Kafka 受付を呼び出します。

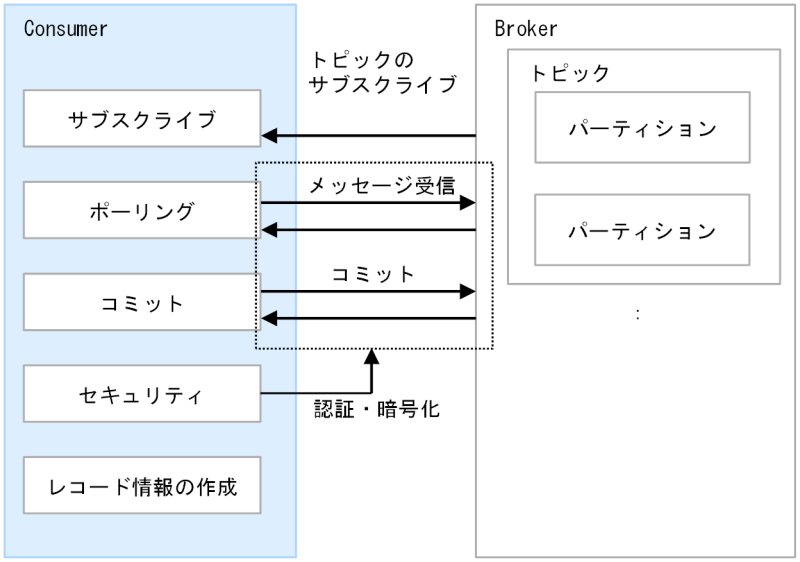
Kafka インバウンドアダプタが接続できる Apache Kafka のバージョンは，2.6.0～3.9.0 です。また，Kafka インバウンドアダプタと連携できるアプリケーションは，サービスプラットフォームが提供する Kafka 受付だけです。

(1)　メッセージの受信機能

メッセージの受信機能は，Broker からメッセージを受信するための機能です。Consumer とも呼ばれます。

メッセージの受信機能（Consumer）の機能概要を次の図に示します。

図 2-98　メッセージの受信機能（Consumer）の機能概要



Consumer は，Kafka 受付の起動の延長で開始します。

(a)　パーティションの割り当て

パーティションの割り当ては，Consumer グループのグループリーダー（グループ内の最初の Consumer）によって動的に行われます。Consumer に手動でパーティションを割り当てることはできません。

Consumer グループについては，「(5)　Kafka インバウンドアダプタの Consumer グループ」を参照してください。

(b)　サブスクライブ

Consumer が，どのトピックからレコードを受信するか設定することをサブスクライブといいます。Consumer のサブスクライブ対象のトピックを次に示します。

- ユーザが作成したトピック
- Kafka が作成した内部トピック

サブスクライブ対象のトピックが存在しない場合でも，Broker 側ではトピックは自動生成されません。この場合，Consumer の処理の実行中にトピックを作成することで，トピック作成後のポーリング処理でデータが読み込まれます。

(c)　データのポーリング

データのポーリングでは，Consumer に割り当てられたトピック内のパーティションからデータを読み込みます。データの取得単位は，レコードバッチ単位です。ポーリング処理は，この機能が停止しない限り繰り返し行われます。

クラスタの初期接続時のポーリング

Consumer は，Kafka クラスタの情報を取得するために，データのポーリング実行時に Kafka クラスタの初期接続を実施します。Kafka クラスタへの初期接続を確立するために使用するホスト／ポートペアのリストのどれかの Broker と初期接続を行い，接続できた Broker から Kafka クラスタの情報を取得します。ポーリング処理中に接続できた場合，以降のポーリングでは初期接続は必要ありません。

初期接続ができない場合，ポーリングの待ち時間内に繰り返し初期接続を行います。

ポーリングの待ち時間内に初期接続ができなかった場合，データのポーリング実行時にエラーは発生しません。

Broker の接続時のポーリング

初期接続が行えた場合は，次に示す Broker に接続します。

- Consumer グループの，グループコーディネータの Broker
- パーティションのリーダーである Broker

ポーリングの待ち時間内に接続ができなかった場合は，ポーリング実行時にエラーは発生しません。

レコードのデシリアライズ

Consumer は，Broker から取得したバイト配列のレコードをデシリアライズします。デシリアライズに使用するデシリアライザを Kafka 受付定義ファイルで設定できます。レコードのデシリアライズに関するプロパティを次に示します。

表 2-30　レコードのデシリアライズに関するプロパティ

プロパティ	説明
consumer.key.deserializer	レコードのキーのデシリアライザを指定します。
consumer.value.deserializer	レコードの値のデシリアライザを指定します。

取得するメッセージの順序

Consumer が取得するメッセージは，単一のパーティション内のメッセージに対してだけ順序が保証されます。トピックが複数のパーティションで構成される場合，取り出されるメッセージが，送信された順序で取得できるとは限りません。

受信するレコード数の上限

ポーリングごとに Broker から受信するレコード数の上限を設定できます。レコード数の上限は，Kafka 受付定義ファイルの consumer.max.poll.records で設定します。

フェッチ要求のレコードのデータサイズ

Consumer からのデータのフェッチ要求に対して，Broker が返す必要のあるデータサイズを設定できます。データサイズとは，1 回のポーリングで Consumer が送信する，すべてのレコードデータのサイズです。

フェッチ要求のレコードのデータサイズについては，「(7)　レコードのデータサイズ」を参照してください。

(d)　受信できるメッセージの圧縮形式

Consumer では，次に示す形式で圧縮されたメッセージが受信できます。Consumer はメッセージを伸張して処理を行います。Consumer が受信できない形式で圧縮されたレコードを受信した場合の動作は保証されません。

表 2-31　Consumer で受信できる圧縮方式

圧縮方式	受信可否
gzip	○
lz4	×
snappy	×
zstd	×
圧縮していないデータ	○

（凡例）

○：受信できます。

×：受信できません。

(e)　コミット

Consumer がどこまでメッセージを読み込んだかを Broker に通知することをコミットといいます。

コミットについての詳細は，「(6)　コミットのタイミング」を参照してください。

(f)　セキュリティ

Kafka インバウンドアダプタでは，TLS を使用した暗号化と認証，ホスト名の検証をサポートしています。TLS で使用するセキュリティ通信用プロトコルについては，Kafka の公式ドキュメントを参照してください。

TLS で使用できる TLS 暗号スイートの制限はありません。ただし，使用する暗号スイートを設定することはできません。

ホスト名の検証は，証明書内のホスト名が接続先のホスト名と一致するかどうかをチェックします。ホスト名の検証を無効にすることはできません。

(g)　レコード情報の作成

Broker から取得したレコードを基に，Kafka 受付に渡すレコード情報を作成します。レコード情報の作成は，1 回のポーリングで受信したレコードごとに行います。レコード情報が作成されるのは，Broker からメッセージが取得できた場合だけです。

取得したメッセージから作成されるレコード情報を次に示します。

表 2-32　取得したメッセージから作成されるレコード情報

レコードの項目	説明
レコードのキー	レコードのキーです。
レコードの値	レコードの値です。
オフセット	対応するパーティション内のレコードの位置です。
パーティション	レコードの受信元のパーティションです。
タイムスタンプ	メッセージ送信時のレコードのタイムスタンプです。
トピック	レコードを受け取ったトピックです。
ヘッダ	レコードのヘッダ情報です。

(2)　Kafka 受付呼び出し機能

Kafka 受付の呼び出しを実行することで，取得したレコード情報をビジネスプロセスで処理できるようになります。

Kafka 受付の呼び出しは，ポーリングで取得したメッセージごとに行われます。そのため，ポーリングで複数のメッセージを受信した場合でも，Kafka 受付に渡されるメッセージは 1 つです。また，Kafka 受付の呼び出しは Broker からメッセージが受信できた場合だけ行われます。

(3)　Kafka インバウンドアダプタの実行

メッセージの受信機能および Kafka 受付呼び出し機能は，J2EE サーバのリソースアダプタのワーク管理を使用して実行します。リソースアダプタのワーク管理については，マニュアル「アプリケーションサーバ 機能解説 基本・開発編(コンテナ共通機能)」の「3.16.2　リソースアダプタのワーク管理」を参照してください。

リソースアダプタのワーク管理では，ワークに割り当てるスレッドをスレッドプールで管理します。スレッドプールの動作は，Connector 属性ファイルで設定します。Kafka インバウンドアダプタのスレッドプールに関するプロパティを次に示します。

表 2-33　スレッドプールに関するプロパティ

プロパティ	説明
MaxTPoolSize	スレッドプールで同時に実行される最大スレッド数
MinTPoolSize	スレッドプールにプールする最小スレッド数
TPoolKeepalive	スレッドプールのスレッド解放までのタイムアウト値

スレッドプールに関する設定については，マニュアル「サービスプラットフォーム システム構築・運用ガイド」の「3.1.2(11)　Kafka インバウンドアダプタのセットアップ」を参照してください。

(4)　配備できるリソースアダプタ数

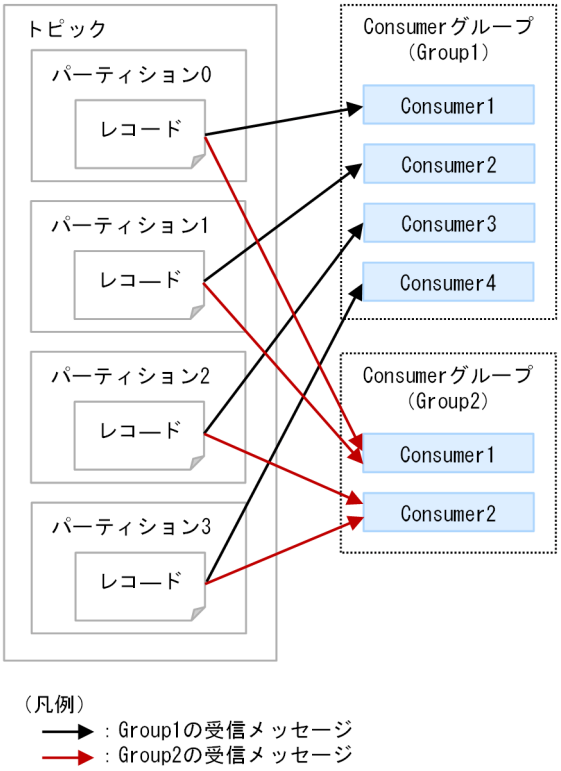
1 つの HCSC サーバに配備できるリソースアダプタ数は，1 つです。リソースアダプタを複数配備した場合の動作は保証されません。

(5)　Kafka インバウンドアダプタの Consumer グループ

Consumer は，Consumer グループと呼ばれるグループに属することができます。Consumer は，初回のポーリングまたは Consumer グループのリバランス時に，Consumer グループのコーディネータである Broker に，グループへの参加要求を行います。

Consumer グループのメッセージの受信例を次に示します。

図 2-99　Consumer グループのメッセージの受信例



この例の場合，「Group1」「Group2」で同一のメッセージが受信されます。

同一グループ内に複数の Consumer がある場合，グループ内のどれか 1 つの Consumer が一意のメッセージを受信できます。

パーティションは Broker から各 Consumer に割り振られますが，Consumer グループ内の各 Consumer に同一のパーティションが割り当てられることはありません。

レコードは各パーティションのどれかに格納されるため，Consumer グループ内の Consumer が同一のメッセージを受信することはありません。

Consumer グループが異なる場合は，Consumer グループ単位で同一のメッセージを受信することができます。

(a) Consumer グループのメンバーへのパーティションの割り当て

パーティション数に応じて，Consumer グループのメンバーへのパーティションは次のように割り当てられます。

表 2-34 Consumer グループのメンバーへのパーティションの割り当て

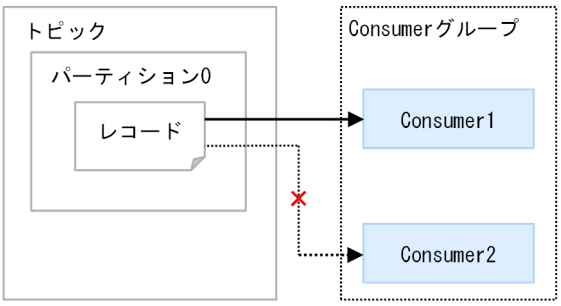
パーティション数と Consumer グループのメンバーの関係	パーティションの割り当て
パーティション数 ＜ グループ内のメンバー	パーティション数がグループ内のメンバーより少ない場合，パーティションが割り当てられないメンバーが存在します。
パーティション数 ＝ グループ内のメンバー	パーティション数がグループ内のメンバーと等しい場合，均等に割り当てられます。
パーティション数 ＞ グループ内のメンバー	パーティション数がグループ内のメンバーより多い場合，割り当てに偏りが出ます。

パーティション数 ＜ グループ内のメンバーの場合

パーティション数がグループ内のメンバーより少ない場合，パーティションが割り当てられない Consumer が存在します。パーティションが割り当てられない Consumer は，リバランスでパーティションが割り当てられない限りメッセージを受信しません。

パーティション数がグループ内のメンバーより少ない場合の割り当て例を次に示します。

パーティション数 ＜ グループ内のメンバーの場合の割り当て例



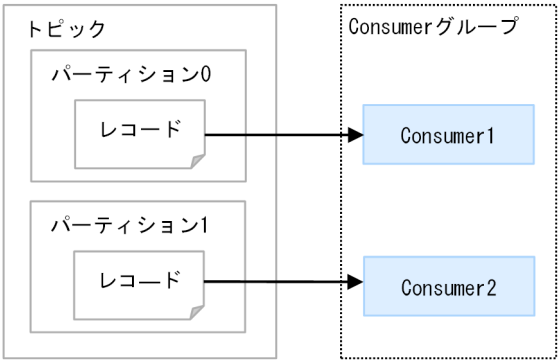
Consumer2 に割り当てるパーティションがないため，Consumer2 はメッセージを受信しません。

パーティション数 ＝ グループ内のメンバーの場合

パーティション数がグループ内のメンバーと等しい場合，グループ内のすべての Consumer がメッセージを受信できる状態となります。

パーティション数がグループ内のメンバーと等しい場合の割り当て例を次に示します。

パーティション数 ＝ グループ内のメンバーの場合の割り当て例



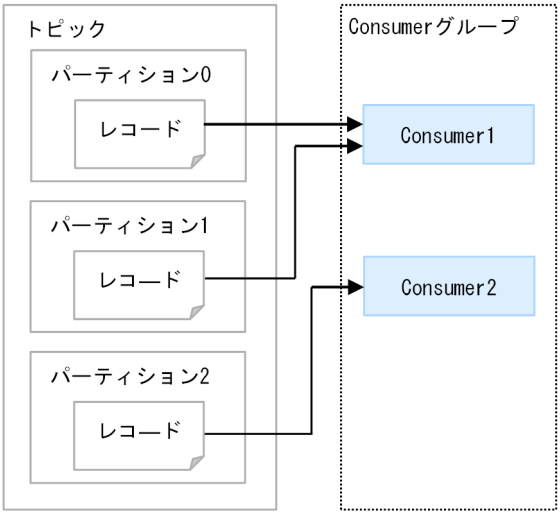
グループ内のすべての Consumer にパーティションが割り当たります。

パーティション数 ＞ グループ内のメンバーの場合

パーティション数がグループ内のメンバーより多い場合，グループ内のどれかの Consumer に，多くのパーティションが割り当てられます。

パーティション数がグループ内のメンバーより多い場合の割り当て例を次に示します。

パーティション数 ＞ グループ内のメンバーの場合の割り当て例



グループ内にパーティションを均等に割り当てることができないため，Consumer1 に 2 つのパーティションが割り当てられます。

(b) パーティションの範囲割り当て

Kafka では，Consumer がグループ構成になっている場合，トピック内のパーティションをグループのメンバーに分散して割り当てます。これを範囲割り当てといいます。

範囲割り当ては，次の順序で行われます。

1. トピックごとに使用できるパーティションを番号順に，Consumer を辞書式順序でレイアウトします。
2. パーティションの数を Consumer の総数で割り，各 Consumer に割り当てるパーティションの数を決定します。

   均等に分割できない場合，最初の Consumer から順にパーティションを 1 つ追加で割り当てます。

範囲割り当ての例

2 つの ConsumerC0 と C1，2 つのトピック t0 と t1 があり，各トピックに 3 つのパーティション p0，p1，p2 がある場合，パーティションは t0p0，t0p1，t0p2，t1p0，t1p1，および t1p2 の 6 つとなります。

トピック t0 のパーティションは 3 つあり，これらのパーティションを 2 つの Consumer で分割します。均等に分割できないため，C0 にパーティションが 1 つ追加で割り当てられます。トピック t1 のパーティションも同様に割り当てが行われます。

そのため，Consumer ごとのパーティションの割り当ては次のようになります。

- C0：t0p0，t0p1，t1p0，t1p1
- C1：t0p2，t1p2

(c) Consumer グループのメンバーシップ

Consumer グループのメンバーシップは動的メンバーシップとなります。静的メンバーシップに変更することはできません。

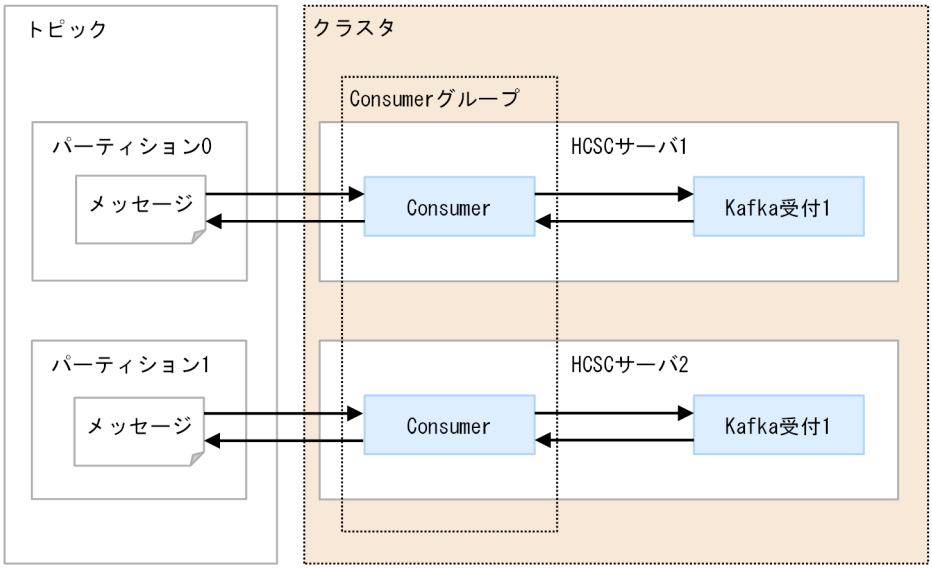
(d) Consumer グループのリバランス

Consumer グループへの Consumer の加入や除外などを契機として，Broker がグループ内のリバランスを行います。Consumer グループのリバランス時は，ポーリングが停止し，リバランス完了後にポーリングが再開します。

(e) 冗長構成

パーティションは Consumer ごとに割り当てられるため，冗長構成にすることができます。Consumer グループがクラスタ内の同一の Kafka 受付を使用した冗長構成の例を次の図に示します。

図 2-100 Consumer グループがクラスタ内の同一の Kafka 受付を使用した冗長構成の例



ただし，Consumer ごとにパーティションが割り当てられるため，冗長構成にしても特定の HCSC サーバの負荷を分散することはできません。

(f) 注意事項

Consumer グループについての注意事項を次に示します。

- Consumer グループに対する操作は，Consumer グループの情報表示に関する操作，およびオフセットのリセットだけ許可されています。それ以外の操作を行った場合の動作は保証されません。
- サービスプラットフォームで構成した Consumer グループに参加できるのは，サービスプラットフォームの Consumer だけです。その他の製品の Consumer がグループに参加した場合の動作は保証されません。

(6) コミットのタイミング

Consumer がコミット処理を行うことで，どこまでメッセージを読み込んだか Broker に通知されます。メッセージをどこまで読んだかを管理する単位をオフセットといいます。

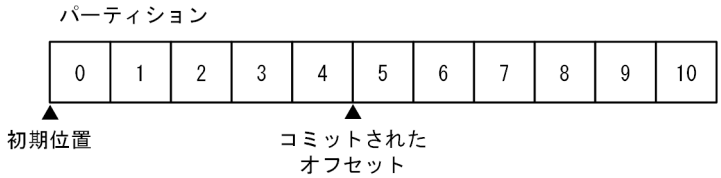
オフセットには，次の 2 つがあります。

- 現在のオフセット
  Consumer が読み取る次のメッセージのオフセットを示します。
- コミットされたオフセット
  パーティションの最後にコミットされたオフセットを示します。

オフセットが初期化されている場合，初回のポーリングでは割り当てられたパーティションの「コミットされたオフセット」以降のメッセージを受信します。2 回目以降のポーリングでは「現在のオフセット」以降のメッセージを受信します。オフセットが初期化されていない場合，または「現在のオフセット」がサーバ上にない場合は，オフセットの自動リセットを行います。Consumer は，パーティション内のメッセージの読み込み位置を指定することはできません。

オフセットされたコミットの例を次に示します。

図 2-101　オフセットされたコミットの例



オフセットの初期位置が 0 の場合，5 通のメッセージを受信して，コミットを行うと，コミットされたオフセットは「5」となります。

コミットのタイミングは，Consumer 単位で制御できます。コミットのタイミングは，次のどちらかを選択できます。

- Broker からのメッセージ受信後
- メッセージ処理後（Kafka 受付からの応答後）

コミットのタイミングの違いによる動作の違いを次に示します。

表 2-35　コミットタイミングによる動作の違い

コミットのタイミング	一度のポーリングで受信できるメッセージ数	同一メッセージを受信するケース	ポーリングの最大遅延時間内にメッセージ処理が終了しない場合	同一メッセージの再送制限
メッセージ受信後	1 つ以上	コミットが完了するまでにエラーが発生した場合，またはコミットが実行されなかった場合※1 は，リバランスでパーティションを引き継いだ Consumer が同一メッセージを受信します。 リバランスが発生して該当するパーティションがグループ内の Consumer に引き継がれた場合は，パーティションを引き継いだ Consumer はコミットされたオフセットからメッセージを受信するため，同一のメッセージを受信します。	処理を続行します。 Consumer はグループから除外されます。この場合，次回のポーリング処理で再度グループに加入するようにリバランスされます。	制限できない
メッセージ処理後	1 つだけ	メッセージ受信後にエラーが発生した場合，またはコミットが実行されなかった場合※1 は，再送の上限数まで同一の Consumer が受信します。 また，このあとにリバランスが発生した場合は，パーティションを引き継いだ Consumer が同一メッセージを受信します。	Consumer を停止します。 Consumer はグループから除外されます。グループから除外された場合，割り当てられていたパーティションがグループ内の他の Consumer に割り当てられます。※2	制限できる※3

注※1

ポーリング後に Kafka 受付を停止すると，コミットされずに，同一メッセージが再送される場合があります。

コミットタイミングの設定によって，次の動作となることがあります。

・メッセージ受信後コミット

コミットが実行されません。

・メッセージ処理後コミット

メッセージ処理は実行されますが，コミットが実行されないことがあります。

注※2

グループから除外された Consumer は，割り当てられていないパーティションに対してコミットを実行することがあります。この場合，コミット処理が失敗し，エラーメッセージを出力します。

パーティションが引き継がれた Consumer は，コミットされたオフセットが変更されていないため，再度同じメッセージを受信します。

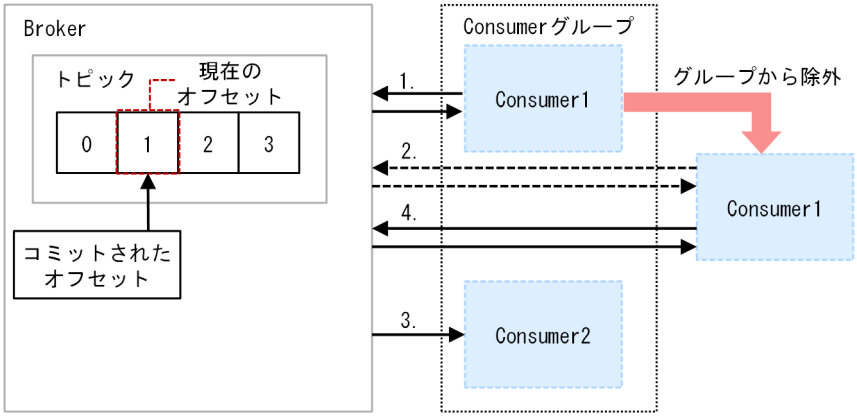
注※3

Kafka 受付からメッセージ処理の失敗の応答が返却された場合，次回のポーリングで同一のメッセージを受信して，Kafka 受付に送信します。Consumer では，同一メッセージの受信をリトライする回数を制御できます。リトライ回数内に，メッセージ処理が正常終了しない場合は Consumer を停止します。

また，一時的な障害ではなくメッセージの不正など，ユーザが対処しないと障害が回復しない場合は，リバランスでパーティションを割り当てられた Consumer も同様の動作となるため，グループ内のすべての Consumer が停止します。

パーティションを引き継いだ場合の例を次に示します。

図 2-102　パーティションを引き継いだ場合の例



図中の 1.～4.について説明します。

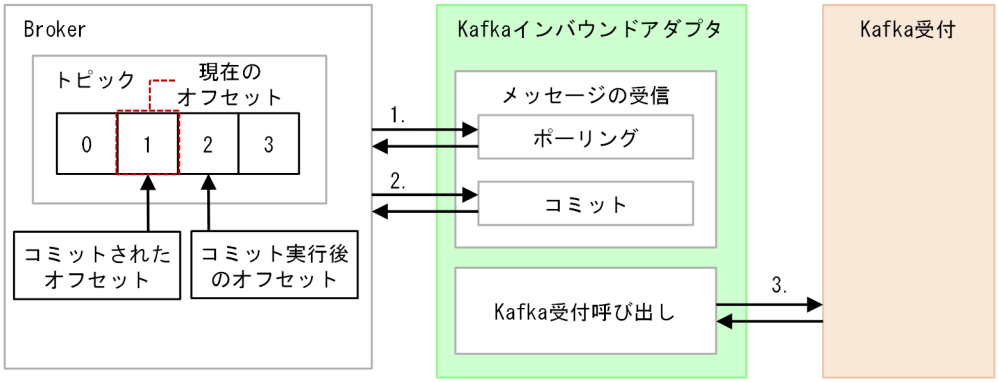
1. Consumer1 がポーリング処理を行います。
   現在のオフセットが「1」のため，オフセット「1」のメッセージを受信します。
2. ポーリングの最大遅延時間内に次のポーリング処理を呼べなかった場合，Consumer1 がグループから除外されます。
3. リバランス処理によって Consumer2 にパーティションが引き継がれます。
   コミットされたオフセットが「1」のため，Consumer2 ではオフセット「1」のメッセージを受信します。
4. Consumer1 がメッセージをコミットします。
   Consumer1 は割り当てられていないパーティションに対してコミットするため，コミットが失敗します。

(a)　メッセージ受信後のコミット

メッセージの処理順序を考慮する必要がない場合に設定します。この場合，ビジネスプロセスのメッセージ処理が失敗したときに，該当するメッセージ処理の回復処理を行わなくても，該当するメッセージよりあとのオフセットのメッセージが処理できます。

Kafka 受付からの応答が正常か，異常かに関わらず，「現在のオフセット」および「コミットされたオフセット」が更新されるため，次回のポーリングでは，前回受信したメッセージの次のオフセットのメッセージが受信されます。

図 2-103　メッセージ受信後のコミット処理の流れの例



図中の 1.～3.について説明します。

1. ポーリング処理でメッセージを受信します。

   受信するメッセージ数は 1 つです。現在のオフセットは「1」であるため，オフセット「1」のメッセージを受信します。メッセージ受信後に，現在のオフセットは「2」になります。

2. メッセージをコミットします。

   コミット実行後のオフセット（コミットされたオフセット）は「2」となります。

3. オフセット「1」のメッセージを Kafka 受付に送信します。

   Kafka 受付からの応答が正常か，異常かに関わらず，現在のオフセットは「2」のため，次回のポーリングではオフセット「2」のメッセージを受信します。

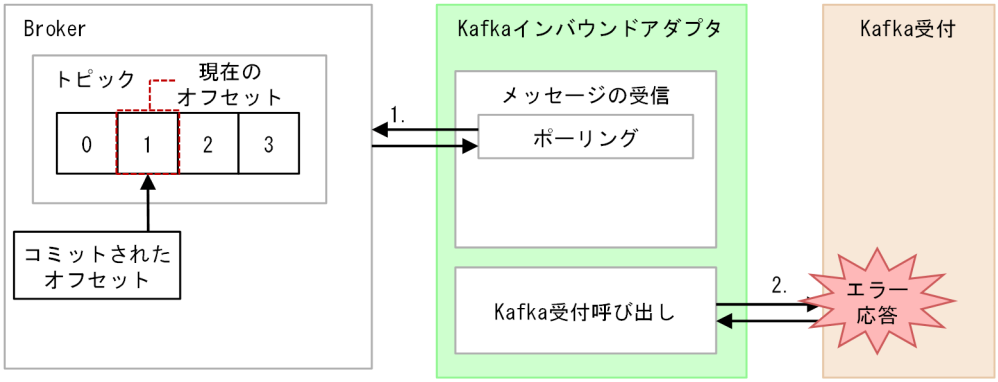
   また，コミット後にリバランスが発生して，該当するパーティションがグループ内の Consumer に引き継がれた場合は，パーティションを引き継いだ Consumer は 2.でコミットされたオフセットからメッセージを受信します。

(b)　メッセージ処理後のコミット

ビジネスプロセスのメッセージ処理が失敗した場合，該当するメッセージ処理の回復処理を行ってから，次のオフセットのメッセージを処理する必要があります。この場合，該当するメッセージは再送されます。

メッセージ処理後のコミットでは，Kafka 受付からの応答が正常な場合だけコミットを行います。メッセージ受信後の処理が失敗した場合や，Kafka 受付からエラー応答が返ってきた場合はコミットしません。また，次回のポーリングでは前回受信したメッセージを再度受信します。

図 2-104　メッセージ処理後コミットの処理の流れの例



図中の 1.～2.について説明します。

1. ポーリング処理を行い，メッセージを受信します。

   現在のオフセットは「1」のため，オフセット「1」のメッセージを受信します。メッセージ受信後に現在のオフセットは「2」となります。

2. オフセット「1」のメッセージを Kafka 受付に送信します。

   Kafka 受付からエラー応答があった場合，コミット処理が省略されるため，コミットされたオフセットは更新されません。

   また，次回のポーリングで同一のメッセージを受信するために，現在のオフセットをエラーが発生したメッセージのオフセットに戻します。

(c)　オフセットの自動リセット

オフセットが初期化されていない場合，または「現在のオフセット」がサーバ上にない場合は，オフセットの自動リセットを行います。自動リセットは，Kafka 受付定義ファイルの consumer.auto.offset.reset で設定します。

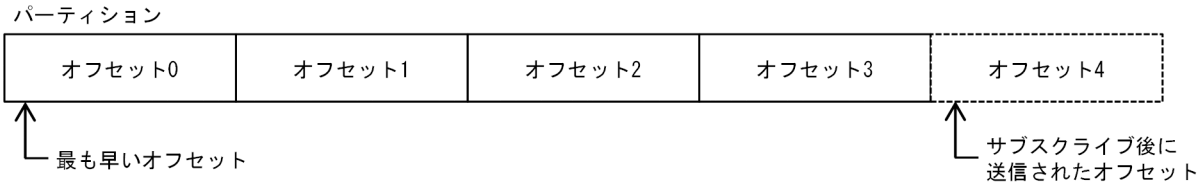
自動リセットの方法に応じて取得されるデータを次に示します。

表 2-36　自動リセットの方法に応じて取得されるデータ

プロパティ	設定値	説明
consumer.auto.offset.reset	earliest	最も早いオフセットに自動的にリセットします。パーティションの先頭からすべてのメッセージを取得します。
	latest	自動的に最新のオフセットにリセットします。サブスクライブのあとに送信されたメッセージを取得します。

自動リセットの設定に応じた自動リセットで取得できるメッセージの例を次に示します。

図 2-105　自動リセットで取得できるメッセージの例



earliest（最も早いオフセットにリセット）を設定した場合，オフセットが 0 にリセットされるため，オフセット 0 以降のメッセージを受信できます。

latest（最新のオフセットにリセット）を設定した場合，サブスクライブのあとに送信されたメッセージであるオフセット 4 以降のメッセージが受信できます。サブスクライブ前にパーティションに存在していたオフセット 0～3 のメッセージは受信しません。

(7)　レコードのデータサイズ

レコードのデータサイズは Kafka 受付定義ファイルで設定します。データは，レコードバッチ単位で取得します。

レコードのデータサイズを設定するプロパティを次に示します。

表 2-37　レコードのデータサイズに関するプロパティ

プロパティ	説明
consumer.fetch.min.bytes※	サーバがフェッチ要求に対して返す必要のあるデータの最小量を指定します。
consumer.fetch.max.bytes	サーバがフェッチ要求に対して返す必要のあるデータの最大量を指定します。
consumer.max.partition.fetch.bytes	サーバが返すパーティションごとのデータの最大量を指定します。

注※

利用できるデータに満たない場合，Broker はその量のデータが蓄積されるまで待ってから Consumer に応答します。データが蓄積されるまでの待ち時間は，consumer.fetch.max.wait.ms（フェッチ要求のブロック時間）で設定します。この待ち時間を経過した場合，最小サイズに満たなくても Broker から応答が返却されます。

利用できるデータを満たす場合

最小サイズを 20 バイトと設定し，次に示すパーティションの，オフセット 0 のメッセージから受信する場合の例を次に示します。データの取得単位は，1 レコード 1 バッチです。

図 2-106　利用できるデータを満たす場合の例

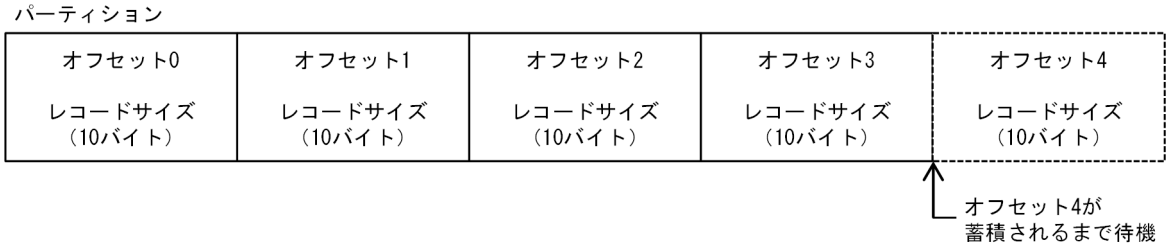


最小サイズを満たすデータがあるため，最大サイズまでのデータが受信できます。

利用できるデータに満たない場合

最小サイズを 50 バイトと設定し，次に示すパーティションの，オフセット 0 のメッセージから受信する場合の例を次に示します。データの取得単位は，1 レコード 1 バッチです。

図 2–107　利用できるデータに満たない場合の例



最小サイズに満たないため，データが蓄積されるまで待ちます。

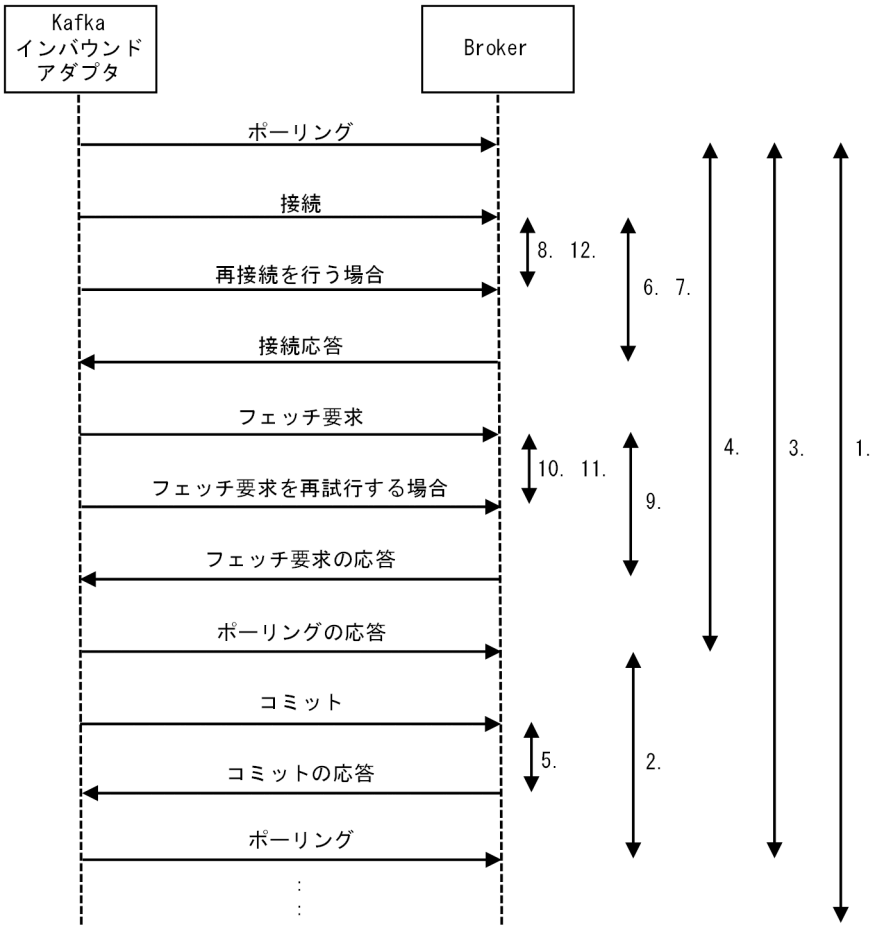
待ち時間内に最小サイズを満たすデータが蓄積されない場合，オフセット 0～3 のメッセージが受信されます。

待ち時間内に最小サイズを満たすデータが蓄積された場合，オフセット 4 のメッセージまで受信できます。

2.12.4　Kafka インバウンドアダプタのタイムアウト監視

Kafka インバウンドアダプタのタイムアウトを監視するポイントを次に示します。

図 2-108　タイムアウトを監視するポイント



（凡例）

1. アイドルタイムアウト※
2. ポーリングの最大遅延時間
3. メタデータの更新を強制するまでの期間※
4. ポーリングのブロック時間
5. コミットのタイムアウト
6. 最初のソケット接続が確立されるまでクライアントが待機する最大時間
7. 最初のソケット接続が確立されるまで待機する時間
8. 繰り返し接続に失敗した Broker に再接続するときに待機する最大時間
9. フェッチ要求のブロック時間
10. パーティションへの要求の待機時間
11. パーティションへの要求の最大待機時間
12. 特定のホストへの再接続を試行する前に待機する基本時間

注※

フェッチ要求の延長で，タイムアウトが経過したか監視しています。

タイムアウトは次に示す関係で設定することを推奨します。

```
1. > 3. > 4. > 6. > 7. > 8. > 9. > 10. > 11. > 12.
1. > 2. > 5.
```

また，次の関係で設定することを推奨します。

```
4. > (6. + 8. + 9. + 10. + 11. + 12.)
```

(1) Kafka のグループ管理機能のタイムアウト

Kafka のグループ管理機能は，Broker が Consumer グループを管理するための機能です。Kafka クラスタのサーバにあるグループコーディネータが，Consumer グループのメンバーが生存しているか把握できるように，Consumer から定期的にハートビートを送信しています。Consumer がグループコーディネータにハートビートを一定期間送信できなかった場合，グループコーディネータはハートビートが送信できなかった Consumer をグループから除外します。

グループ管理機能に関するタイムアウトについて，次の表に示します。

表 2-38 グループ管理機能に関するタイムアウト

タイムアウト項目	説明	該当する Kafka 受付定義ファイルのプロパティ
セッションタイムアウト	Consumer セッションのタイムアウトです。	consumer.session.timeout.ms
ハートビート間隔	ハートビートの送信間隔です。	consumer.heartbeat.interval.ms

タイムアウト発生後の動作

セッションタイムアウトの期間内に，ハートビートが Broker に送信されなかった場合は，ポーリング処理がエラーとなります。エラーメッセージをメッセージログに出力して，Consumer を停止します。

(2) ポーリング処理のタイムアウト

ポーリング処理に関するタイムアウトについて，次の表に示します。

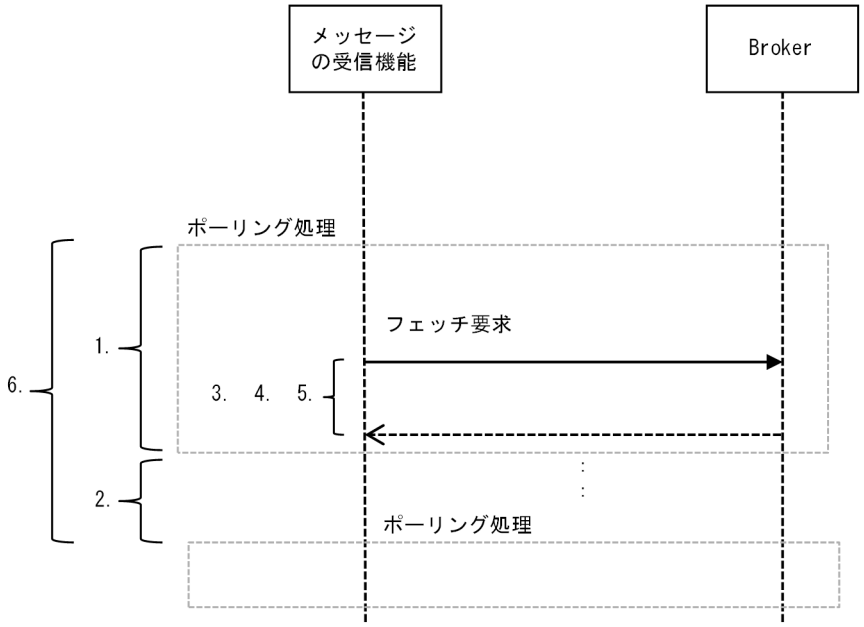
表 2-39 ポーリング処理のタイムアウト

タイムアウト項目	説明	該当する Kafka 受付定義ファイルのプロパティ	タイムアウト発生後の動作
ポーリングのブロック時間	利用できるレコードがない場合にポーリングをブロックする最大時間です。	urecp-kafka.poll-timeout.ms	空のレコードセットを返却し，ポーリング処理を続行します。

タイムアウト項目	説明	該当する Kafka 受付定義ファイルのプロパティ	タイムアウト発生後の動作
ポーリングのブロック時間	利用できるレコードがある場合は，ブロックしません。	urecp-kafka.poll-timeout.ms	空のレコードセットを返却し，ポーリング処理を続行します。
ポーリングの最大遅延時間	ポーリング実行後に，次のポーリングが行われるまでの最大遅延時間です。	consumer.max.poll.interval.ms	ポーリングの最大遅延時間のタイムアウト発生後の動作は，「2.12.3(6)　コミットのタイミング」を参照してください。
フェッチ要求のブロック時間	フェッチ要求でサーバが返却する最小データ量（fetch.min.bytes）を，すぐに満たすのに十分なデータがない場合に，サーバがフェッチ要求に応答する前にブロックする最大時間です。	consumer.fetch.max.wait.ms	フェッチ要求が終了し，ポーリング処理を続行します。
パーティションへの要求の待機時間	パーティションへの要求が失敗した場合に，失敗した要求を再試行する前に待機する時間です。	consumer.retry.backoff.ms	失敗した要求を再試行し，ポーリング処理を続行します。
パーティションへの要求の最大待機時間	パーティションへの要求が失敗した場合に，失敗した要求を再試行する前に待機する最大時間です。	consumer.retry.backoff.max.ms	失敗した要求を再試行し，ポーリング処理を続行します。
メタデータの更新を強制するまでの期間	メタデータを強制リフレッシュするための最大待ち時間です。Broker やパーティションの追加を行った場合，この時間経過後に Broker やパーティションが検出されます。	consumer.metadata.max.age.ms	メタデータを強制リフレッシュし，ポーリング処理を続行します。

ポーリング処理のタイムアウトの範囲を次の図に示します。

図 2-109　ポーリング処理のタイムアウトの範囲



（凡例）

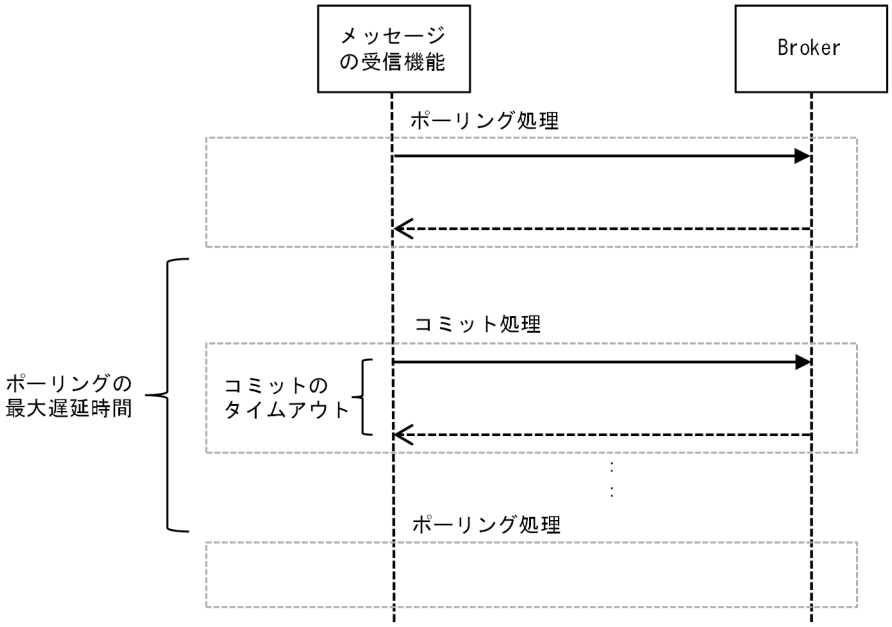
1. ポーリングのブロック時間
2. ポーリングの最大遅延時間
3. フェッチ要求のブロック時間
4. パーティションへの要求の待機時間
5. パーティションへの要求の最大待機時間
6. メタデータの更新を強制するまでの期間

(3) コミットのタイムアウト

コミットのタイムアウトは，オフセットコミットの完了を待機する最大時間です。Kafka 受付定義ファイルの urecp-kafka.commit-timeout.ms で設定します。

タイムアウトの範囲を次の図に示します。コミット時間を設定する場合，ポーリングの最大遅延時間を考慮する必要があります。

図 2-110　タイムアウトの範囲



コミット処理はポーリングの後に行われるため，ポーリングの最大遅延時間内に含まれます。そのため，ポーリングの最大遅延時間よりコミットのタイムアウトが長い場合は，ポーリングの最大遅延時間が経過したあとにコミットが行われるときがあります。このときは，ポーリングの最大遅延時間が経過したあとに Consumer がグループから除外されるため，コミットが失敗するおそれがあります。コミットのタイムアウトは，ポーリングの最大遅延時間を超えないように設定してください。

タイムアウト発生後の動作

オフセットコミットが正常に完了する前にタイムアウトが切れた場合は，Consumer を停止し，エラーメッセージをメッセージログに出力して，処理を終了します。

(4) アイドルタイムアウト

アイドルタイムアウトは，アイドル状態の接続を閉じる際に使用されるタイムアウトです。Kafka 受付定義ファイルの consumer.connections.max.idle.ms で設定します。

タイムアウト発生後の動作

アイドル状態の接続を閉じ，Consumer を続行します。

(5) Consumer インスタンスのクローズタイムアウト

Consumer インスタンスをクローズするときに使用されるタイムアウトです。タイムアウト値は，30 秒です。Consumer インスタンスのクローズタイムアウトのタイムアウト値は変更できません。

タイムアウト発生後の動作

エラーメッセージを出力して，処理を続行します。

(6) Broker 接続

Broker 接続のタイムアウトについて，次の表に示します。Broker 接続はポーリング時に行われるため，これらのタイムアウトが，ポーリングの最大遅延時間を超えないように設定する必要があります。

表 2-40　Broker 接続のタイムアウト

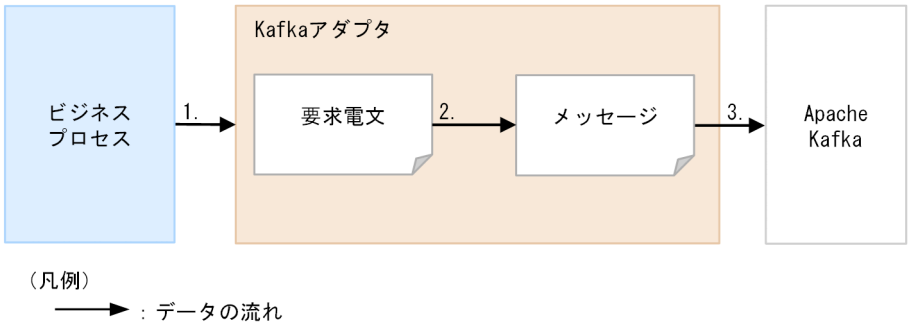
タイムアウト項目	説明	該当する Kafka 受付定義ファイルのプロパティ	タイムアウト発生後の動作
最初のソケット接続が確立されるまで待機する時間	最初のソケット接続が確立されるまで待機する時間です。	consumer.socket.connection.setup.timeout.ms	最大時間が経過していない場合，Broker に再接続して，Consumer を続行します。
最初のソケット接続が確立されるまでクライアントが待機する最大時間	最初のソケット接続が確立されるまでクライアントが待機する最大時間です。	consumer.socket.connection.setup.timeout.max.ms	最大時間が経過していない場合，Broker に再接続して，Consumer を続行します。
再接続を試行する前に待機する基本時間	特定のホストへの再接続を試行する前に待機する基本時間です。	consumer.reconnect.backoff.ms	最大時間が経過していない場合，Broker に再接続して，Consumer を続行します。
再接続するときに待機する最大時間	繰り返し接続に失敗した Broker に再接続するときに待機する最大時間です。	consumer.reconnect.backoff.max.ms	最大時間が経過していない場合，Broker に再接続して，Consumer を続行します。

2.12.5　Kafka アダプタの機能

Kafka アダプタは，ビジネスプロセスからのメッセージ送信要求に従い，Kafka にメッセージを送信するためのサービスアダプタです。Kafka アダプタが利用できる Apache Kafka のバージョンは 2.6.0～3.9.0 です。

Kafka アダプタを利用したメッセージの送信の例を次に示します。

図 2-111　Kafka アダプタを利用したメッセージの送信



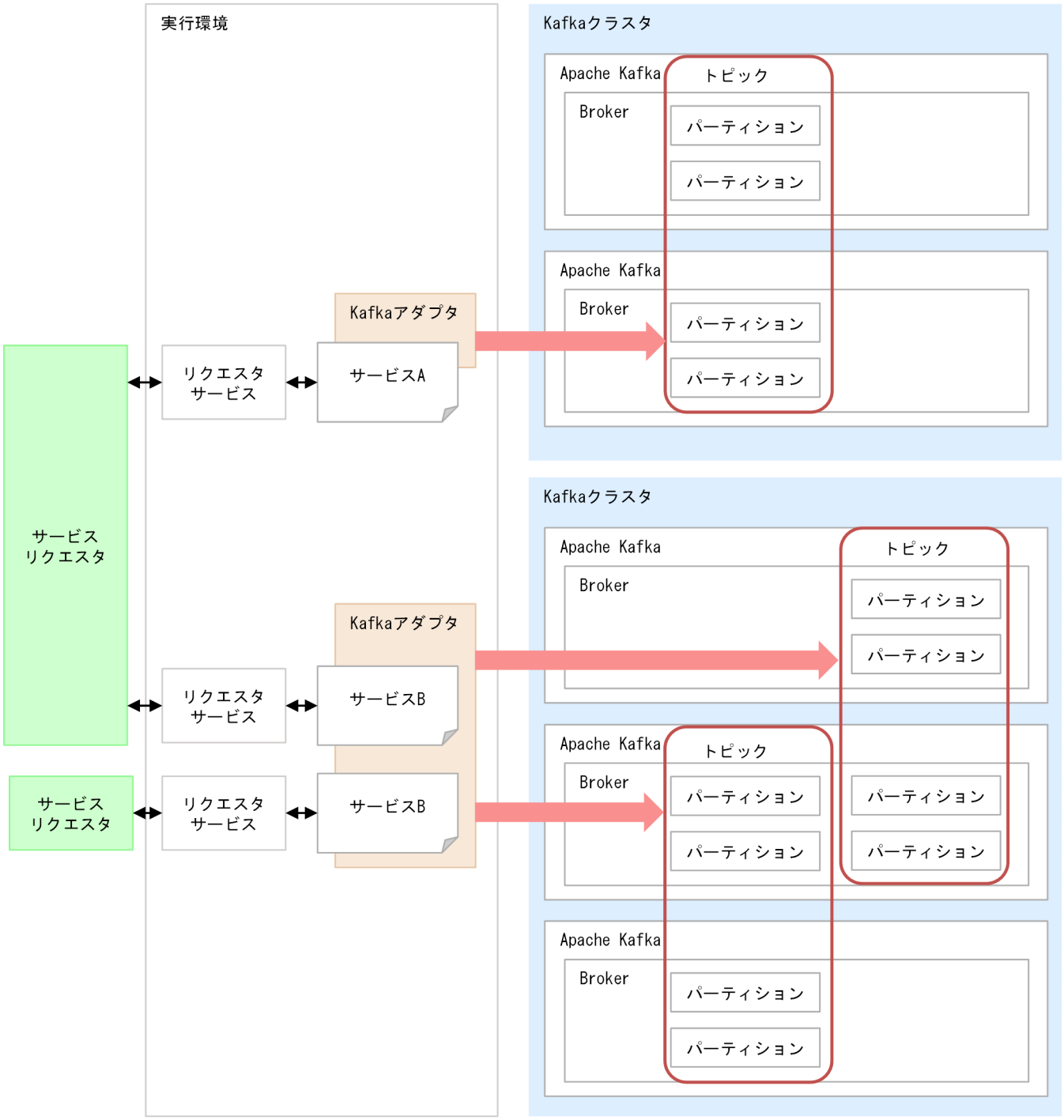
図中の 1.～3.について説明します。

1. ビジネスプロセスから要求電文が送信されます。
2. Kafka アダプタで要求電文をメッセージに変換します。
   メッセージの変換については，「(3)　メッセージの変換」を参照してください。
3. Kafka アダプタから Kafka にメッセージが送信されます。
   メッセージの送信については，「(4)　メッセージの送信」を参照してください。

(1)　Kafka アダプタと Apache Kafka との関係

Kafka アダプタと Apache Kafka との関係を次の図に示します。

図 2-112　Kafka アダプタと Apache Kafka との関係



Apache Kafka

複数台のサーバで大量のデータを処理する分散メッセージングシステムです。Kafka とも表記します。

Broker

Kafka の構成要素の一つで，データを受信，および配信するサービスです。

メッセージ

Kafka 内で扱うデータの単位です。レコードともいいます。メッセージはキーバリュー形式で構成されています。

トピック

メッセージを種別ごとに管理するためのストレージです。Broker 上に配置され，管理されます。Broker で設定された auto.create.topics.enable が true で，かつ送信先の Broker に指定したトピックがない場合，Apache Kafka は，自動でトピックを生成します。このときも Kafka アダプタでは，メッセージ送信処理を続行します。トピックの自動生成を無効にするには，Broker の auto.create.topics.enable を false に設定してください。Broker で設定するプロパティについては Kafka の公式ドキュメントを参照してください。

パーティション

Broker 上のデータの読み書きを行う単位です。トピックは複数のパーティションで構成されます。パーティションの切り分けは次の順に決定します。

1. メッセージの partition 要素に有効なパーティション ID が指定されている場合，そのパーティション ID を持つパーティションに送信されます。ヘッダ要求電文の<kafka-header-send-request>-<record>-<partition>で設定します。
2. メッセージの partition 要素にパーティション ID が指定されていない場合，メッセージのキーを指定することで，キーのハッシュ値に応じて送信先のパーティションに送信されます。同一のキーを持つメッセージは同一のパーティション ID を持つパーティションに送信されます。メッセージのキーはヘッダ要求電文の<kafka-header-send-request>-<record>-<key>で設定します。
3. メッセージのキーに何も指定されていない場合，デフォルトの設定に応じて複数のパーティションにメッセージが送信されます。

Kafka クラスタ

複数の Broker によって構成されたクラスタシステムです。

Kafka はトピックと呼ばれる単位で，種別ごとにメッセージを管理しています。Kafka アダプタは，ビジネスプロセスから送られる要求電文からメッセージを作成し，定義ファイルで設定した内容を基に指定した Apache Kafka 内にメッセージを格納します。メッセージのキーと値の型の構成が同一，かつ，同じ Kafka クラスタ内にメッセージを送信する場合，1 つの Kafka アダプタで Apache Kafka にメッセージを送信できます。また，1 つの Kafka アダプタで同じ Kafka クラスタ内の異なるトピックにメッセージを送信することもできます。

Kafka アダプタと Kafka は，ローカル環境でもリモート環境でも接続できます。

(2) Kafka アダプタで使用できる通信モデル

Kafka アダプタの通信モデルは，同期通信です。通信モデルに「非同期」を設定した場合は，Kafka アダプタの呼び出し時にエラーになります。

(3) メッセージの変換

ビジネスプロセスから渡された要求電文を Kafka アダプタでメッセージに変換します。Kafka アダプタの要求電文の形式については，マニュアル「サービスプラットフォーム 開発ガイド 受付・アダプタ定義編」の「3.3.16　Kafka アダプタを定義する」を参照してください。

Kafka では，メッセージのキーおよび値はシリアライズされています。Kafka で使用できる形式のメッセージを作成するため，Kafka アダプタではメッセージのシリアライズを行います。シリアライズ形式には，バイナリ形式と文字列形式が使用できます。メッセージのシリアライズ形式の指定は，Kafka アダプタ定義ファイルの次のプロパティで指定します。

表 2-41　メッセージの変換に関するプロパティ

プロパティ	説明
producer.key.serializer	メッセージのキーのシリアライザクラスを指定します。
producer.value.serializer	メッセージの値のシリアライザクラスを指定します。

(4)　メッセージの送信

Kafka アダプタで変換したメッセージを Kafka に送信します。Kafka アダプタでは，要求電文から生成されたメッセージを，1 つのメッセージごとに Kafka に送信します。ただし，マルチスレッドでの送信のため，メッセージの送信順序は保証されません。

Kafka は，Kafka アダプタから送られてくるメッセージを受信するとメッセージ受信完了通知（Ack）を Kafka アダプタに返します。Kafka アダプタは，Ack を受信すると，メッセージ送信の完了処理を行います。

メッセージの送信では，メッセージの配信保証レベル，再送，圧縮を Kafka アダプタ定義ファイルのプロパティで設定できます。

(a)　メッセージの配信保証レベル

メッセージ配信の保証レベルを設定できます。メッセージ配信の保証レベルは Kafka アダプタ定義ファイルの producer.acks で設定します。

Kafka アダプタで保証しているメッセージ配信の種類を次に示します。

表 2-42　Kafka アダプタのメッセージ配信の保証

保証レベルの種類	説明	再送	重複除去
At Most Once	一回はメッセージの配信を試みます。重複メッセージは発生しません。メッセージがロストする場合があります。	×	×
At Least Once	少なくとも一回はメッセージを配信します。重複メッセージが発生する場合があります。デフォルトでは At Least Once が設定されています。	○	×

（凡例）

○：保証する

×：保証しない

注意事項

Kafka にメッセージ送信リクエストを実行して完了通知を受け取る間にクラッシュした場合，Kafka アダプタはそのメッセージの送信完了前という状態になり，再度ビジネスプロセスから送信要求を実行した場合に同じメッセージを再送することがあります。そのため，メッセージ配信の保証の設定に関わらず，メッセージの重複を前提として設計してください。

(b) メッセージの再送

一時的なエラーで送信に失敗した場合，メッセージを再送できます。メッセージの再送は，Kafka アダプタ定義ファイルの次のプロパティで設定します。

表 2-43 メッセージの再送に関するプロパティ

プロパティ	説明
producer.acks	Kafka アダプタからの送信リクエストに，Kafka のメッセージ受信完了通知（Ack）を返すタイミングを指定します。
producer.retries	一時的なエラーで送信に失敗したメッセージを再送信する場合の再試行回数を指定します。
producer.delivery.timeout.ms	Kafka アダプタからの送信リクエストの呼び出しが戻ったあと，Kafka へのメッセージ送信の成功または失敗を応答する時間の上限を指定します。

ネットワークエラーが発生した場合，メッセージを再送しようとすると重複が発生するときがあります。メッセージが送信される前にネットワークエラーが発生した場合は，重複は発生しません。ただし，メッセージが Kafka に格納されたあと応答を送信者に配信する前にネットワークエラーが発生した場合は，Kafka アダプタはエラーを検出できません。

(c) メッセージの圧縮

送信するメッセージを指定した圧縮方式で圧縮できます。メッセージの圧縮は，Kafka アダプタ定義ファイルの producer.compression.type で設定します。

(5) Kafka アダプタのセキュリティ設定

Kafka アダプタでは，TLS を使用した暗号化と認証，ホスト名の検証をサポートしています。TLS で使用するセキュリティ通信用プロトコルについては，Kafka の公式ドキュメントを参照してください。

TLS を使用した暗号化と認証に関する定義を設定するには，Kafka アダプタ実行環境（共通）プロパティファイルに次のプロパティを設定します。

表 2-44 TLS を使用した暗号化と認証に関するプロパティ

プロパティ	説明
producer.security.protocol	Broker との通信に使用されるプロトコルの指定

プロパティ	説明
producer.ssl.key.password	キーストアファイル内の秘密鍵のパスワード
producer.ssl.keystore.key	キーストアファイルの秘密キー
producer.ssl.keystore.location	キーストアファイルの格納場所
producer.ssl.keystore.password	キーストアファイルのストアパスワード
producer.ssl.truststore.location	トラストストアファイルの格納場所
producer.ssl.truststore.password	トラストストアファイルのパスワード
producer.ssl.enabled.protocols	SSL 接続が有効になっているプロトコルのリスト
producer.ssl.keystore.type	キーストアファイルのファイル形式
producer.ssl.protocol	SSL コンテキストの生成に使用される SSL プロトコル
producer.ssl.provider	SSL 接続に使用されるセキュリティプロバイダの名前
producer.ssl.truststore.type	トラストストアファイルのファイル形式

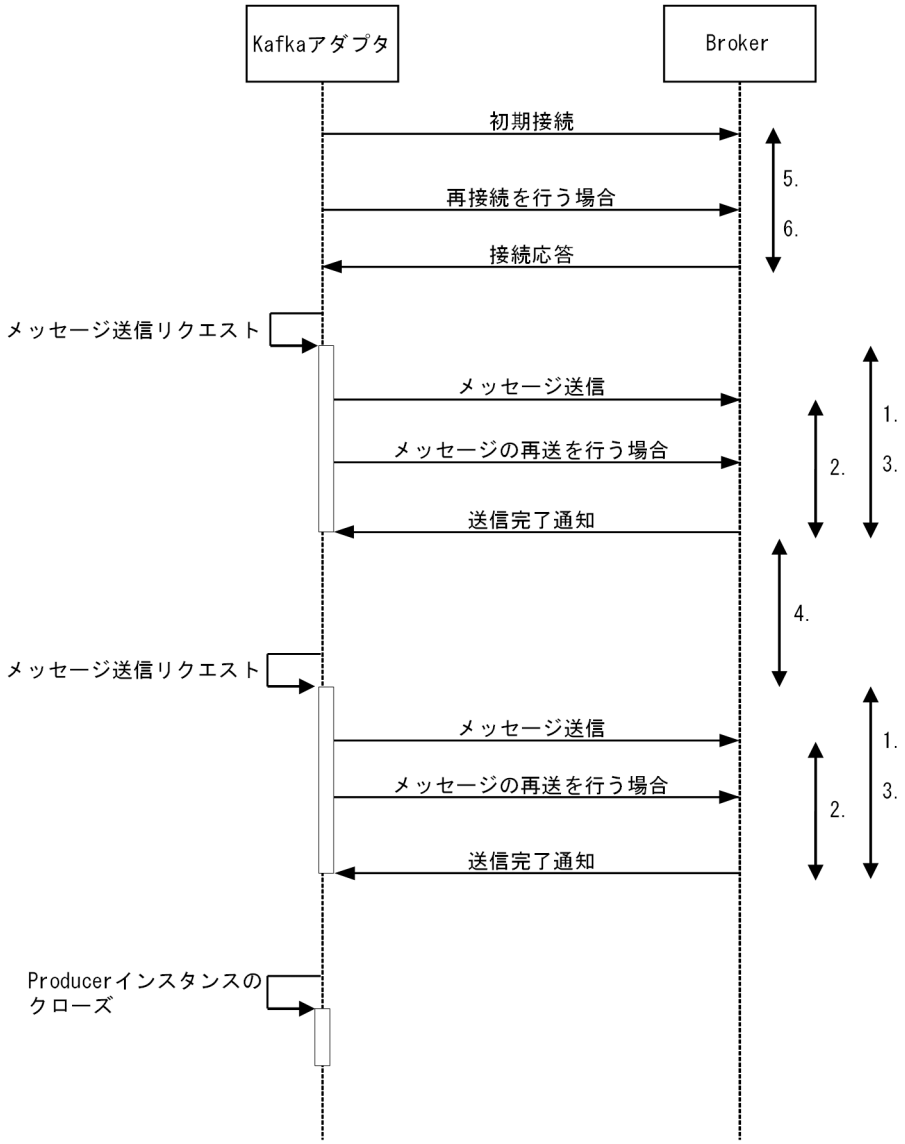
TLS で使用できる TLS 暗号スイートの制限はありません。ただし，使用する暗号スイートを設定することはできません。

Kafka の TLS を使用した暗号化と認証の設定については，Kafka の公式ドキュメントを参照してください。

2.12.6　Kafka アダプタのタイムアウト監視

Kafka アダプタのタイムアウトを監視するポイントを次に示します。

図 2-113　Kafka アダプタのタイムアウトを監視するポイント



（凡例）
1. ：送信リクエストの呼び出しが待機する時間のタイムアウト
2. ：リクエストタイムアウト
3. ：配信タイムアウト
4. ：アイドルタイムアウト
5. ：ソケット接続が確立されるのを待機する最大時間
6. ：最初のソケット接続が確立されるまで待機する時間

ここでは，各タイムアウトの設定方法と発生後の動作について説明します。

(1)　メッセージ送信時のタイムアウト

(a)　送信リクエストの呼び出しが待機する時間のタイムアウト

送信リクエストの呼び出しが待機する時間です。

設定方法

Kafka アダプタ定義ファイルの producer.max.block.ms で設定します。

タイムアウト発生後の動作

送信リクエストの呼び出しが待機したあと，開放される前にタイムアウトが切れた場合は，処理が続行できないと判断し，エラーメッセージを出力して，メッセージの送信を停止します。

(b)　リクエストタイムアウト

Kafka アダプタが送信要求の応答を待機する最大時間です。

設定方法

Kafka アダプタ定義ファイルの producer.request.timeout.ms で設定します。

タイムアウト発生後の動作

Kafka からの要求の応答が返ってくる前にタイムアウトが切れた場合の状態によって，次のように動作が変わります。

- メッセージが再送できる場合は，リクエストが再試行されます。
- メッセージが再送できない場合は，処理が続行できないと判断し，エラーメッセージを出力して，メッセージの送信を停止します。

メッセージの再送については，「2.12.5(4)(b)　メッセージの再送」を参照してください。

(c)　配信タイムアウト

送信リクエストの呼び出しが返されてから成功または失敗をレポートする時間です。

設定方法

Kafka アダプタ定義ファイルの producer.delivery.timeout.ms で設定します。

タイムアウト発生後の動作

送信リクエストの呼び出しが返されてから成功または失敗をレポートするまでにタイムアウトが切れた場合は，処理が続行できないと判断し，エラーメッセージを出力して，メッセージの送信を停止します。

(2)　アイドルタイムアウト

アイドル状態の接続を閉じるときに使用されるタイムアウトです。

設定方法

Kafka アダプタ定義ファイルの producer.connections.max.idle.ms で設定します。

タイムアウト発生後の動作

アイドル状態の接続が閉じます。メッセージの送信は続行します。

(3) Broker 接続時のタイムアウト

(a) ソケット接続が確立されるのを待機する最大時間

ソケット接続が確立されるまでクライアントが待機する最大時間です。

設定方法

Kafka アダプタ定義ファイルの producer.socket.connection.setup.timeout.max.ms で設定します。

タイムアウト発生後の動作

ノードが停止していると判断し，他のノードに接続を切り替えます。メッセージの送信は続行します。

(b) ソケット接続が確立されるまで待機する時間

最初のソケット接続が確立されるまで待機する時間です。

設定方法

Kafka アダプタ定義ファイルの producer.socket.connection.setup.timeout.ms で設定します。

タイムアウト発生後の動作

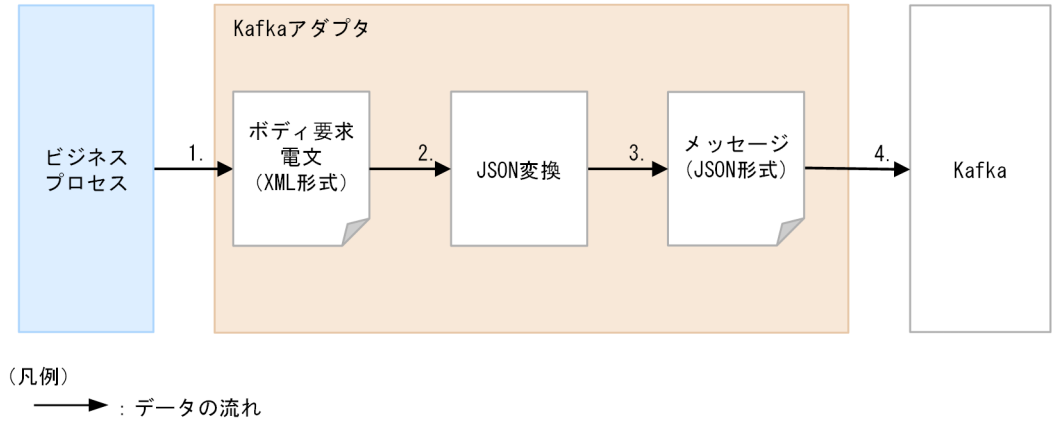
ノードが停止していると判断し，他のノードに接続を切り替えます。メッセージの送信は続行します。

2.12.7 Kafka アダプタの JSON-XML 変換機能

Kafka アダプタのボディ要求電文に設定された XML 形式のデータを JSON 形式に変換します。JSON-XML 変換した JSON 形式データは，Kafka に送信するメッセージの値に格納されます。JSON 形式で使用できる文字コードは UTF-8 です。

Kafka アダプタで JSON 形式データを送信する場合のデータの流れを次に示します。

図 2-114 Kafka アダプタで JSON 形式データを送信する場合のデータの流れ



図中の 1.～4.について説明します。

1. ビジネスプロセスから Kafka アダプタにボディ要求電文（XML 形式）が渡されます。
2. Kafka アダプタの JSON-XML 変換機能を使用して，XML 形式のボディ要求電文を JSON 形式に変換します。
3. Kafka アダプタが生成したメッセージの値に JSON 形式のデータを設定します。
4. Kafka アダプタがメッセージを送信します。

Kafka アダプタで JSON-XML 変換をするどうかは，Kafka アダプタ定義ファイルで設定します。Kafka アダプタ定義ファイルの設定については，マニュアル「サービスプラットフォーム リファレンス」の「3.13.1　Kafka アダプタ定義ファイル」を参照してください。

JSON-XML 変換機能では，変換処理の動作定義を設定できます。Kafka アダプタ定義ファイルに JSON-XML 変換定義ファイルが指定されていた場合，JSON-XML 変換定義ファイルの設定が JSON-XML 変換機能に反映されます。なお，JSON-XML 変換定義ファイルが指定されていない場合は，デフォルト値が設定されます。JSON-XML 変換定義ファイルは，Kafka アダプタの起動時に読み込まれます。

JSON-XML 変換定義ファイルの詳細については，マニュアル「サービスプラットフォーム リファレンス」の「3.18.1　JSON-XML 変換定義ファイル」を参照してください。

2.13　gRPC プロトコルを利用した接続

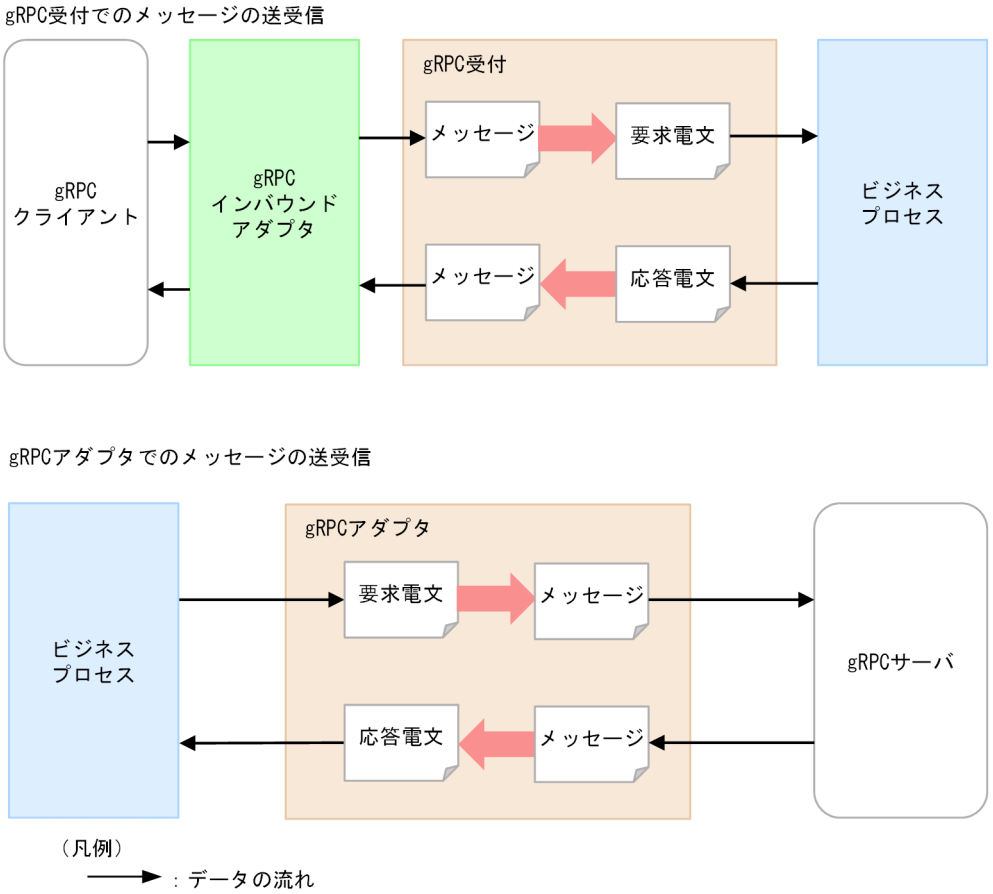
gRPC 受付，gRPC インバウンドアダプタ，gRPC アダプタを利用して，gRPC プロトコルでメッセージを送受信できます。

2.13.1　gRPC プロトコルを利用したシステム

サービスプラットフォームでは，gRPC プロトコルを利用してメッセージの送受信をする機能を提供しています。

gRPC プロトコルを利用したシステムの概要を次の図に示します。

図 2-115　gRPC プロトコルを利用したシステムの概要



gRPC 受付は，gRPC インバウンドアダプタから渡されたメッセージを要求電文に変換してビジネスプロセスに渡します。

gRPC アダプタは，ビジネスプロセスから受け取ったメッセージを gRPC サーバに送信します。

なお，サービスプラットフォームと gRPC プロトコルとの接続を利用する場合，次の制限事項があります。

- proto ファイルの完全修飾名（「パッケージ名.サービス名」）が重複するサービス名は，gRPC 受付，および gRPC インバウンドアダプタでは使用できません。
- gRPC が規定している通信方式のうち，サービスプラットフォームでは「Unary RPC」だけを使用できます。
- サービスプラットフォームでサポートしている Protocol Buffers のバージョンは，Protocol Buffers version 3 です。Protocol Buffers version 2 仕様の GROUP タイプのメッセージは使用できません。

gRPC との接続に必要な設定について，次に説明します。

(1) gRPC プロトコルを利用した接続に必要な OSS

gRPC プロトコルを利用した接続では，次の OSS を使用します。

表 2-45 gRPC プロトコルを利用した接続に必要な OSS

項番	OSS 名	概要
1	animal-sniffer-annotations-1.24.jar	Animal Sniffer Annotations
2	annotations-4.1.1.4.jar	Google Android Annotations Library
3	checker-qual-3.42.0.jar	Checker Qual
4	commons-codec-1.11.jar※1	The Apache Commons Codec package contains simple encoder and decoders for various formats such as Base64 and Hexadecimal. In addition to these widely used encoders and decoders, the codec package also maintains a collection of phonetic encoding utilities.
5	commons-logging-1.2.jar※1	Apache Commons Logging is a thin adapter allowing configurable bridging to other, well known logging systems.
6	error_prone_annotations-2.28.0.jar	Error Prone Annotations
7	failureaccess-1.0.2.jar	Guava InternalFutureFailureAccess and InternalFutures
8	google-auth-library-credentials-1.23.0.jar※1	Google Auth Library For Java Credentials
9	google-auth-library-oauth2-http-1.23.0.jar※1	Google Auth Library For Java OAuth2 HTTP
10	google-http-client-1.43.3.jar※1	Google HTTP Client Library for Java. Functionality that works on all supported Java platforms, including Java 7 (or higher) desktop (SE) and web (EE), Android, and Google App Engine.
11	google-http-client-gson-1.43.3.jar※1	GSON Extensions to The Google HTTP Client Library For Java.
12	grpc-api-1.68.2.jar	GRPC API
13	grpc-auth-1.68.2.jar※1	gRPC: Auth
14	grpc-census-1.68.2.jar※2	gRPC のトレース機能を補助するモジュール

項番	OSS 名	概要
15	grpc-context-1.68.2.jar	GRPC Context
16	grpc-core-1.68.2.jar	GRPC Core
17	grpc-netty-shaded-1.68.2.jar	GRPC Netty Shaded
18	grpc-protobuf-1.68.2.jar	GRPC Protobuf
19	grpc-protobuf-lite-1.68.2.jar	GRPC Protobuf lite
20	grpc-stub-1.68.2.jar	GRPC Stub
21	grpc-util-1.68.2.jar	GRPC Utilities
22	gson-2.11.0.jar	Gson
23	guava-33.2.1-android.jar	Guava: Google Core Libraries For Java
24	httpclient-4.5.14.jar※1	HTTPClient provides an efficient, up-to-date, and feature-rich package implementing the client side of the most recent HTTP standards and recommendations.
25	httpcore-4.4.16.jar※1	Apache HttpComponents Core (blocking I/O)
26	j2objc-annotations-3.0.0.jar	J2ObjC Annotations
27	jsr305-3.0.2.jar	FindBugs JSR305
28	listenablefuture-9999.0-empty-to-avoid-conflict-with-guava.jar	Guava ListenableFuture Only
29	opencensus-api-0.31.1.jar※1, ※2	トレース・メトリックス用ライブラリモジュール
30	opencensus-contrib-http-util-0.31.1.jar※1	HTTP utilities for Opencensus
31	opencensus-contrib-grpc-metrics-0.31.1.jar※2	Opencensus 用 gRPC メトリックスライブラリモジュール
32	perfmark-api-0.27.0.jar	PerfMark API
33	protobuf-java-3.25.5.jar	Protocol Buffers [Core]
34	proto-google-common-protos-2.41.0.jar	Proto Google Common Protos

注※1

gRPC アダプタで使用する場合は必要です。

注※2

gRPC 通信ログを出力する場合は必要です。

(2) gRPC ライブラリ（OSS）のインストール

gRPC との接続には，表 2-45 に記載された gRPC ライブラリ（OSS）の jar ファイルが必要です。

前提 jar ファイルをインストールする手順を次に示します。

1. Service Platform の添付品 CD から表 2-45 の jar ファイルを，次のフォルダにコピーします。
   <サービスプラットフォームのインストールディレクトリ>¥CSC¥lib¥external
2. J2EE サーバ用オプション定義ファイル（usrconf.cfg）を開き，次の gRPC ライブラリ（OSS）のクラスパスを指定します。

```
add.class.path=<gRPCライブラリ(OSS)のjarファイルのインストールディレクトリ>¥各gRPCライブラリ(OSS)のjarファイル名
```

(3) protoc コマンドの取得と proto ファイルの出力

gRPC との接続には，protoc コマンドが必要です。

proto ファイルは，protocol buffer IDL 仕様に従って記述されたファイルです。

(a) protoc コマンドの取得

サービスプラットフォームの開発環境（Service Architect）を使用する場合は，protoc コマンドを取得してください。

protoc コマンドを取得する手順を次に示します。

1. Service Platform の添付品 CD から protoc-21.8-win64.zip を取得します。
2. 取得した protoc-21.8-win64.zip を任意のディレクトリに展開します。
3. 展開したディレクトリから，bin ディレクトリ，および include ディレクトリをコピーし，次のディレクトリに格納します。
   <サービスプラットフォームのインストールディレクトリ>¥CSCTE¥lib¥external¥gRPC

protoc-21.8-win64.zip のディレクトリ構成を次に示します。

表 2-46 protoc-21.8-win64.zip のディレクトリ構成

ディレクトリとファイル	説明
ルートディレクトリ	protoc-21.8-win64.zip を展開した任意のディレクトリです。
bin	protoc コマンドを格納するディレクトリです。
protoc.exe	protoc コマンドです。gRPC 受付・アダプタ作成時に，使用します。
include	Well-Known-Types の proto ファイルを格納するディレクトリです。
google	Google が提供する Well-Known-Types（汎用メッセージタイプ）の proto ファイルおよびディレクトリです。
protobuf	
*.proto	
compiler	

ディレクトリとファイル						説明
					plugin.proto	Google が提供する Well-Known-Types（汎用メッセージタイプ）の proto ファイルおよびディレクトリです。

(b) proto ファイルの出力

gRPC 受付または gRPC アダプタの作成時に入力した proto ファイルは，ユーザ定義受付定義画面（詳細）またはサービスアダプタ定義画面（詳細）の proto ファイルの［出力］ボタンから取得できます。

proto ファイルは，ZIP ファイルに圧縮されて出力されます。ZIP ファイルのディレクトリ構成を次に示します。

表 2-47　出力した proto ファイルのディレクトリ構成

ディレクトリとファイル		説明
ルートディレクトリ		出力した ZIP ファイルのルートディレクトリです。
	proto_path1	proto_path ディレクトリおよび proto ファイルが格納されます。配下のサブディレクトリは，gRPC 受付または gRPC アダプタの作成時に指定した proto_path のディレクトリ構成で格納されます。
	proto_path2	
	:	
	proto_path*N*	

(c) proto ファイル使用時の注意事項

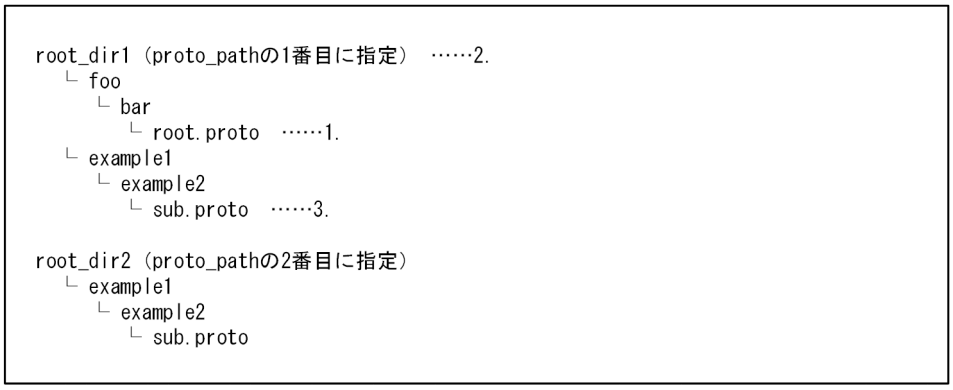
proto ファイル使用時の注意事項を次に示します。

- proto ファイル内に記述する import 文のパスは，proto ファイルを管理するプロジェクトの起点ディレクトリからの相対パスを記述してください。
- proto ファイル内に定義する package 名は，proto ファイルを管理するプロジェクトの起点ディレクトリからのファイルシステム上の相対パスと一致させることを推奨します。
- gRPC 受付または gRPC アダプタの作成時に入力する proto_path は，proto ファイルを管理するプロジェクトの起点ディレクトリを指定してください。
- proto_path や proto ファイルのパスに，260 バイト以上のパスは設定できません。
- proto ファイルのパスに，マルチバイト文字を含むパスは設定できません。
- Eclipse のワークスペースのパスに，マルチバイト文字を含むパスは設定できません。ワークスペースにマルチバイト文字を含むパスを設定し，gRPC 受付またはアダプタを作成した場合はエラーとなります。
- proto_path ディレクトリ以下には，gRPC 受付，gRPC アダプタの作成に使用する必要なファイルだけを格納してください。不要なファイルを格納した場合，proto ファイル解析処理中に次のことが発生するおそれがあります。
  - proto ファイル解析の処理時間が増加する。

- proto ファイル解析処理の途中でエラーが発生する。

- proto ファイルをインポートする場合，proto_path に指定した順番で proto ファイルが検索されます。
  例えば，次のディレクトリ構成のとき，proto_path に複数のディレクトリ（root_dir1，および root_dir2）を指定し，proto ファイル（foo/bar/root.proto）からほかの proto ファイル（example1/example2/sub.proto）をインポートした場合，foo/bar/root.proto がインポートする proto ファイルは，root_dir1 内にある example1/example2/sub.proto となります。

図 2-116　proto ファイルをインポートする場合の動作



1. foo/bar/root.protoからexample1/example2/sub.protoをインポートする。
2. proto_pathに指定した順番でprotoファイルが検索される。
3. proto_pathの1番目に指定されたroot_dir1内にあるexample1/example2/sub.protoがインポートされる。

(4)　HTTP/2 の設定

gRPC を利用したシステムでは，HTTP/2 を使用して gRPC 電文の送受信を行います。サービスプラットフォームで設定できる HTTP/2 に関するパラメタについては，マニュアル「サービスプラットフォームシステム構築・構築ガイド」の「3.1.2　実行環境に必要なソフトウェアを設定する」の gRPC インバウンドアダプタの設定に関する説明を参照してください。

HTTP/2 の詳細については，RFC の HTTP/2 の説明を参照してください。

(5)　保護区リストファイルの設定

gRPC プロトコルを使用する場合，保護区リストファイル（criticalList.cfg）を変更する必要があります。

保護区リストファイルに次の要素を追加してください。

```
org.codehaus.mojo.animal_sniffer.*
android.annotation.*
org.checkerframework.*
org.apache.commons.codec.*
org.apache.commons.logging.*
com.google.*
io.grpc.*
org.apache.http.*
com.fasterxml.jackson.core.*
io.opencensus.*
io.perfmark.*
```

保護区リストファイルについては，マニュアル「アプリケーションサーバ リファレンス 定義編(サーバ定義)」の「2.2.5　criticalList.cfg（保護区リストファイル）」を参照してください。

(6)　gRPC 通信時に出力されるログメッセージについての注意事項

gRPC インバウンドアダプタ，gRPC 受付，または gRPC アダプタ内で利用する OSS からログメッセージ（標準出力および標準エラー出力）を出力した場合，標準出力は J2EE サーバのユーザ出力ログ（usr_out[n].log），標準エラー出力はユーザエラーログ（usr_err[n].log）に出力されます。

標準出力，および標準エラー出力へのログメッセージの出力を抑制したい場合は，J2EE サーバ用ユーザプロパティファイル（usrconf.properties）の java.util.logging.config.file に設定したファイル，または java.util.logging 設定ファイル（%JAVA_HOME%/conf/logging.properties）に次のプロパティを追加します。

- io.grpc.level = OFF
- org.apache.commons.logging.level = OFF
- com.google.auth.level = OFF
- com.google.api.client.level = OFF
- com.google.common.level = OFF
- org.apache.http.level = OFF
- io.opencensus.level = OFF
- com.google.protobuf.level = OFF

これらのプロパティを追加すると，OSS のログメッセージは出力されません。

ただし，gRPC に関する機能だけでなく，設定したパッケージを使用するすべての機能でログが出力されなくなります。

2.13.2　gRPC 電文（Protocol Buffers 形式）と XML 形式の変換

gRPC 電文（Protocol Buffers 形式）と XML 形式データの変換について説明します。

gRPC 電文と XML 形式データの変換は，proto ファイルに定義した gRPC 電文のメッセージ構造を基に行います。

(1)　gRPC 電文（Protocol Buffers 形式）と XML 形式データの関係

gRPC 電文（Protocol Buffers 形式）と XML 形式データの関係を次に示します。

表 2-48　gRPC 電文（Protocol Buffers 形式）と XML 形式データの関係

Protocol Buffers 形式	XML 形式

package "mypackage";
message MyMessage {
子要素
…
}
	<mypackage.MyMessage>
	子要素
	…
	</mypackage.MyMessage>

gRPC 電文のメッセージ構造を定義した proto ファイルで定義した「パッケージ名」と「メッセージ名」を.（ピリオド）で結合し，XML の要素名とします。

gRPC 電文のメッセージ構造を定義した proto ファイルで定義した各フィールド値は，フィールド名を XML 要素名とします。XML の出力内容は，子要素のフィールドタイプによって異なります。

子要素のフィールドタイプごとの XML 出力内容を次に示します。

表 2-49　子要素のフィールドタイプごとの XML の出力内容

フィールドタイプ	XML の出力内容
double float	• 文字列表現（指数表記） • NaN • INF，-INF
int32 int64 uint32 uint64 sint32 sint64 fixed32 fixed64 sfixed32 sfixed64	文字列表現
bool	• true • false
string	UTF-8 文字列
bytes	バイナリデータを base64 変換した文字列
enum	列挙子 同一列挙値に複数の列挙子が割当てられている場合は，先に定義された列挙子を出力します。
map	• キー • 値

フィールドタイプ	XML の出力内容
message	内包するフィールドのフィールド値の規則に従います。

proto ファイルで repeated が指定されたフィールドは，同一要素名を持つ XML 要素が連続します。

(2) gRPC 電文と XML 形式データのマッピング例

相互変換を行う gRPC 電文と XML 形式データのマッピング例について説明します。

(a) int32 型のマッピング例

int32 型のマッピング例を次に示します。

●proto ファイルに定義した Protocol Buffers IDL

```
package mypackage;
message MyMessage {
  int32 intvar = 1;
}
```

●gRPC 電文

```
{
  intvar: 100
}
```

●XML 文書

```
<mypackage.MyMessage>
  <intvar>100</intvar>
</mypackage.MyMessage>
```

(b) bytes 型のマッピング例

bytes 型のマッピング例を次に示します。

●proto ファイルに定義した Protocol Buffers IDL

```
package mypackage;
message MyMessage {
  bytes bytesField = 1;
}
```

●gRPC 電文

```
{
  bytesField: [0x01,0x02,0x03,0x04]
}
```

●XML 文書

```
<mypackage.MyMessage>
  <bytesField>AQIDBA==</bytesField>
</mypackage.MyMessage>
```

(c) ネストしたメッセージのマッピング例

ネストしたメッセージのマッピング例を次に示します。

●proto ファイルに定義した Protocol Buffers IDL

```
package mypackage;
message MyMessage {
  mesage SubMessage {
    string stringField = 1;
  }
  SubMessage messageField = 1;
}
```

●gRPC 電文

```
{
  messageField: {stringField: "hello"}
}
```

●XML 文書

```
<mypackage.MyMessage>
  <messageField>
    <stringField>hello</stringField>
  </messageField>
</mypackage.MyMessage>
```

(d) repeated 型のフィールドのマッピング例

repeated 型のフィールドのマッピング例を次に示します。

●proto ファイルに定義した Protocol Buffers IDL

```
package mypackage;
message MyMessage {
  repeated string stringField = 1;
}
```

●gRPC 電文

```
{
  stringField: ["hello","goodbye"]
}
```

●XML 文書

```
<mypackage.MyMessage>
  <stringField>hello</stringField>
```

```
  <stringField>goodbye</stringField>
</mypackage.MyMessage>
```

(e) enum 型のフィールドのマッピング例

enum 型のフィールドのマッピング例を次に示します。

●proto ファイルに定義した Protocol Buffers IDL

```
package mypackage;
message MyMessage {
  enum EnumType {
    ZERO = 0;
    ONE = 1;
    ALT_ONE = 1;
  }
  EnumType enumField = 1;
}
```

●gRPC 電文

```
{
  enumField: ZERO
}
```

●XML 文書

```
<mypackage.MyMessage>
  <enumField>ZERO</enumField>
</mypackage.MyMessage>
```

(f) enum 型のフィールドのマッピング例（先に定義された列挙子を出力する場合）

enum 型のフィールドのマッピング例（先に定義された列挙子を出力する場合）を次に示します。この例では，同一列挙値（=1）の ONE および ALT_ONE の 2 つの列挙子を定義しているため，定義順が先の列挙子に変換されます。

●proto ファイルに定義した Protocol Buffers IDL

```
package mypackage;
message MyMessage {
  enum EnumType {
    ZERO = 0;
    ONE = 1;
    ALT_ONE = 1;
  }
  EnumType enumField = 1;
}
```

●gRPC 電文

```
{
  enumField: ALT_ONE
}
```

●XML 文書

```
<mypackage.MyMessage>
  <enumField>ONE</enumField>
</mypackage.MyMessage>
```

(g) map 型のフィールドのマッピング例

map 型のフィールドのマッピング例を次に示します。

●proto ファイルに定義した Protocol Buffers IDL

```
package mypackage;
message MyMessage {
  map<string, uint32> mapField = 1;
}
```

●gRPC 電文

```
{
  mapField: [{"key1",1234},{"key2",9999}]
}
```

●XML 文書

```
<mypackage.MyMessage>
  <mapField>
    <key>key1</key>
    <value>1234</value>
  </mapField>
  <mapField>
    <key>key2</key>
    <value>9999</value>
  </mapField>
</mypackage.MyMessage>
```

(3) proto-XML スキーマの出力ファイル形式

proto-XML スキーマの出力ファイル形式を次に示します。

XML スキーマは，タイプ定義部およびルート要素定義部によって構成されます。

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    targetNamespace="メッセージタイプ名"
    xmlns="メッセージタイプ名"
    elementFormDefault="qualified">
 タイプ定義部[0..n]
 ルート要素定義部[1..1]
</xs:schema>
```

(a) タイプ定義部

タイプ定義部は 0 個以上のメッセージ型定義，または列挙型定義によって構成されます。

■メッセージ型定義

```
<xs:complexType name="完全メッセージ型名">
 <xs:sequence>
 項目定義[1..n]
 </xs:sequence>
</xs:complexType>
```

「完全メッセージ型名」は，「パッケージ名.[親メッセージタイプ名称.]メッセージタイプ名称」の形式を持つ文字列です。「親メッセージタイプ名」の部分は，再帰的に適用されます。

■列挙型定義

```
<xs:simpleType name="完全列挙型名">
 <xs:restriction base="xs:string">
 列挙値定義[1..n]
 </xs:restriction>
</xs:simpleType>
```

「完全列挙型名」は，「パッケージ名.[親メッセージタイプ名称.]列挙型」の形式を持つ文字列です。「親メッセージタイプ名」の部分は再帰的に適用されます。

「列挙値定義」は，次の内容で生成されます。

```
<xs:enumeration value="列挙値"/>
```

(b) ルート要素定義部

ルート要素定義部は次の内容で生成されます。

```
<xs:element name="ルート要素名" type="ルート要素タイプ"/>
```

「ルート要素名」および「ルート要素タイプ」は，「パッケージ名.メッセージタイプ名称」の形式を持つ文字列です。

(c) 項目定義

項目定義は単純項目定義，および ONEOF 定義のどちらかで生成されます。

■単純項目定義

単純項目定義は次の内容で生成されます。

- repeated が指定されていない場合

  `<xs:element name="項目名" type="タイプ名" minOccurs="0"/>`

- repeated が指定されている場合

  `<xs:element name="項目名" type="タイプ名" minOccurs="0" maxOccurs="unbounded"/>`

「タイプ名」の一覧を次に示します。

表 2-50　単純項目定義のタイプ名一覧

proto ファイルに定義した項目のタイプ名	生成される XML スキーマの項目のタイプ名
string	xs:string
bytes	xs:base64Binary
int32	xs:int
sint32	
sfixed32	
int64	xs:long
sint64	
sfixed64	
uint32	xs:unsignedInt
fixed32	
uint64	xs:unsignedLong
fixed64	
bool	xs:boolean
float	xs:float
double	xs:double
型参照	メッセージ型名，または列挙型名

■ONEOF 定義

ONEOF 定義は，次の内容で生成されます。

```
<xs:element name="ONEOF名">
 <xs:complexType>
 <xs:choice>
 項目定義[1..n]
 </xs:choice>
 </xs:complexType>
</xs:element>
```

「項目定義」内の minOccurs 属性は出力されません。

(d)　XML スキーマ変換例

ここでは，proto ファイルから XML スキーマを生成した場合の変換例について説明します。

■例 1　すべてのメッセージタイプの一括変換

descriptor set ファイル内に定義されたすべての rpc が使用するメッセージタイプの XML スキーマを生成する場合の変換例を次に示します。

●Descriptor Set ファイルの作成に使用した proto ファイル

```
syntax = "proto3";

option java_multiple_files = true;
option java_package = "io.grpc.examples.sampledata";
option java_outer_classname = "SampleDataProto";
option objc_class_prefix = "HLW";

package sampledata;

// サービス定義
service UserInfoManager {
  rpc GetUserInfo (UserInfo) returns (UserData) {}
  rpc UpdateUserInfo (UserData) returns (UserData) {}
}

service ProductInfoManager {
  rpc UpdateProductInfo (ProductData) returns (ProductData) {}
}

// リクエストまたはレスポンスメッセージ
message UserInfo {
  string userid = 1;
}

message UserData {
  string name = 1;
  string age = 2;
}

message ProductData {
  string name = 1;
  string price = 2;
}
```

●生成された XML スキーマファイル

・UserInfo

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    targetNamespace="sampledata.UserInfo"
    xmlns="sampledata.UserInfo"
    elementFormDefault="qualified">

    // タイプ定義部
    <xs:complexType name="sampledata.UserInfo">
        <xs:sequence>
            <xs:element name="userid" type="xs:string" minOccurs="0" />
        </xs:sequence>
    </xs:complexType>

    // ルート要素定義部
    <xs:element name="sampledata.UserInfo" type="sampledata.UserInfo"/>
</xs:schema>
```

・UserData

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    targetNamespace="sampledata.UserData"
    xmlns="sampledata.UserData"
    elementFormDefault="qualified">

    // タイプ定義部
    <xs:complexType name="sampledata.UserData">
        <xs:sequence>
            <xs:element name="name" type="xs:string" minOccurs="0" />
            <xs:element name="age" type="xs:string" minOccurs="0" />
        </xs:sequence>
    </xs:complexType>

    // ルート要素定義部
    <xs:element name="sampledata.UserData" type="sampledata.UserData"/>
</xs:schema>
```

・ProductData

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    targetNamespace="sampledata.ProductData"
    xmlns="sampledata.ProductData"
    elementFormDefault="qualified">

    // タイプ定義部
    <xs:complexType name="sampledata.ProductData">
        <xs:sequence>
            <xs:element name="name" type="xs:string" minOccurs="0" />
            <xs:element name="price" type="xs:string" minOccurs="0" />
        </xs:sequence>
    </xs:complexType>

    // ルート要素定義部
    <xs:element name="sampledata.ProductData" type="sampledata.ProductData"/>
</xs:schema>
```

■例 2　特定のサービスが使用するメッセージタイプの変換

引数で指定したサービスに定義された，すべての rpc が使用するメッセージタイプの XML スキーマを生成する場合の変換例を次に示します。

●Descriptor Set ファイルの作成に使用した proto ファイル

例 2 の Descriptor Set ファイルの作成に使用した proto ファイルは，例 1 と同じです。

●生成された XML スキーマファイル

・UserInfo

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    targetNamespace="sampledata.UserInfo"
    xmlns="sampledata.UserInfo"
    elementFormDefault="qualified">

    // タイプ定義部
    <xs:complexType name="sampledata.UserInfo">
```

```
        <xs:sequence>
            <xs:element name="userid" type="xs:string" minOccurs="0" />
        </xs:sequence>
    </xs:complexType>

    // ルート要素定義部
    <xs:element name="sampledata.UserInfo" type="sampledata.UserInfo"/>
</xs:schema>
```

・UserData

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    targetNamespace="sampledata.UserData"
    xmlns="sampledata.UserData"
    elementFormDefault="qualified">

    // タイプ定義部
    <xs:complexType name="sampledata.UserData">
        <xs:sequence>
            <xs:element name="name" type="xs:string" minOccurs="0" />
            <xs:element name="age" type="xs:string" minOccurs="0" />
        </xs:sequence>
    </xs:complexType>

    // ルート要素定義部
    <xs:element name="sampledata.UserData" type="sampledata.UserData"/>
</xs:schema>
```

■例３　特定のサービスの中の rpc が使用するメッセージタイプの変換

引数で指定した rpc が使用するメッセージタイプの XML スキーマを生成する場合の変換例を次に示します。

●生成された XML スキーマファイル

・ProductData

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    targetNamespace="sampledata.ProductData"
    xmlns="sampledata.ProductData"
    elementFormDefault="qualified">

    // タイプ定義部
    <xs:complexType name="sampledata.ProductData">
        <xs:sequence>
            <xs:element name="name" type="xs:string" minOccurs="0" />
            <xs:element name="price" type="xs:string" minOccurs="0" />
        </xs:sequence>
    </xs:complexType>

    // ルート要素定義部
    <xs:element name="sampledata.ProductData" type="sampledata.ProductData"/>
</xs:schema>
```

■例 4　google/protobuf/struct.proto から作成した Descriptor Set ファイルを基に生成した場合の変換例

Google がサポートしている google/protobuf/struct.proto から作成した Descriptor Set ファイルを基に，XML スキーマを生成した場合の変換例を次に示します。

なお，メッセージタイプの変換例のため，サービスの定義は省略します。

●struct.proto

```
syntax = "proto3";

package google.protobuf;

option csharp_namespace = "Google.Protobuf.WellKnownTypes";
option cc_enable_arenas = true;
option go_package =
"google.golang.org/protobuf/types/known/structpb";
option java_package = "com.google.protobuf";
option java_outer_classname = "StructProto";
option java_multiple_files = true;
option objc_class_prefix = "GPB";

message Struct {
  // Unordered map of dynamically typed values.
  map<string, Value> fields = 1;
}

message Value {
  // The kind of value.
  oneof kind {
    // Represents a null value.
    NullValue null_value = 1;
    // Represents a double value.
    double number_value = 2;
    // Represents a string value.
    string string_value = 3;
    // Represents a boolean value.
    bool bool_value = 4;
    // Represents a structured value.
    Struct struct_value = 5;
    // Represents a repeated `Value`.
    ListValue list_value = 6;
  }
}

enum NullValue {
  // Null value.
  NULL_VALUE = 0;
}

message ListValue {
  // Repeated field of dynamically typed values.
  repeated Value values = 1;
}
```

●生成された XML スキーマファイル

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
    targetNamespace="google.protobuf.Struct"
    xmlns="google.protobuf.Struct"
    elementFormDefault="qualified">

    // タイプ定義部
    <xs:simpleType name="google.protobuf.NullValue">
        <xs:restriction base="xs:string">
            <xs:enumeration value="NULL_VALUE" />
        </xs:restriction>
    </xs:simpleType>
    <xs:complexType name="google.protobuf.ListValue">
        <xs:sequence>
            <xs:element name="values" type="google.protobuf.Value"
                minOccurs="0" maxOccurs="unbounded" />
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="google.protobuf.Value">
        <xs:sequence>
            <xs:element name="kind">
                <xs:complexType>
                    <xs:choice>
                        <xs:element name="null_value"
                            type="google.protobuf.NullValue" />
                        <xs:element name="number_value" type="xs:double" />
                        <xs:element name="string_value" type="xs:string" />
                        <xs:element name="bool_value" type="xs:boolean" />
                        <xs:element name="struct_value"
                            type="google.protobuf.Struct" />
                        <xs:element name="list_value"
                            type="google.protobuf.ListValue" />
                    </xs:choice>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType
        name="google.protobuf.Struct.FieldsEntry">
        <xs:sequence>
            <xs:element name="key" type="xs:string" minOccurs="0" />
            <xs:element name="value" type="google.protobuf.Value"
                minOccurs="0" />
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="google.protobuf.Struct">
        <xs:sequence>
            <xs:element name="fields"
                type="google.protobuf.Struct.FieldsEntry" minOccurs="0"
                maxOccurs="unbounded" />
        </xs:sequence>
    </xs:complexType>

    // ルート要素定義部
    <xs:element name="google.protobuf.Struct" type="google.protobuf.Struct"/>

</xs:schema>
```

タイプ定義部

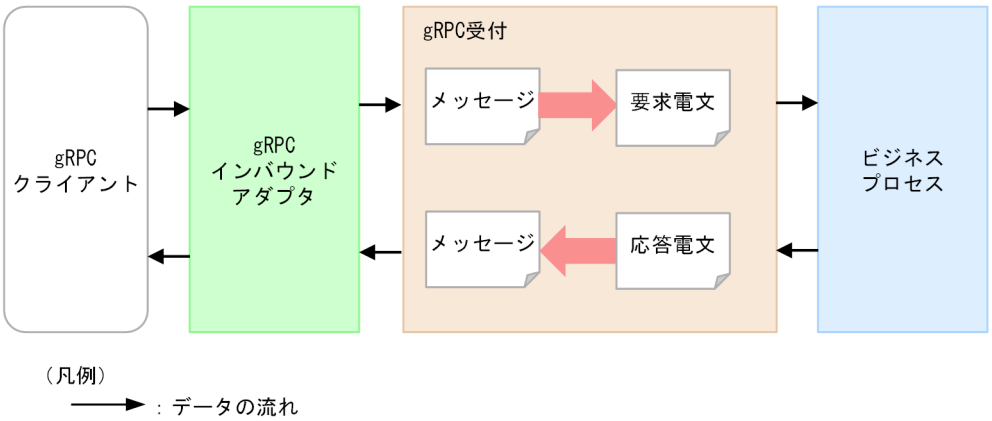
ルート要素定義部

2.13.3 gRPC 受付の機能

gRPC 受付では，gRPC インバウンドアダプタから渡されたメッセージから，要求電文を作成し，ビジネスプロセスに渡します。

また，ビジネスプロセスから渡された応答電文から返却するメッセージを作成し，インバウンドアダプタに返します。

図 2-117　gRPC 受付を利用した処理の流れ



なお，1 つの gRPC インバウンドアダプタから複数の gRPC 受付に接続できます。複数の gRPC 受付から 1 つのビジネスプロセスに接続することもできます。

gRPC 受付の通信モデルは，同期通信です。

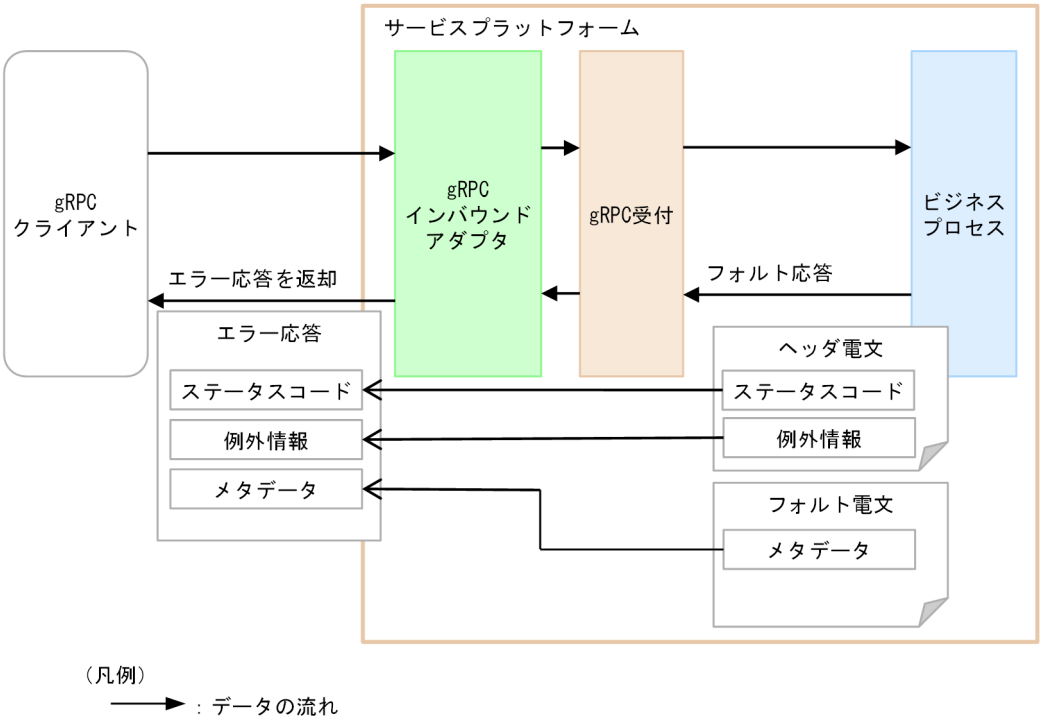
gRPC 受付では，ビジネスプロセスの処理時間（要求電文を渡して応答電文が返ってくるまでの処理時間）は監視しません。

gRPC 受付の定義の詳細については，マニュアル「サービスプラットフォーム 開発ガイド 受付・アダプタ定義編」の「2.10.2　gRPC 受付を追加する」を参照してください。

(1)　フォルト応答

gRPC クライアントに任意の gRPC のエラー応答（ステータスコードやエラー情報など）を返す場合，ビジネスプロセスからフォルトが発生したことを示す応答電文を送信します。

図 2-118　gRPC 受付のフォルト応答



gRPC 受付では，ビジネスプロセスから返却されたフォルト電文を gRPC のエラー情報（gRPC のオブジェクト）に変換し，gRPC インバウンドアダプタにデータを渡します。

また，gRPC クライアントに返すエラー情報には，gRPC のメタデータを含むことができます。メタデータを含む場合は，フォルト電文にメタデータの設定内容を定義します。メタデータの設定については，「(2) メタデータ」を参照してください。

フォルト電文の設定の詳細については，マニュアル「サービスプラットフォーム 開発ガイド 受付・アダプタ定義編」の「2.10.3　フォルト電文フォーマットを修正する」を参照してください。

(2)　メタデータ

gRPC クライアントから送信されたメタデータの内容を，ビジネスプロセスで参照できます。また，gRPC クライアントにメタデータを返却できます。

gRPC クライアントからメタデータが送信された場合，gRPC 受付はヘッダ電文にメタデータの内容を設定し，ビジネスプロセスにヘッダ電文を渡します。

gRPC クライアントにメタデータを返す場合，ビジネスプロセスからヘッダ電文を返します。

ビジネスプロセスから返却されたヘッダ電文が，次の条件をすべて満たす場合，gRPC 受付にヘッダ電文が返却されたとみなされます。

- ヘッダ電文の名前空間接頭辞が「grc」である。
- ヘッダ電文のルート要素が接頭辞「grc」で修飾されている。

また，上記に該当するヘッダ電文内の，接頭辞「grc」で修飾されていない要素は，要素が指定されていないとみなされます。

gRPC 受付のヘッダ電文はサービスプラットフォームが提供するスキーマを使用します。そのため，ヘッダ電文の作成は不要です。

送受信するメタデータの値は，テキスト形式またはバイナリ形式で設定できます。どちらのデータ形式かは，メタデータのキー名によって判断します。

- バイナリ形式：メタデータのキー名が-bin（大文字小文字を区別する）で終わる場合
- テキスト形式：メタデータのキー名が-bin 以外で終わる場合

データ形式はメタデータのキーごとに設定できます。

(3) 流量制御

gRPC クライアントとサービスプラットフォームを接続する場合，gRPC 受付で流量制御を設定できます。流量制御の設定については，gRPC 受付定義ファイルで設定します。

gRPC 受付定義ファイルの設定の詳細については，マニュアル「サービスプラットフォーム リファレンス」の「3.17.1 gRPC 受付定義ファイル」を参照してください。

流量制御で設定できる項目を次に示します。

表 2-51 流量制御の設定項目

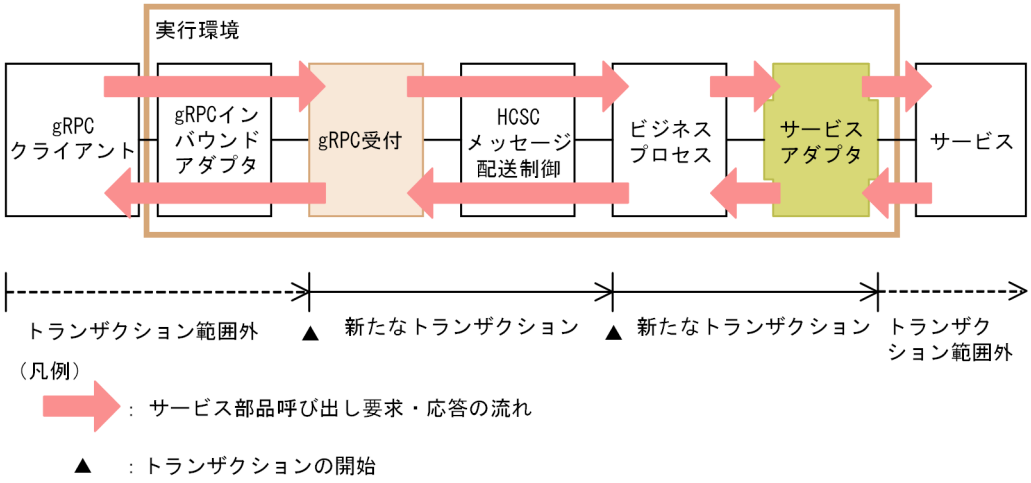
項目	オブジェクト	説明
プール内インスタンスの最小数	urecp-grpc.pooled-instance.minimum	カスタム受付フレームワークのプール内インスタンスの最小数を指定します。
プール内インスタンスの最大数	urecp-grpc.pooled-instance.maximum	受付処理およびカスタム受付フレームワークのプール内インスタンスの最大数を指定します。

(4) gRPC 受付使用時のトランザクション

gRPC 受付で開始される新たなトランザクションは，受付処理が操作します。

ビジネスプロセスを呼び出すときの gRPC 受付のトランザクションの範囲を次に示します。

図 2-119　gRPC 受付のトランザクションの範囲



2.13.4　gRPC インバウンドアダプタの機能

gRPC インバウンドアダプタは，RPC プロトコル「gRPC」を利用する gRPC クライアントからの要求電文を受信し，gRPC 受付から返される応答電文を gRPC クライアントに返信します。JCA1.5 に準拠しています。

(1)　gRPC インバウンドアダプタが使用できる OSS の機能

gRPC インバウンドアダプタでは，次の OSS の機能を使用できます。

- メッセージ圧縮機能（gRPC の要求電文，応答電文の双方でメッセージ圧縮機能をサポート）
- メタデータの送受信（gRPC の要求電文の受信，応答電文の送信，双方でメタデータをサポート）
- SSL/TLS
- フローコントロール
- キープアライブ
- HTTP/2 Rapid Reset Attack 対策

OSS の機能の一部は gRPC インバウンドアダプタの RAR 属性ファイルで設定できます。設定方法については，マニュアル「サービスプラットフォーム システム構築・運用ガイド」の「3.1.2　実行環境に必要なソフトウェアを設定する」を参照し，gRPC インバウンドアダプタの RAR 属性ファイルの説明を確認してください。

(2)　サポートする gRPC 通信種別

gRPC インバウンドアダプタでは，gRPC 通信種別の「Simple-RPC(Unary)」をサポートしています。「Simple-RPC(Unary)」は，gRPC クライアントからの 1 つのリクエストに対して，gRPC サーバが 1 つのレスポンスを返す通信種別です。

(3)　gRPC インバウンドアダプタの機能一覧

gRPC インバウンドアダプタの機能を次に示します。

機能	説明
運用機能	gRPC インバウンドアダプタの開始・停止など
gRPC サーバ機能	gRPC サーバの開始・停止など※
ログ出力機能	gRPC インバウンドアダプタに関するログの出力※

注※

設定方法については，マニュアル「サービスプラットフォーム システム構築・運用ガイド」の「3.1.2　実行環境に必要なソフトウェアを設定する」を参照し，gRPC インバウンドアダプタの RAR 属性ファイルの説明を確認してください。

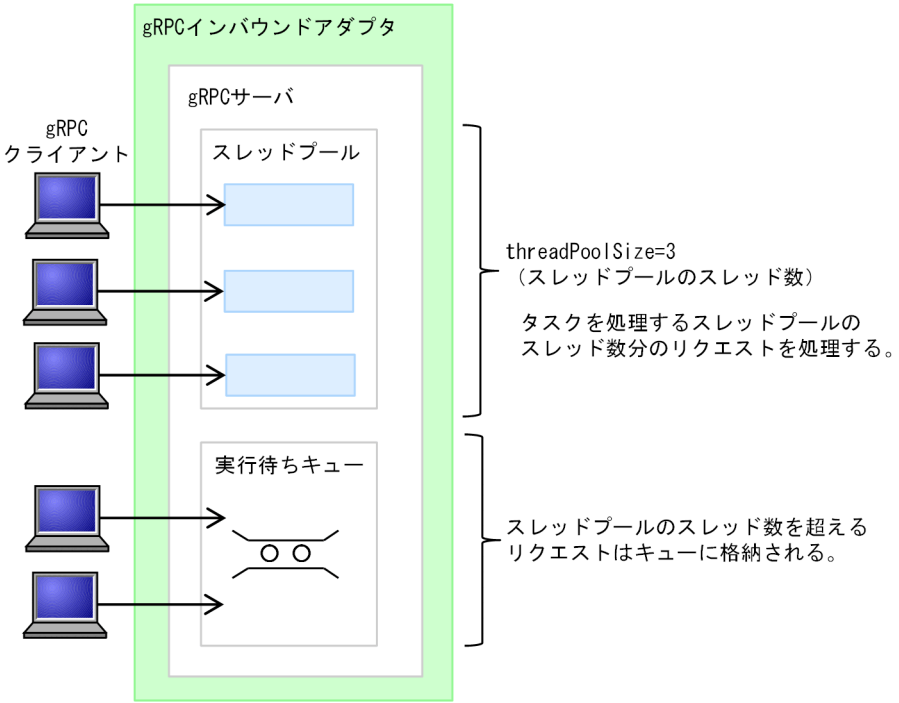
(4)　gRPC インバウンドアダプタの流量制御

gRPC インバウンドアダプタでは，gRPC クライアントからのリクエストを受信した場合，スレッドプールからスレッドを割り当ててリクエスト処理を実行します。スレッドプールのスレッド数を指定し，同時に実行できるスレッドを制御することで，パフォーマンスの低下やリソースの枯渇を抑制できます。

RAR 属性ファイルの threadPoolSize（リクエストを処理するスレッドプールのスレッド数）には，gRPC クライアントの同時実行数をスレッド数として設定します。

threadPoolSize については，マニュアル「サービスプラットフォーム システム構築・運用ガイド」の「3.1.2　実行環境に必要なソフトウェアを設定する」の threadPoolSize について説明している個所を参照してください。

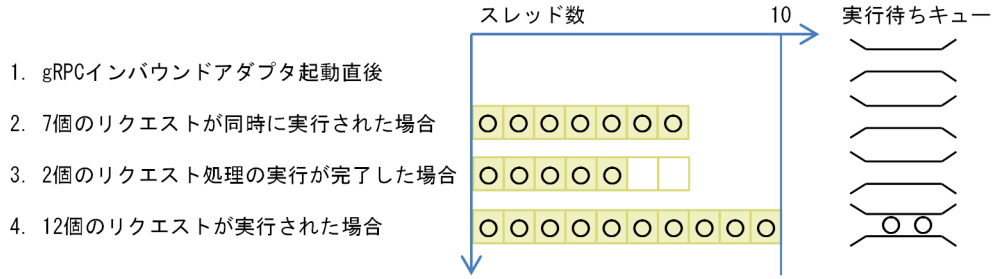
図 2-120　gRPC インバウンドアダプタの流量制御



gRPC インバウンドアダプタでは，スレッドプールを使用し，一度作成したスレッドを再利用して処理を実行します。スレッドプール内のスレッド数は，属性ファイルから設定できます。

タスクを処理するスレッドプールのスレッド数に 10 を設定した場合の遷移の例を次の図に示します。

図 2-121　gRPC インバウンドアダプタのスレッド数の遷移の例



この例では，属性ファイルに次のように設定しています。

```
threadPoolSize=10
```

1. gRPC インバウンドアダプタ起動直後は，スレッドは生成されません。
2. 7 つのリクエストが同時に実行された場合，7 つのスレッドが作成され，実行中スレッドとなります。
3. リクエスト処理の実行が完了した場合，実行が完了したスレッドはアイドルスレッドとなります。リクエスト処理が完了してもスレッドは破棄されません。
4. スレッドプールのスレッド数（threadPoolSize）を超えるリクエストが同時に実行された場合，threadPoolSize に指定した値を超えた分のリクエスト処理要求は，実行待ちキューにタスクとして追加され，処理を待機します。実行中のスレッドの処理が終了し，スレッドに空きが発生するとキューからタスクが取り出されます。

(5)　gRPC インバウンドアダプタ使用時の注意事項

gRPC インバウンドアダプタ使用時の注意事項を次に示します。

- gRPC インバウンドアダプタがサポートする gRPC 電文フォーマットの仕様は proto3 以上です。proto2 以前の電文フォーマットは使用できません。
- 同一完全修飾サービス名を持つ複数の gRPC サービスは登録できません。同一完全修飾サービス名を持つ gRPC サービスは，先に登録されたサービスだけが有効となり，それ以外は無効となります。

2.13.5　gRPC アダプタの機能

gRPC アダプタは，gRPC で作成されたサービスにメッセージを送信するためのサービスアダプタです。gRPC アダプタがサポートする gRPC 電文フォーマットは，proto3 以上です。proto2 以前の電文フォーマットはサポートしません。

また，gRPC アダプタを使用するには，OSS のインストールが必要です。必要な OSS については，「2.13.1(1)　gRPC プロトコルを利用した接続に必要な OSS」を参照してください。

gRPC アダプタには，次の機能があります。

- サーバ呼び出し機能

  gRPC サーバと通信を行い，gRPC サービスにメッセージを送受信します。リクエスト処理の延長で gRPC サーバと接続するため，gRPC アダプタの起動時に接続先の gRPC サーバを起動しておく必要がありません。gRPC アダプタが呼び出すメソッドは，gRPC アダプタの開発時に指定した proto ファイルに定義されているサービス定義のメソッドです。

- proto-XML 変換機能

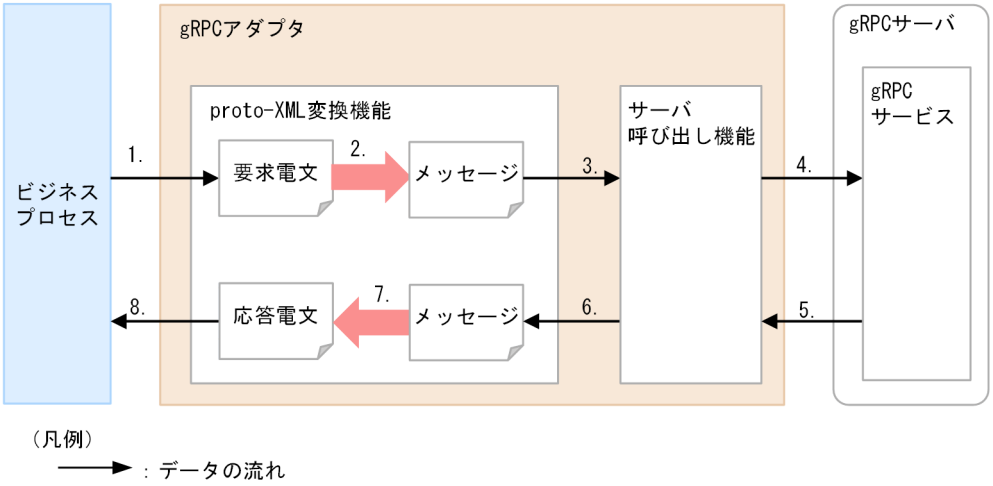
  ビジネスプロセスから渡された要求電文を gRPC サービスに送信し，メッセージ（プロトコルバッファ）に変換します。メッセージは，メソッド定義の引数に沿って作成されます。

  また，gRPC サービスから返されたメッセージ（プロトコルバッファ）を応答電文に変換します。応答電文は，開発環境で設定された応答電文スキーマに沿って作成されます。

gRPC アダプタと gRPC サーバ間では，メッセージはプロトコルバッファと呼ばれるシリアライズ方式を用いて変換されます。

gRPC アダプタのメッセージの処理の流れを次に示します。

図 2-122　gRPC アダプタのメッセージの処理の流れ



1. ビジネスプロセスから gRPC アダプタに要求電文が渡されます。
2. 要求電文を gRPC サーバに送信するメッセージに変換します。

   要求電文が電文フォーマット（要求電文ボディ）に沿っていない場合，メッセージの作成でエラーとなり，リクエストが失敗します。
3. 変換したメッセージをサーバ呼び出し機能に渡します。
4. gRPC サービスにメッセージを送信します。
5. gRPC サービスから返却されたメッセージを受け取ります。

6. proto-XML 変換機能にメッセージを渡します。
7. メッセージを応答電文に変換します。

   応答メッセージがメソッド定義の戻り値に沿っていない場合，応答メッセージの受信でエラーとなり，リクエストが失敗します。
8. 応答電文をビジネスプロセスに渡します。

gRPC アダプタのオペレーション（サービスアダプタ定義画面の「オペレーション」）は，gRPC アダプタの開発時に指定したサービス定義のメソッド（proto ファイルのサービス定義）で設定します。

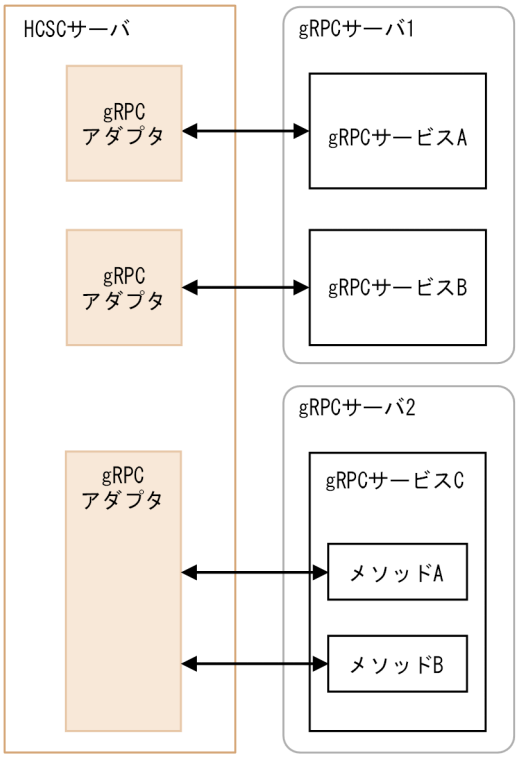
例えば，サービス定義内にメソッド「SayHello」「SayGoodbye」が定義されている場合，gRPC アダプタのオペレーションに「SayHello」「SayGoodbye」が定義されます。サービスアダプタ定義画面でオペレーション「SayHello」を呼び出した場合は，gRPC サービスのメソッド「SayHello」が呼び出されます。

gRPC アダプタの定義の詳細については，マニュアル「サービスプラットフォーム 開発ガイド 受付・アダプタ定義編」の「3.2.17　gRPC アダプタを新規に追加する」を参照してください。

(1)　gRPC アダプタの構成

gRPC サーバは，複数の gRPC サービスで構成されている場合があります。gRPC アダプタがメッセージを送信する gRPC サービスは，gRPC アダプタの開発時に決定するため，リクエスト時に変更することはできません。

図 2-123　gRPC アダプタの構成



- gRPC アダプタと gRPC サーバの数量関係は 1:N です。
- gRPC アダプタと gRPC サービスの数量関係は 1:1 です。

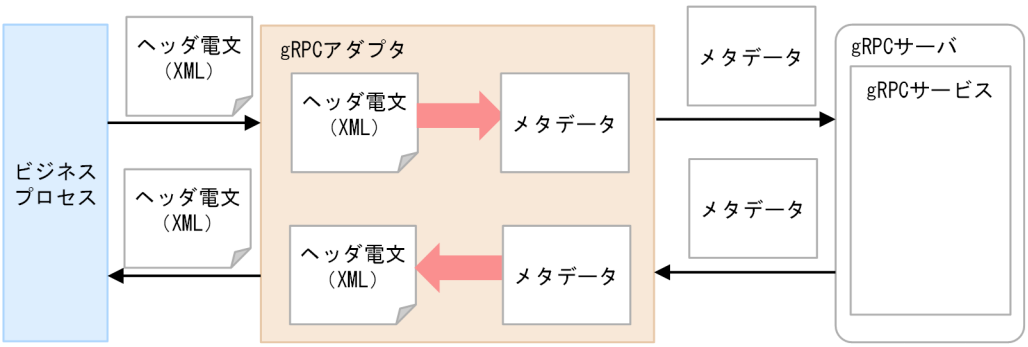
- gRPC アダプタとメソッドの数量関係は 1:N です。ただし，1 回のリクエストで呼び出せるメソッドは 1 つだけです。
- メソッドは同期で呼び出します。非同期呼び出しは設定できません。

(2) メタデータ

gRPC アダプタは，メッセージ（プロトコルバッファ）に加え，メタデータ（キーと値のペアの任意のセット）を送受信できます。

メタデータ送受信の処理の流れを次に示します。

図 2-124 メタデータ送受信の処理の流れ



gRPC サービスへ送信するメタデータ

ビジネスプロセスから渡されたヘッダ電文の設定内容からメタデータを作成し，gRPC サービスにメタデータを送信します。

データ形式がバイナリ形式のメタデータを送信する場合，ヘッダ電文内の該当するメタデータの値には Base64 でエンコードした値を設定します。

gRPC アダプタ内で，デコードした値をメタデータの値に設定します。

gRPC サービスから受信したメタデータ

gRPC サービスから，ステータスコード「OK」が返された場合は，gRPC サービスから受信したメタデータの設定内容からヘッダ電文を作成し，ビジネスプロセスに渡します。ヘッダ電文に設定するメタデータは，gRPC サービスが設定したメタデータ，および gRPC 内で設定されたメタデータとします。

また，フォルト応答で，gRPC サービスから返却された例外にメタデータが設定されている場合，例外に設定されているメタデータをフォルト電文に設定します。

メタデータのデータ形式がバイナリ形式の場合，gRPC アダプタは該当するメタデータの値を Base64 でエンコードした値をヘッダ電文に設定します。

送受信するメタデータの値は，テキスト形式またはバイナリ形式で設定できます。どちらのデータ形式かは，メタデータのキー名によって判断します。

- バイナリ形式：メタデータのキー名が-bin（大文字小文字を区別する）で終わる場合
- テキスト形式：メタデータのキー名が-bin 以外で終わる場合

データ形式はメタデータのキーごとに設定できます。

メタデータの設定内容は，ヘッダ電文で扱われます。ただし，gRPC サービスから返された例外に設定されているメタデータは含まれません。

ヘッダ電文の設定については，マニュアル「サービスプラットフォーム 開発ガイド 受付・アダプタ定義編」の「3.3.17　gRPC アダプタを定義する」を参照してください。

(3)　フォルト応答

gRPC アダプタでは，次のフォルト応答を設定できます。

- ステータスコードに応じたフォルト応答
- システム例外発生時のフォルト応答

それぞれのフォルト応答について説明します。

(a)　ステータスコードに応じたフォルト応答

gRPC アダプタでは，gRPC 呼び出し実行時に返されたステータスコードが「OK」以外の場合に，フォルト応答としてフォルトを返す設定ができます。フォルト応答は，gRPC アダプタ独自定義ファイルの adpgrpc.response-code.fault で設定します。

gRPC サービスから返されたステータスコードが，gRPC アダプタ独自定義ファイルの adpgrpc.response-code.fault に指定したステータスコードのどれかと一致する場合にフォルト応答を行います。

gRPC アダプタのフォルト応答では，例外情報を基にフォルト電文を作成し，フォルト応答としてフォルト電文をビジネスプロセスに返します。

フォルト電文の設定については，マニュアル「サービスプラットフォーム 開発ガイド 受付・アダプタ定義編」の「3.3.17　gRPC アダプタを定義する」を参照してください。

ステータス情報には詳細メッセージが設定されている場合があります。詳細メッセージが，gRPC が提供している proto ファイルである error_details.proto（エラーの詳細に関する標準エラーペイロードのセット）内に定義されているメッセージのとき，gRPC アダプタはフォルト電文内の詳細メッセージを設定するための要素に，メッセージの内容を設定します。対象のメッセージを次に示します。

- RetryInfo
- DebugInfo
- QuotaFailure
- ErrorInfo
- PreconditionFailure
- BadRequest
- RequestInfo

- ResourceInfo
- Help
- LocalizedMessage

error_details.proto 内に定義されていないメッセージの場合，フォルト電文の additional-message タグにメッセージの内容を設定します。

(b) システム例外発生時のフォルト応答

gRPC アダプタでシステム例外が発生した場合，または gRPC 呼び出し実行時に例外が発生した場合，システム例外をフォルトに変換して，ビジネスプロセスにフォルト応答を返すことができます。

システム例外発生時のフォルト応答を行うには，次の条件をすべて満たす必要があります。

- gRPC アダプタ独自定義ファイルの adpgrpc.response-code.fault に指定したステータスコードのどれとも一致しない。
- サービスアダプタ定義画面で［システム例外をフォルトに変換する］をチェックしている。

(4) メッセージの圧縮と解凍

gRPC サービスに gzip 形式で圧縮したメッセージを送信できます。また，gRPC サービスから gzip 形式で圧縮されたメッセージが返された場合，メッセージを解凍します。メッセージの圧縮は，gRPC アダプタ独自定義ファイルの adpgrpc.protocol.message-compression.enabled で設定します。gRPC サービスから返されたメッセージが gzip 形式以外で圧縮されたデータの場合，リクエストはエラーとなります。

(5) gRPC アダプタのセキュリティ

gRPC アダプタと gRPC サーバ間で交換されるすべてのデータを暗号化できます。gRPC アダプタでは，次の認証方式をサポートしています。

- SSL/TLS
- Token-based authentication with Google

(a) SSL/TLS の設定

SSL/TLS 認証を設定する場合，gRPC アダプタ実行環境（共通）プロパティファイルの adpgrpc.request.use-plaintext に false を設定する必要があります。

また，設定する項目は，認証方式によって異なります。認証方式ごとの設定項目を次に示します。

表 2-52 認証方式ごとの設定項目

認証方式	設定項目	プロパティ	設定の要否
サーバ認証	サーバのルート証明書	adpgrpc.security.tls.certificate-chain	任意

認証方式	設定項目	プロパティ	設定の要否
双方向認証※	サーバのルート証明書	adpgrpc.security.tls.certificate-chain	任意
	クライアントの証明書チェーン	adpgrpc.security.tls.client.certificate-chain	必須
	クライアントの秘密鍵	adpgrpc.security.tls.client.private-key	必須
	クライアントの秘密鍵のパスワード	adpgrpc.security.tls.client.private-key-password	任意

注※

双方向認証をする場合，必須の情報がすべて設定されている必要があります。設定されていない場合は，双方向認証は行われません。

(b) Token-based authentication with Google の設定

gRPC アダプタから Google API（Google 社が提供する各種サービスにアクセスする機能）を呼び出す場合に必要な認証です。

Token-based authentication with Google の設定は，gRPC アダプタ実行環境（共通）プロパティファイルで設定します。Token-based authentication with Google についての設定項目を次に示します。

表 2-53 Token-based authentication with Google 認証の設定項目

設定項目	プロパティ	設定の要否	説明
スコープ	adpgrpc.security.token-based-auth.scope	任意	Google API へのアクセスに必要となる OAuth 2.0 スコープを指定します。 認証されたユーザを承認するかどうかを判定するために設定します。
クレデンシャル	adpgrpc.security.token-based-auth.credentials	必須	サービスアカウントキーファイルを指定します。ファイルの内容が JSON 形式でない場合の動作は保証されません。

アクセストークンの有効期限が切れた場合，自動的にアクセストークンはリフレッシュされます。そのため，ユーザがアクセストークンを手動で再取得する必要はありません。

(6) gRPC アダプタのタイムアウト

gRPC アダプタのタイムアウトは，gRPC アダプタ独自定義ファイルで設定します。

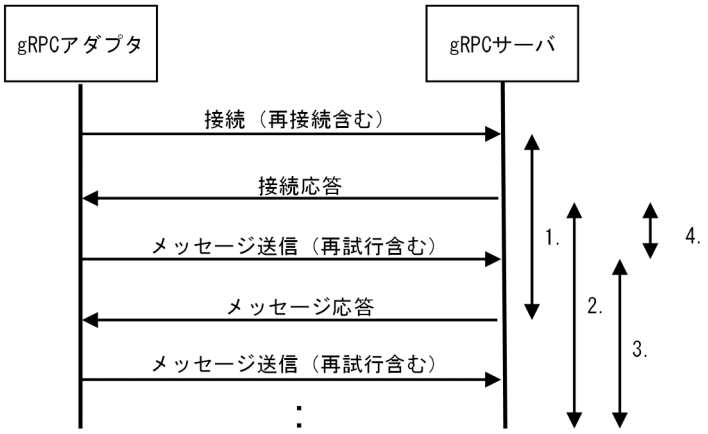
表 2-54 gRPC アダプタのタイムアウトの設定項目

設定項目		プロパティ	説明
キープアライブ	PING の送信間隔	adpgrpc.protocol.keepalive-time	キープアライブ PING の送信間隔を指定します。

設定項目		プロパティ	説明
キープアライブ	タイムアウト時間	adpgrpc.protocol.keepalive-timeout	キープアライブタイムアウト時間を指定します。 設定した時間が経過しても接続先から応答がない場合，接続を終了します。再度 RPC 要求があるまで再接続はしません。
	未処理の RPC がない場合の動作	adpgrpc.protocol.keepalive-without-calls	未処理の RPC がない場合に，キープアライブを実行するかどうかを指定します。
アイドルタイムアウト時間		adpgrpc.protocol.idle-timeout	チャネルの状態をアイドル状態に移行するまでの時間を指定します。 アイドル状態になった場合，再度 RPC 要求があるまで再接続はしません。
リクエストの実行期限		adpgrpc.protocol.deadline	RPC が DEADLINE_EXCEEDED でエラー終了する前の，RPC の完了待ち時間を指定します。 gRPC アダプタの接続先が利用できるようになるまで，gRPC 呼び出しを待機します。実行期限が経過しても RPC が完了しなかった場合，リクエストはエラーとなり，ステータスコード「DEADLINE_EXCEEDED」が返されます。

gRPC で指定できるタイムアウトの関係を次に示します。

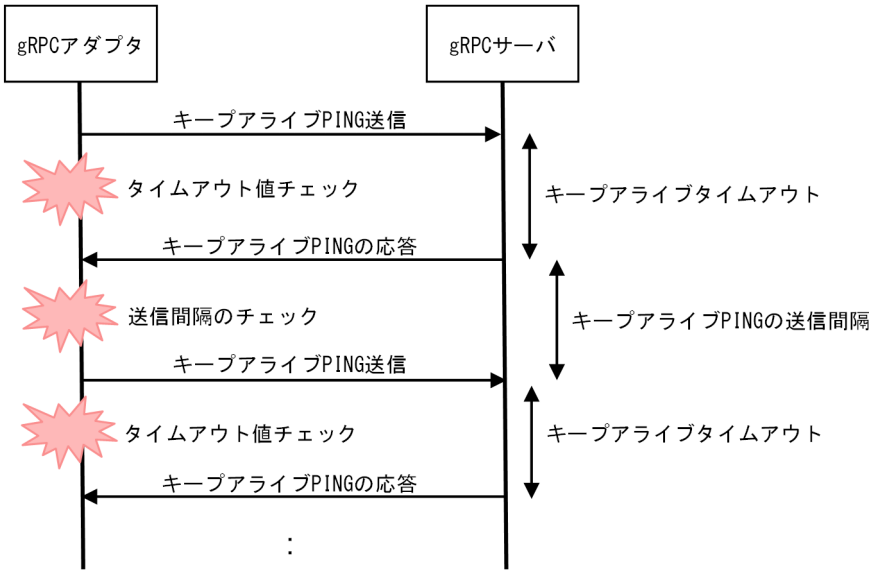
図 2-125　gRPC で指定できるタイムアウトの関係



1. リクエストの実行期限
2. キープアライブ（未処理の RPC がない場合に，キープアライブを実行する場合）
3. キープアライブ（未処理の RPC がない場合に，キープアライブを実行しない場合）
4. アイドルタイムアウト時間

キープアライブタイムアウトの流れを次に示します。

図 2-126　キープアライブタイムアウトの流れ



キープアライブ PING の送信間隔が 10 分の場合，「キープアライブ PING の応答」の 10 分後に，gRPC アダプタはキープアライブ PING を送信します。

キープアライブタイムアウトが 20 秒の場合，gRPC アダプタがキープアライブ PING を送信してから 20 秒以内に「キープアライブ PING の応答」がないときはタイムアウトしたとみなされます。

2.14　メールサーバとの接続

メールアダプタを利用することで，サービスプラットフォームとメールサーバとを接続できます。

2.14.1　前提条件

メールアダプタを利用するための前提条件を次に示します。

- メールサーバは HCSC サーバと同じ LAN 上に配置する必要があります。
- メールアダプタは JavaMail の仕様に従い，SMTP プロトコルをサポートした（RFC821 に準拠した）メールサーバをサポートしています。

2.14.2　メールアダプタを使用したシステム

メールアダプタは，SMTP プロトコルをサポートしたメールサーバをサービスプラットフォーム上でサービスとして呼び出すことができます。

メールアダプタを利用することで，ビジネスプロセスが正常に完了した場合や，システムに障害が発生した場合に，ユーザやシステム管理者にメールで通知できるようになります。また，FTP 受付や FTP アダプタと連携して利用することで，任意のファイルをメールに添付して送信することもできます。

メールアダプタを利用したシステムの例を次に示します。

- FTP クライアントから転送されたファイルをメールに添付して送信する
- FTP サーバから取得したファイルをメールに添付して送信する
- ワークフロー案件の承認依頼をメールで送信する
- 非同期のビジネスプロセスの完了通知をメールで送信する
- ビジネスプロセスの実行中に障害が発生したことをメールで通知する

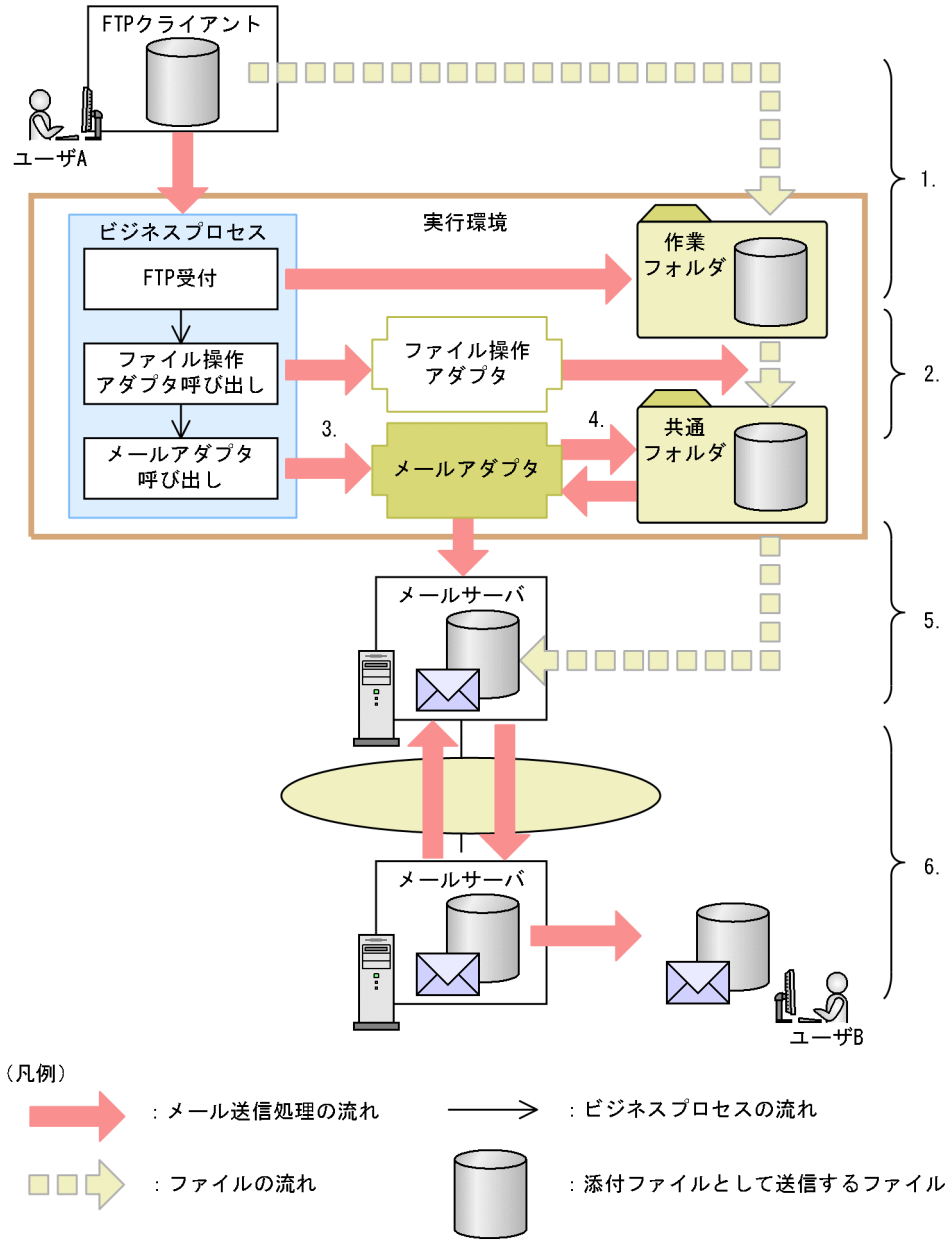
それぞれのシステムの例について説明します。なお，FTP 受付や FTP アダプタの機能については，「8. FTP と連携してファイルを転送する機能（FTP 連携）」を参照してください。

また，TLS を利用した暗号化をサポートしているメールサーバとの連携に関する設定方法については，マニュアル「サービスプラットフォーム 開発ガイド 受付・アダプタ定義編」の「付録 J　メールアダプタで必要なセキュリティの設定」を参照してください。

(1)　FTP クライアントから転送されたファイルをメールに添付して送信する

この例では，ユーザが FTP クライアントから転送したファイルをメールに添付して別のユーザに送信しています。

図 2-127　メールアダプタを利用したシステムの例（1）



メール送信および添付ファイルの流れを次に示します。なお，次の番号は図中の番号と対応しています。

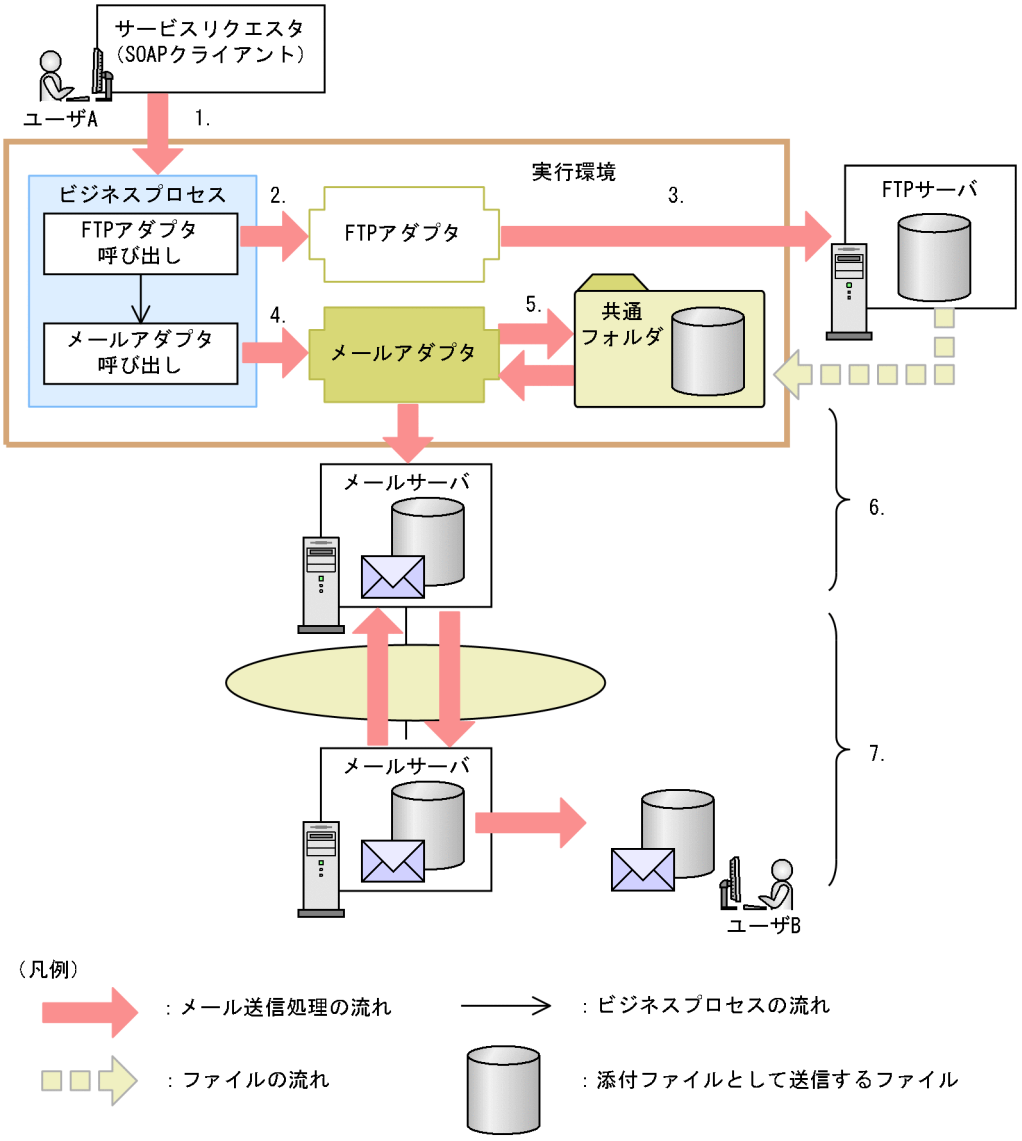
1. ユーザ A の FTP クライアントから PUT オペレーションで任意のファイルをサービスプラットフォームに転送します。転送されたファイルは，FTP 受付が作成した HCSC サーバ内の作業フォルダに保存されます。
2. ビジネスプロセスから呼び出されたファイル操作アダプタによって，作業フォルダのファイルが共通フォルダにコピーされます。また，必要に応じてファイル操作アダプタでフォーマット変換／ファイル変換が実施されます。
3. ビジネスプロセスからメールアダプタが呼び出されます。
4. 共通フォルダのファイルがメールの添付ファイルとして設定されます。
5. メールアダプタから同一 LAN 内のメールサーバにメールが送信されます。

6. 添付ファイルが設定されたメールがネットワークを介してメールサーバからユーザ B に送信されます。

(2) FTP サーバから取得したファイルをメールに添付して送信する

この例では，SOAP クライアントからビジネスプロセスを開始したあと，FTP アダプタ経由で FTP サーバから取得されたファイルをメールに添付して，別のユーザに送信しています。

図 2-128 メールアダプタを利用したシステムの例（2）



メール送信および添付ファイルの流れを次に示します。なお，次の番号は図中の番号と対応しています。

1. ユーザ A のサービスリクエスタ（SOAP クライアント）からサービスプラットフォームの Web サービスを呼び出します。
2. ビジネスプロセスから FTP アダプタが呼び出されます。
3. FTP アダプタの GET オペレーションによって，FTP サーバから共通フォルダにファイルが転送されます。

4. ビジネスプロセスからメールアダプタが呼び出されます。

5. 共通フォルダに保存されたファイルがメールの添付ファイルとして設定されます。

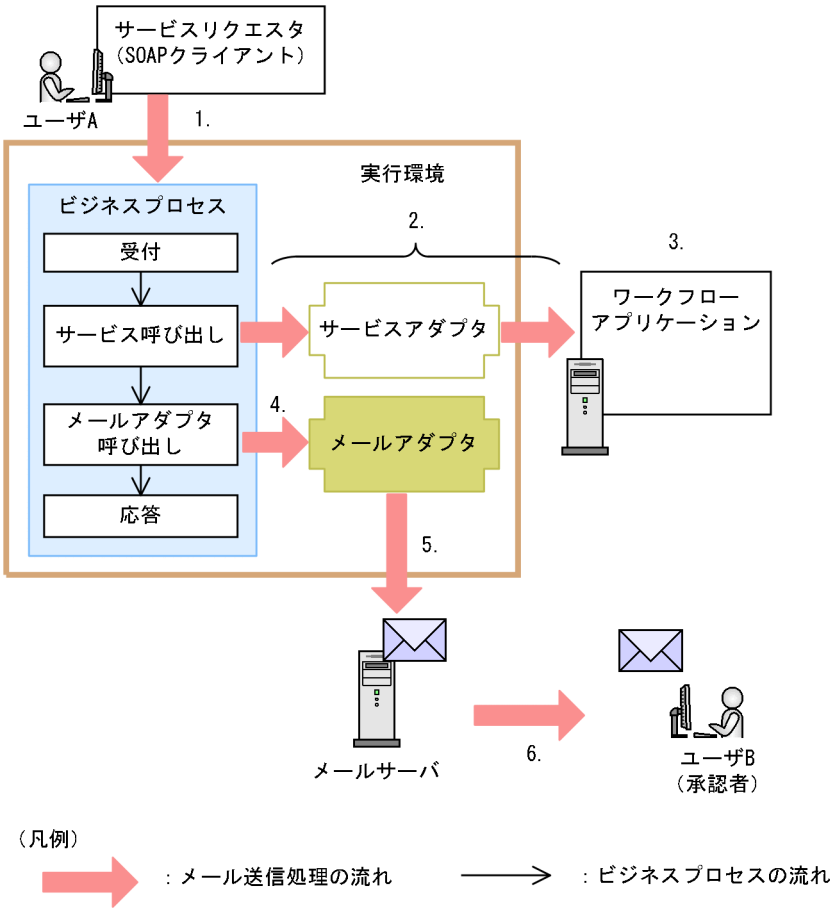
6. メールアダプタから同一 LAN 内のメールサーバにメールが送信されます。

7. 添付ファイルが設定されたメールがネットワークを介してメールサーバからユーザ B に送信されます。

(3)　ワークフロー案件の承認依頼をメールで送信する

この例では，ビジネスプロセスからサービスアダプタ経由でワークフローアプリケーションが呼び出されたあと，承認者に承認依頼メールを通知してワークフローを処理しています。

図 2-129　メールアダプタを利用したシステムの例（3）



メール送信の流れを次に示します。なお，次の番号は図中の番号と対応しています。

1. ユーザ A のサービスリクエスタ（SOAP クライアント）からサービスプラットフォームの Web サービスを呼び出します。

2. ビジネスプロセスからサービスアダプタを経由してワークフローアプリケーションが呼び出されます。

3. ワークフローの処理が開始され，承認プロセス待ちの状態になります。

4. ビジネスプロセスからメールアダプタが呼び出されます。

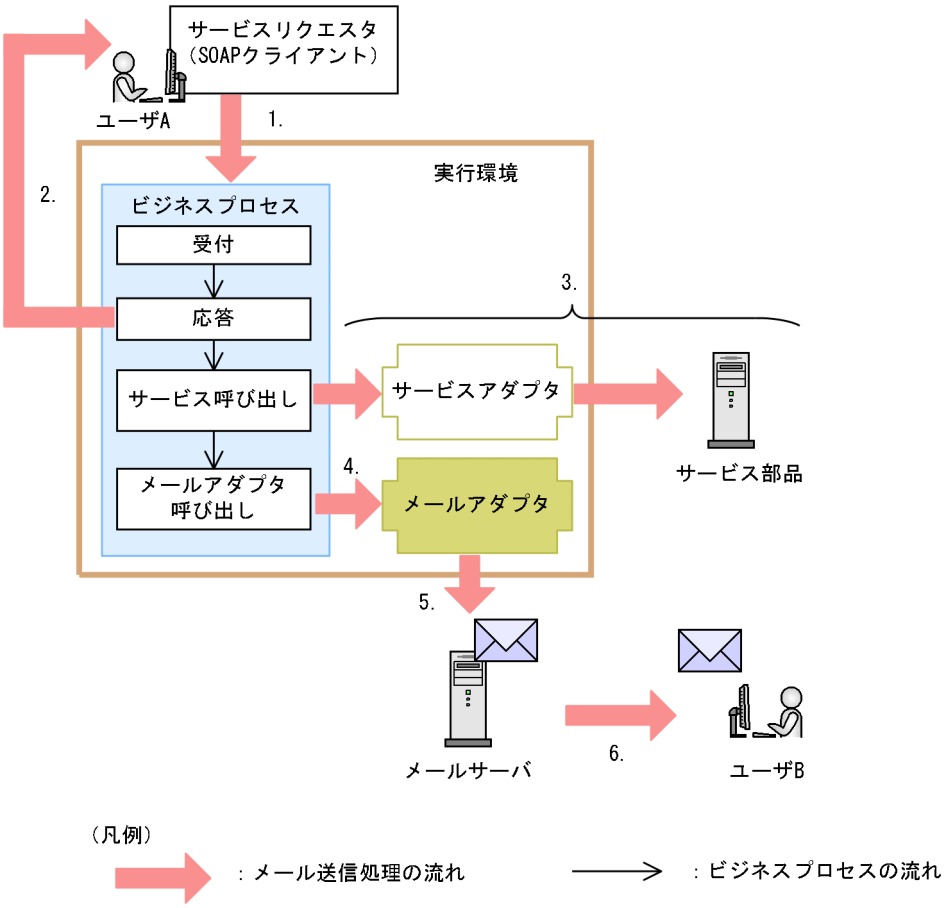
5. 承認サイトの URL がメール本文に設定されます。

6. メールがユーザ B（承認者）に送信されたあと，ユーザ B（承認者）は，メール本文に記載された URL をクリックして承認プロセスを実施します。

(4) 非同期のビジネスプロセスの完了通知をメールで送信する

非同期のビジネスプロセスの場合，サービスリクエスタへの応答後にビジネスプロセスの処理が続行しても，ユーザには以降の処理の完了が通知されません。この例では，メールアダプタを使用して，非同期のビジネスプロセスが完了したことをユーザにメールで通知しています。

図 2-130 メールアダプタを利用したシステムの例（4）



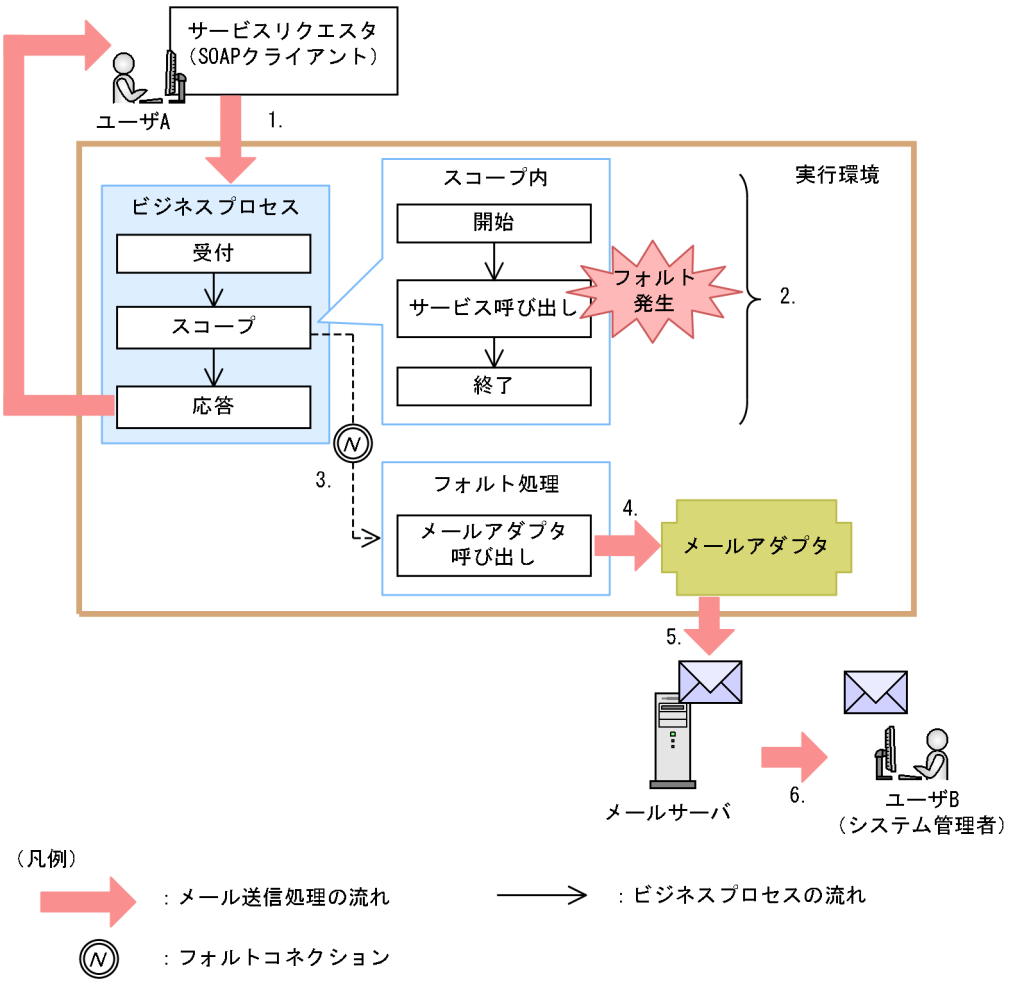
メール送信の流れを次に示します。なお，次の番号は図中の番号と対応しています。

1. ユーザ A のサービスリクエスタ（SOAP クライアント）からサービスプラットフォームの Web サービスを呼び出します。
2. ユーザ A のサービスリクエスタに応答電文が返されます。
3. 非同期のビジネスプロセスからサービスアダプタが呼び出され，サービスを実行します。
4. 非同期のビジネスプロセスからメールアダプタが呼び出されます。
5. サービス呼び出しの処理が正常に完了したことがメール本文に記載されます。
6. ユーザ B にメールが送信されたあと，ユーザ B は処理が完了したことを確認します。

(5)　ビジネスプロセスの実行中に障害が発生したことをメールで通知する

この例では，ビジネスプロセスのサービス呼び出し中にエラーが起きた場合に，メール経由でシステム管理者にエラーを通知しています。

図 2-131　メールアダプタを利用したシステムの例（5）



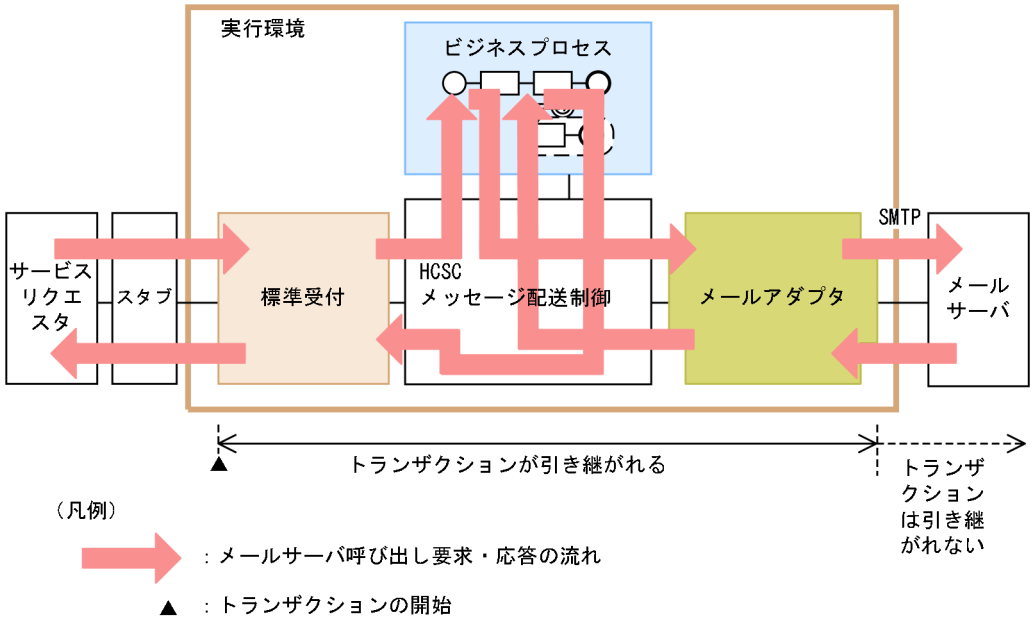
メール送信の流れを次に示します。なお，次の番号は図中の番号と対応しています。

1. ユーザ A のサービスリクエスタ（SOAP クライアント）からサービスプラットフォームの Web サービスを呼び出します。
2. ビジネスプロセスのスコープ内でサービス呼び出し中に障害が発生し，サービスリクエスタにフォルト電文が返されます。
3. ビジネスプロセスの処理がエラー時のフロー（フォルト処理）に遷移します。
4. フォルト処理内でメールアダプタが呼び出されます。
5. エラーメッセージがメール本文に記載されます。
6. ユーザ B（システム管理者）にメールが送信されたあと，ユーザ B（システム管理者）はエラーの発生を確認し，トラブルシュートを実施します。

2.14.3　メールアダプタ使用時のトランザクション

メールアダプタを使用した場合のトランザクションの範囲を次の図に示します。

図 2-132　メールアダプタを使用した場合のトランザクションの範囲



メールサーバへの処理（メール送信）が完了したあとにビジネスプロセスの処理でエラーが発生した場合，トランザクションはロールバックされ，ビジネスプロセスにエラーが返されます。しかし，メールサーバに送信したメールは引き戻すことはできません。

2.14.4　メールアダプタで設定できるメール情報

メールアダプタでは，メールを送信するときに必要な情報を要求電文またはメールアダプタ実行環境プロパティファイルで設定できます。要求電文とメールアダプタ実行環境プロパティファイルの両方で指定された場合は，要求電文が優先されます。

メールアダプタで設定できるメール情報を次の表に示します。

表 2-55　設定できるメール情報

大分類	小分類	設定内容
アドレス	アドレス種別	メールのアドレス種別（TO/CC/BCC），および宛先に指定する電子メールアドレスを設定できます。
メールの主題	－	メールの主題を設定できます。
メールの本文	メール本文の内容	次の方法でメールの本文を設定できます。 • メール本文の内容を直接記述する • メール本文の内容が記述されたファイルのパスを指定する※

大分類	小分類	設定内容
メールの本文	メール形式	メールの形式を指定できます。 メールアダプタがサポートする形式を次に示します。 • テキスト形式 • HTML 形式 • XML 形式
	エンコード	メール送信時のエンコードを指定できます。 エンコードを指定しなかった場合，ISO-2022-JP が使用されます。
	文字セット	メール本文の内容が記述されたファイルを読み込んでメール本文に設定する場合，テキストの文字セットを指定できます。 文字セットを指定しなかった場合，UTF-8 が使用されます。
添付ファイル	－	メールに添付するファイルのパスを指定できます。※
メールヘッダ	－	メールの重要度，返信要求の有無など，ユーザが定義したメールヘッダのオプションを設定できます。

（凡例）

－：該当する項目はありません。

注※

メールアダプタでは，メール本文として使用するファイルおよび添付ファイルに，共通フォルダルートの配下のサブフォルダを指定できます。共通フォルダルートの配下のサブフォルダは，メールアダプタ要求電文の/request/mail-body/file/mail-body-file-name，または/request/attachment-file/item/attach-file-name に共通フォルダルートからファイル名までの相対パスを指定します。指定例を次に示します。

```
<request>
      :
  <folder-name common="true">default</folder-name>
  <mail-body-file-name>/folder1/folder2/body.txt</mail-body-file-name>
      :
  <attach-folder common="true">default<attach-folder>
  <attach-file-name>/folder1/folder2/attach.txt</attach-file-name>
      :
</request>
```

要求電文で設定できる項目の詳細は，マニュアル「サービスプラットフォーム 開発ガイド 受付・アダプタ定義編」の「3.3.12　メールアダプタを定義する」の adpmail_smtp_request.xsd（要求電文フォーマットファイル）について説明している個所を参照してください。メールアダプタ実行環境プロパティファイルで設定できる項目の詳細は，マニュアル「サービスプラットフォーム リファレンス」の「6.13.3　メールアダプタ実行環境プロパティファイル」を参照してください。

2.14.5　メールアダプタのタイマ管理

ここでは，メールアダプタのタイマ管理の概要，およびタイムアウト値の設定方法について説明しています。

メールアダプタでは，再接続タイムアウトおよび排他管理タイムアウトを設定できます。各タイムアウト値は，メールアダプタ実行環境プロパティファイルで設定できます。メールアダプタ実行環境プロパティファイルの詳細は，マニュアル「サービスプラットフォーム リファレンス」の「6.13.3　メールアダプタ実行環境プロパティファイル」を参照してください。

(1)　再接続タイムアウト

メールサーバがダウンした場合や，メールサーバ接続中またはメール送信中にネットワーク障害が発生した場合，メールアダプタはメールサーバ接続タイムアウトに指定されたタイムアウト値までメールサーバへの接続をリトライします。また，メール送信タイムアウトを設定することで，メール送信に失敗するまでのタイムアウト時間を設定できます。

メールアダプタのリクエスト送信時に例外メッセージ「KDJE59101-E」または「KDJE59105-E」が出力され，例外の原因がタイムアウトだった場合は，次に示すプロパティの設定を見直してください。

設定できるパラメタを次の表に示します。

表 2-56　メールサーバ接続タイムアウトおよびメール送信タイムアウトの設定

値	キー名称	デフォルト値
メールサーバ接続タイムアウト（秒）	mailadp.smtp.connectiontimeout	180
メール送信タイムアウト（秒）	mailadp.smtp.timeout	180

(2)　排他管理タイムアウト

添付ファイル付きのメールを送信する場合，要求電文で指定した共通フォルダ内の添付ファイルのパスにアクセスするとき，指定したファイルに共有ロックが設定されます。

要求電文で作業フォルダを指定したとき，およびメールアダプタ実行環境プロパティファイルで指定したパスのファイルにアクセスするときは，共有ロックが設定されません。

排他管理タイムアウトでは，メールアダプタが共有ロックの取得に失敗したときのリトライ回数およびリトライ間隔を設定できます。

なお，共有ロックの取得に失敗したときのリトライ状態はメンテナンスログに出力されます。メンテナンスログの詳細は，マニュアル「サービスプラットフォーム システム構築・運用ガイド」の「7.3.2　障害情報の取得方法（サービスプラットフォーム）」の保守情報の取得方法について説明している個所を参照してください。

設定できるパラメタを次の表に示します。

表 2-57　排他管理タイムアウトの設定

値	キー名称	デフォルト値
排他管理ロックリトライ回数（回）	mailadp.read-lock.retry.count	0
排他管理ロックリトライ間隔（秒）	mailadp.read-lock.retry.interval	1

2.14.6　メールアダプタからメールを送信するときの認証

ここでは，メールアダプタからメールを送信するときの SMTP 認証の指定方法，および認証に必要なユーザ名およびパスワードの設定方法について説明します。

(1)　SMTP 認証の指定

SMTP 認証の形式には，LOGIN または PLAIN の 2 種類があります。

LOGIN
　メールアダプタアカウント定義ファイルに指定されたユーザ名とパスワードを使って認証されます。

PLAIN
　メール送信時に認証されません。

認証形式は要求電文またはメールアダプタ実行環境プロパティファイルのどちらかで指定できます。要求電文の場合，<mail-auth>タグで指定します。メールアダプタ実行環境プロパティファイルの場合，mailadp.smtp.auth キーで指定します。

要求電文で設定できる項目については，マニュアル「サービスプラットフォーム 開発ガイド 受付・アダプタ定義編」の「3.3.12　メールアダプタを定義する」の adpmail_smtp_request.xsd（要求電文フォーマットファイル）について説明している個所を参照してください。メールアダプタ実行環境プロパティファイルで設定できる項目についてはマニュアル「サービスプラットフォーム リファレンス」の「6.13.3 メールアダプタ実行環境プロパティファイル」を参照してください。

要求電文とメールアダプタ実行環境プロパティファイルの両方で指定された場合は，要求電文が優先されます。

要求電文とメールアダプタ実行環境プロパティファイルの両方に指定がない場合，PLAIN 認証でメールが送信されます。

(2)　ユーザ名およびパスワードの設定

LOGIN 認証に必要なユーザ名およびパスワードは，要求電文またはプロパティファイルを利用して設定できます。ユーザ名およびパスワードの設定方法を次に示します。

ユーザ名
　要求電文またはメールアダプタ実行環境プロパティファイルで設定できます。

要求電文の場合，<mail-user>タグで指定します。メールアダプタ実行環境プロパティファイルの場合，mailadp.user キーで指定します。

パスワード

csmmailaddusr コマンドを使用してユーザ名に対応するパスワードをメールアダプタアカウント定義ファイルに追加します。

2.15　HTTP 通信を使用したクライアントとの接続

HTTP 受付を利用することで，HTTP クライアントから HTTP 通信を使用してビジネスプロセスを呼び出すことができます。

携帯端末や Web ブラウザからサービスプラットフォームに接続要求を出す場合，HTTP 受付を使用すると，Web フロントシステムや SOAP 受付を経由しないで，直接ビジネスプロセスを呼び出せるようになります。

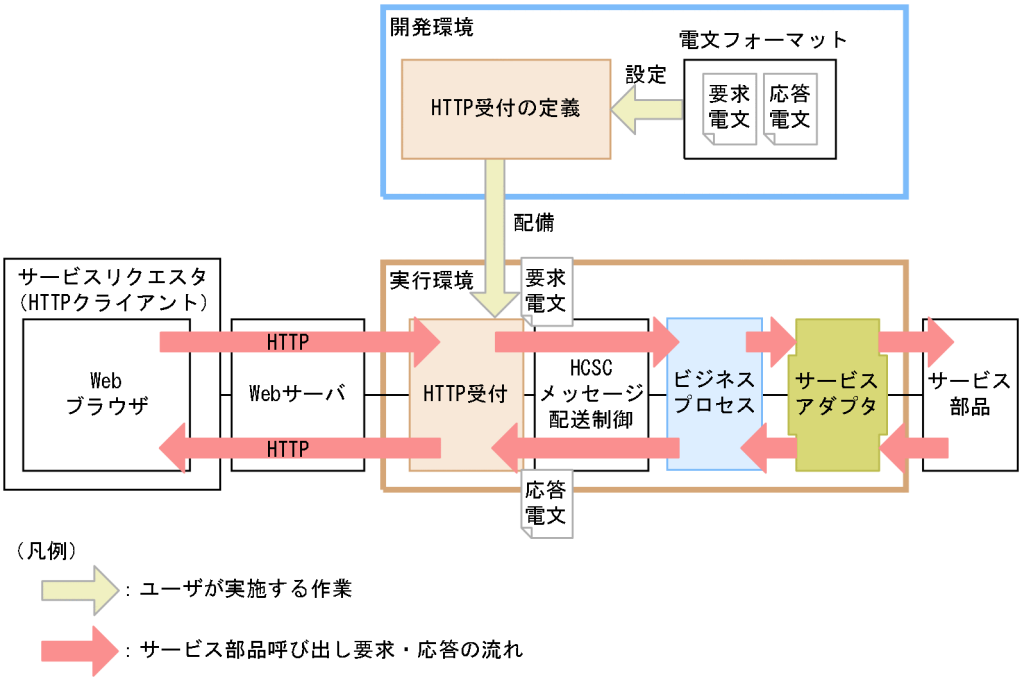
2.15.1　HTTP 受付を使用したサービス部品の呼び出し

HTTP クライアントとして利用する Web ブラウザから発信された HTTP リクエストは，Web サーバを経由して HTTP 受付で受け付けられます。そのあと，HTTP リクエストは要求電文（サービス部品呼び出し要求）に変換され，ビジネスプロセス上でサービス部品を呼び出します。

応答時は，ビジネスプロセス上の応答電文が HTTP 受付で HTTP レスポンスの形式に変換され，Web ブラウザに応答を返します。

HTTP 受付を使用したサービス部品呼び出し要求・応答の流れを次の図に示します。

図 2-133　HTTP 受付を使用したサービス部品呼び出し要求・応答の流れ



HTTP 受付でサポートするメソッド

HTTP 受付でサポートするメソッドを次に示します。

GET メソッド

HTTP クライアントからサービスプラットフォームにクエリストリングの情報を渡して，テキストデータおよびファイルのデータをダウンロードするためのメソッドです。

POST メソッド

HTTP クライアントからサービスプラットフォームにテキストデータの送信およびファイルのデータをアップロードするためのメソッドです。レスポンスとしてテキストデータの受信およびファイルのデータをダウンロードすることもできます。

HEAD メソッド

HTTP クライアントからサービスプラットフォームにクエリストリングの情報を渡して，HTTP ヘッダをアップロードするためのメソッドです。

PUT メソッド

HTTP クライアントからサービスプラットフォームにテキストデータの送信およびファイルのデータをアップロードするためのメソッドです。レスポンスとしてテキストデータの受信およびファイルのデータをダウンロードすることもできます。テキストデータ，およびファイルを更新または作成する目的で使用します。

DELETE メソッド

テキストデータおよびファイルのデータを削除するためのメソッドです。

OPTIONS メソッド

利用できるメソッドの一覧を取得するためのメソッドです。

HTTP 受付でサポートする Web サーバおよび HTTP クライアント

HTTP 受付でサポートする Web サーバおよび HTTP クライアントを次に示します。

Web サーバ

HTTP 1.1 に準拠した，Component Container で動作する Web サーバをサポートしています。

HTTP クライアント

上記の Web サーバがサポートする HTTP クライアントをサポートしています。

2.15.2 HTTP 受付を使用したオンラインシステムの例

ここでは，テキストデータ，ファイル，およびバイナリデータを扱うシステムを例に HTTP 受付の利用方法を紹介します。

(1) テキストデータを送受信するシステムの場合

タブレット端末の Web ブラウザからオンラインで商品を購入するシステムを例に説明します。

この例で使用する Web ブラウザは，ユーザ情報を認証するための［ログイン画面］，購入する商品などの情報を入力するための［入力画面］，および購入結果を表示する［結果画面］で構成されています。ユーザが商品を購入する場合，次の流れで Web ブラウザからリクエストを送信します。

1. GET メソッドのリクエストを送信して商品のリストを取得します。
2. POST メソッドのリクエストを送信してリストから選択した商品を送信し，購入処理を呼び出します。

それぞれのリクエストの処理と HTTP 受付の関係を説明します。

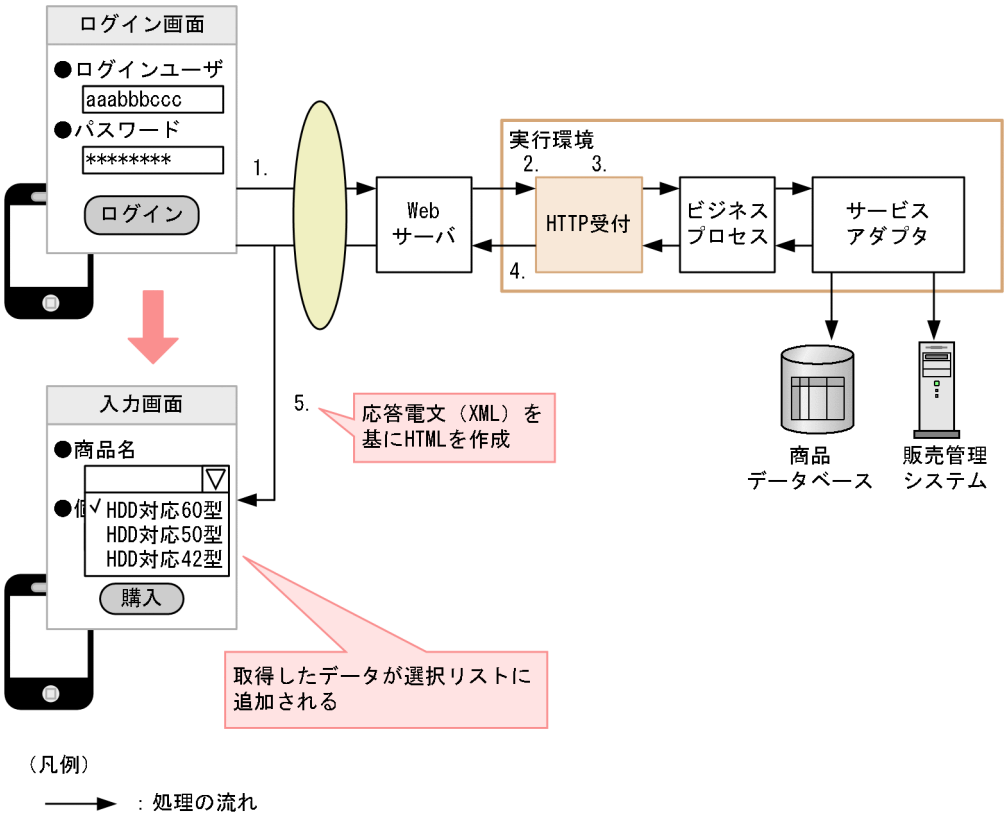
(a) GET メソッドのリクエストを送信する例

このリクエストの例では，Web ブラウザで入力した情報が GET メソッドのリクエストとしてサービスプラットフォームに送信されたあと，対応する HTTP レスポンスのデータを基に生成された画面が Web ブラウザ上に表示される処理が実行されます。

GET メソッドのリクエストを送信する流れを次に示します。なお，次の番号は図中の番号と対応しています。

1. Web ブラウザの［ログイン画面］でユーザが入力した情報が Web サーバに送信されます。
2. Web サーバから HTTP 受付に HTTP リクエストが渡されます。
3. HTTP 受付で HTTP ヘッダと HTTP ボディが，それぞれ要求電文（ヘッダ）と要求電文（ボディ）に変換されます。変換されたデータはサービスプラットフォーム内のビジネスプロセスに渡され，ビジネスプロセスから商品データベースなど，さまざまなサービスを呼び出します。そのあと，応答電文は HTTP 受付で HTTP レスポンスの形式に変換されます。
4. HTTP レスポンスの HTTP ボディには XML 形式の応答電文が設定されます。
5. 応答電文の内容を基に HTML を作成（JavaScript などを利用，またはフロントエンドサーバを経由）し，クライアント側で表示されます。この例では，取得したデータが［入力画面］の選択リストに追加されています。

図 2-134　HTTP 受付を利用したシステムの例（GET メソッドのリクエストを送信した場合）



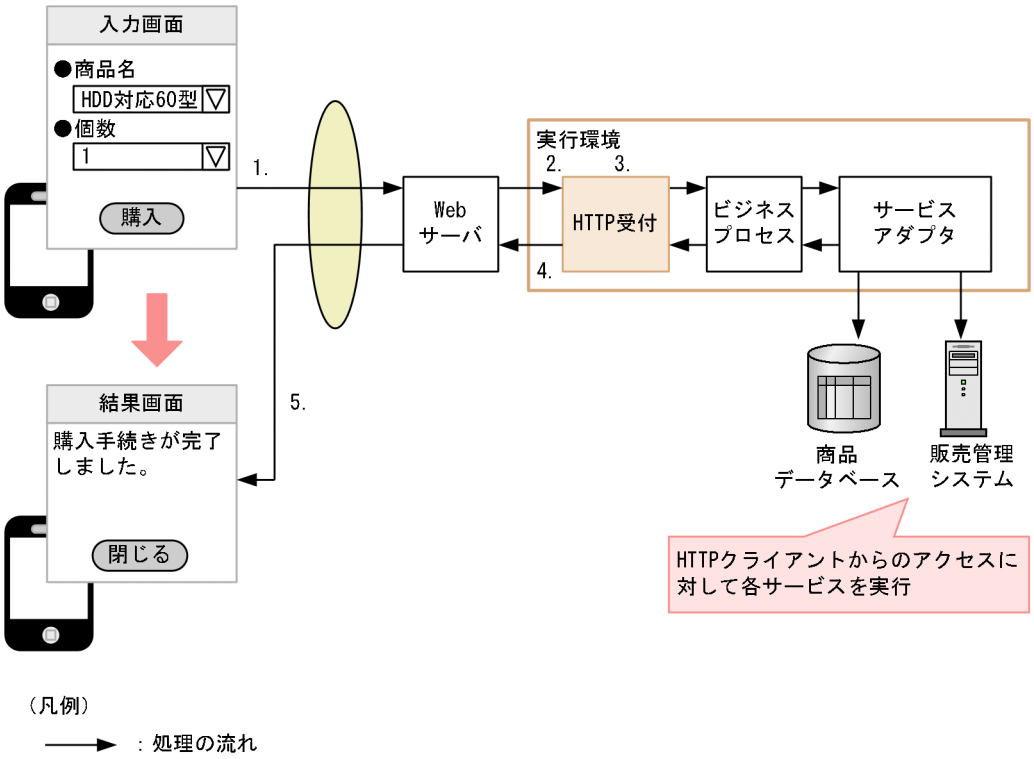
(b)　POST メソッドのリクエストを送信する例

このリクエストの例では，Web ブラウザで入力した情報が POST メソッドのリクエストとしてサービスプラットフォームに送信されたあと，対応する HTTP レスポンスのデータを基に生成された画面が Web ブラウザ上に表示される処理が実行されます。

POST メソッドのリクエストを送信する流れを次に示します。なお，次の番号は図中の番号と対応しています。

1. Web ブラウザの［入力画面］でユーザが入力したデータが Web サーバに送信されます。
2. Web サーバから HTTP 受付に HTTP リクエストが渡されます。
3. GET メソッドのリクエストを送信する場合と同様に，HTTP 受付で HTTP ヘッダと HTTP ボディが，それぞれ要求電文（ヘッダ）と要求電文（ボディ）に変換され，ビジネスプロセスを通じてさまざまなサービスを呼び出します。そのあと，応答電文は HTTP 受付で HTTP レスポンスの形式に変換されます。
4. HTTP レスポンスの HTTP ボディには XML 形式の応答電文が設定されます。
5. 応答電文の内容を基に HTML を作成（JavaScript などを利用，またはフロントエンドサーバを経由）し，クライアント側で表示されます。この例では，商品購入の処理が完了したことを示す［結果画面］が表示されます。

図 2-135　HTTP 受付を利用したシステムの例（POST メソッドのリクエストを送信した場合）



(2)　ファイルを送受信するシステムの場合

HTTP 受付のファイル転送機能を利用して，HTTP クライアント，サービスプラットフォーム，サーバ間でデータを送受信するシステムを例に説明します。

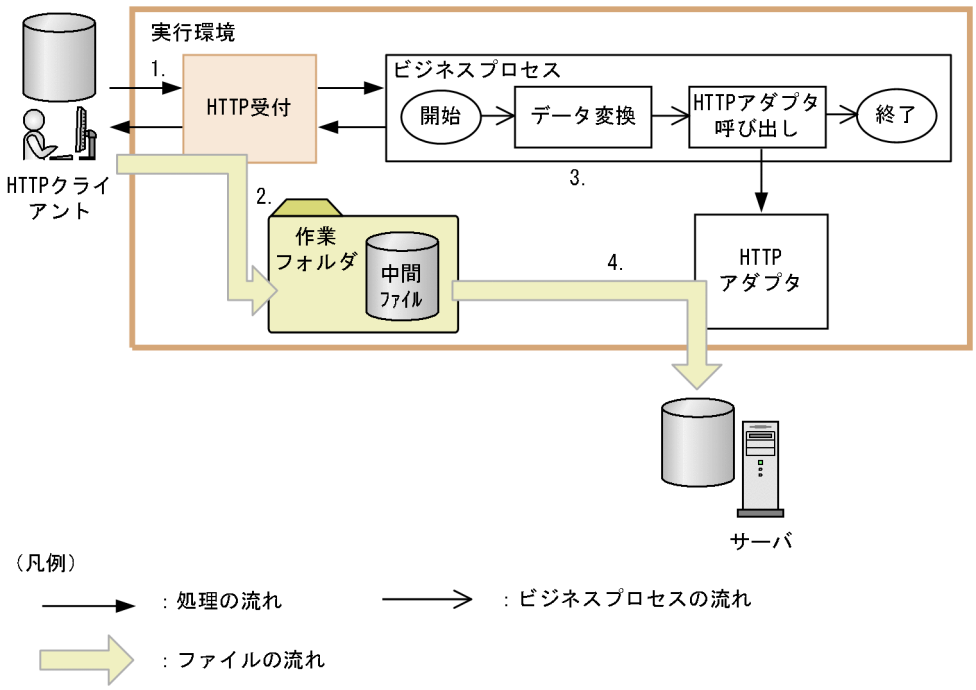
(a)　HTTP クライアントからサーバにファイルを送信する例

このリクエストでは，HTTP クライアントから POST メソッドでアップロードしたファイルを，ビジネスプロセスを介して呼び出した HTTP アダプタから宛先のサーバに送信する処理を実行しています。

HTTP クライアントからサーバにファイルを送信する流れを次に示します。なお，次の番号は図中の番号と対応しています。

1. HTTP クライアントのプログラムから POST メソッドで HTTP リクエストを送信します。
2. HTTP 受付は受信した HTTP リクエストを解析し，ファイルを処理するための作業フォルダを作成します。アップロードされたファイルは中間ファイルとして作業フォルダに生成されます。
3. ビジネスプロセス内のデータ変換アクティビティで，HTTP アダプタの要求電文に宛先サーバと送信データ（HTTP 受付が作業フォルダに保存した中間ファイル）の情報を指定します。
4. ビジネスプロセスから呼び出された HTTP アダプタが宛先のサーバに作業フォルダ内のファイルを送信します。

図 2-136　HTTP 受付を利用したシステムの例（HTTP クライアントからサーバにファイルを送信する場合）



HTTP アダプタの機能については，「2.16　HTTP 通信を使用したサービスとの接続」を参照してください。

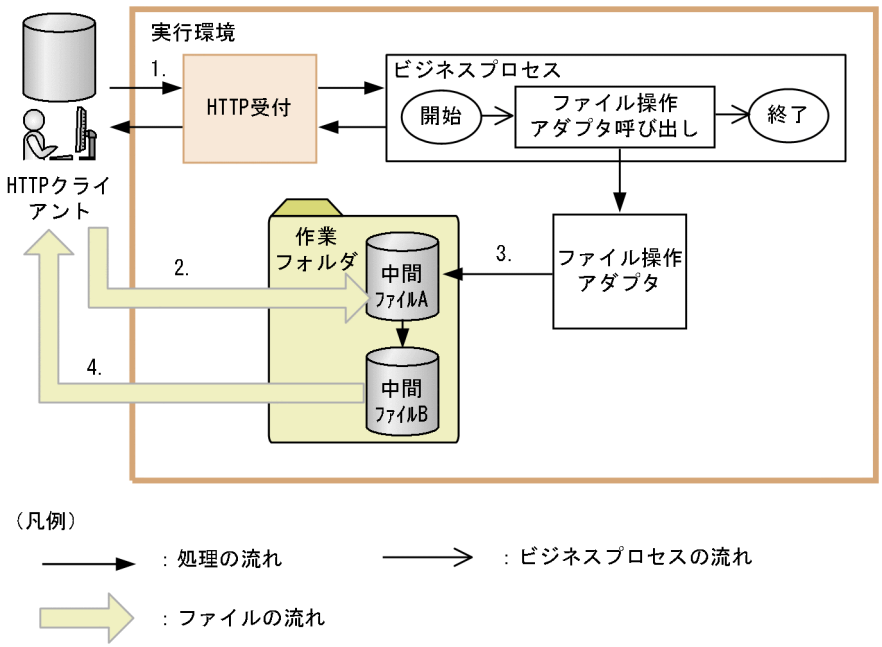
(b)　HTTP クライアントから送信したファイルを変換する例

このリクエストでは，HTTP クライアントから POST メソッドでアップロードしたファイルを，ビジネスプロセスを介して呼び出したファイル操作アダプタで変換してから，HTTP クライアントにダウンロードする処理を実行しています。

HTTP 受付を利用したファイル送信の流れを次に示します。なお，次の番号は図中の番号と対応しています。

1. HTTP クライアントのプログラムから POST メソッドで HTTP リクエストを送信します。
2. HTTP 受付は受信した HTTP リクエストを解析し，ファイルを処理するための作業フォルダを作成します。アップロードされたファイルは中間ファイル A として作業フォルダに生成されます。
3. ビジネスプロセスから呼び出されたファイル操作アダプタが作業フォルダ内の中間ファイル A を中間ファイル B に変換します。
4. HTTP 受付は，HTTP クライアントに変換後の中間ファイル B をダウンロードします。

図 2-137　HTTP 受付を利用したシステムの例（HTTP クライアントから送信したファイルを変換する場合）



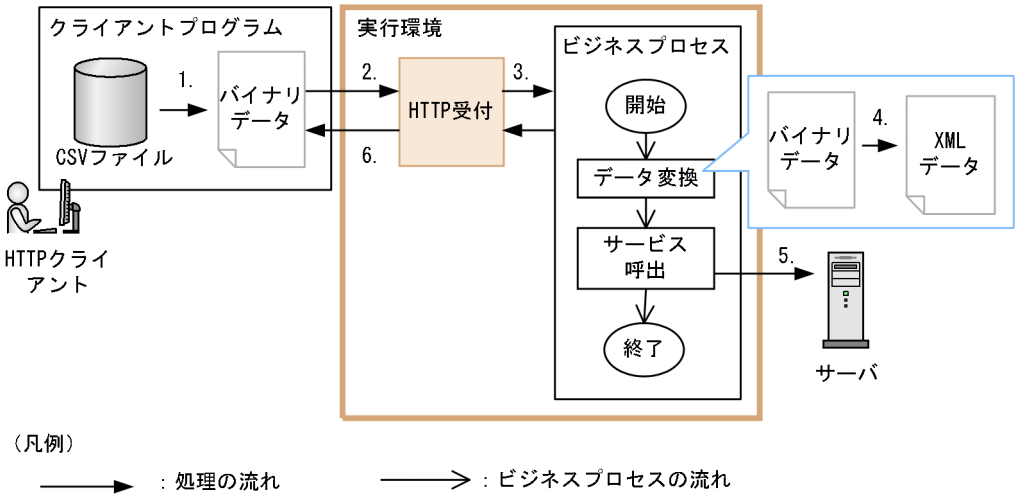
(3)　バイナリデータを送受信するシステムの場合

このリクエストでは，バイナリデータをビジネスプロセスで処理する例として，CSV 形式のファイルを変換したバイナリデータを HTTP クライアントからビジネスプロセスに送信する処理を実行しています。

HTTP クライアントからビジネスプロセスにバイナリデータを送信する流れを次に示します。なお，次の番号は図中の番号と対応しています。

1. HTTP クライアントのプログラムで CSV 形式のファイルをバイナリデータに変換し，HTTP リクエストボディに設定します。
2. HTTP クライアントのプログラムから，POST メソッドで HTTP リクエストを送信します。
3. HTTP 受付は受信した HTTP リクエストを解析し，ビジネスプロセスを呼び出します。
4. ビジネスプロセス内のデータ変換アクティビティで，要求電文に設定されたバイナリデータを XML データ（XML 電文）に変換します。
5. サービス呼出アクティビティを使用して XML データ（XML 電文）を各サービスに送信します。
6. HTTP 受付が HTTP レスポンスを HTTP クライアントに返します。

図 2-138　HTTP 受付を利用したシステムの例（HTTP クライアントからビジネスプロセスにバイナリデータを送信する場合）



2.15.3　HTTP 受付の構成例

HTTP 受付では，送受信する次のデータ形式に応じてさまざまなシステム構成を実現できます。

- テキストデータまたはバイナリデータを送受信する場合
- ファイルのデータを送受信する場合

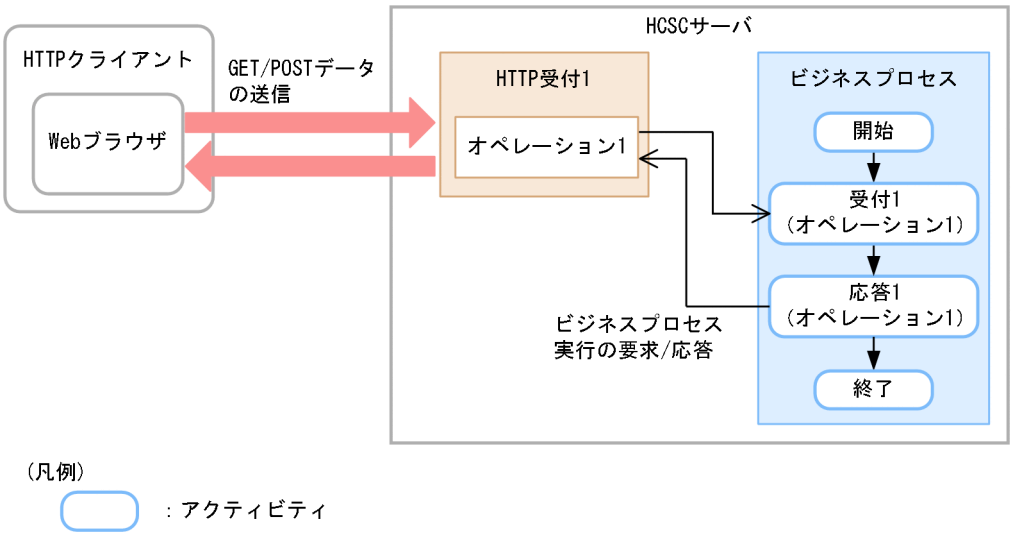
HTTP 受付およびビジネスプロセスの構成例を次に示します。

(1)　テキストデータまたはバイナリデータを送受信する場合

(a)　ビジネスプロセスと単一の HTTP 受付の構成例

単一の HTTP 受付とビジネスプロセスの構成の例を次の図に示します。

図 2-139　単一の HTTP 受付をビジネスプロセスに関連づける場合

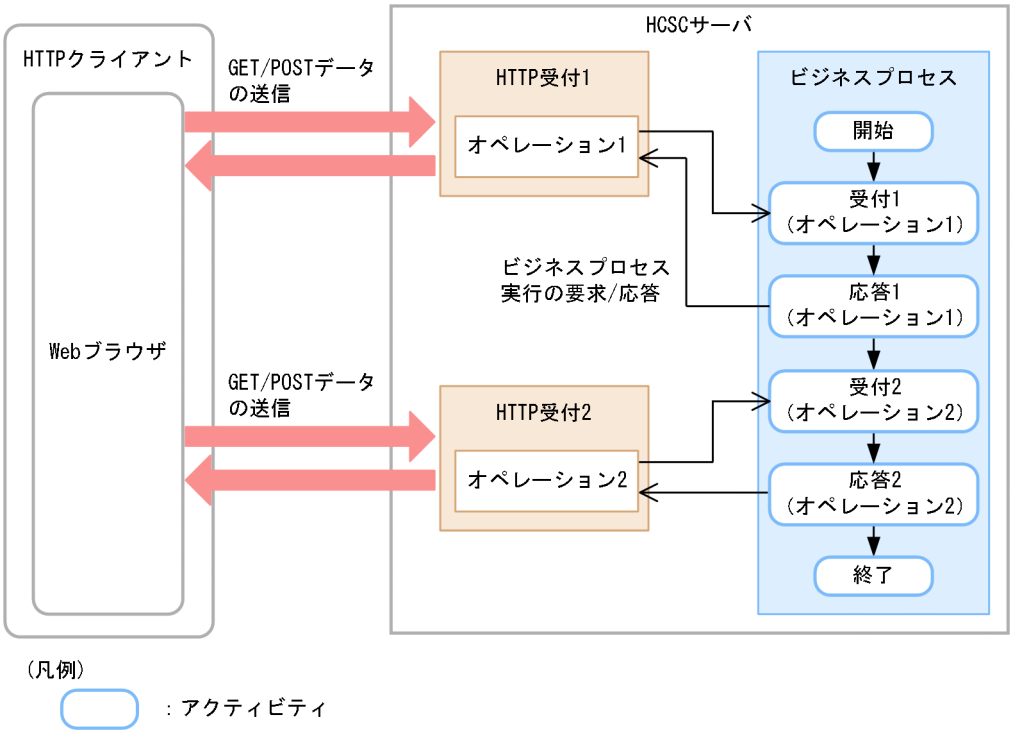


(b)　ビジネスプロセスと複数の HTTP 受付の構成例

ビジネスプロセス内で複数のリクエスト情報を管理したい場合などには，複数の HTTP 受付をビジネスプロセスに関連づけることができます。

複数の HTTP 受付とビジネスプロセスの構成の例を次の図に示します。

図 2-140　複数の HTTP 受付をビジネスプロセスに関連づける場合



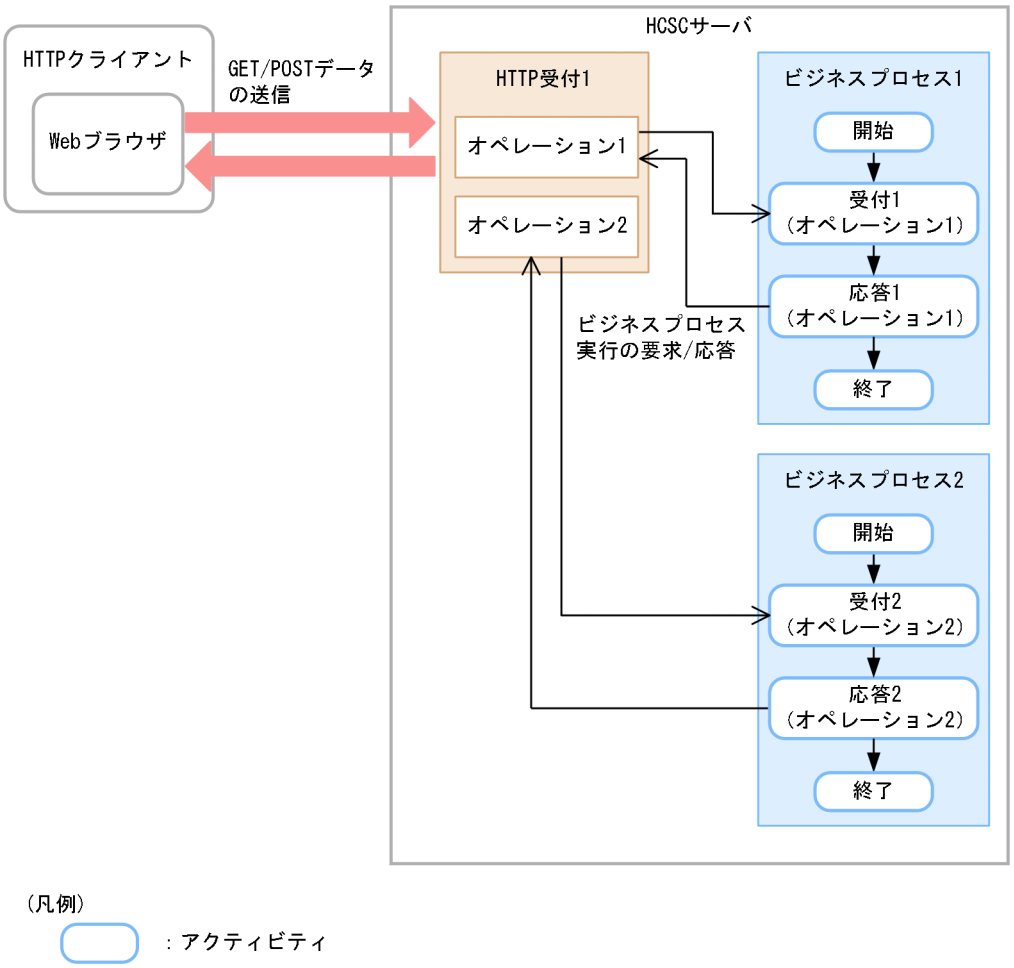
この構成の場合，HTTP リクエストの URL に指定されたコンテキストルートおよびオペレーション名に応じて，ビジネスプロセスに関連づけられた HTTP 受付のオペレーションが呼び出されます。

なお，この構成を選択する場合，永続化するビジネスプロセスとして定義する必要があります。

(c)　複数のビジネスプロセスと単一の HTTP 受付の構成例

複数のビジネスプロセスと単一の HTTP 受付の構成の例を次の図に示します。

図 2-141　複数のビジネスプロセスに単一の HTTP 受付を関連づける場合

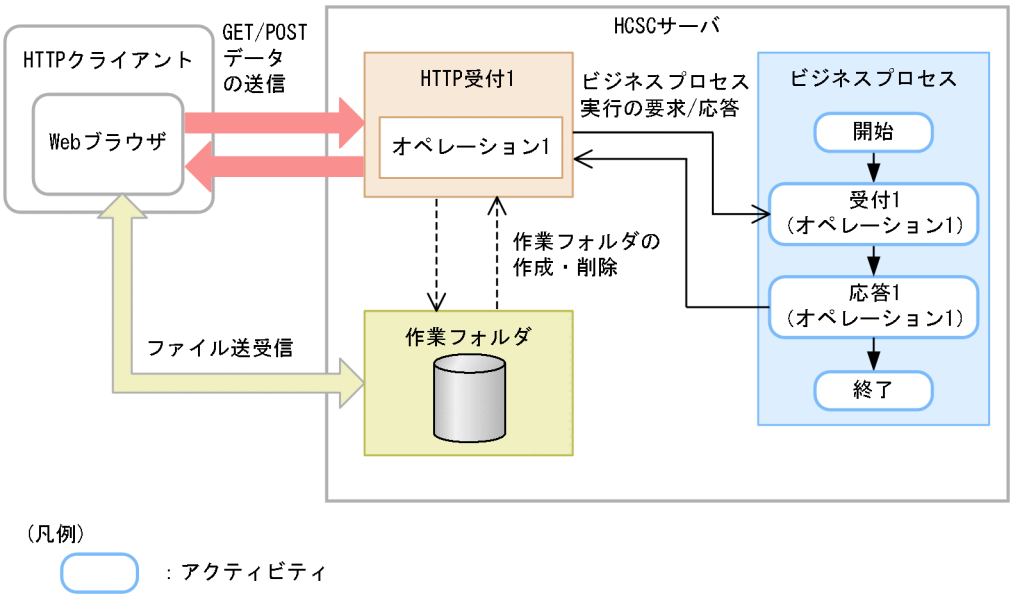


(2)　ファイルのデータを送受信する場合

(a)　ビジネスプロセスと単一の HTTP 受付（単一オペレーション）の構成例

単一のオペレーションを持つ単一の HTTP 受付とビジネスプロセスの構成の例を次の図に示します。

図 2-142　単一の HTTP 受付（単一オペレーション）をビジネスプロセスに関連づける場合

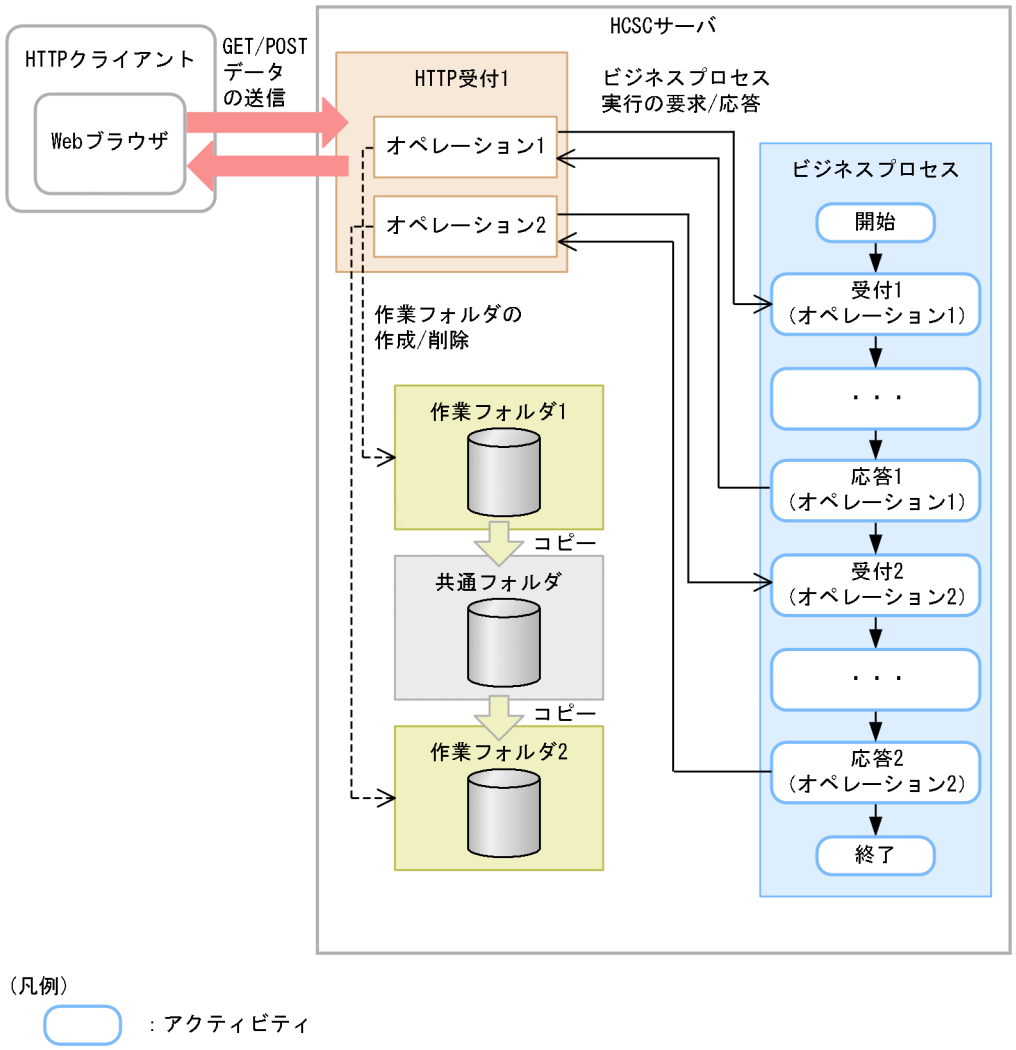


HTTP 受付は HTTP クライアントから HTTP リクエストを受け付けると，HCSC サーバ内にファイルのデータを中間ファイルとして処理するための作業フォルダを作成します。作業フォルダは HTTP レスポンスの処理が完了したあと自動で削除されます。ビジネスプロセスの処理中や HTTP 受付の処理中にエラーが発生した場合は，ファイルは削除されません。

(b)　ビジネスプロセスと単一の HTTP 受付（複数オペレーション）の構成例

複数のオペレーションを持つ単一の HTTP 受付とビジネスプロセスの構成の例を次の図に示します。

図 2-143　単一の HTTP 受付（複数オペレーション）をビジネスプロセスに関連づける場合



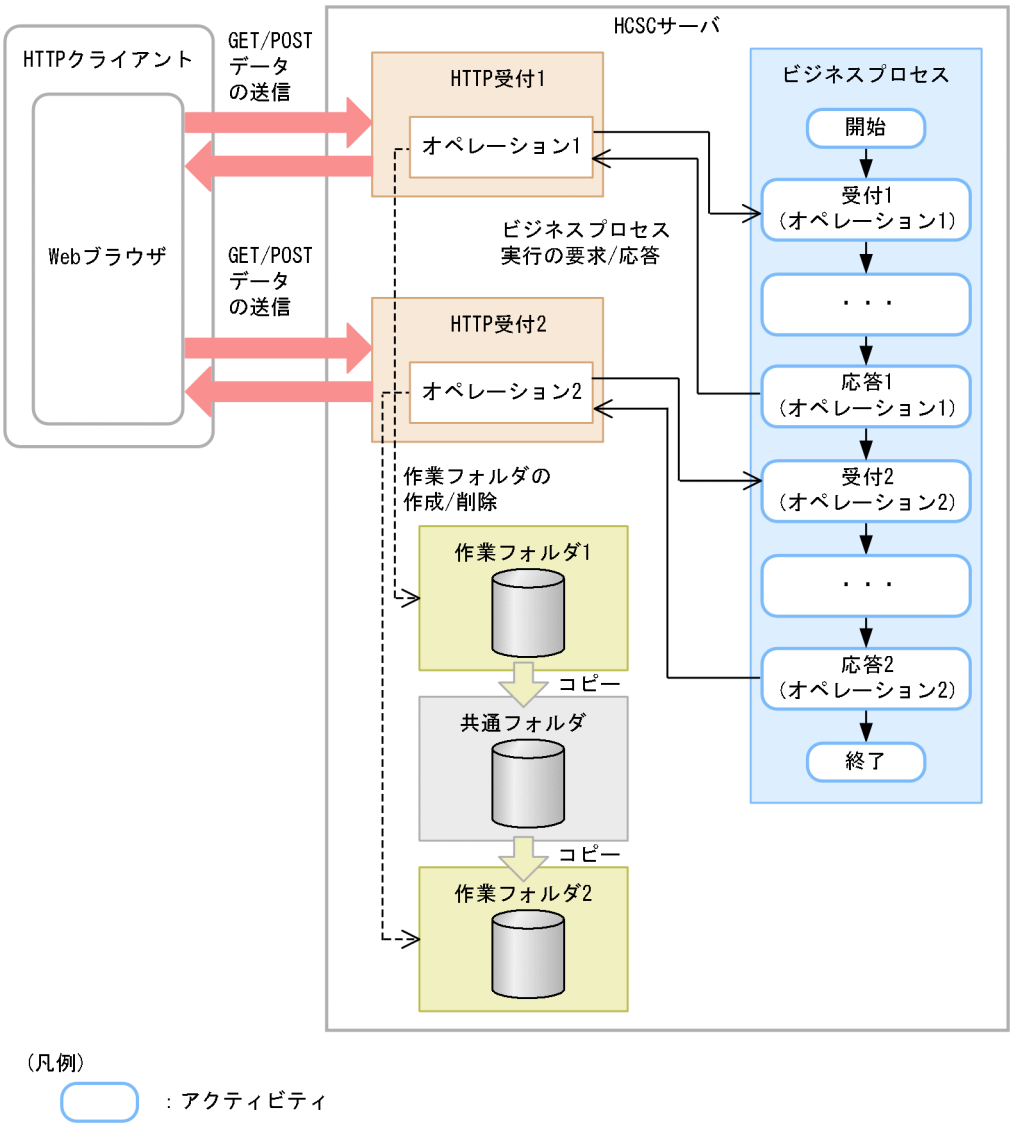
この構成例の場合，作成される作業フォルダはオペレーション（HTTP リクエスト）ごとに異なります。1 回目のオペレーションで作業フォルダに生成されたファイルを 2 回目のオペレーションの処理で参照したいときは，ファイル操作アダプタを使用して作業フォルダ・共通フォルダ間でファイルをコピーする必要があります。

ファイル操作アダプタについては，「8.7　ファイル操作アダプタの機能」を参照してください。共通フォルダについては，「8.8.2　共通フォルダを利用したファイルアクセス」を参照してください。

(c)　ビジネスプロセスと複数の HTTP 受付の構成例

複数の HTTP 受付とビジネスプロセスの構成の例を次の図に示します。

図 2-144　複数の HTTP 受付をビジネスプロセスに関連づける場合



この構成例の場合，複数のオペレーションを持つ単一の HTTP 受付の構成の場合と同様に，作成される作業フォルダはオペレーション（HTTP リクエスト）ごとに異なります。このため，各 HTTP リクエストで同一のファイルを共有したい場合は，ファイル操作アダプタを使用して作業フォルダ・共通フォルダ間でファイルをコピーする必要があります。

2.15.4　HTTP 受付のリクエスト処理

ここでは，HTTP 受付を実行するときに HTTP リクエストがどのように処理されるかについて説明しています。

HTTP クライアントが送信する HTTP リクエストは，次のように分割して HTTP 受付内で処理されます。

リクエストライン

HTTP リクエストメッセージの先頭行に位置し，HTTP メソッド，リクエスト URL，および HTTP バージョンが格納されます。

HTTP リクエストヘッダ

HTTP クライアントで受信する言語，認証情報などの情報が HTTP ヘッダフィールドで指定されます。

HTTP リクエストボディ

POST メソッドの場合に送信するメッセージボディが格納されます。

HTTP 受付では，電文フォーマットと定義ファイルを使用して HTTP リクエストのリクエストライン，HTTP リクエストヘッダ，および HTTP リクエストボディを処理できます。

なお，ここで説明する電文の例では，名前空間接頭辞を省略しています。

(1) リクエストラインの処理

HTTP 受付のリクエストラインの処理について説明します。

(a) コンテキストルート

URL のコンテキストルートは，HTTP 受付定義ファイルの urecp-http.context-root プロパティで指定できます。

urecp-http.context-root プロパティの指定がない場合，ユーザ定義受付定義画面で設定する HTTP 受付の受付 ID がコンテキストルートとして使われます。HTTP 受付定義ファイルの詳細は，マニュアル「サービスプラットフォーム リファレンス」の「3.11.3　HTTP 受付定義ファイル」を参照してください。

HTTP リクエストの送信時，URL は次の形式で指定します。

- http://<ホスト名>:<ポート番号>※/<コンテキストルート>/<HTTP 受付のオペレーション名>
- https://<ホスト名>:<ポート番号>※/<コンテキストルート>/<HTTP 受付のオペレーション名>

注※

Web サーバの URL です。

ただし，ユーザ定義受付（呼出先選択）として定義した HTTP 受付の場合は，オペレーション名は HTTP メソッドから取得するため，コンテキストルートのあとに指定する必要はありません。

URL の指定例を次に示します。

GET メソッドの場合

urecp-http.context-root プロパティが「Test001」のとき，URL は次の形式で指定します。

```
http://<ホスト名またはIPアドレス>/Test001/get?type=A&size=7
```

「get」はオペレーション名，「type=A&size=7」はクエリ文字列を示します。

POST メソッドの場合

urecp-http.context-root プロパティが「Test002」のとき，URL は次の形式で指定します。

```
http://<ホスト名またはIPアドレス>/Test002/post
```

「post」はオペレーション名を示します。

ユーザ定義受付（呼出先選択）として定義した HTTP 受付の場合

ユーザ定義受付（呼出先選択）として定義した HTTP 受付で，HTTP 受付定義ファイルのコンテキストルートに「Test001」を指定したとき，URL は次の形式で指定します。

```
http://<ホスト名またはIPアドレス>/Test001?type=A&size=7
```

「Test001」はコンテキストルート，「type=A&size=7」はクエリ文字列を示します。

オペレーション名とは，ユーザ定義受付定義画面で設定する HTTP 受付のオペレーションの名称です。オペレーション名はビジネスプロセスを呼び出すときに使用されます。

なお，オペレーション名のあとに「/」が指定された場合，「/」の後ろに文字列があるときは，「/」の後ろの部分もパスとして扱われます。例えば，URL に「http://localhost/contextroot/get/abc123?type=A&size=7」が指定された場合，「/contextroot/get/abc123」がロケーション情報（<location>）にパスとして格納されます。

注意事項

コンテキストルートは，Component Container の作業ディレクトリの中でディレクトリ名として使用されます。そのため，作業ディレクトリ全体のパス長が OS の上限に達しないようにコンテキストルートを指定する必要があります。詳細は，「2.15.15　HTTP 受付を使用するときの注意」を参照してください。

(b)　ロケーション情報とクエリ文字列部

リクエストラインで指定された URL は，次の要素に分割して要求電文（ヘッダ）に格納されます。

- ロケーション情報（<location>）※1
- クエリ文字列部（<query>）※2

例えば，リクエストラインが「http://localhost/httprecp/get?type=A&size=7」の場合，要求電文には次のように変換されます。

```
<url>
  <location>/httprecp/get</location>
  <query>type=A&amp;size=7</query>
</url>
```

注※1

ロケーション情報とは，コンテキストパスから拡張パスまでのパスです。

ユーザ定義受付（呼出先選択）として定義したHTTP受付の場合，コンテキストルートのあとの「/」に続く文字は，URLの分析中に無視されますが，<location>タグに格納されます。

注※2

クエリ文字列に複数の「?」が存在する場合，最初の「?」がクエリ文字列の区切り文字として使用されます。

(2) HTTPリクエストヘッダの処理

HTTP受付で受け付けられたHTTPヘッダ部分のすべての情報は，any型で定義されたhttp-header要素に格納して，ビジネスプロセスに渡されます。

HTTPヘッダ部分の変換例を次に示します。

- 変換前のHTTPリクエストヘッダ

```
Accept-Language: ja
Host: aaa.bbb.ccc.ddd
Connection: Keep-Alive
Authorization: Basic aG9nZTpmdWdh
```

- 変換後の要求電文（ヘッダ）

```
<http-header>

<content-type>application/x-www-form-urlencoded; charset=utf-8</content-type>
<Accept-Language> ja</Accept-Language>
<Host> aaa.bbb.ccc.ddd </Host>
<Connection> Keep-Alive</Connection>
<Authorization> Basic aG9nZTpmdWdh</Authorization>

</http-header>
```

HTTP受付で変換されたHTTPヘッダ部分のXML要素は，SOAPアダプタのHTTPヘッダ部とany種別同士でマッピングできます。

(a) Authorizationヘッダの変換

ユーザの認証情報を格納するHTTPリクエストヘッダのAuthorizationヘッダフィールドは，Base64形式にエンコーディングされます。このため，HTTP受付は，Authorizationヘッダフィールドを Base64形式からプレーン形式にデコードし，要求電文のXML要素に設定します。

注意事項

- HTTPヘッダ部分にユーザ名とパスワード情報の両方が指定された場合，HTTP受付では，ユーザ名だけが設定されます。
- HTTP受付ではベーシック認証だけをサポートしています。ほかの認証タイプを指定した場合，ユーザ名は設定されません。

- HTTP ヘッダ部分の Authorization ヘッダフィールドを指定して値を指定しなかった場合，要求電文として Authorization の空要素だけ生成され，auth 要素および username 要素は作成されません。
- HTTP ヘッダ部分の Authorization ヘッダフィールドでパスワードを指定してユーザ名を指定しなかった場合，auth 要素および username 要素が作成されますが，username 要素の値は空となります。

(b) Content-Type ヘッダの変換

受信した Content-Type ヘッダの値のメディアタイプを content-type 要素として要求電文（ヘッダ）に格納します。このとき，charset 属性がある場合は，content-type 要素の charset 属性として格納します。

Content-Type ヘッダがない場合は，content-type 要素は生成しません。また，Content-Type ヘッダのメディアタイプの指定だけがあって，charset 属性がない場合は，属性を生成しません。

Content-Type ヘッダの charset 属性は，クエリまたはフォームデータのテキストデータの文字コードとして使用します。Content-Type ヘッダの charset 属性が存在しない場合，または Content-Type ヘッダがリクエストに存在しない場合は，HTTP 受付定義ファイルの httprecp.http.charset プロパティの値によって処理されます。ただし，この場合も，charset 属性がないため，content-type 要素の charset 属性を生成しません。

注意事項

Content-Type ヘッダの content-type 要素が存在しない，または content-type 要素の charset 属性が生成されていないデータを受信した場合に，テキストデータの文字コードを確認する必要があるときは，httprecp.http.charset プロパティ値の文字コードで判断してください。

(3) HTTP リクエストボディの処理

HTTP リクエストのパート区分およびデータ形式は，HTTP リクエストヘッダの指定に応じて次のように HTTP 受付で処理されます。

- HTTP リクエストのパート区分

  HTTP リクエストヘッダの Content-Type ヘッダが multipart/form-data の場合は，マルチパート型として扱われます。それ以外の場合は単一パート型（テキストデータまたはバイナリデータ）として扱われます。なお，HTTP 受付で処理できるマルチパート型の上限数は 1,024 です。

- HTTP リクエストのデータ形式

  各パート内の Content-Disposition ヘッダの filename 属性が存在する場合はファイルのデータがアップロードされたものとして扱います。filename 属性がない場合はテキストデータまたはバイナリデータとして扱います。

HTTP 受付で扱うデータ形式ごとに，HTTP リクエストボディの処理について説明します。

(a) テキストデータとして扱う場合

HTTP 受付は，クエリ，フォームデータ，マルチパート型の形式で Content-Disposition ヘッダの filename 属性がないパート部分をテキストデータとして扱います。テキストデータとして扱う場合，XML 形式の要求電文（ボディ）に格納されたあと，ボディ変数としてビジネスプロセスに渡されます。

HTTP リクエストのデータ形式がテキストデータだけの場合は，メッセージボディをビジネスプロセスに渡すときのモードとして，次のどちらかを選択できます。

標準モード

標準モードでは，メッセージボディは要求電文の要素に変換されてビジネスプロセスに渡されます。メッセージボディを格納する XML 形式の要求電文フォーマット（ボディ変数用（詳細））は任意にカスタマイズできます。

パススルーモード

パススルーモードでは，メッセージボディ中で msg キーに対応する値がそのままビジネスプロセスに渡されます。XML 形式の要求電文フォーマットはメッセージボディの形式に応じてユーザが作成する必要があります。msg キーに対応する値が null，または空文字の場合，ビジネスプロセスに空文字の要求電文を渡します。

パススルーモードは，ビジネスプロセスをテストする場合，SOAP 受付のダミーとして HTTP 受付を使用するときなどに使用します。

なお，HTTP リクエストがマルチパート型の場合，パススルーモードを指定しても無効になります。

メッセージボディをビジネスプロセスに渡すときのモードは，HTTP 受付定義ファイルの httprecp.switchover.pass-through.mode プロパティで選択できます。また，httprecp.telegram-auto-convert プロパティで true を指定している場合で content-type HTTP ヘッダが application/xml，または application/json のときもパススルーモードが適用されます。HTTP 受付定義ファイルの詳細は，マニュアル「サービスプラットフォーム リファレンス」の「3.11.3　HTTP 受付定義ファイル」を参照してください。

標準モードとパススルーモードの変換例を次の表に示します。

表 2-58　標準モードとパススルーモードの変換例

モード	説明	HTTP リクエスト（メッセージボディ）	変換後の要求電文（ボディ）
標準モード	メッセージボディは「キー名=値」から「<キー名>値</キー名>」の形式に変換されます。※1	product1=uCSP&product2=HiRDB%20Single%20Server	<HTTPBody> <product1>uCSP</product1> <product2>HiRDB Single Server</product2> </HTTPBody>
パススルーモード	メッセージボディ部分は「msg（固定）=値」形式で指定します。※2	msg=%3cHTTPBody%3e%3cproduct1%3euCSP%3c%2fproduct1%3e%3cproduct2%3eHiRDB%20Single%20	<HTTPBody> <product1>uCSP</product1> <product2>HiRDB Single Server</product2>

モード	説明	HTTP リクエスト（メッセージボディ）	変換後の要求電文（ボディ）
パススルーモード	値部分は，メッセージボディのまま XML 形式の要求電文（ボディ）に設定されます。	Server%3c%2fproduct2%3e%3c%2fHTTPBody%3e	</HTTPBody>

注※1

値が半角スペース 1 文字だけの場合，空文字として処理します。

注※2

HTTP 受付定義ファイルの httprecp.pass-through.parameter-use プロパティで，msg キーの指定を省略するかどうかを設定できます。

msg キーを複数指定した場合の動作は保証されません。

標準モードの場合，HTTP リクエストは要求電文の変換時に HTTP 受付でサニタイズされてビジネスプロセスに渡されます。

パススルーモードの場合，HTTP リクエストはサニタイズされません。このため，あらかじめユーザ自身でサニタイズする必要があります。

(b) ファイルのデータとして扱う場合

HTTP リクエストのパート区分がマルチパート型で，かつパート内の Content-Disposition ヘッダの filename 属性が存在するパートのデータは，HTTP 受付で各パートに分割され，パートごとに中間ファイルとして作業フォルダに格納されます。

なお，ファイルのデータとして処理した場合，各パートのヘッダ情報は，次の表に示すように要求電文（ヘッダ）の file 要素にファイルのメタ情報として格納されます。

表 2-59 要求電文（ヘッダ）に格納される各パートのヘッダ情報

HTTP リクエストのヘッダ	ヘッダの属性	任意/必須	説明
Content-Type	－	任意	各パートに指定されているファイルのメディアタイプです。 このヘッダは省略できます。
	charset	任意	各パートに指定されているファイルの文字コードです。 この属性は省略できます。
Content-Disposition※1	－	必須	Disposition タイプを示すヘッダです。 必ず「form-data」を指定します。※2
	name	必須	各パートの名称を示す属性です。 この属性は必ず指定します。※3
	filename	任意	アップロードするファイルのデータのファイル名を示す属性です。 この属性は省略できます。この属性を省略した場合，テキストデータ扱いになります。

（凡例）

－：該当する項目はありません。

注※1

Content-Disposition ヘッダに size 属性，modification-date 属性など，この表に示した属性以外の属性が指定されている場合，それらの属性値は無視されます。

注※2

マルチパート型としてファイルのデータをアップロードする場合，Content-Disposition ヘッダには必ず form-data を指定してください。form-data 以外の場合，HTTP 受付では処理の対象となりません。

注※3

name 属性を省略した場合，省略したパートのデータは，HTTP 受付では処理の対象となりません。

注意事項

- Content-Disposition ヘッダの name 属性および filename 属性にマルチバイト文字を使用した場合，動作を保証しません。
- Content-Disposition ヘッダの属性に「filename」で始まる任意の属性を指定した場合，および filename 属性の値に「"」が含まれるファイル名（パス部分は除く）を指定した場合は，動作を保証しません。

なお，複数のファイルのデータをアップロードする場合，これらのメタ情報はファイルごとに要求電文の file 要素に繰り返して格納されます。

(c) バイナリデータとして扱う場合

HTTP リクエストボディにバイナリデータを設定することで，バイナリ形式の要求電文をビジネスプロセスに送信できます。CSV 形式のデータをビジネスプロセスで処理したい場合などに活用できます。

HTTP 受付でバイナリデータとして扱うには，次の条件をすべて満たす必要があります。

- パススルーモードを使用していること（HTTP 受付定義ファイルに httprecp.switchover.pass-through.mode=true を指定していること）
- パススルーモードで msg キーを省略する設定にしていること（HTTP 受付定義ファイルに httprecp.pass-through.parameter-use=false を指定していること，および httprecp.telegram-auto-convert=false を指定していること）
- HTTP 受付の要求電文の電文フォーマットにバイナリ形式（拡張子：fdx），または任意形式（any 形式）を指定していること
- HTTP リクエストボディのパート区分が単一パートであること

バイナリ形式の要求電文の文字エンコーディングは，ビジネスプロセスのデータ変換時に行われます。

2.15.5　HTTP 受付のレスポンス処理

ここでは，HTTP 受付に返されたビジネスプロセスからの応答電文が，どのように HTTP レスポンスへと処理されるかについて説明しています。

HTTP クライアントに応答を返す HTTP レスポンスは次のように分割して HTTP 受付内で処理されます。

ステータスライン
　HTTP バージョン，ステータスコード，および理由フレーズが格納されます。

HTTP レスポンスヘッダ
　HTTP レスポンスの Content-Length ヘッダや Content-Type ヘッダなどの情報が格納されます。

HTTP レスポンスボディ
　HTTP クライアントに応答を返すテキストデータまたはバイナリデータが格納されます。

(1)　ステータスラインの処理

HTTP レスポンスのステータスラインは，次の形式で出力されます。

[HTTPバージョン※1][空白][ステータスコード※2][空白][理由フレーズ]

注※1
　HTTP バージョンは「HTTP/1.1」で固定です。

注※2
　応答電文（ヘッダ）の status-code 要素の値がステータスコードに設定されます。ステータスコードの詳細は，「(4) ステータスコード」を参照してください。

(2)　HTTP レスポンスヘッダの処理

HTTP 受付に指定されたすべてのヘッダ情報は HTTP レスポンスヘッダに設定されます。

HTTP レスポンスヘッダの処理について説明します。

(a)　Content-Type と charset

HTTP レスポンスヘッダに使用される Content-Type ヘッダおよび charset 属性は，次の場合によって異なります。

- テキストデータを返す場合
- ファイルのデータを返す場合
- 複数データを返す場合（マルチパート型）
- バイナリデータを返す場合

●テキストデータを返す場合

HTTP レスポンスヘッダに使用される Content-Type ヘッダおよび charset 属性は，応答電文（ヘッダ）の content-type 要素，http-header 要素，HTTP レスポンスヘッダ定義ファイルなどで設定できます。

応答電文フォーマット（ヘッダ変数用）をビジネスプロセスに設定した場合で，Content-Type ヘッダおよび charset 属性を複数指定したときは，次の表に示すように応答電文フォーマット（ヘッダ変数用）の指定が優先されます。

表 2-60　HTTP レスポンスヘッダに使用される Content-Type および charset（応答電文フォーマット（ヘッダ変数用）が設定されている場合）

設定項目	応答電文に content-type 要素が存在する場合※1		応答電文の http-header 要素に content-type 要素が存在する場合		応答電文に content-type 要素および charset 属性が存在しない場合
	content-type 要素に charset 属性が存在するとき	content-type 要素に charset 属性が存在しないとき	http-header 要素に charset 属性が存在するとき	http-header 要素に charset 属性が存在しないとき	
Content-Type ヘッダ	応答電文の content-type 要素の値が有効になります。	応答電文の content-type 要素の値が有効になります。	応答電文の http-header 要素の content-type 要素の値が有効になります。※3	応答電文の http-header 要素の content-type 要素の値が有効になります。※3	「text/xml」が設定されます。
charset 属性	応答電文の content-type 要素の charset 属性の値が有効になります。※2	HTTP 受付定義ファイルの指定値が有効になります。	応答電文の http-header 要素の charset 属性の値が有効になります。※3	HTTP 受付定義ファイルの指定値が有効になります。	HTTP 受付定義ファイルの指定値が有効になります。

注※1

要素が存在していても値が空の場合は，要素が存在しないと判断されます。

注※2

属性が存在していても値が空の場合は，属性が存在しないと判断されます。

注※3

応答電文（ヘッダ）の http-header 要素に複数の content-type 要素が指定された場合，指定のうちどれか 1 つの値が有効になります。ただし，どの値が設定されるかは保証されません。

また，応答電文フォーマット（ヘッダ変数用）をビジネスプロセスに設定していない場合，次の表に示すように，HTTP レスポンスヘッダ定義ファイルの値が有効になります。

表 2-61　HTTP レスポンスヘッダに使用される Content-Type ヘッダおよび charset 属性（応答電文フォーマット（ヘッダ変数用）を設定していない場合）

設定項目	HTTP レスポンスヘッダ定義ファイルが存在している場合			HTTP レスポンスヘッダ定義ファイルが存在しない場合
	HTTP レスポンスヘッダ定義ファイルに content-type キーが存在するとき		HTTP レスポンスヘッダ定義ファイルに content-type キーが存在しないとき	
	content-type キーの指定値に charset 属性が存在する	content-type キーの指定値に charset 属性が存在しない	HTTP レスポンスヘッダ定義ファイルに charset キーが存在しない	
Content-Type ヘッダ	content-type キーの指定値が有効になります。※	content-type キーの指定値が有効になります。※	「text/xml」が設定されます。	「text/xml」が設定されます。
charset 属性	content-type キーの指定値に存在する charset 属性の値が有効になります。※	HTTP 受付定義ファイルの指定値が有効になります。	HTTP 受付定義ファイルの指定値が有効になります。	HTTP 受付定義ファイルの指定値が有効になります。

注※

HTTP レスポンスヘッダ定義ファイルに複数の content-type キーと値のペアを指定した場合，指定のうちどれか 1 つの値が有効になります。ただし，どの値が設定されるかは保証されません。

HTTP レスポンスヘッダ定義ファイルは，HTTP 受付定義ファイルから呼び出される任意のファイルです。HTTP レスポンスヘッダ定義ファイルの詳細は，マニュアル「サービスプラットフォーム リファレンス」の「3.11.1　HTTP レスポンスヘッダ定義ファイル」を参照してください。

●ファイルのデータを返す場合

HTTP レスポンスヘッダに使用される Content-Type ヘッダおよび charset 属性は，次に示すように応答電文（ヘッダ）の file 要素配下の content-type 要素で設定できます。

表 2-62　HTTP レスポンスヘッダに使用される Content-Type ヘッダおよび charset 属性

設定項目	応答電文の file 要素配下に content-type 要素が存在する場合		応答電文の file 要素配下に content-type 要素が存在しない場合
	file 要素配下の content-type 要素に charset 属性が存在するとき	file 要素配下の content-type 要素に charset 属性が存在しないとき	
Content-Type ヘッダ	応答電文の file 要素配下の content-type 要素の値が有効になります。	応答電文の file 要素配下の content-type 要素の値が有効になります。	設定されません。
charset 属性	応答電文の file 要素配下の content-type 要素の charset 属性の値が有効になります。	設定されません。	設定されません。

注

要素が存在していても値が空の場合は，要素が存在しないと判断されます。

属性が存在していても値が空の場合は，属性が存在しないと判断されます。

●複数データを返す場合（マルチパート型）

HTTP クライアントに返すデータが，テキストデータと 1 ファイルのデータ，または 2 ファイルのデータのように複数となる場合は，応答電文（ヘッダ）および HTTP レスポンスヘッダ定義ファイルの指定に関係なく，Content-Type ヘッダのメディアタイプには multipart/form-data が設定されます。各データの Content-Type ヘッダは，マルチパート型のレスポンスボディの各パートのヘッダに指定されます。各データに指定される値は，「●テキストデータを返す場合」，および「●ファイルのデータを返す場合」の説明にそれぞれ従います。

●バイナリデータを返す場合

テキストデータを返す場合と同様に HTTP レスポンスヘッダに使用される Content-Type ヘッダおよび charset 属性は，応答電文（ヘッダ）の content-type 要素，http-header 要素，HTTP レスポンスヘッダ定義ファイルなどで設定できます。

Content-Type ヘッダおよび charset 属性を複数指定した場合も，テキストデータを返すときと同様に，応答電文フォーマット（ヘッダ変数用）の指定が優先されます。そのため，複数指定した場合に有効となる Content-Type ヘッダおよび charset 属性は，「表 2-60」，または「表 2-61」を参照してください。

ただし，次の場合は Content-Type ヘッダに「application/octet-stream」が設定されます。

- 応答電文に content-type 要素および charset 属性が存在しない
- HTTP レスポンスヘッダ定義ファイルに content-type キーが存在しない
- HTTP レスポンスヘッダ定義ファイルが存在しない

(b) Content-Disposition

HTTP レスポンスの Content-Disposition ヘッダは，HTTP レスポンスのパート区分およびデータ数に応じて次の値が設定されます。ファイルをダウンロードしない場合，HTTP レスポンスには Content-Disposition ヘッダが設定されません。

表 2-63 HTTP 受付が設定する Content-Disposition ヘッダの値

パート区分	データ数	設定個所	Content-Disposition ヘッダの値	name 属性	filename 属性
単一パート型	1 ファイル	HTTP レスポンスヘッダ	attachment※	設定されません。	応答電文（ヘッダ）の filename 要素の値
マルチパート型	1 ファイル+テキストデータ	HTTP レスポンスの各パートのヘッダ	form-data	応答電文（ヘッダ）の partID 要素の値	
	複数ファイル				

注※

HTTP 受付定義ファイルの httprecp.response.download.disposition-type プロパティで変更できます。

応答電文（ヘッダ）の http-header 要素内に content-disposition 要素が含まれていた場合，その値は無視されます。なお，HTTP レスポンスヘッダ定義ファイルで Content-Disposition ヘッダが指定されている場合はその値が有効になります。

(c) Content-Length

テキストデータだけを HTTP レスポンスで返す場合，コンテンツ長を表す Content-Length ヘッダを HTTP 受付で生成するかどうかは，HTTP 受付定義ファイルの httprecp.response.generate.content-length プロパティで指定できます。また，コンテンツ長を計算するときに使用する文字コードは，HTTP 受付定義ファイルの httprecp.http.charset プロパティで指定できます。

HTTP 受付定義ファイルの詳細は，マニュアル「サービスプラットフォーム リファレンス」の「3.11.3 HTTP 受付定義ファイル」を参照してください。

HTTP 受付定義ファイルの指定がある場合とない場合の，HTTP クライアントに渡される Content-Length ヘッダの値を次の表に示します。

表 2-64　HTTP クライアントに渡される Content-Length ヘッダ

httprecp.response.generate.content-length プロパティの値	HTTP レスポンスヘッダ定義ファイルの content-length キーまたは応答電文（ヘッダ）の http-header 要素の content-length 要素の指定有無	HTTP クライアントに渡される Content-Length ヘッダの値
true	○	HTTP レスポンス定義ファイルまたは応答電文（ヘッダ）で指定した値は，HTTP 受付が計算したデータ長に上書きされます。
	×	
false	○	HTTP レスポンスヘッダ定義ファイルまたは応答電文（ヘッダ）に指定したデータ長を Content-Length ヘッダに設定します。※
	×	HTTP クライアントに Content-Length ヘッダを渡しません。

（凡例）

○：指定している場合です。

×：指定していない場合です。

注※

HTTP 受付定義ファイルの httprecp.response.generate.content-length プロパティが「false」で，応答電文（ヘッダ）の http-header 要素で複数の content-length 要素が指定された場合，または HTTP レスポンスヘッダ定義ファイルで複数の content-length キーと値のペアが指定された場合，指定のうちどれか 1 つの値が設定されます。ただし，どの値が設定されるかは保証されません。

なお，ファイルを 1 つ以上ダウンロードする場合は，Content-Length ヘッダをクライアントに送信する設定にしていた場合でも無効になります。また，ファイルをダウンロードしない場合でも，応答電文（ヘッダ）の ignore-bodymsg 要素または HTTP 受付定義ファイルの httprecp.response.ignore-bodymsg プロパティで応答電文（ボディ）を無視する設定にしているときは，Content-Length ヘッダが設定されません。

(3) HTTP レスポンスボディの処理

HTTP クライアントに送信するデータ形式ごとに，HTTP レスポンスボディの処理について説明します。

また，レスポンスボディを gzip 形式で圧縮する HTTP レスポンス圧縮機能について説明します。

(a) テキストデータだけの場合

ビジネスプロセスの処理が完了したあと，HTTP 受付には応答電文（ボディ）変数が XML 型の文字列として返り，HTTP レスポンスボディにテキストデータとして設定されます。設定する際，指定された文字コードによってエンコードされます。

システム例外が発生した場合は，エラーメッセージ ID とメッセージの内容が応答電文（ボディ）の要素に設定されます。

テキストデータで返せるデータは応答アクティビティで指定した応答電文（ボディ）の部分だけです。このため，マルチパート型のレスポンスを返す場合，テキストデータが存在するパートは 1 つだけとなります。

注

XML 要素に設定するテキストデータは，HTTP 受付定義ファイルで指定した文字コードごとにエンコードする必要があります。

なお，ビジネスプロセスから受信した応答電文（ボディ）を HTTP 受付が無視する（HTTP レスポンスに設定しない）かどうかを，応答電文（ヘッダ）の ignore-bodymsg 要素または HTTP 受付定義ファイルの httprecp.response.ignore-bodymsg プロパティで設定できます。応答電文（ヘッダ）と HTTP 受付定義ファイルの両方で指定した場合，応答電文（ヘッダ）の指定が優先されます。この要素（またはプロパティ）を指定し，かつダウンロードするファイルのデータが 1 つもない場合は，空の HTTP レスポンスボディを HTTP クライアントに返します。

(b) ファイルのデータの場合

ファイルのデータをダウンロードする場合，応答電文（ヘッダ）の file 要素内でダウンロード対象のファイル名などを指定します。応答電文（ヘッダ）で指定する要素と対応する HTTP レスポンスヘッダを次に示します。

表 2-65 応答電文（ヘッダ）で指定する要素

応答電文（ヘッダ）で指定する要素名	説明	対応する HTTP レスポンスヘッダ（マルチパート型の場合は各パートに対応するヘッダ）	省略した場合の値
partID	マルチパート型の各パートを表すパート識別名です。	Content-Disposition ヘッダの name 属性	local-file-name 要素で指定した中間ファイル名が設定されます。マルチパート型にならない場合は指定しても無視され，name 属性は付与されません。

応答電文（ヘッダ）で指定する要素名	説明	対応する HTTP レスポンスヘッダ（マルチパート型の場合は各パートに対応するヘッダ）	省略した場合の値
file-name	HTTP クライアントにダウンロードする際のダウンロードファイル名です。	Content-Disposition ヘッダの filename 属性	なし（filename 属性が設定されません）。
local-file-name	作業フォルダ内のダウンロード対象の中間ファイル名です。指定した名称のファイルが存在しない場合はエラーになります。	なし	省略できません。
content-type※	ダウンロードファイルのメディアタイプを指定します。	Content-Type ヘッダ	なし（Content-Type ヘッダが設定されません）。
charset	ダウンロードファイルの文字コードを指定します。	charset 属性	なし。

注※

content-type 要素の指定を省略した場合，ダウンロードするファイルの Content-Type ヘッダは HTTP 受付で設定しません。なお，ダウンロードするファイルが複数ある場合，またはファイルとテキストデータのマルチパート型となる場合，HTTP レスポンスヘッダの Content-Type ヘッダは multipart/form-data となり，各ファイルのデータの Content-Type ヘッダはパートごとのヘッダに指定されます。

参考

HTTP レスポンス時にテキストデータを返さないでファイルを 1 つだけダウンロードしたい場合などのために，応答電文（ボディ）で指定されたテキストデータを HTTP 受付が無視するかどうかを設定できます。この設定は，応答電文（ヘッダ）の ignore-bodymsg 要素，または HTTP 受付定義ファイルの httprecp.response.ignore-bodymsg プロパティで変更できます。HTTP レスポンスごとに設定を動的に変更したい場合は応答電文（ヘッダ）で設定します。

応答電文（ヘッダ）と HTTP 受付定義ファイルの両方で指定した場合，応答電文（ヘッダ）の指定が優先されます。

ただし，応答電文（ボディ）を無視する設定の場合でも，ビジネスプロセスの応答アクティビティのボディ割当変数には，任意の変数を設定しておく必要があります。このため，サービスプラットフォームが提供している応答電文フォーマット（ボディ変数用）のダミーフォーマット（urecp_http_dummy_body_response.xsd）をボディ変数用のダミーとして使用することを推奨します。

(c) バイナリデータの場合

HTTP 受付の応答電文の電文フォーマットにバイナリ形式または任意形式（any 形式）を指定することで，HTTP レスポンスボディにバイナリデータを設定してクライアントに応答を返すことができます。

HTTP レスポンスボディにバイナリデータが設定されるのは，次の条件をすべて満たす場合だけです。

- ビジネスプロセスから送信された応答電文（ボディ）を HTTP レスポンスボディに利用する設定であること（HTTP 受付定義ファイルに httprecp.response.ignore-bodymsg=false を指定していること）
- HTTP 受付の応答電文の電文フォーマットにバイナリ形式（拡張子：fdx）または任意形式（any 形式）を指定していること
- 応答電文（ヘッダ）に file 要素が存在しないこと

注意事項

- ビジネスプロセスの応答電文（ボディ）を無視する設定（HTTP 受付定義ファイルに httprecp.response.ignore-bodymsg=true を指定している）の場合，HTTP レスポンスボディにはバイナリデータが設定されません。
- HTTP 受付の応答電文の電文フォーマットに XML 形式（拡張子：xsd）を指定している場合はテキストデータとして処理されます。
- 応答電文（ヘッダ）に file 要素が存在する場合，バイナリデータを HTTP レスポンスボディへ設定することはできません。
- HTTP レスポンスボディに設定するバイナリデータは文字エンコーディングが行われません。

(d)　HTTP レスポンス圧縮機能

HTTP 受付のレスポンス処理では，アプリケーションサーバの HTTP レスポンス圧縮機能を使用して，レスポンスボディを gzip 形式で圧縮できます。

HTTP レスポンス圧縮機能については，マニュアル「アプリケーションサーバ 機能解説 互換編」の「9.2 HTTP レスポンス圧縮機能」を参照してください。

HTTP レスポンス圧縮機能を使用するには，HTTP 受付定義ファイルに HTTP レスポンス圧縮規則を指定する必要があります。HTTP 受付定義ファイルで次のプロパティを設定してください。

- httprecp.response.compression.url-pattern<URL パターン識別番号>
- httprecp.response.compression.rule<圧縮規則識別番号>.url-mapping<URL マッピング識別番号>
- httprecp.response.compression.rule<圧縮規則識別番号>.condition<圧縮条件識別番号>

HTTP クライアントから HTTP リクエストを送信する場合は，gzip 形式で圧縮された HTTP レスポンスを受信できることを示す HTTP ヘッダを付けてください。

HTTP 受付定義ファイルの設定方法については，マニュアル「サービスプラットフォーム リファレンス」の「3.11.3　HTTP 受付定義ファイル」を参照してください。

(4)　ステータスコード

HTTP リクエストおよび HTTP レスポンスのステータスコードについて説明します。

(a) 応答時のステータスコード

応答時のステータスコードは，応答のタイミングおよび種類によって設定される値が異なります。ステータスコードの設定値と応答のタイミングを次の表に示します。

表 2-66 設定されるステータスコード

タイミング	応答の種類	応答電文（ヘッダ）での status-code 要素の指定	
		あり	なし
ビジネスプロセスを呼び出す前	HTTP 受付の検証でエラーが発生した場合	－	エラーの状況に応じて HTTP 受付がステータスコードを設定します。
ビジネスプロセスを呼び出したあと	正常終了の場合	応答電文（ヘッダ）の status-code 要素で設定したステータスコードが使用されます。※1	200※2
	フォルトの場合	応答電文（ヘッダ）の status-code 要素で設定したステータスコードが使用されます。※1	200※3
	システム例外の場合	－	HTTP 受付定義ファイルの httprecp.system-exception.status-code プロパティの値が有効になります。※4

（凡例）

－：該当しません。

注※1

ステータスコードが正常終了の場合やエラーの場合であるかどうかは検証されません。このため，応答電文（ヘッダ）にエラーの範囲（例：400～599）のステータスコードを設定した場合，HTTP クライアントにはステータスコードがそのまま返されます。

注※2

ステータスコードを設定していない場合のデフォルト値です。

注※3

フォルトの場合でステータスコードを設定していないときは，正常終了の場合とみなされて，デフォルト値が設定されます。

注※4

HTTP 受付定義ファイルの指定がない場合は，デフォルト値の 500 が設定されます。

(b) エラー発生時のステータスコード

HTTP リクエストおよび HTTP レスポンスの処理でエラーが発生した場合の，メッセージ ID とステータスコードの対応を次の表に示します。

表 2-67　リクエスト処理でエラーが発生した場合のステータスコード一覧

メッセージ ID	エラー発生時の状況	ステータスコード
KDEC80424	内部エラーが発生した場合（HTTP ヘッダまたは HTTP ボディの XML 形式の変換時にエラーが発生した）	500
KDEC80438	HTTP メソッドが GET，POST，HEAD，PUT，DELETE，または OPTIONS 以外の場合	405
KDEC80439	URL で指定されたオペレーション名が，HTTP 受付で指定されたオペレーション名と一致しない場合	404
KDEC80440	クエリ文字列，HTTP ボディ，または HTTP ヘッダで無効な文字が指定された場合	400
KDEC80441	パススルーモードの場合に，リクエストパラメタで msg キーが指定されていない，または大文字で指定された場合	400
KDEC80442	HTTP リクエストヘッダまたは HTTP 受付定義ファイルで指定したエンコードに従った URL のデコーディング中にエラーが発生した場合	500
KDEC80451	次のどちらかの原因で一意となるリクエスト ID を生成できなかった場合 • リクエスト ID を生成するための時刻，日付，HTTP クライアントの IP アドレスなどのパラメタの取得に失敗した • 同一時刻に同一の IP アドレスから大量のリクエストを受信して使用できる ID がなくなった	500
KDEC80452	HCSC サーバランタイム定義ファイルで指定された作業フォルダルートのディレクトリが存在しない場合	500
KDEC80453	作業フォルダの作成に失敗した場合	500
KDEC80454	マルチパート型 HTTP リクエストのデータの受信・解析に失敗した場合	400
KDEC80456	HTTP メソッドが，ユーザ定義受付（呼出先選択）として定義した HTTP 受付のオペレーション名と一致しない場合	404
KDEC80457	中間ファイルの出力中にエラーが発生した場合	500
KDEC80464	HTTP 受付ルーティング定義ファイルの path 要素の pattern 属性に指定した URL パターン以外を使用してリクエストが実行された場合	404
	HTTP 受付ルーティング定義ファイルの HTTP メソッド要素に指定した HTTP メソッド以外を使用してリクエストが実行された場合	405

表 2-68　レスポンス処理でエラーが発生した場合のステータスコード一覧

メッセージ ID	エラー発生時の状況	ステータスコード
KDEC80424	内部エラーが発生した場合（HTTP レスポンスボディのストリーム取得に失敗した，または HTTP レスポンスボディへの値の書き込み中に例外が発生した）	500
KDEC80443	受付時に，HTTP ヘッダまたは HTTP ボディの解析でエラーが発生した場合	500
KDEC80444	応答電文のエンコーディング時にエラーが発生した場合	500

メッセージ ID	エラー発生時の状況	ステータスコード
KDEC80445	ビジネスプロセスから返されたステータスコードが，3 桁より大きいか，または小さい桁数だった場合	500
KDEC80458	中間ファイルの読み込みに失敗した場合	500
KDEC80459	作業フォルダの削除に失敗した場合	200

2.15.6　HTTP 受付のファイル管理

ここでは，HTTP 受付でファイルのデータを送受信するときのファイル管理について説明します。

HTTP 受付はファイルのデータを受信すると，受信したデータを作業フォルダ内に中間ファイルとして出力します。レスポンス時にファイルをダウンロードする場合は，ユーザが応答電文（ヘッダ）で指定した作業フォルダの中間ファイルを各クライアントに送信します。

(1)　作業フォルダの作成と削除

作業フォルダは，HTTP 受付が送受信するデータを一時的に格納するためのフォルダです。

(a)　作業フォルダの作成

HTTP 受付は，HTTP クライアントから HTTP リクエストを受け付けると，リクエストごとに作業フォルダを作成します。

作業フォルダを作成するディレクトリ（作業フォルダルート）は HCSC サーバランタイム定義ファイルの work-folder プロパティで設定できます。詳細は，マニュアル「サービスプラットフォーム リファレンス」の「6.5.6　HCSC サーバランタイム定義ファイル」を参照してください。

また，HTTP 受付は，HTTP クライアントからのリクエストを受け付けると，リクエストごとにリクエスト ID を生成します。ビジネスプロセスの処理は，生成したリクエスト ID を指定して実行されます。

リクエスト ID は次の形式で生成されます。

<受付 ID><HTTP クライアントの IP アドレス><日付><時刻><追番><分割キー><HCSC サーバ名>

リクエスト ID のそれぞれの項目について次の表に示します。

表 2-69　リクエスト ID の項目

項目	桁数	内容
<受付 ID>	8 桁	HTTP 受付の受付 ID です。受付 ID が 8 桁に満たない場合は，受付 ID の右側に足りない分の「-」が設定されます。
<HTTP クライアントの IP アドレス>	IPv4 の場合：12 桁	要求元の HTTP クライアントの IP アドレス（IPv4 の場合は 10 進数表記形式，IPv6 の場合は 16 進数表記形式）です。IPv4 アド

項目	桁数	内容
<HTTP クライアントの IP アドレス>	IPv6 の場合：32 桁	レスのそれぞれのバイトが 3 桁に満たない場合は，それぞれのバイトの左側に足りない分の「0」が設定されます。IPv6 アドレスはそれぞれのバイトが 4 桁に満たない場合は，それぞれのバイトの左側に足りない分の「0」が設定されます。英字は小文字で表記されます。
<日付>	8 桁	HTTP 受付がリクエストを受け付けた日付が yyyymmdd の形式で設定されます。
<時刻>	9 桁	HTTP 受付がリクエストを受け付けた時刻が hhmmssSSS の形式で設定されます。
<追番>	3 桁	フォルダ名を一意にするための整数値が「000」からの連番で設定されます。<日付><時刻><追番>がすでにあるリクエスト ID と重複する場合は追番を 1 つ繰り上げます。 「001」から「999」までのどの整数値を設定しても重複する場合，<日付>および<時刻>を変更して追番には「000」を設定します。
<分割キー>	1 桁	HCSC サーバ名が付加されていることを示す分割キーです。半角スラッシュ（/），半角アットマーク（@）または半角ピリオドが使用されます。 HCSC サーバランタイム定義ファイルで変更可能です。 詳細は，マニュアル「サービスプラットフォーム リファレンス」の「6.5.6　HCSC サーバランタイム定義ファイル」を参照してください。
<HCSC サーバ名>	最大 8 桁	HTTP 受付が配備されている HCSC サーバの名称です。HCSC サーバ名が 8 桁に満たない場合，不足分は補われません。

なお，リクエスト ID から<受付 ID>，<分割キー>，<HCSC サーバ名>を除いた文字列が作業フォルダ名となります。

参考

デフォルトの状態では，マルチパート型以外の HTTP リクエストの場合は，作業フォルダが作成されません。ただし，マルチパート型以外の HTTP リクエストの場合でも，HTTP レスポンス時にファイルをダウンロードするときのために，HTTP リクエスト時にマルチパート型かどうかに関係なく作業フォルダを必ず作成するように設定できます。この設定は，HTTP 受付定義ファイルの httprecp.switchover.file-transfer.mode プロパティで有効にできます。

(b)　作業フォルダの削除

作業フォルダおよび作業フォルダ内の中間ファイルは，次の条件をすべて満たす場合に自動で削除されます。

- HTTP 受付定義ファイルの httprecp.work-dir.auto-delete プロパティの値が true である
- HTTP 受付が呼び出したビジネスプロセスから受け取った応答電文の status-code 要素の値が 200，または status-code 要素の指定を省略している

プロセスが異常終了した場合，および cscpireexec コマンド（プロセスインスタンスの再実行）などでビジネスプロセスを再実行した場合は，作業フォルダおよび作業フォルダ内の中間ファイルは自動で削除されません。なお，作業フォルダが自動で削除されない場合，次の方法で作業フォルダを削除する必要があります。

- HTTP 受付定義ファイルの httprecp.work-dir.auto-delete プロパティの値が false，かつビジネスプロセスが非同期型のとき

  ビジネスプロセスからファイル操作アダプタを呼び出して，作業フォルダおよび中間ファイルを削除してください。または，cscfswls コマンドで作業フォルダを参照し，削除対象かどうか確認したあと，cscfswrm コマンドで作業フォルダを削除してください。

- 上記以外のとき

  cscfswls コマンドで作業フォルダを参照し，削除対象かどうか確認したあと，cscfswrm コマンドで作業フォルダを削除してください。

(2) 一時ファイルの作成と削除

HTTP リクエスト時にファイルのデータを送信する場合，メモリの圧迫を回避するため，処理用に一時ファイルが作成されます。また，一時ファイルを保存するフォルダとして，HTTP 受付の起動時に，次の場所に一時ファイル保存フォルダが必ず作成されます。

<作業フォルダルート>¥<HCSC サーバ名>¥<受付 ID>¥<00000000000000000000000000000000>

注意事項

一時ファイルはリクエスト処理の最後に自動で削除されます。ただし，次の場合，一時ファイルは削除されません。

- 一時ファイルが開かれたままの場合
- J2EE サーバが正常に停止されなかった場合
- システムがダウンした場合

一時ファイルが削除されない場合，cscfswrm コマンドで一時ファイル保存フォルダを削除してください。なお，一時ファイル保存フォルダは，HTTP 受付を停止してから削除してください。HTTP 受付が起動しているときに削除すると，ファイルのデータを送信する HTTP リクエストはすべてエラーとなります。

なお，小さいサイズのデータを送信するなど，一時ファイルへの出力が不要な場合，マルチパート型の各パートに対して一時ファイルに出力するかどうかのサイズの閾値を設定できます。この閾値は HTTP 受付定義ファイルの httprecp.file-trans.temp-file.partsize-threshold プロパティで設定できます。

(3) HTTP リクエストのサイズ上限の設定

サイズが大きい不正なデータが送信されることを防ぐために，マルチパート型の HTTP リクエストのサイズに上限を設定できます。HTTP リクエストのサイズが上限を超えた場合は，リクエスト処理が中止し HTTP クライアントにエラーを返します。

サイズ上限は次の項目についてそれぞれ設定できます。

- リクエスト全体

  HTTP ヘッダ，HTTP ボディ（全パートのヘッダおよびボディ分）を含む HTTP リクエスト全体のサイズ上限を設定できます。リクエスト全体のサイズ上限は，HTTP 受付定義ファイルの httprecp.file-trans.maxsize.request プロパティで設定できます。

- パート単位

  各パート単位に指定されたボディ部分のサイズ上限を設定できます。テキストデータの場合でも同様の上限です。パート単位のサイズ上限は，HTTP 受付定義ファイルの httprecp.file-trans.maxsize.part プロパティで設定できます。

(4) パス長に関する注意事項

OS によってパス長に制限がある場合，作業フォルダルートのパス長は適切な長さに設定してください。それぞれのパス長は次のとおりです。

表 2-70 作業フォルダルートのパス長

項目	パス長
作業フォルダルート	HCSC サーバランタイム定義ファイルの work-folder プロパティで設定します。
HCSC サーバ名	最大 8 桁です。
受付 ID	最大 8 桁です。
作業フォルダ名	IPv4 の場合：32 桁で固定です。 IPv6 の場合：52 桁で固定です。
中間ファイル	最大 23 桁です。
一時ファイル保存フォルダ名	32 桁で固定です。
一時ファイル	最大 53 桁です。

2.15.7 HTTP リクエストと HTTP レスポンスのエンコーディング/デコーディング

HTTP リクエストと HTTP レスポンスのエンコーディングおよびデコーディングの処理について説明します。

HTTP リクエスト時，および HTTP レスポンス時に使用される文字コードを次に示します。

HTTP リクエスト時

クエリ文字列およびメッセージボディをデコーディングするための文字コードは，次の方法で指定できます。

- HTTP リクエストヘッダの Content-Type ヘッダの charset 属性で指定する方法
- HTTP 受付定義ファイルの httprecp.http.charset プロパティで指定する方法

HTTP リクエストヘッダと HTTP 受付定義ファイルの両方で指定した場合，HTTP リクエストヘッダの指定が優先されます。

HTTP レスポンス時

メッセージボディのエンコーディングに使用される文字コードについては，「2.15.5(2)(a) Content-Type と charset」を参照してください。

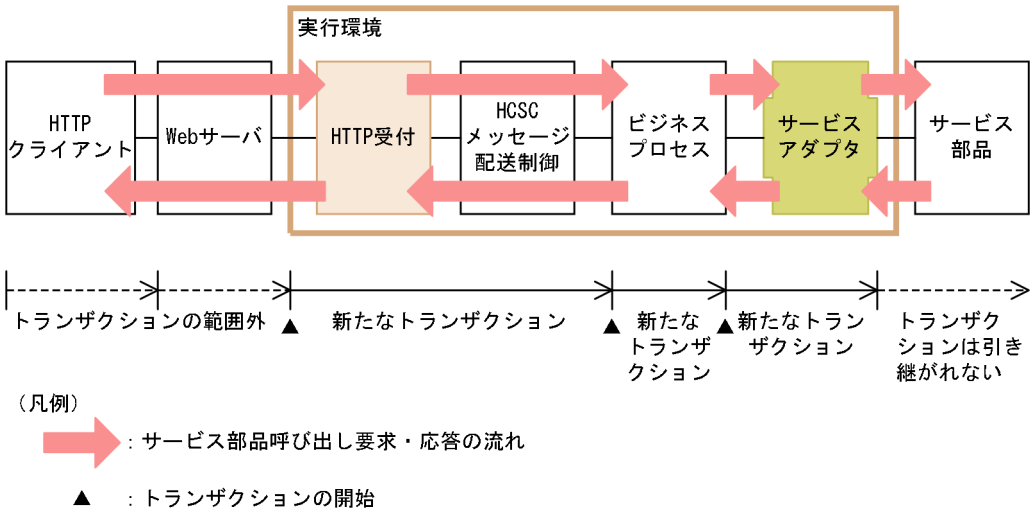
注意事項

- URL のデコードとメッセージボディのデコードには同じ文字コードが使用されます。
- テキストデータとして利用する応答電文（ボディ）のデータは，XML 宣言タグの encoding 属性で設定した文字コードが，HTTP レスポンスボディと HTTP レスポンスヘッダに設定される文字コードと同じものになります。
- 次の場合，HTTP ヘッダの値はユーザが URL エンコーディングする必要があります。
  - ・日本語が含まれる HTTP リクエストヘッダを HTTP クライアントから HTTP 受付に送信する場合
  - ・応答電文（ヘッダ）に日本語の HTTP レスポンスヘッダが含まれる場合
  - ・HTTP レスポンスヘッダ定義ファイルの値に日本語の HTTP レスポンスヘッダが含まれる場合
- HTTP 受付は，HTTP クライアントからの HTTP リクエストをビジネスプロセスに渡すときに，HTTP ヘッダの値をエンコードおよびデコードしません。また，ビジネスプロセスから HTTP クライアントに応答を返すときも，HTTP ヘッダの値をエンコードおよびデコードしません。
- パススルーモードの場合，送信する要求電文の XML 宣言タグの encoding 属性には，必ず「UTF-8」を指定してください。
- バイナリデータから生成したバイナリ形式または任意形式（any 形式）の要求電文を XML 形式の電文に変換する際は，データ変換アクティビティなどを使用してデコードする必要があります。

2.15.8　HTTP 受付使用時のトランザクション

HTTP 受付を使用した場合のトランザクションの範囲を次の図に示します。

図 2-145　HTTP 受付を使用した場合のトランザクションの範囲



HTTP クライアントから HTTP 受付が呼び出されると，HTTP 受付では，HTTP クライアントおよび Web サーバのトランザクションとは異なる新たなトランザクションが開始されます。また，ビジネスプロセス内で，HTTP 受付のトランザクションとは異なる新たなトランザクションが開始されます。

2.15.9　HTTP 受付のタイマ管理

HTTP 受付では，トランザクションタイムアウトを利用してリクエスト処理のタイマ監視ができます。

トランザクションタイムアウトでは，HTTP 受付がビジネスプロセスを呼び出してから，HTTP 受付に応答を返すまでの時間を秒単位で設定できます。処理時間がタイムアウトの設定値を超えた場合は，メッセージが出力されます。

トランザクションタイムアウト値は，HTTP 受付定義ファイルで設定できます。

設定できるパラメタを次の表に示します。

表 2-71　トランザクションタイムアウトの設定

値	キー名称	デフォルト値
トランザクションタイムアウト（秒）	urecp-http.ejb-transaction-timeout	0※

注※

0 を設定した場合，J2EE サーバに設定されたデフォルトのタイムアウト値で動作します。

2.15.10　HTTP 受付の JSON-XML 変換機能

HTTP 受付の JSON-XML 変換機能とは，HTTP 受付で JSON 形式の要求電文を XML 形式に，または XML 形式の応答電文を JSON 形式に変換する機能です。

HTTP 受付の JSON-XML 変換機能には，XML スキーマを使用して変換する拡張モードと，XML スキーマを使用しないで変換する標準モードがあります。どのモードを使って変換するかは，HTTP 受付定義ファイルの httprecp.json-xml-convert.mode プロパティで設定します。HTTP 受付定義ファイルの詳細は，マニュアル「サービスプラットフォーム リファレンス」の「3.11.3　HTTP 受付定義ファイル」を，標準モードと拡張モードの差異については，マニュアル「サービスプラットフォーム 開発ガイド 基本開発編」の「11.　JSON データの変換」を参照してください。

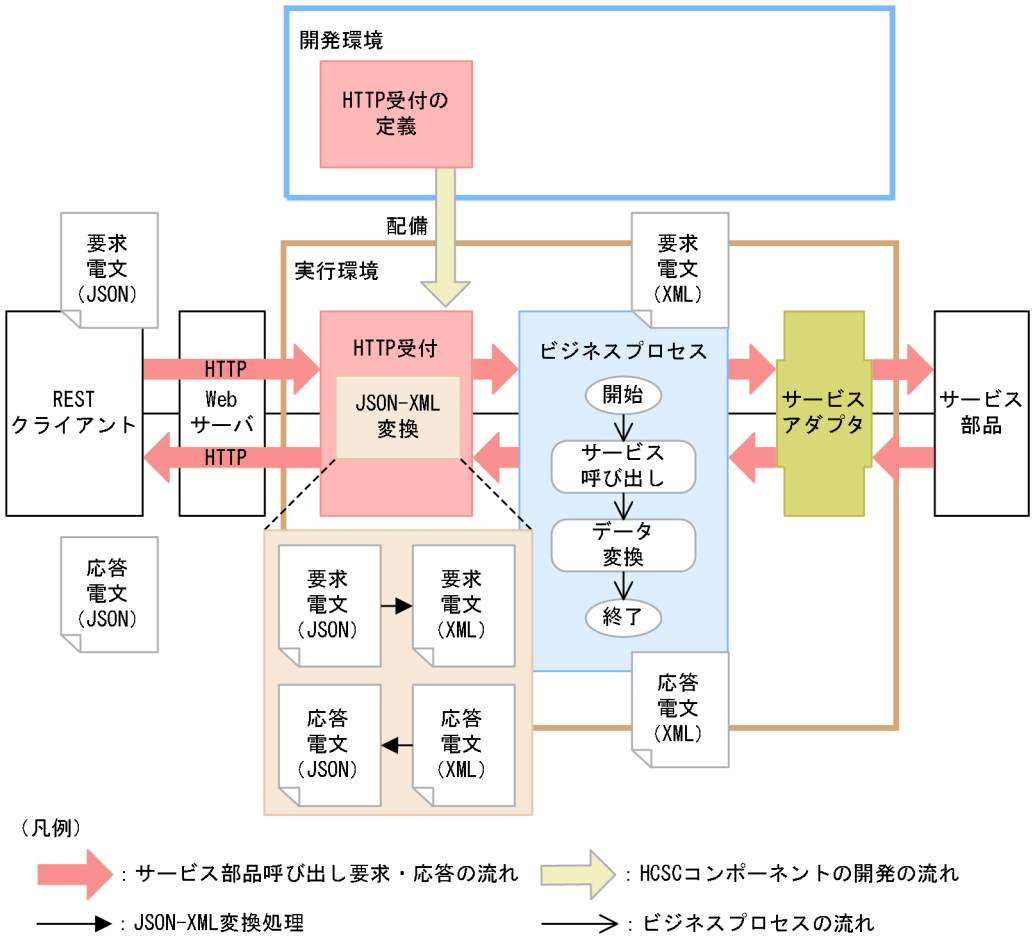
また，使用するモードに合わせて，対応する JSON 変換ツールを選択する必要があります。

JSON 変換ツールについては，マニュアル「サービスプラットフォーム 開発ガイド 基本開発編」の「14. JSON 変換ツールを使用したデータ変換」を参照してください。

(1)　HTTP 受付の JSON-XML 変換機能の概要

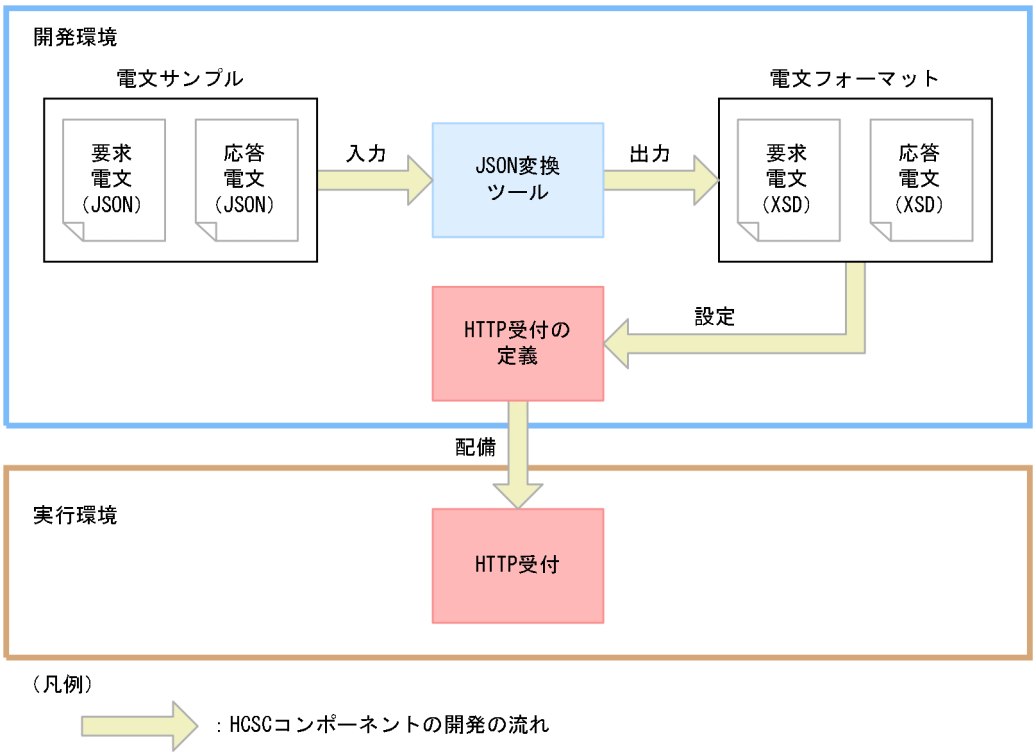
REST クライアントから送信された JSON 形式の要求電文を，HTTP 受付で XML 形式の要求電文に変換します。また，ビジネスプロセスから応答された XML 形式の応答電文を JSON 形式に変換し，REST クライアントへ応答します。HTTP 受付の JSON-XML 変換機能を使用したサービス部品呼び出し要求および応答の流れを次の図に示します。

図 2-146　JSON-XML 変換機能を使用したサービス部品呼び出し要求・応答の流れ（HTTP 受付）



HTTP 受付に設定する電文フォーマット（XML スキーマ）は，JSON 変換ツールで生成します。JSON 変換ツールは JSON 形式の電文のサンプルを受け取り，JSON-XML 変換後の XML 形式の電文に適合する電文フォーマットを生成します。JSON-XML 変換機能を使用する HTTP 受付の開発の流れを次の図に示します。

図 2-147　JSON-XML 変換機能を使用する HTTP 受付の開発の流れ



HTTP 受付の定義の詳細については，マニュアル「サービスプラットフォーム 開発ガイド 受付・アダプタ定義編」の「2.5　HTTP 受付の定義」を，JSON 変換ツールの詳細については，マニュアル「サービスプラットフォーム 開発ガイド 基本開発編」の「14.　JSON 変換ツールを使用したデータ変換」を参照してください。

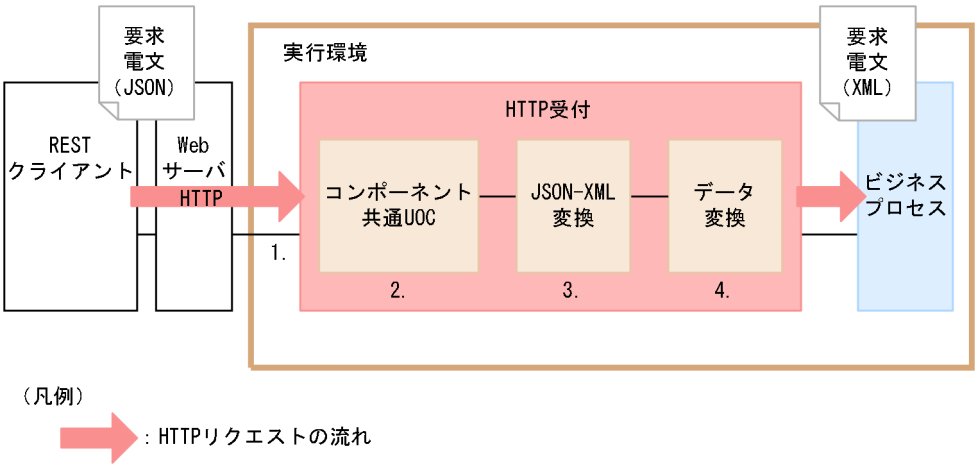
(2)　HTTP 受付のリクエスト処理時の JSON-XML 変換機能

HTTP 受付のリクエスト処理時の JSON-XML 変換のタイミングや設定について説明します。

(a)　HTTP 受付のリクエスト処理時の JSON-XML 変換のタイミング

HTTP 受付のリクエスト処理時の，JSON-XML 変換のタイミングを次の図に示します。

図 2-148　リクエスト処理時の JSON-XML 変換のタイミング（HTTP 受付）



図中の 1.～4.について説明します。

1. HTTP 受付が REST クライアントから要求電文を受信します。
2. コンポーネント共通 UOC では，REST クライアントから送信された JSON 形式の要求電文を扱います。
3. HTTP 受付では，コンポーネント共通 UOC の呼び出しとデータ変換の間で JSON-XML 変換が実行されます。
4. データ変換では，JSON-XML 変換機能で変換された XML 形式の要求電文を扱います。

(b)　HTTP 受付のリクエスト処理時に JSON-XML 変換をする場合の設定

HTTP 受付定義ファイルの httprecp.telegram-auto-convert プロパティが「true」で，要求電文の content-type HTTP ヘッダが application/json の場合，または httprecp.request.switchover.json-transfer.mode プロパティの値が「true」の場合，HTTP 受付のリクエスト処理時に JSON-XML 変換が実行されます。要求電文（ヘッダ）では，JSON-XML 変換を実行するかどうかの切り替えはできません。プロパティの設定と JSON-XML 変換の実行可否を次の表に示します。

表 2-72　リクエスト処理時の JSON-XML 変換の実行可否（HTTP 受付）

HTTP 受付定義ファイルに設定したプロパティの値	JSON-XML 変換の実行
－（プロパティを設定していない）	実行されない
false	実行されない
true	実行される

参考

パススルーモードで，HTTP リクエストボディに msg キーを使用するかどうかの設定に関わらず，JSON-XML 変換機能を使用できます。

また，拡張モードで JSON-XML 変換を行う場合，HTTP 受付定義ファイルの httprecp.json-xml-convert.mode プロパティに「advanced」を設定し，HTTP 受付の要求電文フォーマットに，拡張モードの JSON 変換ツールで生成した XML スキーマファイルを設定する必要があります。

注意事項

JSON 変換ツール以外で作成した XML スキーマファイルを指定した場合，動作は保証されません。

XML スキーマファイルの設定方法は，HTTP 受付の定義内容によって異なります。HTTP 受付の定義ごとの XML スキーマファイルの設定方法を次に示します。

表 2-73　HTTP 受付の定義ごとの XML スキーマファイルの設定方法

HTTP 受付の定義	ルーティング定義の設定	XML スキーマファイルの設定
呼出先固定	－	HTTP 受付の要求電文フォーマットに，XML スキーマファイルを設定します。
呼出先選択	使用しない	
	使用する	HTTP 受付定義ファイルに，XML スキーマファイルを設定します。また，HTTP 受付ルーティング定義ファイルに，XML スキーマファイルのファイル名を設定します。

(c)　HTTP 受付の HTTP リクエストヘッダに関連する処理

JSON-XML 変換機能に関連している HTTP 受付の HTTP リクエストヘッダについて説明します。

HTTP 受付の HTTP リクエストヘッダの処理の詳細については，「2.15.4　HTTP 受付のリクエスト処理」を参照してください。

Content-Type ヘッダ

JSON-XML 変換は，HTTP リクエストヘッダの Content-Type ヘッダの値に関わらず実行されます。また，Content-Type ヘッダが HTTP リクエストヘッダに設定されていない場合は，要求電文（ヘッダ）の content-type 要素は生成されません。

Content-Type ヘッダの charset 属性

HTTP リクエストヘッダの Content-Type ヘッダに charset 属性が設定されていない場合は，要求電文（ヘッダ）の charset 属性に HTTP 受付定義ファイルの httprecp.http.charset プロパティの値が設定されます。httprecp.http.charset プロパティが設定されていないときは，要求電文（ヘッダ）の charset 属性に「UTF-8」が設定されます。

HTTP リクエストヘッダおよび HTTP 受付定義ファイルのプロパティの設定と要求電文（ヘッダ）の関係を次の表に示します。

表 2-74　要求電文（ヘッダ）の charset 属性の設定（HTTP 受付）

HTTP リクエスト ヘッダの設定		HTTP 受付定義ファイル httprecp.http.charset プロパティの設定	要求電文（ヘッダ） charset 属性の設定
Content-Type ヘッダ	charset 属性		
○	○	○	HTTP リクエストヘッダの charset 属性の値が設定されます。
○	○	×	
○	×	○	httprecp.http.charset プロパティの値が設定されます。
○	×	×	「UTF-8」が設定されます。
×	×	○	要求電文（ヘッダ）の charset 属性は設定されません。

（凡例）

○：値が設定されている

×：値が設定されていない

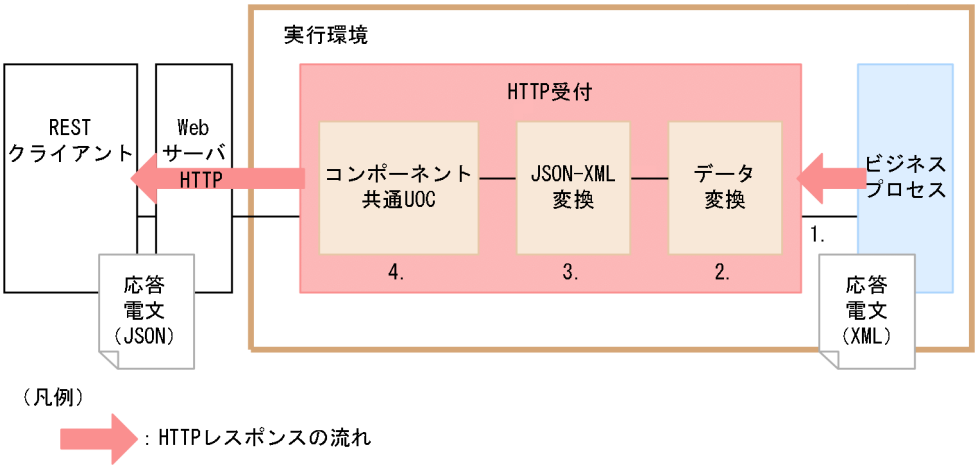
(3)　HTTP 受付のレスポンス処理時の JSON-XML 変換機能

HTTP 受付のレスポンス処理時の JSON-XML 変換のタイミングや設定について説明します。

(a)　HTTP 受付のレスポンス処理時の JSON-XML 変換のタイミング

HTTP 受付のレスポンス処理時の JSON-XML 変換のタイミングを次の図に示します。

図 2-149　レスポンス処理時の JSON-XML 変換のタイミング（HTTP 受付）



図中の 1.～4.について説明します。

1. HTTP 受付がビジネスプロセスから応答電文を受信します。
2. データ変換では，ビジネスプロセスから返却された XML 形式の応答電文を扱います。

3. HTTP 受付では，データ変換とコンポーネント共通 UOC の呼び出しの間で JSON-XML 変換が実行されます。
4. コンポーネント共通 UOC では，JSON-XML 変換機能で変換された JSON 形式の応答電文を扱います。

(b) HTTP 受付のレスポンス処理時に JSON-XML 変換をする場合の設定

HTTP 受付定義ファイルの httprecp.telegram-auto-convert プロパティが「true」で，要求電文の accept HTTP ヘッダが application/json の場合，または httprecp.response.switchover.json-transfer.mode プロパティの値が「true」の場合は，HTTP 受付のレスポンス処理時に JSON-XML 変換が実行されます。応答電文（ヘッダ）の json-transfer 要素の値が「false」の場合は，HTTP 受付のレスポンス処理時に JSON-XML 変換は実行されません。HTTP 受付定義ファイルのプロパティおよび応答電文（ヘッダ）の設定と JSON-XML 変換の実行可否を次の表に示します。

表 2-75 レスポンス処理時の JSON-XML 変換の実行可否（HTTP 受付）

HTTP 受付定義ファイルに設定したプロパティの値	応答電文（ヘッダ） json-transfer 要素	JSON-XML 変換の実行
－（プロパティを設定していない）	要素なし	実行されない
	false	
	true	
false	要素なし	実行されない
	false	
	true	
true	設定なし	実行される
	false	実行されない
	true	実行される

また，拡張モードで JSON-XML 変換を行う場合，HTTP 受付定義ファイルの httprecp.json-xml-convert.mode プロパティに「advanced」を設定し，HTTP 受付の要求電文フォーマットに，拡張モードの JSON 変換ツールで生成した XML スキーマファイルを設定する必要があります。

注意事項

JSON 変換ツール以外で作成した XML スキーマファイルを指定した場合，動作は保証されません。

XML スキーマファイルの設定方法は，HTTP 受付の定義内容によって異なります。HTTP 受付の定義ごとの XML スキーマファイルの設定方法を次に示します。

表 2-76　HTTP 受付の定義ごとの XML スキーマファイルの設定方法

HTTP 受付の定義	ルーティング定義の設定	XML スキーマファイルの設定
呼出先固定	−	HTTP 受付の応答電文フォーマットに，XML スキーマファイルを設定します。
呼出先選択	使用しない	
	使用する	HTTP 受付定義ファイルに，XML スキーマファイルを設定します。また，HTTP 受付ルーティング定義ファイルに，XML スキーマファイルのファイル名を設定します。

(c)　HTTP 受付の HTTP レスポンスヘッダに関連する処理

JSON-XML 変換機能に関連している HTTP 受付の HTTP レスポンスヘッダについて説明します。

HTTP 受付の HTTP レスポンスヘッダの処理の詳細については，「2.15.5　HTTP 受付のレスポンス処理」を参照してください。

Content-Type ヘッダ

JSON-XML 変換は，応答電文（ヘッダ）に設定されている Content-Type の値に関わらず実行されます。また，応答電文（ヘッダ）に Content-Type が設定されていない場合は，「text/xml」が HTTP レスポンスヘッダの Content-Type ヘッダに設定されます。

Content-Type ヘッダの charset 属性

応答電文（ヘッダ）の Content-Type に charset 属性が設定されていない場合は，HTTP レスポンスヘッダに設定される Content-Type ヘッダの charset 属性に，HTTP 受付定義ファイルの httprecp.http.charset プロパティの値が設定されます。httprecp.http.charset プロパティが設定されていないときは，Content-Type ヘッダの charset 属性に「UTF-8」が設定されます。

応答電文（ヘッダ）および HTTP 受付定義ファイルのプロパティの設定と HTTP レスポンスヘッダの charset 属性の関係を次の表に示します。

表 2-77　HTTP レスポンスヘッダの charset 属性の設定（HTTP 受付）

応答電文（ヘッダ）の設定		HTTP 受付定義ファイル httprecp.http.charset プロパティの設定	HTTP レスポンスヘッダ charset 属性の設定
Content-Type ヘッダ	charset 属性		
○	○	○	応答電文（ヘッダ）の charset 属性の値が設定されます。
○	○	×	
○	×	○	httprecp.http.charset プロパティの値が設定されます。
○	×	×	「UTF-8」が設定されます。

応答電文（ヘッダ）の設定		HTTP 受付定義ファイル httprecp.http.charset プロパティの設定	HTTP レスポンスヘッダ charset 属性の設定
Content-Type ヘッダ	charset 属性		
×	×	○	HTTP レスポンスヘッダの charset 属性は設定されません。

（凡例）

○：値が設定されている

×：値が設定されていない

(4)　拡張モードの JSON-XML 変換で変換できない要素があった場合の処理

拡張モードの JSON-XML 変換で変換できない要素があった場合，次の処理のどちらかを選択できます。

- 変換できる要素だけを対象にして，部分的に変換を行ってリクエスト処理を続行させる。
- リクエスト処理をエラーにする。

変換できない要素があった場合の処理は，HTTP 受付定義ファイルの httprecp.json-transfer.allow-partial-conversion で設定できます。httprecp.json-transfer.allow-partial-conversion の設定については，マニュアル「サービスプラットフォーム リファレンス」の「3.11.3　HTTP 受付定義ファイル」を参照してください。

ただし，変換できた要素がない場合は，設定に関係なくリクエスト処理はエラーとなります。

変換結果に問題がある場合は，メッセージを確認し，変換前のデータを修正して，再度リクエストを実行してください。変換できる要素だけを対象にして，部分的に変換を行ってリクエスト処理を続行させるときは，メッセージ「KDEC00065-W」を出力します。リクエスト処理をエラーにするときは，メッセージ「KDEC00060-E」を出力します。メッセージの引数「error message」に，変換に失敗したすべての要素と，それぞれの発生原因が出力されます。

2.15.11　HTTP 受付の呼び出しサービス選択機能

HTTP 受付から呼び出すビジネスプロセスをあらかじめ HTTP 受付サービス選択定義ファイルに設定しておくことで，リクエスト処理時に HTTP 受付サービス選択定義ファイルに設定したビジネスプロセスを呼び出すことができる機能です。

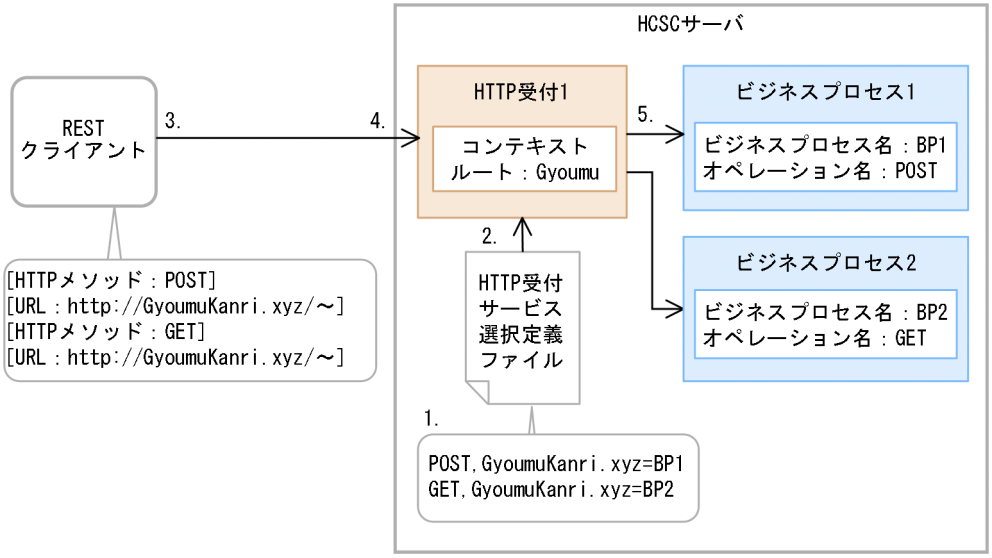
HTTP 受付とビジネスプロセスを 1 対 N で定義し，HTTP 受付から呼び出すサービスを条件（HTTP メソッド名，およびホスト名）によって選択できます。1 つの HTTP 受付にひもづくビジネスプロセス（業務）を，複数のビジネスプロセス（業務）に分割できるので，ビジネスプロセス（業務）を変更するときの影響範囲を小さくします。

呼び出しサービス選択機能の設定については，マニュアル「サービスプラットフォーム 開発ガイド 受付・アダプタ定義編」の「付録 L　ユーザ定義受付（呼出先選択）の定義」を参照してください。HTTP 受付

サービス選択定義ファイルについては，マニュアル「サービスプラットフォーム リファレンス」の「3.11.2　HTTP 受付サービス選択定義ファイル」を参照してください。

HTTP メソッドとホストを使用して，呼び出すビジネスプロセスを選択する処理を次の図に示します。

図 2-150　呼び出しサービス選択機能を使用したビジネスプロセスの選択処理



図中の 1.～5.について説明します。

1. HTTP 受付に HTTP 受付サービス選択定義ファイルを設定します。
2. HTTP 受付の起動時に，設定した HTTP 受付サービス選択定義ファイルが読み込まれます。
3. REST クライアントから HTTP リクエストヘッダに，HTTP メソッド，およびホストの情報を含むリクエストを送信します。
4. URL に指定されたコンテキストルートに該当する HTTP 受付で，HTTP リクエストを受け付けます。
5. リクエストから取得した HTTP メソッドおよびホスト名を条件に，2.の HTTP 受付サービス選択定義ファイルの情報を検索し，条件に一致したビジネスプロセスを呼び出します。

2.15.12　HTTP 受付のルーティング機能

HTTP 受付のルーティング機能を使用して，ホスト名，URL パターン，および HTTP メソッド単位で HTTP 受付を実行できます。

HTTP 受付に HTTP 受付ルーティング定義ファイルを設定し，リクエスト処理時に，HTTP リクエストが保持する情報（ホスト名，URL パターン，HTTP メソッド）が，HTTP 受付ルーティング定義ファイルに設定した条件と合致した場合，HTTP 受付ルーティング定義ファイルに設定した内容で HTTP 受付が実行されます。

(1) HTTP 受付のルーティング機能で使用するファイル

HTTP 受付のルーティング機能を使用する場合，次の定義ファイルを設定します。

- HTTP 受付定義ファイル

  HTTP 受付定義ファイルの httprecp.use-routing-definition プロパティに true を設定します。

  HTTP 受付定義ファイルの設定項目の詳細については，マニュアル「サービスプラットフォーム リファレンス」の「3.11.3　HTTP 受付定義ファイル」を参照してください。

- HTTP 受付ルーティング定義ファイル

  HTTP 受付定義ファイル，および HTTP 受付サービス選択定義ファイルの設定項目の中から，RESTful Web サービスと連携するリクエストを処理する場合に使用する設定項目を，HTTP 受付ルーティング定義ファイルに設定します。

  HTTP 受付ルーティング定義ファイルの設定項目を次に示します。

  表 2-78　HTTP 受付ルーティング定義ファイルの設定項目

設定項目	指定する要素	property 要素の key 属性に設定する値
ビジネスプロセス名	invoke 要素	－
パススルーモードまたは標準モードの指定	property 要素	httprecp.switchover.pass-through.mode
パススルーモードで，HTTP リクエストボディに msg キーを使用するかどうか		httprecp.pass-through.parameter-use
リクエスト処理時に JSON-XML 変換するかどうか		httprecp.request.switchover.json-transfer.mode
レスポンス処理時に JSON-XML 変換するかどうか		httprecp.response.switchover.json-transfer.mode

  HTTP 受付ルーティング定義ファイルの設定項目の詳細については，マニュアル「サービスプラットフォーム リファレンス」の「3.11.4　HTTP 受付ルーティング定義ファイル」を参照してください。

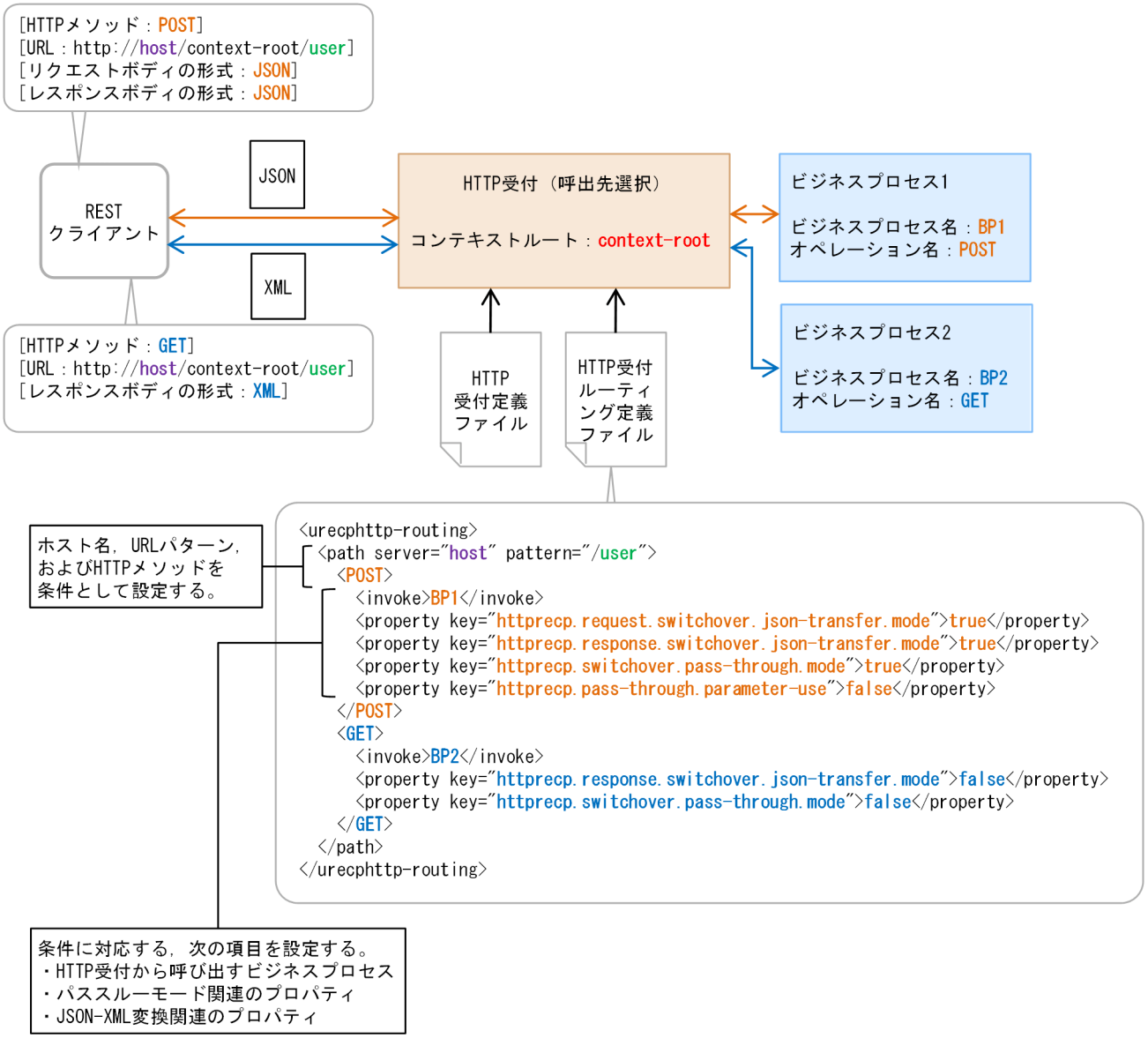
なお，HTTP 受付ルーティング定義ファイルを使用した場合，HTTP 受付サービス選択定義ファイルは読み込まれません。

(2) HTTP 受付のルーティング機能の定義方法

HTTP 受付のルーティング機能の定義例を基に，定義方法について説明します。

HTTP 受付ルーティング定義ファイルにホスト名，URL パターン，および HTTP メソッドを条件として設定し，HTTP 受付から呼び出すビジネスプロセス，JSON-XML 変換，およびパススルーモードを設定する場合の定義例を次に示します。

図 2-151　HTTP 受付のルーティング機能の定義例



HTTP 受付のルーティング機能は，次の手順で設定します。

1. 開発環境で，ユーザ定義受付（呼出先選択）に HTTP 受付を追加します。

   HTTP 受付ルーティング定義ファイルは，HTTP 受付の追加時に HTTP 受付の独自定義ファイルとして追加されます。

   ユーザ定義受付（呼出先選択）の設定については，マニュアル「サービスプラットフォーム 開発ガイド 受付・アダプタ定義編」の「2.5　HTTP 受付の定義」を参照してください。

2. HTTP 受付ルーティング定義ファイルを編集して，設定項目を追加します。

   拡張モードで JSON-XML 変換を行う場合，HTTP 受付ルーティング定義ファイルに，XML スキーマファイルのファイル名を設定します。

3. HTTP 受付定義ファイルの httprecp.use-routing-definition プロパティに true を設定します。

   拡張モードで JSON-XML 変換を行う場合，HTTP 受付定義ファイルの httprecp.json-xml-convert.mode で拡張モード（advanced）を設定します。

注意事項

プロパティに設定される値は，次の優先順位で適用されます。

1. HTTP 受付ルーティング定義ファイルに設定したプロパティの値
2. HTTP 受付定義ファイルに設定したプロパティの値
3. HTTP 受付定義ファイルに設定するプロパティのデフォルト値

2.15.13　HTTP 受付の電文自動変換機能

HTTP 受付の電文自動変換機能では，次の 2 つの変換が行われます。

- HTTP クライアントから送信される Content-Type HTTP リクエストヘッダの値から要求電文の形式を判断して，必要な変換を行う。
- HTTP クライアントから送信される Accept HTTP リクエストヘッダの値から応答電文の形式を判断して，必要な変換を行う。

要求電文は Content-Type の内容によって，次の表のとおりに自動変換が行われます。

表 2-79　要求電文の自動変換の内容

項番	Content-Type	変換処理の内容
1	application/json	JSON-XML 変換が行われます。
2	application/xml	パススルーモードになります。変換は行われません。
3	上記以外の場合	標準モードで変換が行われます。

応答電文は Accept の内容によって，次の表のとおりに自動変換が行われます。

表 2-80　応答電文の自動変換の内容

項番	Accept	変換処理の内容
1	application/json	XML-JSON 変換が行われます。
2	上記以外の場合	変換は行われません。

HTTP 受付の電文自動変換機能は，HTTP 受付定義ファイルの httprecp.telegram-auto-convert プロパティが「true」の場合に有効になります。「false」の場合は無効になります（デフォルトは無効です）。

また，HTTP 受付の自動変換機能が有効である場合，次の 3 つのプロパティの設定値に関係なく，電文の自動変換が行われます。

- httprecp.switchover.pass-through.mode プロパティ

- httprecp.request.switchover.json-transfer-mode プロパティ
- httprecp.response.switchover.json-transfer-mode プロパティ

ただし，HTTP 受付ルーティング定義ファイルの property 要素でオペレーション（HTTP メソッド）ごとに上記プロパティの設定値が指定された場合は，HTTP 受付ルーティング定義ファイルの設定が優先されます。

2.15.14　OpenAPI 仕様書から HCSC コンポーネントを作成する OpenAPI 仕様書インポート機能

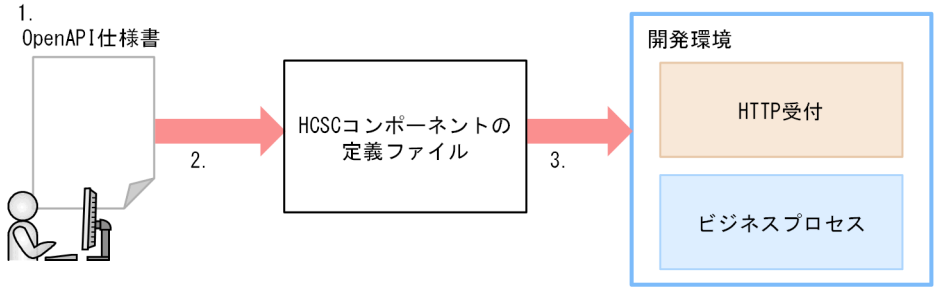
OpenAPI 仕様書インポート機能とは，OpenAPI 仕様書を HTTP 受付やビジネスプロセスなどの HCSC コンポーネントに変換する機能です。

OpenAPI 仕様書インポート機能を利用して作成した HCSC コンポーネントの定義ファイルをサービス定義のインポート機能を使用して，開発環境に取り込みます。

(1)　OpenAPI 仕様書インポート機能の概要

OpenAPI 仕様書を HCSC コンポーネントに変換する流れを次に示します。

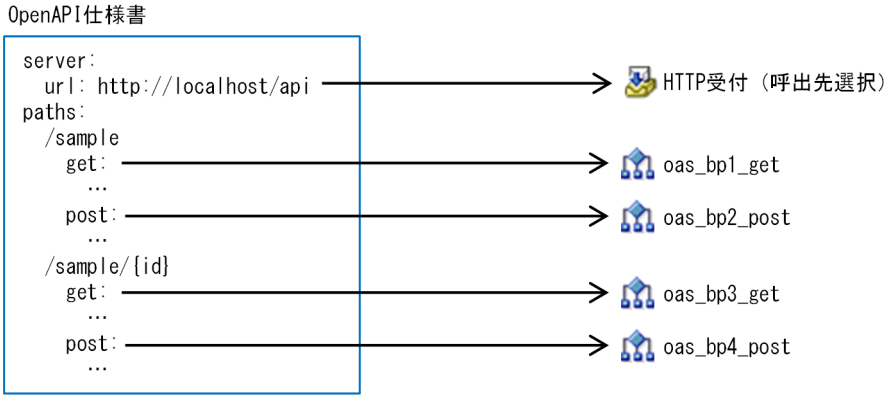
図 2-152　OpenAPI 仕様書を HCSC コンポーネントに変換する流れ



1. Swagger Editor などのツールを使用し，OpenAPI 仕様書を作成します。
   HCSC コンポーネントと OpenAPI 仕様書との対応については，「(3)　HCSC コンポーネントと OpenAPI 仕様書との対応」を参照してください。
2. OpenAPI 仕様書インポート機能を利用して，OpenAPI 仕様書を HCSC コンポーネントの定義ファイル（サービス定義 XML ファイル，電文のスキーマファイル，独自定義ファイル）に変換します。
   OpenAPI 仕様書の変換については，「(5)　OpenAPI 仕様書の変換」を参照してください。
3. サービス定義のインポート機能を利用して，開発環境に HCSC コンポーネント（HTTP 受付，ビジネスプロセス）を作成します。
   サービス定義のインポート機能については，マニュアル「サービスプラットフォーム 開発ガイド 基本開発編」の「6.4　サービス定義のインポート」を参照してください。

OpenAPI 仕様書から作成された HCSC コンポーネントの生成イメージを次に示します。

図 2-153　HCSC コンポーネントの生成イメージ



OpenAPI 仕様書での Server は HTTP 受付，Operation はビジネスプロセスにそれぞれ対応します。

(2)　前提条件

OpenAPI 仕様書インポート機能を使用する場合の前提条件を次に示します。

- OpenAPI 仕様書インポート機能で対応する OpenAPI Specification のバージョンは 3.0 以降です。
- OpenAPI 仕様書インポート機能で使用する前提 OSS を次の表に示します。
  前提 OSS の jar ファイルは，次に示すディレクトリの下に配置してください。

  ```
  <サービスプラットフォームのインストールディレクトリ>¥CSCTE¥lib¥external
  ```

表 2-81　前提 OSS 一覧

項番	OSS 名	バージョン	jar ファイル名	ライセンス
1	commons-io	2.7	commons-io-2.7.jar	Apache License 2.0
2	commons-lang3	3.9	commons-lang3-3.9.jar	Apache License 2.0
3	jackson-annotations	2.11.2	jackson-annotations-2.11.2.jar	Apache License 2.0
4	jackson-core	2.11.2	jackson-core-2.11.2.jar	Apache License 2.0
5	jackson-databind	2.11.2	jackson-databind-2.11.2.jar	Apache License 2.0
6	jackson-dataformat-yaml	2.11.2	jackson-dataformat-yaml-2.11.2.jar	Apache License 2.0
7	jackson-datatype-jsr310	2.11.2	jackson-datatype-jsr310-2.11.2.jar	Apache License 2.0
8	jakarta.activation-api	1.2.2	jakarta.activation-api-1.2.2.jar	EDL 1.0

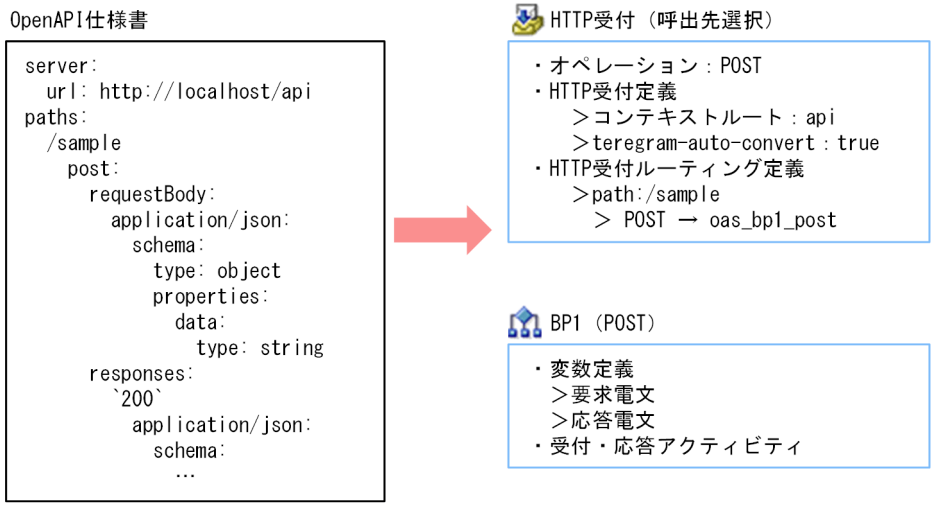
項番	OSS 名	バージョン	jar ファイル名	ライセンス
9	jakarta.validation-api	2.0.2	jakarta.validation-api-2.0.2.jar	Apache License 2.0
10	jakarta.xml.bind-api	2.3.3	jakarta.xml.bind-api-2.3.3.jar	EDL 1.0
11	slf4j-api	1.7.25	slf4j-api-1.7.25.jar	MIT
12	snakeyaml	1.26	snakeyaml-1.26.jar	Apache License 2.0
13	swagger-annotations	2.2.26	swagger-annotations-2.2.26.jar	Apache License 2.0
14	swagger-core (io.swagger.core.v3)	2.2.26	swagger-core-2.2.26.jar	Apache License 2.0
15	swagger-inflector	2.0.5	swagger-inflector-2.0.5.jar	Apache License 2.0
16	swagger-models	2.2.26	swagger-models-2.2.26.jar	Apache License 2.0
17	swagger-parser-core	2.1.23	swagger-parser-core-2.1.23.jar	Apache License 2.0
18	swagger-parser-safe-url-resolver	2.1.23	swagger-parser-safe-url-resolver-2.1.23.jar	Apache License 2.0
19	swagger-parser-v2-converter	2.1.23	swagger-parser-v2-converter-2.1.23.jar	Apache License 2.0
20	swagger-parser-v3	2.1.23	swagger-parser-v3-2.1.23.jar	Apache License 2.0
21	swagger-parser	2.1.23	swagger-parser-2.1.23.jar	Apache License 2.0

(3) HCSC コンポーネントと OpenAPI 仕様書との対応

OpenAPI 仕様書インポート機能では，OpenAPI 仕様書を変換して，HCSC コンポーネントの HTTP 受付（呼出先選択）とビジネスプロセスを作成できます。

OpenAPI 仕様書を HCSC コンポーネントに変換したときの例を次に示します。

図 2-154　OpenAPI 仕様書からの HCSC コンポーネントの変換例



OpenAPI 仕様書インポート機能で作成した HTTP 受付（呼出先選択）には，OpenAPI 仕様書の定義に従った HTTP クライアントから接続できます。

HTTP 受付（呼出先選択）およびビジネスプロセスと，OpenAPI 仕様書について，次に説明します。

(a)　HTTP 受付（呼出先選択）と OpenAPI 仕様書との対応

HTTP 受付（呼出先選択）と OpenAPI 仕様書との対応を次の表に示します。

表 2-82　HTTP 受付（呼出先選択）と OpenAPI 仕様書との対応

項番	HTTP 受付（呼出先選択）の設定	OpenAPI 仕様書のフィールド	サービス定義 XML		
			タグ	設定値	フィールドが定義されていない場合の設定値
1	受付名	Server#x-csd-reception-name	/service-definition/selectable-reception/name	Server#x-csd-reception-name に設定した名称	デフォルトの受付名を設定します。
2	受付 ID	Server#x-csd-reception-id	/service-definition/selectable-reception/reception-id	Server#x-csd-reception-id に設定した ID	デフォルトの ID を設定します。
3	オペレーション	Path Item#get	/service-definition/selectable-reception/protocol-definition/operations/operation/name	GET	operation 要素を省略します。
4		Path Item#put		PUT	
5		Path Item#post		POST	
6		Path Item#delete		DELETE	

項番	HTTP 受付（呼出先選択）の設定	OpenAPI 仕様書のフィールド	サービス定義 XML		
			タグ	設定値	フィールドが定義されていない場合の設定値
7	オペレーション	Path Item#options	/service-definition/selectable-reception/protocol-definition/operations/operation/name	OPTIONS	operation 要素を省略します。
8		Path Item#head		HEAD	
9	要求電文ボディの電文フォーマット	Request Body#content	/service-definition/selectable-reception/protocol-definition/operations/operation/input-body/format/file-path	XSD のファイルパス※	file-path 要素を省略します。
10	応答電文ボディの電文フォーマット	Response#content	/service-definition/selectable-reception/protocol-definition/operations/operation/output-body/format/file-path	XSD のファイルパス※	file-path 要素を省略します。
11	HTTP 受付定義	−	/service-definition/selectable-reception/protocol-definition/custom-definition-files/file-path	HTTP 受付定義のファイルパス	−
12	urecp-http.context-root	Servers#/{url}	−	{path}の内容 (変数展開を行います)	HCSC コンポーネントを作成するための情報がないため，エラーメッセージ（KECT94017-E）が出力されます。
13	httprecp.telegram-auto-convert	−	−	true	−
14	HTTP 受付ルーティング定義	−	/service-definition/selectable-reception/protocol-definition/custom-definition-files/file-path	HTTP 受付ルーティング定義のファイルパス	−
15	<path pattern="<パスパターン>">	Paths#/{path}	−	パスパターン	−

項番	HTTP 受付（呼出先選択）の設定		OpenAPI 仕様書のフィールド	サービス定義 XML		
				タグ	設定値	フィールドが定義されていない場合の設定値
16		<HTTP メソッド名>=<ビジネスプロセス名>	Path Item	－	項番 3～8 のオペレーション=ビジネスプロセス名	プロパティを省略します。

（凡例）－：該当しません。

注※

Media Type を優先順位に従って検索し，優先度の高い Media Type の Schema を選択します。次のケースでは file-path 要素を省略します。

・優先順位に合致する Media Type が存在しなかった場合

・Media Type が multipart の場合

この表に記載がない HTTP 受付の設定については OpenAPI 仕様書では定義できません。サービス定義 XML ファイルのデフォルト値が採用されます。サービス定義 XML ファイルについては，マニュアル「サービスプラットフォーム 開発ガイド 基本開発編」の「6. サービス定義のインポート・エクスポート」を参照してください。

(b) ビジネスプロセスと OpenAPI 仕様書との対応

ビジネスプロセスと OpenAPI 仕様書との対応を次の表に示します。

表 2-83 ビジネスプロセスと OpenAPI 仕様書との対応

項番	ビジネスプロセスの設定			OpenAPI 仕様書のフィールド	サービス定義 XML		
					タグ	設定値	フィールドが定義されていない場合の設定値
1	ビジネスプロセス名			Operation#x-csd-business-process-name，または Operation#operationId	/service-definition/business-process/overview/name	Operation#x-csd-business-process-name，または Operation#operationId に設定した名称	デフォルトのビジネスプロセス名を設定します。
2	サービス ID			Operation#x-csd-business-process-id	/service-definition/business-process/overview/service-id	Operation#x-csd-business-process-id に設定した名称	デフォルトの ID を設定します。
3	変数定義	要求電文ヘッダ	変数名	Operation#x-csd-request-header-variable-name	/service-definition/business-process/variables/global/variable/name	Operation#x-csd-request-header-variable-name に設定した名称	デフォルトの変数名を設定します。

項番	ビジネスプロセスの設定			OpenAPI 仕様書のフィールド	サービス定義 XML		
					タグ	設定値	フィールドが定義されていない場合の設定値
4	変数定義	要求電文ヘッダ	種別	−	/service-definition/business-process/variables/global/variable/type	XML	−
5			電文フォーマット	−	/service-definition/business-process/variables/global/variable/file-path	<サービスプラットフォームのインストールディレクトリ>/CSC/custom-reception/http/schema/urecp_http_header_request.xsd	−
6		要求電文ボディ	変数名	Media Type#x-csd-variable-name	/service-definition/business-process/variables/global/variable/name	Media Type#x-csd-variable-name に設定した名称	デフォルトの変数名を設定します。
7			種別	−	/service-definition/business-process/variables/global/variable/type	XML	−
8			電文フォーマット	Request Body#content Media Type#schema	/service-definition/business-process/variables/global/variable/file-path	XSD のファイルパス	−
9		応答電文ヘッダ	変数名	Operation#x-csd-response-header-variable-name	/service-definition/business-process/variables/global/variable/name	Operation#x-csd-response-header-variable-name に設定した名称	デフォルトの変数名を設定します。
10			種別	−	/service-definition/business-process/variables/global/variable/type	XML	−
11			電文フォーマット	−	/service-definition/business-process/variables/global/variable/file-path	<サービスプラットフォームのインス	−

項番	ビジネスプロセスの設定			OpenAPI 仕様書のフィールド	サービス定義 XML		
					タグ	設定値	フィールドが定義されていない場合の設定値
11	変数定義	応答電文ヘッダ	電文フォーマット	−	/service-definition/business-process/variables/global/variable/file-path	トールディレクトリ>/CSC/custom-reception/http/schema/urecp_http_header_response.xsd	−
12		応答電文ボディ	変数名	Media Type#x-csd-variable-name	/service-definition/business-process/variables/global/variable/name	Media Type#x-csd-variable-name に設定した名称	デフォルトの変数名を設定します。
13			種別	−	/service-definition/business-process/variables/global/variable/type	XML	−
14			電文フォーマット	Response#content Media Type#schema	/service-definition/business-process/variables/global/variable/file-path	XSD のファイルパス	−
15	受付アクティビティ	アクティビティ名		Operation#x-csd-receive-activity-name	/service-definition/business-process/activities/global/main-sequence/receive-activity/activity-name	Operation#x-csd-receive-activity-name に設定した名称	デフォルトの受付アクティビティ名を設定します。
16		オペレーション名		−	/service-definition/business-process/activities/global/main-sequence/receive-activity/operation-name	Operation に定義した HTTP メソッド名	−
17		ボディ割当変数		−	/service-definition/business-process/activities/global/main-sequence/receive-activity/body-variable-name-ref	項番 6 の変数	−
18		ヘッダ割当変数	割当変数	−	/service-definition/business-process/activities/global/main-sequence/receive-activity/headers/header/variable-name-ref	項番 3 の変数	−

項番	ビジネスプロセスの設定			OpenAPI 仕様書のフィールド	サービス定義 XML		
					タグ	設定値	フィールドが定義されていない場合の設定値
18	受付アクティビティ	ヘッダ割当変数	ルート要素	－	/service-definition/business-process/activities/global/main-sequence/receive-activity/headers/header/root-element-name	http-header-request	－
19	応答アクティビティ	アクティビティ名		Operation#x-csd-reply-activity-name	/service-definition/business-process/activities/global/main-sequence/reply-activity/activity-name	Operation#x-csd-reply-activity-name に設定した名称	デフォルトの応答アクティビティ名を設定します。
20		オペレーション名		－	/service-definition/business-process/activities/global/main-sequence/reply-activity/operation-name	Operation に定義した HTTP メソッド名	－
21		ボディ割当変数※		－	/service-definition/business-process/activities/global/main-sequence/reply-activity/body-variable-name-ref	項番 14 の変数	－
22		ヘッダ割当変数	割当変数	－	/service-definition/business-process/activities/global/main-sequence/reply-activity/headers/header/variable-name-ref	項番 11 の変数	－
			ルート要素	－	/service-definition/business-process/activities/global/main-sequence/reply-activity/headers/header/root-element-name	http-header-response	－

（凡例）－：該当しません。

注※

Response が Responses 内で複数定義されていた場合は要素を省略します。

(4) 設定項目の命名規則

受付名やビジネスプロセス名などの設定項目は，OpenAPI 仕様書に CSD 独自のフィールドを定義することで設定できます。CSD 独自フィールドの設定値は，変換時に検証され，エラーになった場合，または CSD 独自フィールドが定義されていない場合は，規則に従ってデフォルト値が設定されます。

CSD 独自フィールドと設定項目の対応を次に示します。

表 2-84　CSD 独自フィールドと設定項目の対応

HCSC コンポーネントの設定項目	CSD 独自フィールド	デフォルト値	デフォルト値の設定例
受付名	Server#x-csd-reception-name	oas_rcp<連番>※1	oas_rcp1, oas_rcp2, …
受付 ID	Server#x-csd-reception-id	oas<連番>※1※2	oas1, oas2, …
ビジネスプロセス名※5	Operation#x-csd-business-process-name	oas_bp<連番>※1_<HTTP メソッド名>	oas_bp1_get, oas_bp2_put, …
サービス ID	Operation#x-csd-business-process-id	oas<連番>※1※2	oas1, oas2, …
受付アクティビティ名※3	Operation#x-csd-receive-activity-name	receive_activity	－
応答アクティビティ名※3	Operation#x-csd-reply-activity-name	reply_activity	－
変数名※4	• 要求電文ヘッダ Operation#x-csd-request-header-variable-name • 応答電文ヘッダ Operation#x-csd-response-header-variable-name • 要求電文ボディおよび応答電文ボディ Media Type#x-csd-variable-name	• 要求電文ヘッダ http_header_request • 応答電文ヘッダ http_header_response • 要求電文ボディ http_body_request • 応答電文ボディ http_body_response_<ステータスコード>	－

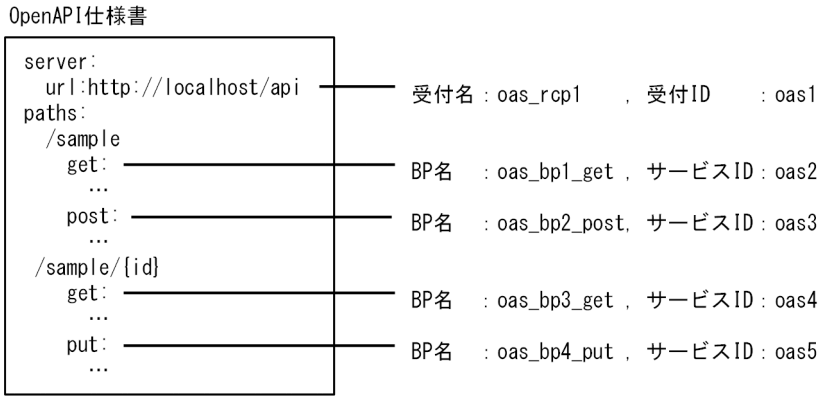
（凡例）－：該当しない。

注※1

受付名およびビジネスプロセス名に利用する連番は変換順に 1 から 1 つずつ繰り上がります。

注※2

受付 ID とサービス ID は共通の連番を使用し，重複しないように設定されます。



注※3

デフォルトの名称がすでに使用されている場合は，連番を付与します。

（例）receive_activity1

注※4

デフォルトの名称がすでに使用されている場合は，アンダーバーと連番を付与します。

（例）http_body_response_200_1

注※5

ビジネスプロセス名を設定する場合は，CSD 独自フィールドに加えて，Operation#operationId 標準フィールドの設定値があわせて使用されます。ビジネスプロセス名は，次の手順で決定されます。

1. CSC 独自フィールド（Operation#x-csd-business-process-name）が設定されている場合は，その設定値がビジネスプロセス名になります。
2. 上記 1.の条件に該当しない場合で，標準フィールド（Operation#operationId）が設定されているときは，その設定値がビジネスプロセス名になります。
3. 上記 1.と 2.の条件に該当しない場合は，コマンド側のデフォルト値がビジネスプロセス名になります。

(5) OpenAPI 仕様書の変換

OpenAPI 仕様書を HCSC コンポーネントに変換するには，cscoas2servicedef コマンドに OpenAPI 仕様書のファイルパスや出力先ディレクトリなどの指定が必要です。

cscoas2servicedef コマンドの詳細については，マニュアル「サービスプラットフォーム リファレンス」の「cscoas2servicedef（OpenAPI 仕様書を HCSC コンポーネントの定義に変換)」を参照してください。

(6) OpenAPI 仕様書の変換時の注意事項

- cscoas2servicedef コマンドに指定した出力先ディレクトリに同名のファイルがある場合は，上書きされます。
- cscoas2servicedef コマンドに指定した出力先ディレクトリがない場合は，新規にディレクトリを作成し，ファイルを出力します。出力先ディレクトリと同名のファイルがあった場合はエラーとなります。

- 入力ファイルに使用できる文字コードは UTF-8 だけです。UTF-8 以外の文字コードのファイルが入力された場合の動作は保証しません。各定義ファイルの出力時に使用する文字コードも UTF-8 となります。
- OpenAPI 仕様書内に記述された OpenAPI Specification のバージョンが 3.0 以降でない場合，およびフィールドが省略されていた場合はエラーとなります。
- 変換結果のファイル名やディレクトリ名として使用するフィールドには Windows のファイル名として利用できない文字および名称は指定できません。指定した場合の動作は保証されません。
- $ref を使用する場合，正しい参照先を指定してください。参照が解決できなかったときは，正しく変換されない場合があります。
- Media Type が json の場合に生成される XML Schema ファイルは，スキーマ内に定義される要素が json のオブジェクト名の昇順にソートされます。
- 受付名などを設定するための CSD の独自フィールドの値は，文字列で指定してください。
- XML Schema の生成で，Schema オブジェクトに items フィールドが定義されている場合は，type フィールドの値に関係なく，array として扱われます。
- XML Schema の生成で，XML オブジェクトの namespace フィールドはルート要素にだけ指定できます。また，namespace フィールドを指定する場合は，prefix フィールドの指定も必要です。
- XML Schema の生成では，XML オブジェクトの wrapped フィールドは指定できません。
- XML Schema の生成で，次のときは XML Schema が生成されません。
  - Media Type が json の場合で，最上位に定義されている Schema オブジェクトの type フィールドが object または array でないとき
  - Media Type が xml の場合で，最上位に定義されている Schema オブジェクトの type フィールドが object でないとき

2.15.15　HTTP 受付を使用するときの注意

- HTTP レスポンスヘッダを応答電文（ヘッダ）の http-header 要素で設定する場合は，有効な HTTP ヘッダを設定してください。HTTP 受付は HTTP ヘッダが妥当かどうかを検証しません。
- HTTP 受付では，HTTP のセッション管理をサポートしていません。
- HTTP リクエストにテキストデータが存在しない場合でも，受付アクティビティに設定する要求電文（ボディ）を設定してください。
- HTTP レスポンスでテキストデータを返さない場合でも，応答アクティビティに設定する要求電文（ボディ）を設定してください。設定するスキーマは製品が提供するスキーマを使用してください。
- HTTP リクエストボディをビジネスプロセスに渡すときのモードにパススルーモードを使用し，かつ HTTP 受付定義ファイルに httprecp.pass-through.parameter-use=true を指定している場合は，次の注意事項があります。また，httprecp.telegram-auto-convert=true を指定している場合で content-type HTTP ヘッダが application/xml，または application/json のときも同様です。

- 「msg=値」のペアは 1 つだけ指定してください。複数の「msg=値」のペアが指定された場合，動作を保証しません。また，msg キーの指定がない場合はエラーになります。
- msg キーはすべて小文字で指定してください。大文字を指定した場合はエラーになります。
- HTTP リクエストボディには有効な XML 文字列を指定してください。HTTP 受付では XML 文書をチェックしません。
- 「msg=msg=abc」のように，クエリ文字列またはメッセージボディに複数の「=」が指定された場合，最初の「=」が区切り文字として使用されます。この場合，最初の「=」の前の部分はキーとして読み込まれ，最初の「=」のあとの文字列全体は値として読み込まれます。

- HTTP 受付を利用してテキストデータの送受信，およびファイルのデータをアップロードまたはダウンロードする場合，次の制限があります。
  - ビジネスプロセスは HTTP 受付に応答を返す必要があります。このため，HTTP 受付の通信モデルは非同期にできません。
  - BPEL の並行処理（FLOW）は使用できません。
  - HTTP 受付に対して 1 リクエストで同時に処理できるファイル数の上限は 1,024 個です。
  - 応答電文（ヘッダ）の partID 要素および file-name 要素にマルチバイト文字を指定した場合，動作を保証しません。
  - HTTP レスポンスでファイルのデータを送信する場合，指定できるファイル数の上限は 1,024 です。指定したファイル数が 1,024 を超えた場合は，動作を保証しません。
- HTTP 受付を利用してファイルのデータを送受信する場合は，次の点を考慮して，HTTP 受付ごとに作業フォルダのディスクサイズを十分に割り当ててください。
  - 同時実行数

    サイズが大きいファイルを扱う HTTP リクエストが一度に大量に送信される可能性がある場合，極端なリソース消費とならないように同時実行数を考慮してください。
  - 想定される最大の HTTP リクエストサイズ

    ファイルのデータの処理時に作成される一時ファイルは，中間ファイルと同じサイズおよび個数生成されます。このため，最大でどの程度の HTTP リクエストが送信されるか事前に見積もっておく必要があります。
  - HTTP 受付に関連づいたビジネスプロセスの処理

    HTTP 受付から呼び出されたビジネスプロセスで，ファイル操作アダプタを利用して作業フォルダ内にファイルを生成したり，HTTP アダプタなどで取得したファイルを作業フォルダに格納したりする場合は，その処理によって生成されるファイルのサイズも考慮しておく必要があります。また，中間ファイルおよび一時ファイルが削除されるまでの処理時間を考慮してビジネスプロセスを設計してください。
- コンテキストルートは，Component Container の作業ディレクトリの中でディレクトリ名として使用されます。そのため，作業ディレクトリ全体のパス長が OS の上限に達しないようにコンテキストルートを指定する必要があります。OS のパス長の上限については，使用する各 OS のドキュメントを参照してください。

Component Container の作業ディレクトリ全体の構成を次に示します。

Component Container の作業ディレクトリ全体の構成	備考
Component Container の作業ディレクトリまでのパス¥	−
Component Container の作業ディレクトリ¥	デフォルトのパスは次のとおりです。 • Windows の場合 <サービスプラットフォームのインストールディレクトリ>¥CC¥server¥public • UNIX の場合 /opt/Cosminexus/CC/server/public コンテキストルート名にスラッシュ（/）などが指定されている場合，値が 3 桁の半角文字に置き換えられます。
web¥	−
<J2EE サーバ名>¥	−
<コンテキストルート名>¥	−
WEB-INF¥	−
classes¥	−
HTTP 受付のパッケージ名のディレクトリ階層¥	ディレクトリ階層は，jp/co/Hitachi/soft/csc/msg/message/reception/http/impl です。
HTTP 受付のクラスファイル	最も長いファイル名は，"CSCMsgHTTPServletContextListener.class" です。

（凡例）

−：該当する説明はありません。

Component Container の作業ディレクトリ全体のパス長は，次の見積もり式で算出できます。このパス長が OS の最大パス長を超えないようにコンテキストルートを指定してください。

Component Container の作業ディレクトリ全体のパス長=A+2×B+C+D+E+F+25

A：Component Container の作業ディレクトリまでのパスの長さ

B：Component Container の作業ディレクトリ名の長さ

C：最も長い J2EE サーバ名の長さ

D：コンテキストルートの長さ（「/」などを含む場合は文字×3 で換算します）

E：HTTP 受付のパッケージ名のディレクトリ階層の長さ

F：HTTP 受付のクラスファイルのうち，最も長いファイル名の長さ

2.16　HTTP 通信を使用したサービスとの接続

HTTP アダプタを利用することで，Web サーバ上に公開されているリソースや，REST スタイルで公開されている Web サービス（RESTful Web サービス）をビジネスプロセスから呼び出すことができます。

2.16.1　前提条件

HTTP アダプタを利用するための前提条件を次に示します。

- HTTP アダプタを使用するときは，SOAP モードを SOAP1.1/1.2 併用モードにしてください。
- HTTP アダプタは，REST サービスや Web リソースとの通信に JAX-RS 機能を使用します。
  HTTP アダプタに設定したパラメタから HTTP リクエストを生成して送信する処理や，HTTP レスポンスを受信して解析する処理は，JAX-RS 機能の動作に従います。

注意事項

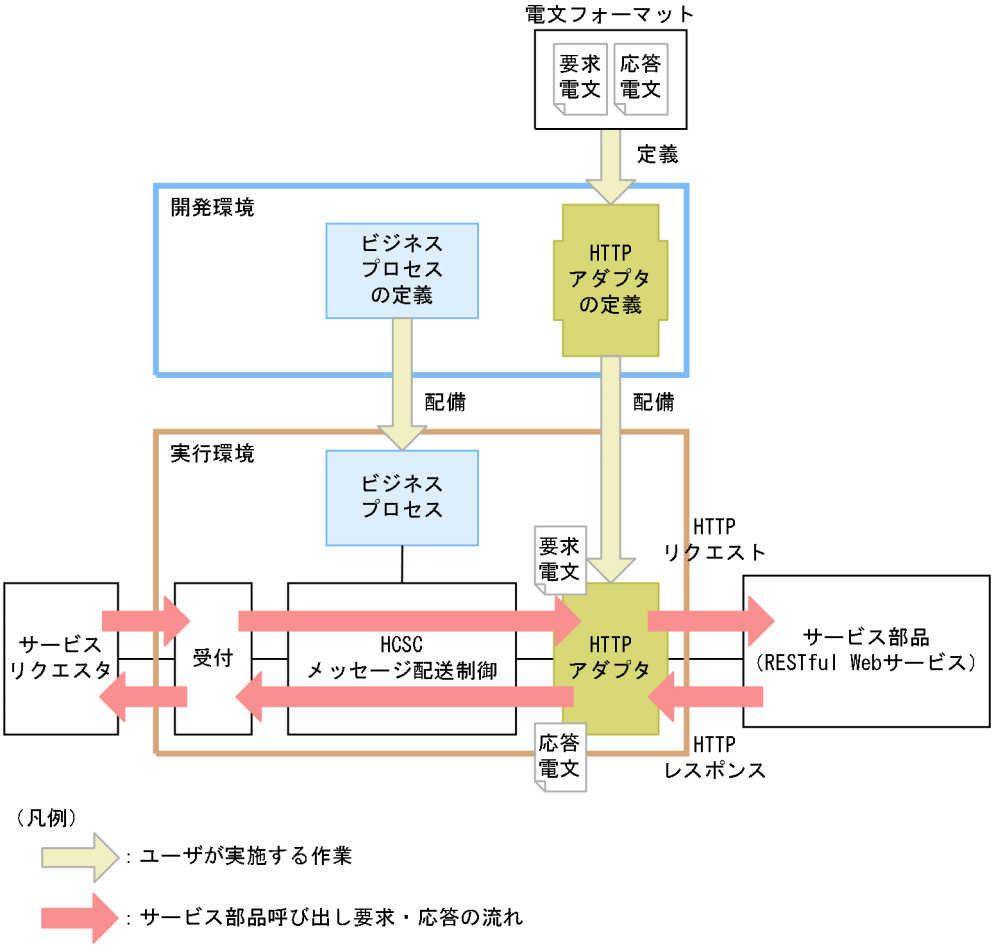
HTTP アダプタの設定は，このマニュアルに記載された方法に従ってください。

HTTP アダプタで使用する JAX-RS 機能の設定を直接変更した場合は，動作の保証はできません。

2.16.2　HTTP アダプタを使用したサービス部品の呼び出し

HTTP アダプタを使用したサービス部品呼び出し要求・応答の流れを次の図に示します。

図 2-155　HTTP アダプタを使用したサービス部品呼び出し要求・応答の流れ



サービス部品呼び出し要求時，ビジネスプロセスから渡された要求電文は HTTP アダプタで HTTP リクエストに変換されてサービス部品に送信されます。また，応答時はサービス部品から送信された HTTP レスポンスが HTTP アダプタで応答電文に変換され，ビジネスプロセスに渡されます。

HTTP アダプタは Cookie を利用したセッション維持やファイルのアップロード・ダウンロードに対応しています。また，HTTP 受付と組み合わせて利用することで，HTTP クライアントからアップロードしたファイルをサーバに送信したり，サーバからダウンロードしたファイルを HTTP クライアントに送信したりすることもできます。

2.16.3　HTTP リクエストと要求電文の関係

ここでは，HTTP アダプタの要求電文から HTTP リクエストへの変換の仕組みについて説明します。

ビジネスプロセスから渡された要求電文は，HTTP アダプタで次のように分割して HTTP リクエストに変換されます。

- リクエストライン
- HTTP リクエストヘッダ

- HTTP リクエストボディ

HTTP リクエストの設定方法および各項目の詳細について説明します。

なお，ここで説明する電文の例では，名前空間接頭辞を省略しています。

(1) HTTP リクエストの設定方法

(a) 設定個所

HTTP リクエストは，HTTP リクエストの構成要素ごとに，次の場所で設定します。

●リクエストライン

- 要求電文（ヘッダ）
- HTTP アダプタ実行環境プロパティファイル
- HTTP アダプタ実行環境共通プロパティファイル

●リクエストヘッダ

- 要求電文（ヘッダ）
- HTTP アダプタ実行環境プロパティファイル
- HTTP アダプタ実行環境共通プロパティファイル

●リクエストボディ

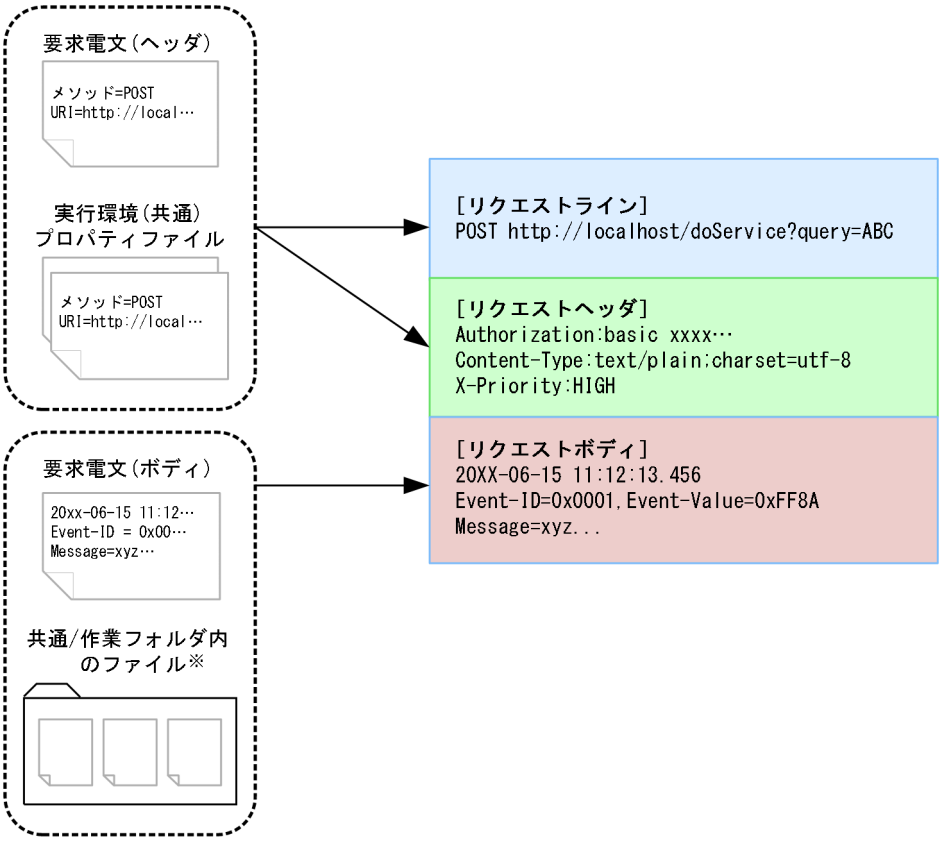
- 要求電文（ボディ）
- 作業フォルダ・共通フォルダ※内のファイル

注※

共通フォルダの場合，配下のサブフォルダを使用できます。

リクエストラインとリクエストヘッダは，上記３つの設定場所の中から任意の場所（単一の場所でも複数の場所の組み合わせでも可能）を使用して設定します。リクエストボディは要求電文（ボディ），または作業フォルダ・共通フォルダ内のファイルを使用して設定します。

図 2-156　HTTP リクエストの設定例



注※
共通フォルダの場合，配下のサブフォルダを使用できます。

(b)　リクエストライン，リクエストヘッダの設定方法

リクエストラインとリクエストヘッダは，要求電文（ヘッダ），HTTP アダプタ実行環境プロパティファイル，HTTP アダプタ実行環境共通プロパティファイルの 3 つの場所で設定します。複数の場所に設定がある場合は，HTTP アダプタは優先度が高い設定を使用して処理をします。

それぞれの設定場所の優先度は次のとおりです。

表 2-85　リクエストラインとリクエストヘッダの設定場所と優先度

優先度		設定場所
高	優先度 1	要求電文（ヘッダ）
｜	優先度 2	HTTP アダプタ実行環境プロパティファイル
低	優先度 3	HTTP アダプタ実行環境共通プロパティファイル

HTTP リクエストの設定は，要求電文（ヘッダ）や各プロパティファイルの中のパラメタ（要求電文（ヘッダ）の各要素やプロパティファイルの個々のプロパティ）単位で有効になります。

それぞれの設定場所には，次のような特長があります。システムの要件に合わせてこれらを適切に組み合わせることで，開発工数の削減や運用の容易化を図れます。

表 2-86　リクエストラインとリクエストヘッダの設定場所と特長

設定場所	特長
要求電文（ヘッダ）	HTTP アダプタの実行時に設定できるため，リクエストごとに動的に値を変更する必要があるパラメタの定義に適しています。 （例）リクエストメソッドや URI の指定，など
HTTP アダプタ実行環境プロパティファイル	HTTP アダプタの 1 定義ごとに固定のパラメタを設定することができるため，アダプタ開発時（または運用時）に静的に決まるパラメタの設定に適しています。 （例）ファイル受信（ダウンロード）時の出力先フォルダの指定，など
HTTP アダプタ実行環境共通プロパティファイル	すべての HTTP アダプタに共通のパラメタを設定することができるため，システム共通の静的なパラメタの設定に適しています。 （例）認証情報やログファイルの出力先の指定，など

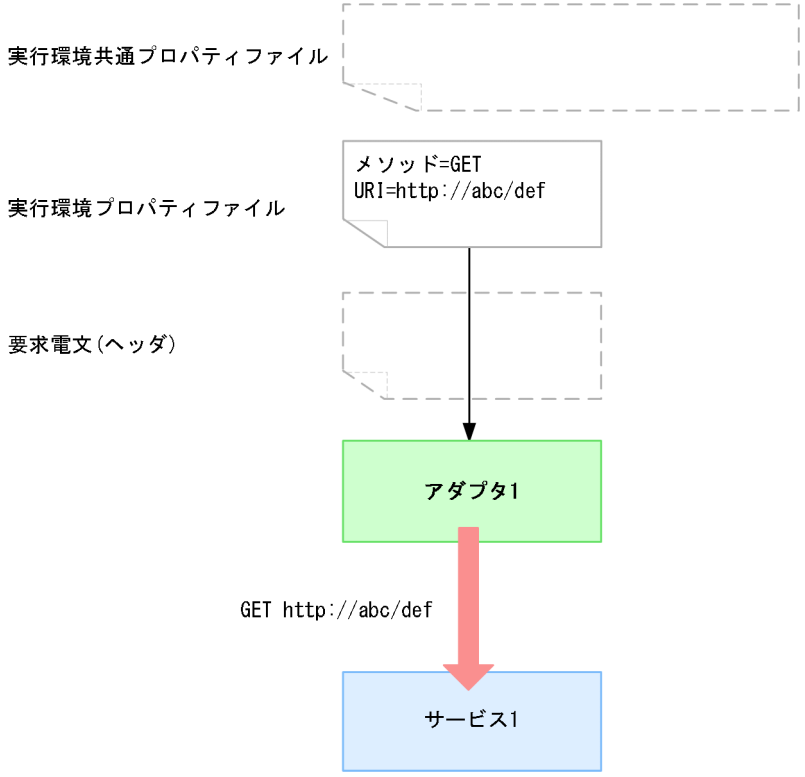
設定例

具体的な使用例を用いて，HTTP リクエストの設定方法を示します。

- パターン 1（常に固定の HTTP リクエストを発行する場合）

  HTTP アダプタから常に固定の HTTP リクエストを発行する場合，最も簡単な設定方法は，実行環境プロパティファイルによるリクエストの設定です。この方法では，HTTP アダプタは自分のサービス ID と同じ名前の実行環境プロパティファイルをアダプタ起動時に読み込み，リクエスト発行の要求があるたびに，実行環境プロパティファイルの内容に基づいて HTTP リクエストを発行します。

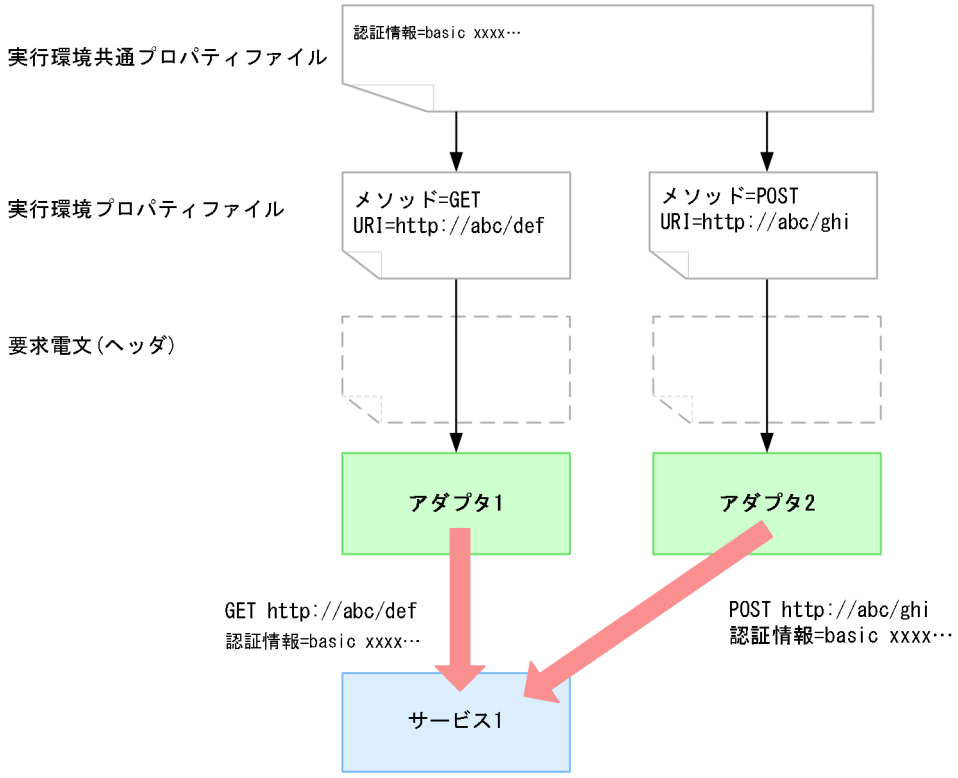
  図 2-157　HTTP リクエストの設定例（常に固定の HTTP リクエストを発行する場合）



- パターン 2（複数の HTTP アダプタの間で一部の設定を共有する場合）

複数の HTTP アダプタの間で一部の設定をシステム共通のパラメタとして共有する場合は，実行環境共通プロパティファイルにシステム共通のパラメタを切り出して設定することが有効です。次の図では，アダプタ 1 およびアダプタ 2 に共通のパラメタとしてサービス 1 に対する WWW 認証の認証情報があり，これを実行環境共通プロパティファイルで定義しています。

図 2-158　HTTP リクエストの設定例（複数の HTTP アダプタの間で一部の設定を共有する場合）



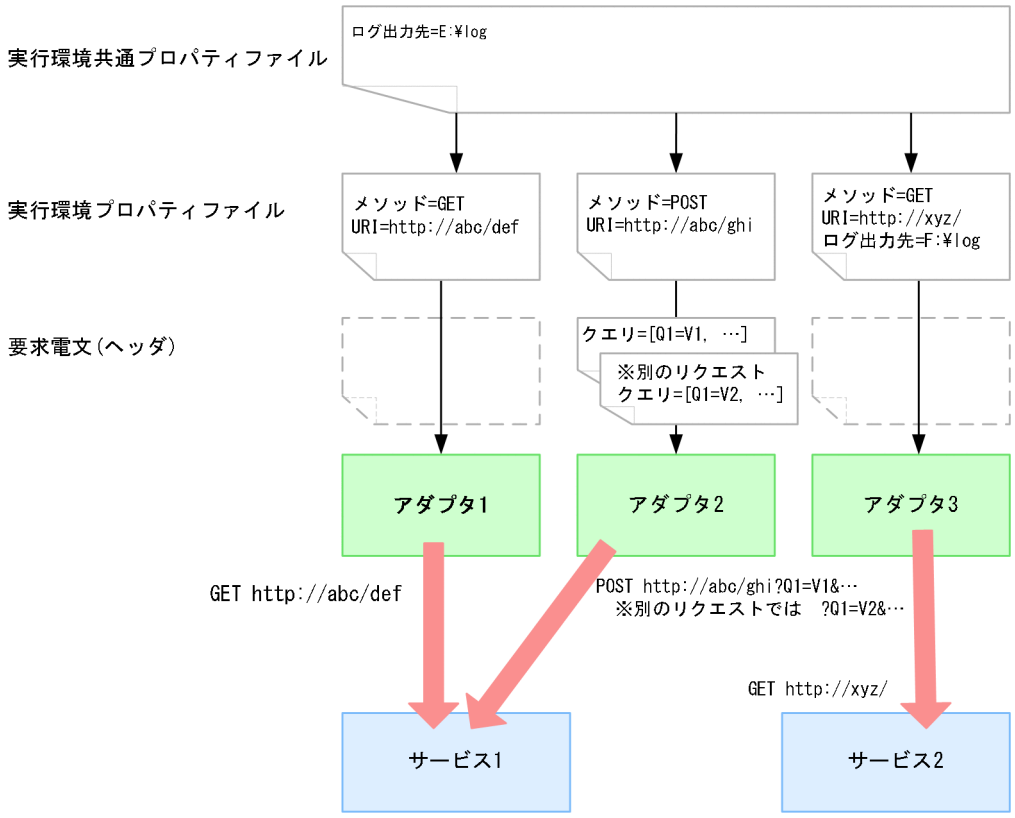
このような設定方法にすることで，共有するパラメタを重複して定義する必要がなくなるため，開発工数を削減できます。また，共通のパラメタが 1 つのプロパティファイルで一元管理されるため，設定の確認や変更作業（本例ではサービス 1 の認証情報の更新に伴う HTTP アダプタの設定変更作業）が容易になります。

- パターン 3（HTTP リクエストの内容を動的に設定する場合）

  HTTP リクエストの内容をリクエスト発行時に動的に設定する必要がある場合（ビジネスプロセスの処理状態に応じてリクエストの内容を変更する場合）は，要求電文（ヘッダ）を利用して HTTP リクエストを設定します。次の図では，ビジネスプロセスからアダプタ 2 に渡す要求電文（ヘッダ）の中にクエリ情報を定義し，HTTP リクエスト発行時にリクエストラインのクエリ情報が動的に生成されるようにしています。

  このようなアダプタ実行時に HTTP リクエストの内容を動的に変更する設定は，要求電文（ヘッダ）を使用した場合だけです。

図 2-159　HTTP リクエストの設定例（HTTP リクエストの内容を動的に設定する場合）



参考

優先度が低い設定場所で定義したパラメタは，優先度が高い設定場所で定義したパラメタで値を置き換える（上書きする）ことができます。例えば上記の図では，実行環境共通プロパティファイルを使用してすべてのアダプタのログ出力先を”E:¥log”に設定していますが，アダプタ 3 だけアダプタ固有の実行環境プロパティファイルを使用して，ログ出力先を”F:¥log”に変更しています。一部の HTTP アダプタだけ例外的な設定が必要な場合や，一時的に特定の HTTP アダプタだけ動作を変更したいなどの場合には，このような設定方法も使用できます。

具体的な設定項目や設定方法

リクエストラインやリクエストヘッダの具体的な設定項目や設定方法については，「(2) リクエストライン」，および「(3) HTTP リクエストヘッダ」を参照してください。

要求電文（ヘッダ）で設定できる要素の詳細については，マニュアル「サービスプラットフォーム 開発ガイド 受付・アダプタ定義編」の「3.3.13　HTTP アダプタを定義する」を参照してください。

HTTP アダプタ実行環境プロパティファイル，および HTTP アダプタ実行環境共通プロパティファイルで設定できるプロパティの詳細については，マニュアル「サービスプラットフォーム リファレンス」の「6.14.2　HTTP アダプタ実行環境プロパティファイル」を参照してください。

要求電文（ヘッダ）と HTTP アダプタ実行環境（共通）プロパティファイルのパラメタの対応関係については，「2.16.14　要求電文（ヘッダ）と HTTP アダプタ実行環境（共通）プロパティファイルのパラメタの対応関係」を参照してください。

注意事項

- 実行環境（共通）プロパティファイルについて

  HTTP アダプタの起動時に実行環境プロパティファイルや HTTP アダプタ実行環境共通プロパティファイルが所定の場所に配備されていない場合は，HTTP アダプタはそのプロパティファイルを使用しないものとして動作します。

  実行環境プロパティファイルと実行環境共通プロパティファイルの両方が配備されていない場合は，HTTP アダプタはこれらのプロパティファイルに対してすべてデフォルト値が指定されたものと仮定して動作します。デフォルト値がないプロパティは未定義状態になるため，必要に応じて，アダプタ実行時に要求電文（ヘッダ）でリクエストの設定を行うようにしてください。

- 要求電文（ヘッダ）について

  HTTP アダプタの開発時に要求電文用のヘッダ割当変数の指定を省略した場合は，HTTP アダプタは要求電文（ヘッダ）を使用しないものとして動作します。

  HTTP アダプタの開発時に要求電文用のヘッダ割当変数を複数指定した場合は，HTTP アダプタの要求電文（ヘッダ）の名前空間を持つヘッダ割当変数が使用されます。

  HTTP アダプタの開発時に HTTP アダプタの要求電文（ヘッダ）の名前空間を持つヘッダ割当変数を複数指定した場合は，実行時にどのヘッダ割当変数が使用されるかは不定です。

- その他全般

  複数の設定場所を使用することでシステムの開発や運用を効率化できます。しかし，必要以上に設定の階層化や複雑化をすると，実行時にどの設定場所のパラメタが使用されるか，理解することが難しくなります。最初にシステムの構成や運用業務の分析を行った上で，適切な効果が得られるように設計や開発を行ってください。

(c) リクエストボディの設定方法

リクエストボディは，リクエストボディに格納するデータと，データ送信種別（データの格納方法）のペアで指定します。それぞれ，次の場所で設定します。

●格納するデータ

要求電文（ボディ），または作業フォルダ・共通フォルダ※内のファイル

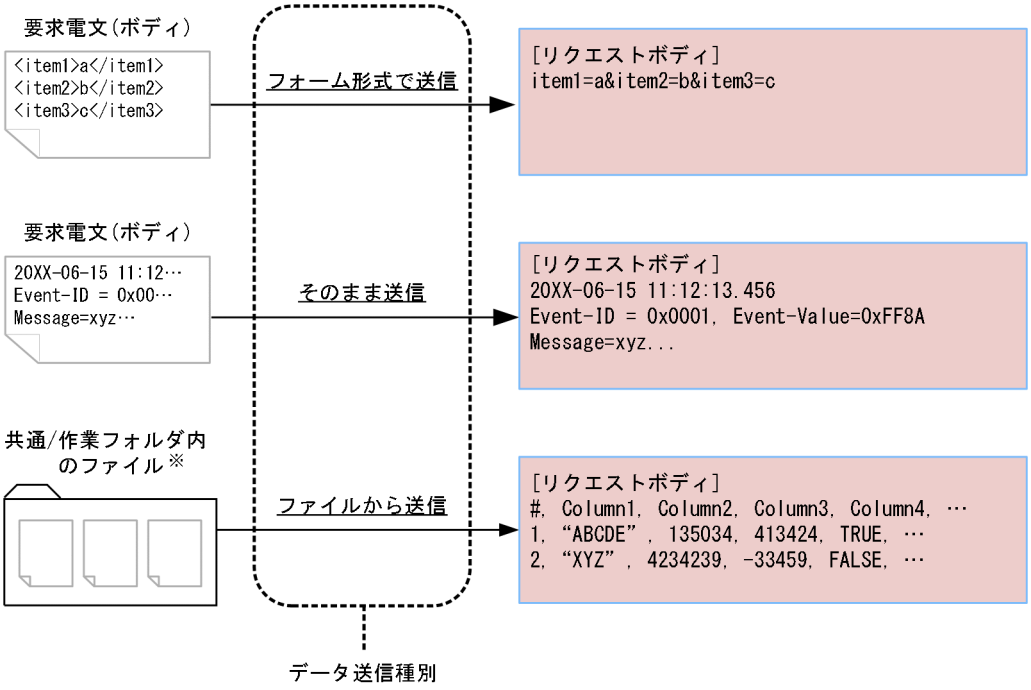
注※

共通フォルダの場合，配下のサブフォルダを使用できます。

●データ送信種別

要求電文（ヘッダ）や HTTP アダプタ実行環境（共通）プロパティファイル

データ送信種別に応じて，要求電文（ボディ），または作業フォルダ・共通フォルダ内のファイルが適切な形に変換された上でリクエストボディに格納され，リクエストとして発行されます。リクエストボディを発行する際の設定イメージを次に示します。



注※
共通フォルダの場合，配下のサブフォルダを使用できます。

具体的な設定項目や設定方法については，「(4) HTTP リクエストボディ」を参照してください。

要求電文（ボディ）で設定できる要素の詳細については，マニュアル「サービスプラットフォーム 開発ガイド 受付・アダプタ定義編」の「3.3.13　HTTP アダプタを定義する」を参照してください。

(2)　リクエストライン

HTTP リクエストの先頭行に記述するリクエストラインは次の形式で送信されます。

```
HTTPメソッド△URI△プロトコル※
```

（凡例）

△：半角スペースです。

注※

プロトコルは「HTTP/1.1」で固定です。

HTTP メソッドおよび URI の指定方法について説明します。

(a)　HTTP メソッド

リクエストラインの HTTP メソッドは，要求電文（ヘッダ），HTTP アダプタ実行環境プロパティファイル，および HTTP アダプタ実行環境共通プロパティファイルに指定します。

HTTP メソッドの指定個所を次に示します。

表 2-87　HTTP メソッドの指定個所

指定方法	項目	指定個所
要求電文（ヘッダ）	HTTP メソッド	method 要素
HTTP アダプタ実行環境プロパティファイル，HTTP アダプタ実行環境共通プロパティファイル	HTTP メソッド	adphttp.request.method プロパティ

注意事項

次のメソッドは HTTP リクエストボディを指定できません。

- GET
- DELETE
- HEAD
- OPTIONS

これらのメソッドを使用するときは，必ずデータ種別を「データなし」にしてください。HTTP リクエストボディのデータ種別については，「(4)(a) データ種別」を参照してください。

要求電文（ヘッダ），HTTP アダプタ実行環境プロパティファイル，および HTTP アダプタ実行環境共通プロパティファイルのどれにも HTTP メソッドが指定されなかった場合はエラーになります。

(b)　URI

リクエストラインの URI は，スキーム・オーソリティ部分，パス部分，クエリ部分に分割して，要求電文（ヘッダ），HTTP アダプタ実行環境プロパティファイル，および HTTP アダプタ実行環境共通プロパティファイルに指定します。

URI の指定個所を次に示します。

表 2-88　URI の指定個所

指定方法	項目	指定個所
要求電文（ヘッダ）	スキーム・オーソリティ	uri-scheme-authority 要素
	パス	uri-path 要素
	クエリ	uri-query 要素
HTTP アダプタ実行環境プロパティファイル，HTTP アダプタ実行環境共通プロパティファイル	スキーム・オーソリティ	adphttp.request.uri-scheme-authority プロパティ
	パス	adphttp.request.uri-path プロパティ
	クエリ	adphttp.request.uri-query.<追番>プロパティ

要求電文（ヘッダ），HTTP アダプタ実行環境プロパティファイル，および HTTP アダプタ実行環境共通プロパティファイルのどれにもスキーム・オーソリティとパスが指定されなかった場合はエラーになります。

スキーム・オーソリティのスキーム部分に「https」を指定した場合，HTTPS を使用して通信します。ただし，HTTPS を使用するには設定が必要です。詳細は，マニュアル「サービスプラットフォーム 開発ガイド 受付・アダプタ定義編」の「付録 I　HTTP アダプタで必要なセキュリティの設定」を参照してください。

URI の指定例を次に示します。どちらの場合も生成される URI は「http://localhost:80/test/selectdata?name=sample&id=001」です。

- 要求電文（ヘッダ）で指定した場合

```
<http-header-request>
    <uri-scheme-authority>http://localhost:80</uri-scheme-authority>
    <uri-path>/test/selectdata</uri-path>
    <uri-query>
        <name>sample</name>
        <id>001</id>
    </uri-query>
</http-header-request>
```

- HTTP アダプタ実行環境プロパティファイルで指定した場合

```
adphttp.request.uri-scheme-authority=http://localhost:80
adphttp.request.uri-path=/test/selectdata
adphttp.request.uri-query.1=name=sample
adphttp.request.uri-query.2=id=001
```

参考

スキーム・オーソリティ部分，パス部分，クエリ部分は独立して指定できます。このため，スキーム・オーソリティ部分を HTTP アダプタ実行環境プロパティファイルに指定して，パス部分とクエリ部分を要求電文（ヘッダ）で指定することで，HTTP リクエストの宛先を動的に変更することもできます。

注意事項

スキーム・オーソリティ部分とパス部分とクエリ部分には，それぞれに適切な値を設定してください。パス部分にスキーム・オーソリティやクエリを指定した場合など，不適切な値を指定した場合は動作を保証しません。

(c)　クエリ

URI のクエリ部分は，要求電文（ヘッダ）に内包されるクエリ用詳細電文フォーマットをカスタマイズして指定できます。

クエリの指定例を次に示します。

- クエリ用詳細電文フォーマットのカスタマイズの例

  (変更前)

```
<xsd:complexType name="uri-query-type">
   <xsd:sequence>
      <xsd:any namespace="##any" processContents="skip" minOccurs="0" maxOccurs="1024" />
   </xsd:sequence>
</xsd:complexType>
```

  (変更後)

```
<xsd:complexType name="uri-query-type">
   <xsd:sequence>
      <xsd:element name="parameter1" type="xsd:string" minOccurs="0"/>
      <xsd:element name="parameter2" type="xsd:string" minOccurs="0"/>
   </xsd:sequence>
 </xsd:complexType>
```

  この例では，xsd:any 型の要素を，クエリ名をローカル名とする xsd:string 型の要素に変更しています。また，出現回数の指定は任意です。

- 要求電文（ヘッダ）の指定例

```
<http-header-request>
    <uri-scheme-authority>http://localhost:80</uri-scheme-authority>
    <uri-path>/test/selectdata</uri-path>
    <uri-query>
      <parameter1>value1</parameter1>
      <parameter2>value2</parameter2>
    </uri-query>
</http-header-request>
```

- 生成されるクエリ

```
http://localhost:80/test/selectdata?parameter1=value1&parameter2=value2
```

注意事項

- uri-query 要素直下に指定した値だけ有効になり，属性を指定しても無視されます。
- 階層構造を持つ要素を指定した場合，その要素が生成するクエリの値については保証しません。
- 同一のキー名を複数定義した場合の出現順は保証しません。
- 「http://user:password@example.jp:80/httprcp1/op1」のように，URI にベーシック認証のアカウント情報が含まれていても，ベーシック認証情報として使用されません。
- 要求電文（ヘッダ）でクエリを指定する場合，クエリのキー名称には NCName として不正な文字を使用できません。

(3) HTTP リクエストヘッダ

要求電文（ヘッダ）の http-header 要素の直下に，ヘッダフィールド名を要素名，フィールド値を要素値とした形式で要求電文を生成します。要素名はローカル名の部分だけ使用するため，任意の名前空間を指定できます。要求電文中の http-header 要素を表すスキーマは，ユーザが変更できます。http-header 要素の直下の値だけが有効となり，属性の指定は無視されます。階層構造を持つ要素を指定した場合，その要素が生成するクエリの値については保証しません。

要求電文（ヘッダ）がない場合，HTTP アダプタ実行環境プロパティファイル（または HTTP アダプタ実行環境共通プロパティファイル）が参照されます。HTTP アダプタ実行環境プロパティファイルで指定したヘッダが要求電文（ヘッダ）の指定に追加されることはありません。

any 型の要素を，ヘッダ名をローカル名とする string 型の要素に変更するスキーマの修正例を次に示します。

- スキーマの修正例

  （変更前）

```
<xsd:complexType name="http-header-type">
   <xsd:sequence>
      <xsd:any namespace="##any" processContents="skip" minOccurs="0" maxOccurs="1024" />
   </xsd:sequence>
</xsd:complexType>
```

  （変更後）

```
<xsd:complexType name="http-header-type">
   <xsd:sequence>
      <xsd:element name="cache-control" type="xsd:string" minOccurs="0"/>
      <xsd:element name="host" type="xsd:string" minOccurs="0"/>
      <xsd:element name="accept-language" type="xsd:string" minOccurs="0"/>
   </xsd:sequence>
 </xsd:complexType>
```

  なお，出現回数の指定は，任意に設定できます。

- インスタンス例

```
<http-header-request>
  <http-header>
    <cache-control>no-cache</cache-control>
    <host>localhost</host>
    <accept-language>ja</accept-language>
  </http-header>
</http-header-request>
```

- 送信される HTTP ヘッダ

```
cache-control: no-cache
host: localhost
accept-language: ja
```

(a) HTTP ヘッダの指定例

HTTP ヘッダの指定例を次に示します。

- 要求電文（ヘッダ）の指定例

```
<http-header-request>
  <http-header>
    <X-PRIORITY>HIGH</X-PRIORITY>
    <X-DOCUMENT-ID>12345-67890</X-DOCUMENT-ID>
  </http-header>
</http-header-request>
```

- HTTP アダプタ実行環境プロパティファイルの指定例

```
adphttp.request.header.userdef.1=X-PRIORITY:HIGH
adphttp.request.header.userdef.2=X-DOCUMENT-ID:12345-67890
```

- 生成される HTTP リクエストヘッダ

```
X-PRIORITY:HIGH
X-DOCUMENT-ID:12345-67890
```

(b) 特殊な HTTP ヘッダの指定

次に示す HTTP ヘッダは，要求電文（ヘッダ）の http-header 要素，および HTTP アダプタ実行環境プロパティファイル（または HTTP アダプタ実行環境共通プロパティファイル）の adphttp.request.header.userdef.<追番>プロパティに指定しても無視されます。

表 2-89 特殊な HTTP ヘッダの指定方法

ヘッダフィールド名	説明
Content-Type	次の個所で指定します。 • 要求電文（ヘッダ） http-header-Content-Type 要素 • HTTP アダプタ実行環境プロパティファイル，HTTP アダプタ実行環境(共通)プロパティファイル adphttp.request.header.content-type プロパティ，adphttp.request.header.content-type.charset プロパティ フォームデータ送信の場合は application/x-www-form-urlencoded 固定，複数ファイルデータ送信の場合は multipart/form-data 固定となります。
Cookie	次の個所で指定します。 • 要求電文（ヘッダ） Cookies 要素 詳細は，「2.16.5 HTTP アダプタを使用した Cookie 情報の引き継ぎ」を参照してください。
Proxy-Authorization	プロキシサーバへの接続に使用する Proxy-Authorization ヘッダは JavaVM のシステムプロパティで設定します。 詳細は，「2.16.6 プロキシサーバを経由した通信」を参照してください。

ヘッダフィールド名	説明
Connection	Connection: Keep-Alive など，コネクション制御に関するヘッダは JavaVM のシステムプロパティで設定します。詳細は，「2.16.7　コネクション継続（Keep-Alive）を使用した通信」を参照してください。
Content-Length	HTTP アダプタが自動的に設定します。
Accept-Encoding	HTTP アダプタが自動的に設定します。
Authorization	次の個所で指定します。 • 要求電文（ヘッダ） http-header-Authorization 要素 • HTTP アダプタ実行環境プロパティファイル，HTTP アダプタ実行環境(共通)プロパティファイル adphttp.request.header.authorization プロパティ， adphttp.request.header.authorization.type プロパティ 詳細は，「2.16.16　WWW 認証を使用したセキュアな接続」を参照してください。
Content-Disposition	Content-Disposition ヘッダには，name および filename 属性だけ指定してください。 複数のファイルデータを送信する場合，Content-Disposition ヘッダに送信するリソースのファイル名を指定します。 HTTP アダプタでは，次の形式でメッセージを作成します。 `Content-Disposition: <Dispositionタイプ>; name=<パートID>; filename=<ファイル名>` 詳細は，「2.15.5(2)(b) Content-Disposition」を参照してください。

(c)　注意事項

HTTP ヘッダ指定時の注意事項を次に示します。

- http-header 要素に指定した値は，http-header 要素直下の値だけ有効になり，属性を指定しても無視されます。
- http-header 要素に階層構造を持つ要素を指定した場合，その要素が生成するヘッダの値については保証しません。
- http-header 要素直下のリクエストヘッダは 1,024 個まで指定できます。1,025 以上の要素を指定した場合は動作を保証しません。
- http-header 要素で HTTP ヘッダを指定する場合，ヘッダフィールド名には NCName として不正な文字を使用できません。
- HTTP アダプタでは HTTP ヘッダの記述順を保証しません。また，同名のヘッダフィールドが複数あった場合，そのまま HTTP リクエストに設定されます。
- 要求電文（ヘッダ），HTTP アダプタ実行環境プロパティファイル，または HTTP アダプタ実行環境共通プロパティファイルのどれにも HTTP ヘッダおよび Content-Type ヘッダを指定しなかった場合，送信内容に応じて自動的に Content-Type ヘッダが付与される場合があります。
- 拡張ヘッダ以外のヘッダフィールド名を指定した場合の動作は保証されません。拡張ヘッダ以外のヘッダフィールド名を指定する場合は，十分に動作検証をした上で利用してください。

(4) HTTP リクエストボディ

(a) データ種別

HTTP リクエストボディで送信するデータ種別は次の中から選択できます。

- フォームデータ（ノーマルモード）
  要求電文（ボディ）に指定したデータをフォームデータに変換して，HTTP リクエストボディに設定します。
- 要求電文（ボディ）（パススルーモード）
  要求電文（ボディ）で指定したデータをそのまま HTTP リクエストボディに設定します。
- 単一ファイルのファイルデータ送信（単一パート型）
  ファイルの内容が HTTP リクエストボディに設定されます。
- 複数ファイルのファイルデータ送信（マルチパート型）
  複数のファイルのデータをマルチパート型で送信できます。マルチパートの一部に要求電文（ボディ）を組み合わせることもできます。
- データなし
  HTTP リクエストボディが設定されません。GET メソッドなど，HTTP リクエストボディを送信しない場合に選択します。

HTTP リクエストボディのデータ種別は，要求電文（ヘッダ）の binding 要素，および HTTP アダプタ実行環境プロパティファイル（または HTTP アダプタ実行環境共通プロパティファイル）の adphttp.request.part.message.binding プロパティで指定できます。

ただし，要求電文（ヘッダ）に files 要素が存在する場合は，上記の要素（またはプロパティ）が指定されていてもファイルデータとして扱われます。同様に，HTTP アダプタ実行環境プロパティファイル（または HTTP アダプタ実行環境共通プロパティファイル）の adphttp.request.part.file.<追番>.input-folder-name プロパティが存在する場合は，adphttp.request.part.message.binding プロパティが指定されていてもファイルデータとして扱われます。

(b) フォームデータ（ノーマルモード）

ノーマルモードでフォームデータを送信する場合，サービスプラットフォームが提供するフォームデータ用電文フォーマットをカスタマイズして指定します。

フォームデータの指定例を次に示します。

- フォームデータ用電文フォーマットのカスタマイズの例
  （変更前）

```
<xsd:complexType name="http-body-form-data-type">
   <xsd:sequence>
      <xsd:any namespace="##any" processContents="skip" minOccurs="0" maxOccurs="unbounde
d" />
```

```
   </xsd:sequence>
</xsd:complexType>
```

（変更後）

```
<xsd:complexType name="http-body-form-data-type">
   <xsd:sequence>
      <xsd:element name="parameter1" type="xsd:string" minOccurs="0"/>
      <xsd:element name="parameter2" type="xsd:string" minOccurs="0"/>
   </xsd:sequence>
</xsd:complexType>
```

- 要求電文（ボディ）の指定例

```
   <http-body-form-data>
      <parameter1>value1<parameter1>
      <parameter2>value2<parameter2>
   </http-body-form-data>
```

- 生成される HTTP リクエストヘッダおよびリクエストボディ

```
Content-Type: application/x-www-form-urlencoded

parameter1=value1&parameter2=value2
```

指定したフォームデータは URL エンコードされます。デフォルトの文字コードは UTF-8 です。

注意事項

- http-body-formdata 要素に指定した値は，http-body-formdata 要素直下の値だけ有効になり，属性を指定しても無視されます。
- http-body-formdata 要素に階層構造を持つ要素を指定した場合，その要素が生成するフォームデータの値については保証しません。
- HTTP アダプタではフォームデータの出現順を保証しません。フォームデータの出現順は要求電文（ボディ）の指定順と必ずしも一致するとは限りません。
- http-body-formdata 要素に指定した値は Content-Type ヘッダの charset 属性に指定した文字コードで URL エンコードされます。このため，必ず URL エンコードをする前の形式で指定してください。
- http-body-formdata 要素でフォームデータを指定する場合，フォームデータのキー名称には NCName として不正な文字を使用できません。

(c) 要求電文（ボディ）（パススルーモード）

パススルーモードで要求電文（ボディ）で指定したデータをそのまま HTTP リクエストボディに設定する場合，送信する HTTP リクエストボディの形式に合わせて要求電文（ボディ）の電文フォーマットを作成します。

(d) 単一ファイルのファイルデータ送信（単一パート型）

作業フォルダまたは共通フォルダ内の 1 ファイルを対象のサーバに送信できます。

複数ファイルを送信する場合は，「(e) 複数ファイルのファイルデータ送信（マルチパート型）」を参照してください。

作業フォルダとは，HTTP 受付または FTP 受付がリクエストごとに HCSC サーバ内に作成する一時フォルダです。作業フォルダは，作成した受付から呼び出されたビジネスプロセス以外からは参照できません。

共通フォルダとは，システム構築時にユーザが作成する任意のフォルダです。複数のビジネスプロセスで共通して利用するファイルを格納することができます。また，共通フォルダの場合，配下のサブフォルダを使用できます。

ファイルデータを送信する場合，要求電文（ヘッダ），および HTTP アダプタ実行環境プロパティファイル（または HTTP アダプタ実行環境共通プロパティファイル）でファイルデータに関する情報を指定します。プロパティファイルで指定する場合は共通フォルダだけ使用できます。

ファイルデータの指定個所を次に示します。

表 2-90　単一ファイルデータの指定方法

指定方法	項目	指定個所
要求電文（ヘッダ）	作業フォルダのリクエスト ID	request-id 要素
	パート型※1	http-body 要素内の style 要素
	使用するフォルダ	input-folder-name 要素の common 属性
	共通フォルダ定義名	input-folder-name 要素
	送信対象のファイル名※	local-file-name 要素
	送信データの Content-Type ヘッダフィールドのメディアタイプ	http-header-Content-Type 要素
	送信データの Content-Type ヘッダフィールドの文字コード属性	http-header-Content-Type 要素の charset 属性
HTTP アダプタ実行環境プロパティファイル，HTTP アダプタ実行環境共通プロパティファイル	作業フォルダのリクエスト ID	−
	ファイルデータ数	−
	使用するフォルダ	−
	共通フォルダ定義名	adphttp.request.part.file.<追番>.input-folder-name※2
	送信対象のファイル名※3	adphttp.request.part.file.<追番>.local-file-name※2

指定方法	項目	指定個所
HTTP アダプタ実行環境プロパティファイル，HTTP アダプタ実行環境共通プロパティファイル	送信データの Content-Type ヘッダフィールドのメディアタイプ	adphttp.request.header.content-type
	送信データの Content-Type ヘッダフィールドの文字コード属性	adphttp.request.header.content-type.charset

（凡例）

－：該当する指定個所はありません。

注※1

単一のファイルデータを送信する場合は，省略または「single」を指定します。

注※2

1 以外の<追番>のプロパティが指定されている場合，該当するプロパティが解析されます。ただし，リクエスト時には<追番>が 1 のプロパティがファイルデータとして使用されます。

注※3

サブフォルダを使用する場合，共通フォルダルートからファイル名までの相対パスを指定してください。

単一ファイルのデータ送信の例を次に示します。

図 2-160　単一ファイルのデータ送信の例（共通フォルダを使用する場合）

●要求電文（ヘッダ）

```
<http-header-request>
  <request-id/>

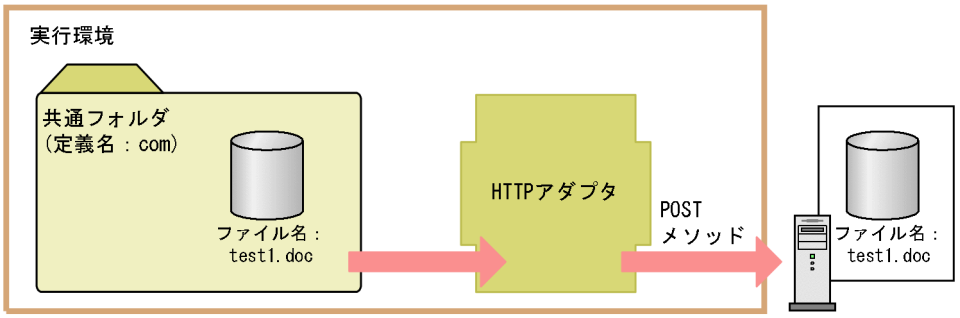
<http-header-Content-Type>application/msword</http-header-Content-Type>
 :
  <http-part>
    <files>
      <file>
        <input-folder-name common="true">com</input-folder-name>
        <local-file-name>test1.doc</local-file-name>
      </file>
    </files>
  </http-part>
</http-header-request>
```

●HTTPリクエスト

```
POST  http://host/test?filename=test1.doc
host:test
Content-Type: application/msword

    :
```

ここにtest1.docのデータが設定されます。



（凡例）

：HTTPリクエストの流れ

ファイルデータを送信する場合，HTTP アダプタに渡された要求電文（ボディ）の値は無視されます。このため，指定する要求電文（ボディ）の電文フォーマットには，空のボディデータを格納するためのファイルデータ用電文フォーマットを使用します。

注意事項

- 要求電文（ヘッダ）で files 要素だけ指定して下位の file 要素を 1 つも指定しなかった場合，HTTP リクエストボディが空の状態で HTTP リクエストが送信されます。
- /http-header-request/http-part/files/file 要素が複数回出現した場合，すべての file 要素が解析されます。ただし，リクエスト時には最初に出現した要素の値をファイルデータとして使用されます。
- ファイルデータを送信する場合，読み込み対象のファイルが格納されたフォルダまたはファイルの指定がない場合や，指定されたフォルダおよびファイルが存在しない場合はエラーとなります。

次に，共通フォルダのサブフォルダを使用して，HTTP リクエストでファイルを送信する場合の指定例を示します。

HTTP アダプタ要求電文（ヘッダ）の/http-header-request/http-part/files/file/local-file-name，または HTTP アダプタ実行環境プロパティファイルの adphttp.request.part.file.<追番>.local-file-name に共通フォルダルートからファイル名までの相対パスを指定します。

- 要求電文（ヘッダ）指定例

```
<http-header-request>
      :
  <input-folder-name common="true">com</input-folder-name>
  <local-file-name>/folder1/folder2/file1.doc</local-file-name>
      :
</http-header-request>
```

- HTTP アダプタ実行環境（共通）プロパティファイルの指定例

```
adphttp.request.part.file.<追番>.input-folder-name=com
adphttp.request.part.file.<追番>.local-file-name=/folder1/folder2/file1.doc
```

(e) 複数ファイルのファイルデータ送信（マルチパート型）

作業フォルダまたは共通フォルダ内の複数のファイルを，マルチパート型の一部としてリクエストボディに設定し，対象のサーバに送信できます。

送信するファイルデータの情報は，要求電文（ヘッダ），または HTTP アダプタ実行環境（共通）プロパティファイルで指定します。

注意事項

複数ファイルのファイルデータを送信する場合，パートが 1 つ以上設定されている必要があります。パートが設定されていない場合，エラーメッセージが出力され，リクエストに失敗します。

要求電文（ヘッダ）での複数ファイルデータの指定

要求電文（ボディ）をマルチパート型の 1 パートとして扱います。要求電文（ボディ）をマルチパート型の 1 パートとして扱う場合，要求電文（ボディ）リクエストボディへのバインド方法（adphttp.request.part.message.binding プロパティ）には「raw」を指定します。「raw」以外を指定した場合，要求電文（ボディ）はマルチパート型の 1 パートとして扱われません。ファイルデータだけを送信したいときは，ボディ要求電文の割当変数に，製品が提供する空データ用フォーマットを指定します。

また，/http-header-request/http-part/files/file 要素ごとにファイル単位で，共通フォルダと作業フォルダのそれぞれに格納したファイルを指定できます。

表 2-91　複数ファイルデータの指定方法（要求電文（ヘッダ））

項目	指定個所	説明
作業フォルダのリクエスト ID	/http-header-request/request-id	作業フォルダ用のリクエスト ID を設定します。作業フォルダを使用しない場合は，省略できます。
HTTP ボディのパート型	/http-header-request/http-body/style	HTTP ボディの型を指定します。マルチパート型の場合は，「multipart/form-data」を指定します。
Content-Disposition ヘッダのパート名	/http-header-request/http-part/message/part-id	要求電文（ボディ）をマルチパート型の 1 パートとして扱う場合，該当するパートに設定する Content-Disposition ヘッダの name 属性の値を指定します。 「csc_http_adapter_telegram_name」で始まる値は，設定できません。設定した場合の動作は保証されません。 次のすべての条件に当てはまる場合に，この要素値に「csc_http_adapter_telegram_name_1」が指定されたと見なします。 • リクエストボディのパート形式が「multipart/form-data」である • 要求電文（ボディ）のリクエストボディへのバインド方法（adphttp.request.part.message.binding プロパティ）が「raw」である • この要素および HTTP アダプタ実行環境プロパティファイルに，「adphttp.request.part.message.part-id」が設定されていない
Content-Type ヘッダの文字コード属性	/http-header-request/http-part/message/Content-Type/@charset	要求電文（ボディ）をマルチパート型の 1 パートとして扱う場合，該当するパートに設定する Content-Type ヘッダの charset 属性の値を指定します。
Content-Type ヘッダのメディアタイプ	/http-header-request/http-part/message/Content-Type	要求電文（ボディ）をマルチパート型の 1 パートとして扱う場合，該当するパートに設定する Content-Type ヘッダのメディアタイプを指定します。
Content-Disposition ヘッダの送信対象のファイル名	/http-header-request/http-part/files/file/part-id	送信対象ファイルをマルチパート型の 1 パートとして扱う場合，該当するパートに設定する Content-Disposition ヘッダの name 属性の値を指定します。 「csc_http_adapter_file_name」で始まる値は設定できません。設定した場合の動作は保証されません。 次のすべての条件に当てはまる場合に，この要素値に「csc_http_adapter_file_name_<1 から始まる連番>」が設定されたと見なします。連番は製品内で採番されます。 • リクエストボディのパート形式が「multipart/form-data」である • 要求電文（ヘッダ）の「/http-header-request/http-part/files/file」要素が出現する • 要求電文（ヘッダ）の「/http-header-request/http-part/files/file/part-id」要素がない，または空要素である
使用するフォルダ	/http-header-request/http-part/files/file/input-folder-name/@common	共通フォルダを使うか，作業フォルダを使うかを指定します。 true：共通フォルダを使用する。 false：作業フォルダを使用する。

項目	指定個所	説明
共通フォルダ論理名	/http-header-request/http-part/files/file/input-folder-name	共通フォルダを使う場合の共通フォルダ論理名を指定します。
送信対象のファイル名	/http-header-request/http-part/files/file/local-file-name	送信対象のファイル名を指定します。 共通フォルダの場合，共通フォルダルート配下のファイル名を指定します。作業フォルダの場合，作業フォルダ直下のファイル名を指定します。
Content-Disposition ヘッダの送信対象のファイル名	/http-header-request/http-part/files/file/file-name	送信対象ファイルをマルチパート型の 1 パートとして扱う場合，該当するパートに設定する Content-Disposition ヘッダの filename 属性の値を指定します。
Content-Type ヘッダの文字コード属性	/http-header-request/http-part/files/file/Content-Type/@charset	送信対象ファイルをマルチパート型の 1 パートとして扱う場合，該当するパートに設定する Content-Type ヘッダの charset 属性の値を指定します。
Content-Type ヘッダメディアタイプ	/http-header-request/http-part/files/file/Content-Type	送信対象ファイルをマルチパート型の 1 パートとして扱う場合，該当するパートに設定する Content-Type ヘッダのメディアタイプを指定します。
送信データの Content-Type ヘッダフィールドのメディアタイプ	/http-header-request/http-header-Content-Type	送信データの Content-Type ヘッダフィールドのメディアタイプを指定します。
送信データの Content-Type ヘッダフィールドの文字コード属性	/http-header-request/http-header-Content-Type/@charset	送信データの Content-Type ヘッダフィールドの charset 属性値を指定します。

HTTP アダプタ実行環境（共通）プロパティファイルでの複数ファイルデータの指定

HTTP アダプタ実行環境（共通）プロパティファイルで指定する際に使用するプロパティを次に示します。

表 2-92　複数ファイルデータの指定方法（HTTP アダプタ実行環境（共通）プロパティファイル）

項目	指定個所	説明
リクエストボディのパート型	adphttp.request.body.style	リクエストボディのパート型を指定します。
リクエストボディのパート ID	adphttp.request.part.message.part-id	リクエストボディのパートに付与する ID を指定します。 各パートのヘッダの Content-Disposition の name パラメタの値として使用します。 「csc_http_adapter_telegram_name」で始まる値は，設定できません。設定した場合の動作は保証されません。 次のすべての条件に当てはまる場合，この定義値に「csc_http_adapter_telegram_name_1」が指定され，

項目	指定個所	説明
リクエストボディのパート ID	adphttp.request.part.message.part-id	「adphttp.request.part.message.part-id」の値に設定されたと見なされます。 • リクエストボディのパート形式が「multipart/form-data」である • 要求電文（ボディ）のリクエストボディへのバインド方法（adphttp.request.part.message.binding プロパティ）が「raw」である • この定義が設定されていない
リクエストボディの Content-Type の文字コード	adphttp.request.part.message.content-type.charset	リクエストボディのパートの Content-Type の文字コードを指定します。 各パートのヘッダの Content-Type の charset パラメタとして使用します。
リクエストボディの Content-Type の値	adphttp.request.part.message.content-type	リクエストボディのパートの Content-Type の値を指定します。 各パートのヘッダの Content-Type の値として使用します。
ファイルのパート ID	adphttp.request.part.file.<追番>.part-id※	ファイルのパートに付与する ID を指定します。 マルチパート形式時に各パートのヘッダの Content-Disposition の name パラメタの値として使用します。 「csc_http_adapter_file_name」で始まる値は，設定できません。設定した場合の動作は保証されません。 次のすべての条件に当てはまる場合，この定義値に「csc_http_adapter_file_name_<1 から始まる連番>」が設定されたと見なされます。連番は製品内で採番されます。 •「adphttp.request.body.style」の値に「multipart/form-data」を指定した •「adphttp.request.part.file.<追番>.input-folder-name」または「adphttp.request.part.file.<追番>.input-folder-name」が設定されている •「adphttp.request.part.file.<追番>.part-id」が未定義，または未設定
共通フォルダ論理名	adphttp.request.part.file.<追番>.input-folder-name※	共通フォルダ論理名を指定します。
送信対象のファイル名	adphttp.request.part.file.<追番>.local-file-name※	送信対象のファイル名を指定します。 共通フォルダルート配下のファイル名を指定する。
送信時のファイル名	adphttp.request.part.file.<追番>.file-name※	送信時のファイルの名称を指定します。 各パートのヘッダの Content-Disposition の filename パラメタの値として使用します。
ファイルの Content-Type の文字コード	adphttp.request.part.file.<追番>.content-type.charset※	ファイルのパートの Content-Type の文字コードを指定します。 各パートのヘッダの Content-Type の charset パラメタとして使用します。
ファイルの Content-Type の値	adphttp.request.part.file.<追番>.content-type※	ファイルのパートの Content-Type の値を指定します。 各パートのヘッダの Content-Type の値として使用します。

項目	指定個所	説明
送信データの Content-Type ヘッダフィールドのメディアタイプ	adphttp.request.header.content-type	送信データの Content-Type ヘッダフィールドのメディアタイプを指定します。
送信データの Content-Type ヘッダフィールドの文字コード属性	adphttp.request.header.content-type.charset	送信データの Content-Type ヘッダフィールドの charset 属性値を指定します。

注※

<追番>は 1 から始まる連番です。連番は次の規則で設定されます。

・数値の前に 0 は付けない（001 など）。

・連続しない数でもよい（例：1，3，5）。

・プロパティキーの一意性を示す。参照順は関係しない。

・プロパティの組で，<追番>は同一の値となる。

複数ファイルのデータ送信の例

共通フォルダを使用した場合の複数ファイルのデータ送信の例を次に示します。

図 2-161　複数ファイルのデータ送信の例（共通フォルダを使用する場合）

●要求電文（ヘッダ）

```
<http-header-request>
  <request-id/>

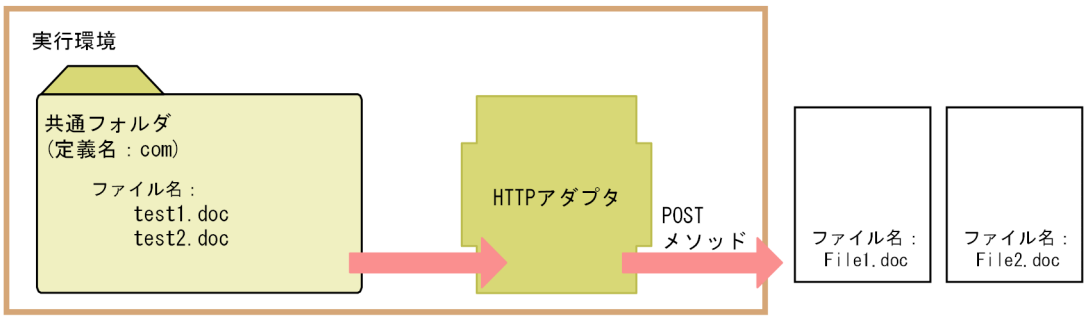
…
<http-body>
    <style>multipart/form-data</style>
</http-body>
  <http-part>
    <files>
      <file>
        <part-id>multipart-file1</part-id>
        <input-folder-name common="true">com</input-folder-name>
        <local-file-name>test1.doc</local-file-name>
        <file-name>File1</file-name>
        <Content-Type charset="UTF-8">text/plain</Content-Type>
      </file>
      <file>
        <part-id>multipart-file2</part-id>
        <input-folder-name common="true">com</input-folder-name>
        <local-file-name>test2.doc</local-file-name>
        <file-name>File2</file-name>
        <Content-Type charset="UTF-8">text/plain</Content-Type>
      </file>
    </files>
  </http-part>
</http-header-request>
```

●HTTPリクエスト

```
POST  http://host/test?filename=test1.doc,test2.doc(クエリでユーザが設定)
host:test
Content-Type: multipart/form-data; boundary="abcde12345"

--abcde12345
Content-Disposition: form-data; name="multipart-file1"; filename="File1"
Content-Type: text/plain
…

--abcde12345
Content-Disposition: form-data; name="multipart-file2"; filename="File2"
Content-Type: text/plain
…
```



（凡例）
：HTTPリクエストの流れ

2.16.4　HTTP レスポンスと応答電文の関係

ここでは，HTTP アダプタの HTTP レスポンスから応答電文への変換の仕組みについて説明します。

サーバから返された HTTP レスポンスは，次の要素ごとに処理されて，応答電文に格納されます。

- ステータスライン

- HTTP レスポンスヘッダ
- HTTP レスポンスボディ

各項目の詳細について説明します。

なお，ここで説明する電文の例では，名前空間接頭辞を省略しています。

(1) ステータスライン

HTTP レスポンスのステータスラインの要素は次のように応答電文に格納されます。

表 2-93 ステータスラインと応答電文の対応

ステータスラインの要素	応答電文との対応
HTTP バージョン	応答電文には格納されません。
ステータスコード	応答電文（ヘッダ）の status-code 要素に格納されます。※
理由フレーズ	応答電文（ヘッダ）の reason-phrase 要素に格納されます。

注※

200 以外のステータスコードはデフォルトでシステム例外となります。ただし，200 以外も正常処理，またはフォルト応答としてフォルトを返すように設定することもできます。詳細は，「(4) ステータスコードの正常処理またはフォルト処理の設定」を参照してください。

(2) HTTP レスポンスヘッダ

HTTP アダプタで受信したすべての HTTP ヘッダは応答電文（ヘッダ）の http-header 要素に格納されます。同一名称の HTTP ヘッダを重複して受信した場合はそのまま複数の要素として格納されます。

HTTP レスポンスヘッダの例と対応する応答電文（ヘッダ）を次に示します。

- HTTP レスポンスヘッダの例

```
Content-Length: 51
Content-Type: image/gif
Last-Modified: Fri, 21 Jul 2006 07:51:32 GMT
Accept-Ranges: bytes
Server: Microsoft-IIS/6.0
Date: Mon, 16 Jul 2012 09:43:57 GMT
```

- 対応する応答電文（ヘッダ）

```
<http-header-response>
  <http-header>
    <Content-Length>51</Content-Length>
    <Content-Type>image/gif</Content-Type>
    <Last-Modified>Fri, 21 Jul 2006 07:51:32 GMT</Last-Modified>
    <Accept-Ranges>bytes</Accept-Ranges>
    <Server>Microsoft-IIS/6.0</Server>
    <Date>Mon, 16 Jul 2012 09:43:57 GMT</Date>
```

```
  </http-header>
</http-header-response>
```

注意事項

HTTP ヘッダの出現順がそのまま XML 要素の出現順になるとは限りません。なお，受信した HTTP ヘッダのヘッダフィールド名に NCName として不正な文字が使用されていた場合，警告メッセージが出力され，該当するヘッダフィールドをスキップして処理が続行されます。

HTTP レスポンスヘッダの Content-Type ヘッダと Content-Disposition ヘッダは応答電文（ヘッダ）の次の要素にも格納されます。

表 2-94　Content-Type ヘッダおよび Content-Disposition ヘッダと応答電文（ヘッダ）の対応

HTTP レスポンスヘッダ	属性	応答電文（ヘッダ）に格納される要素
Content-Type	−	http-header-Content-Type 要素に Content-Type ヘッダフィールドのメディアタイプ部分(*/*)が格納されます。
	charset	http-header-Content-Type 要素の charset 属性に属性値が格納されます。charset 属性の指定がない場合は格納されません。
Content-Disposition	−	HTTP レスポンスで指定があった場合，HTTP レスポンスはファイルデータとして扱われます。Disposition タイプの値は無視されます。
	filename	file 要素の子要素の filename 要素として値が格納されます。指定値はデコードなどの変換は行われません。

（凡例）

−：該当しません。

応答電文（ヘッダ）に格納される要素は次に示す値だけです。これ以外の値はすべて無視されます。

- Content-Type ヘッダのメディアタイプ
- Content-Type ヘッダの charset 属性
- Content-Disposition ヘッダの filename 属性

注意事項

応答電文（ヘッダ）を省略した場合，HTTP レスポンスから応答電文（ヘッダ）に格納された要素はビジネスプロセスで使用できなくなります。

(3)　HTTP レスポンスボディ

HTTP レスポンスボディは，HTTP レスポンスヘッダに Content-Disposition ヘッダが指定されているかどうかによって次のように扱われます。

- ファイルデータ

HTTP レスポンスヘッダに Content-Disposition ヘッダが指定されている場合，HTTP レスポンスボディに指定された任意のデータがファイルに出力されます。出力ファイルのメタ情報は応答電文（ヘッダ）に格納されます。

- ローデータ（パススルーモード）

  HTTP レスポンスヘッダに Content-Disposition ヘッダが指定されていない場合，HTTP レスポンスボディに指定された任意のデータがそのまま応答電文（ボディ）に格納されます。応答電文（ボディ）の電文フォーマットはユーザが作成します。

(a) ファイルデータ

ファイルデータを受信する場合，出力先のディレクトリを要求電文（ヘッダ），HTTP アダプタ実行環境プロパティファイル，または HTTP アダプタ実行環境共通プロパティファイルで指定する必要があります。プロパティファイルで指定する場合は共通フォルダだけ使用できます。

Content-Type が「multipart/form-data」，または「multipart/mixed」の場合，HTTP レスポンスボディを各パートに分割し，各パートの内容をそれぞれファイルとして出力します。ファイルは，指定した 1 つの出力ディレクトリに出力されます。複数の出力先フォルダは，選択できません。

Content-Type が「multipart/form-data」または「multipart/mixed」で，HTTP レスポンスボディに 1 つもパートが含まれていない場合は，リクエストがエラーとなります。

各パートの MIME タイプが省略されている場合は，該当するパートの解析処理では「text/plain」が指定されたと見なして処理が行われます。

要求電文（ヘッダ），HTTP アダプタ実行環境プロパティファイル，および HTTP アダプタ実行環境共通プロパティファイルでのファイルデータの指定個所を次に示します。

表 2-95　ファイルデータの指定個所

指定方法	指定する要素またはプロパティ	説明
要求電文（ヘッダ）	request-id 要素	作業フォルダ用のリクエスト ID を設定します。作業フォルダを使用しない場合は省略できます。
	output-folder-name 要素の common 属性	ファイルデータの出力先として共通フォルダまたは作業フォルダのどちらかを指定します。
	output-folder-name 要素	ファイルデータを出力する共通フォルダ定義名を指定します。作業フォルダを使用する場合は指定を省略できます。
	output-sub-folder 要素	ファイルデータの出力先として共通フォルダのサブフォルダを使用する場合に，共通フォルダルートからサブフォルダまでの相対パスを指定します。
HTTP アダプタ実行環境プロパティファイル，HTTP アダプタ実行環境共通プロパティファイル	adphttp.request.output-folder-name プロパティ	ファイルデータを出力する共通フォルダ定義名を指定します。

指定方法	指定する要素またはプロパティ	説明
HTTP アダプタ実行環境プロパティファイル，HTTP アダプタ実行環境共通プロパティファイル	adphttp.request.output-sub-folder プロパティ	ファイルデータの出力先として共通フォルダのサブフォルダを使用する場合に，共通フォルダルートからサブフォルダまでの相対パスを指定します。

要求電文（ヘッダ），HTTP アダプタ実行環境プロパティファイル，および HTTP アダプタ実行環境共通プロパティファイルのどれにも出力先ディレクトリの指定がなかった場合はエラーになります。

応答電文（ヘッダ）には，出力先ディレクトリおよび出力ファイル名などの情報が格納されます。

応答電文（ヘッダ）に格納されるファイル情報を次に示します。

表 2-96　応答電文（ヘッダ）に格納されるファイル情報

応答電文（ヘッダ）の要素	説明
output-folder-name 要素の common 属性	ファイルデータを出力したフォルダの種類として共通フォルダまたは作業フォルダのどちらかが格納されます。
output-folder-name 要素	ファイルデータを出力したフォルダの定義名が格納されます。共通フォルダに出力した場合は共通フォルダ定義名が格納されます。作業フォルダに出力した場合は，常に空（空文字列）になります。
part-id 要素	Content-Disposition ヘッダの name 属性の値が格納されます。パートを分割したときのパート識別子として使用します。
file-name 要素	HTTP レスポンスに指定された Content-Disposition ヘッダの filename 属性の値が格納されます。ただし，HTTP アダプタでデコードなどの処理はしません。filename 属性が省略された場合は要素が生成されません。
local-file-name 要素	出力先フォルダに保存した中間ファイルの名称が格納されます。
Content-Type 要素の charset 属性	各パートの Content-Type の charset 属性が格納されます。charset 属性の設定がなければ，属性はありません。
Content-Type 要素	各パートの MIME タイプが格納されます。 省略した場合，Content-Type ヘッダは指定されません。

注 1

HTTP アダプタが共通フォルダに出力するファイル名称は，システム内で採番した一意の ID を使用したファイル名となります。次の点に注意してください。

- 一意の ID の生成にサーバのタイムスタンプを利用しているため，時刻の戻りが発生しないようにしてください。NTP ソフトウェアを使用して時刻補正をする場合は，必ず slew モードを使用するようにしてください。時刻補正をする場合の注意事項については，マニュアル「サービスプラットフォーム システム構築・運用ガイド」の「1.4　運用方針の決定」を参照してください。
- 一意の ID の生成に HCSC サーバ名とクラスタ名を使用しているため，同一の共通フォルダに複数の HCSC サーバからアクセスする場合は，これらの名称の組み合わせが重複しないようにシステムを設計してください。

HCSC サーバをロードバランスクラスタ構成や HA クラスタ構成にする場合にも，各 HCSC サーバの HCSC サーバ名やクラスタ名の組み合わせが重複しないようにする必要があります。

例：HCSC サーバ 1 の HCSC サーバ名=HCSC1，クラスタ名=CLS1 の場合

・HCSC サーバ 2 の HCSC サーバ名=HCSC1，クラスタ名=CLS1 → 重複あり

・HCSC サーバ 2 の HCSC サーバ名=HCSC2，クラスタ名=CLS1 → 重複なし

・HCSC サーバ 2 の HCSC サーバ名=HCSC1，クラスタ名=CLS2 → 重複なし

・HCSC サーバ 2 の HCSC サーバ名=HCSC2，クラスタ名=CLS2 → 重複なし

上記の条件を満たすことができない場合は，作業フォルダだけを使用してください。共通フォルダを使用した場合，ファイル名の重複が発生してファイルが上書きされるおそれがあります。

注 2

ファイルデータの出力先に共通フォルダを使用する場合は，HTTP アダプタだけが利用する専用の共通フォルダを必ず使用してください。ほかの用途で使用する共通フォルダと共用しないでください。

HTTP アダプタを利用したファイル受信の例を次に示します。

図 2-162　HTTP アダプタを利用したファイル受信の例（共通フォルダを使用する場合）

●要求電文（ヘッダ）

```
<output-folder-name common="true">com</output-folder-name>
```

●HTTPレスポンス

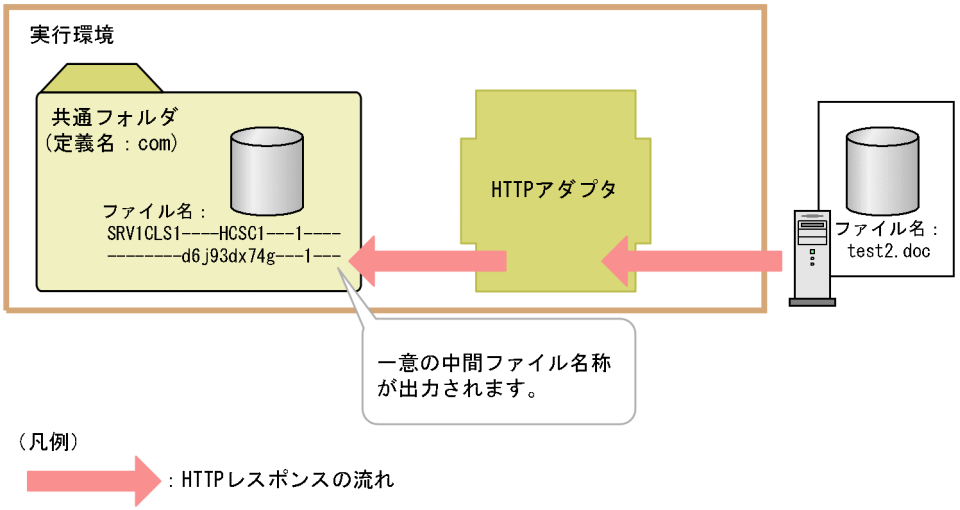
```
HTTP/1.1 200 OK
Content-Disposition: attachment; filename="test2.doc"
Content-Type: application/msword

:
```

●応答電文（ヘッダ）

```
<status-code>200</status-code>
<http-header-Content-Type>application/msword</http-header-Content-Type>

<http-body>
  <output-folder-name common="true">com</output-folder-name>
</http-body>
<http-part>
  <files>
      <file>
          <file-name>test2.doc</file-name>
          <local-file-name>SRV1CLS1----HCSC1---000000000000001000d6j93dx74g0001
          </local-file-name>
      </file>
  </files>
</http-part>
```



次に，共通フォルダのサブフォルダを使用して，HTTP レスポンスでファイルをダウンロードする場合の指定例を示します。

HTTP アダプタ要求電文（ヘッダ）の/http-header-request/output-sub-folder，または HTTP アダプタ実行環境プロパティファイルの adphttp.request.output-sub-folder に共通フォルダルートからサブフォルダまでの相対パスを指定します。

- 要求電文（ヘッダ）指定例

```
<http-header-request>
      :
  <output-folder-name common="true">com</output-folder-name>
  <output-sub-folder>/folder1/folder2</output-sub-folder>
```

```
      :
</http-header-request>
```

- HTTP アダプタ実行環境（共通）プロパティファイルの指定例

```
adphttp.request.output-folder-name=com
adphttp.request.output-sub-folder=/folder1/folder2
```

HTTP アダプタ応答電文（ヘッダ）の/http-headerresponse/http-part/files/file/local-file-name に共通フォルダルートからファイル名までの相対パスが設定されます。

- 応答電文（ヘッダ）の内容

```
<http-header-response>
      :
  <output-folder-name common="true">com</output-folder-name>
      :
    <file-name>file1.doc</file-name>
    <local-file-name>folder1/folder2/SRV1CLS1----HCSC1---0000000000001000d6j93dx74g0001</
local-file-name>
      :
</http-header-response>
```

(b) ローデータ（パススルーモード）

HTTP レスポンスボディのデータを HTTP アダプタの応答電文（ボディ）にそのまま格納する場合，応答電文（ボディ）の電文フォーマットはユーザが任意に作成する必要があります。

必要に応じて，受信したデータを Java 呼出アクティビティやコンポーネント共通 UOC などで加工することもできます。

Content-Type が「multipart/form-data」または「multipart/mixed」の場合は，応答電文（ボディ）にデータを格納できません。

(4) ステータスコードの正常処理またはフォルト処理の設定

HTTP アダプタでは，HTTP レスポンス時に指定したステータスコードを正常処理またはフォルト処理として扱うように設定することができます。

どのステータスコードを正常処理およびフォルト処理として扱うかは HTTP アダプタ定義ファイルで設定できます。

表 2-97　ステータスコードの処理に関する設定

プロパティ名	説明
adphttp.response-code.normal	正常処理として扱うステータスコードを正規表現で指定します。デフォルトは 200 です。
adphttp.response-code.fault	フォルト処理として扱うステータスコードを正規表現で指定します。デフォルトは空文字列です。

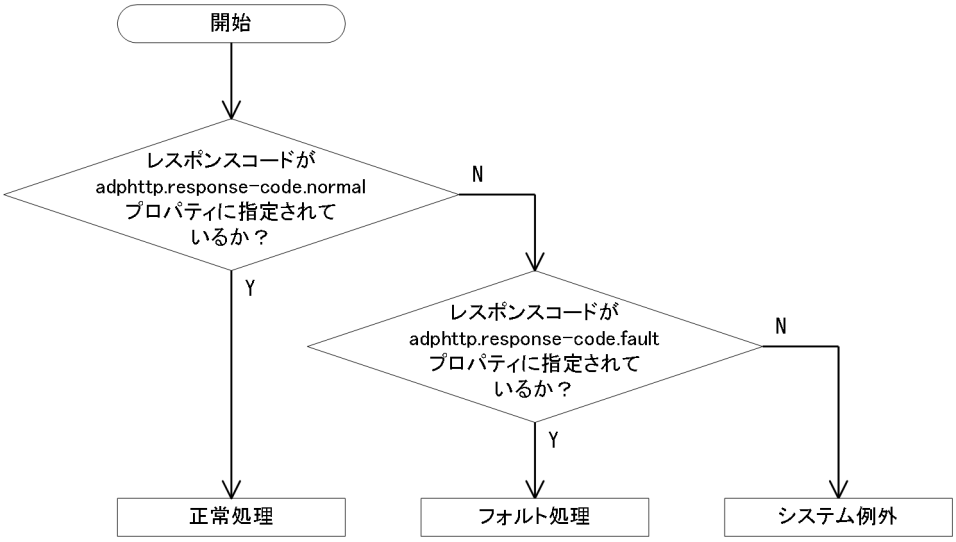
なお，ステータスコードが正規表現に完全一致しない場合は，HTTP レスポンスでステータスコードが返らないため，システム例外として処理されます。

注意事項

フォルト処理として扱う場合，ビジネスプロセス上でキャッチできますが，再実行ができなくなります。このため，フォルト処理を設定する場合はビジネスプロセスへの影響を考慮してから設定してください。

ステータスコードの処理フローを次の図に示します。

図 2-163　ステータスコードの処理フロー



(5)　レスポンスサイズの制限の設定

HTTP アダプタでは，非常に大きなサイズの HTTP レスポンスを受信した場合のメモリ不足およびディスクの圧迫を制御するために，HTTP レスポンスボディのサイズ閾値を設定して，受信サイズを制限することができます。

HTTP レスポンスボディのサイズ閾値は HTTP アダプタ定義ファイルで設定できます。設定するプロパティを次に示します。

表 2-98　HTTP レスポンスボディのサイズ閾値の設定

項目	プロパティ名
パススルーモードで HTTP レスポンスを応答電文（ボディ）に設定する際に許容するデータの最大サイズ	adphttp.data-size-limit.body-message
HTTP レスポンスをファイルに書き出す際に許容するデータの最大サイズ	adphttp.data-size-limit.file

閾値を設定したあとは，HTTP アダプタでは，次の順序でレスポンスサイズの判定をしています。

1. HTTP ヘッダに Content-Length ヘッダがある場合は，ヘッダフィールド値が閾値を超えていないか判定をします。閾値を超えている場合は，HTTP レスポンスボディを受信しないで，ただちにシステム例外となります。
2. HTTP ヘッダに Content-Length ヘッダがない場合（チャンクエンコーディングされたデータを受信する場合など）は，HTTP レスポンスボディの受信処理をし，閾値を超えた時点で受信処理を打ち切り，システム例外となります。

   なお，マルチパート型の受信処理の場合，上限値を超えるバイト数のファイルを受信した時点でシステム例外となります。例外が発生するまでの間に受信したファイルは出力されます。

(6) 不完全受信ファイルの自動削除

HTTP アダプタでファイルを受信中に，ネットワーク障害や I/O エラー，データサイズ制限の超過などが発生した場合，共通フォルダや作業フォルダに不完全な受信ファイルが生成されますが，この不完全な受信ファイルを自動的に削除することができます。

この機能は HTTP アダプタ定義ファイルの adphttp.auto-delete-incomplete-file プロパティで設定できます。

不完全な受信ファイルの削除に失敗した場合は，メッセージログ（KDEC81403-W）が出力されます。なお，この場合，削除に失敗したファイルは手動で削除する必要があります。削除方法については，マニュアル「サービスプラットフォーム システム構築・運用ガイド」の「5.4.20　作業フォルダを削除する」，および「5.4.22　共通フォルダ内のファイルを削除する」を参照してください。

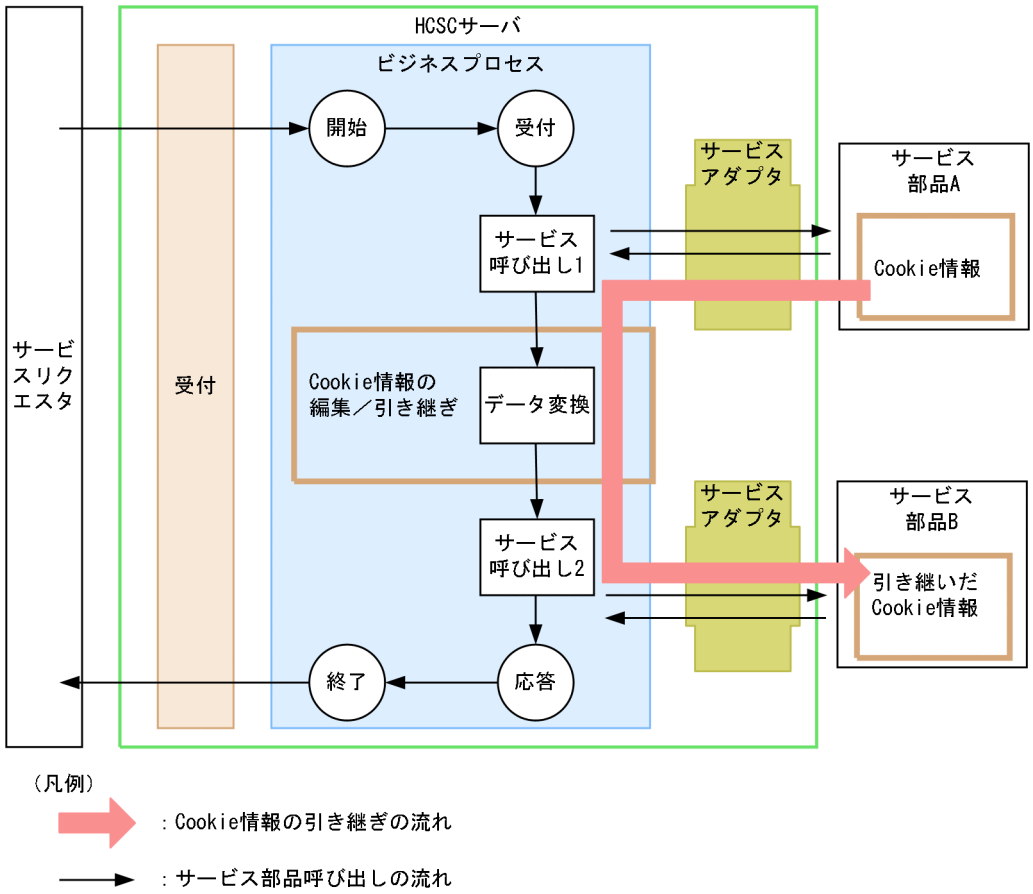
なお，マルチパート型のファイルを受信する場合，受信途中の不完全なファイルが作成された時点で受信処理は中断し，受信途中の不完全なファイルを削除します。受信途中の不完全なファイルが作成されるまでの間に受信したファイルは削除しないため，削除が必要な場合は手動で削除してください。

2.16.5 HTTP アダプタを使用した Cookie 情報の引き継ぎ

HTTP アダプタでは，ビジネスプロセスから複数のサービスを呼び出す場合に，サービス呼び出し間のセッション維持（Cookie 情報の引き継ぎ）ができます。

HTTP アダプタを利用した Cookie 情報の引き継ぎの概要を次の図に示します。

図 2-164　HTTP アダプタを利用した Cookie 情報の引き継ぎ



HTTP アダプタの Cookie 情報の引き継ぎについて，上図を例に説明します。なお，ここで説明する電文の例では，名前空間接頭辞を省略しています。

(1)　Cookie 情報の引き継ぎ方法

2 つのサービス間で Cookie 情報を引き継ぐ場合，Cookie 情報を一括して引き継ぐか，Cookie 名ごとに個別に引き継ぐかを選択できます。どちらの方法を選択するかによって，使用するヘッダ変数用の電文フォーマットが異なります。

Cookie 情報を一括して引き継ぐ場合

- 要求電文フォーマットに adphttp_header_request1.xsd を使用します。
- 応答電文フォーマットに adphttp_header_response1.xsd を使用します。

Cookie 名ごとに編集できる状態で個別に引き継ぐ場合

- 要求電文フォーマットに adphttp_header_request2.xsd を使用します。
- 応答電文フォーマットに adphttp_header_response2.xsd を使用します。

どちらの場合も，サービス部品 A からサービス部品 B に Cookie 情報を引き継ぐには，サービス部品 A の応答電文に含まれる Cookie 情報（Cookie 要素）を，サービス部品 B の要求電文にマッピングして HTTP アダプタを呼び出します。

それぞれの場合の引き継ぎ方法について説明します。

(a) Cookie 情報を一括して引き継ぐ場合

電文フォーマットでは，次のように Cookie 情報がすべて any 型で表現されます。なお，Cookie 要素を表す要素名は「Cookie_types」としています。

```
<xsd:complexType name="http-header-Cookies-type">
  <xsd:sequence>
    <xsd:any namespace="##any"
        processContents="skip" minOccurs="0" maxOccurs="unbounded" />
  </xsd:sequence>
</xsd:complexType>
```

any 型の要素部分同士をサービス部品 A の応答電文からサービス部品 B の要求電文にマッピングすることで，Cookie 情報を一括して引き継ぐことができます。

(b) Cookie 名ごとに編集できる状態で個別に引き継ぐ場合

電文フォーマットでは，次のように Cookie 情報の各属性が XML 要素の属性に分割されます。なお，Cookie 要素を表す要素名は「Cookie_types」としています。

```
<xsd:complexType name="cookie-type">
  <xsd:simpleContent>
    <xsd:extension base="xsd:string">
      <xsd:attribute name="name" type="xsd:string" use="required"></xsd:attribute>
      <xsd:attribute name="host" type="xsd:string" use="required"></xsd:attribute>
      <xsd:attribute name="path" type="xsd:string" use="optional"></xsd:attribute>
    </xsd:extension>
  </xsd:simpleContent>
</xsd:complexType>
```

サービス部品 A の応答電文からサービス部品 B の要求電文へのマッピング時に，Cookie 名などを参照して個別に引き継ぐことができます。

(2) HTTP レスポンス・HTTP リクエストの変換

HTTP レスポンスヘッダから応答電文への変換，および要求電文から HTTP リクエストヘッダへの変換処理について説明します。

HTTP レスポンスヘッダから応答電文への変換処理

サービス部品 A から HTTP レスポンスヘッダとして渡された各 Set-Cookie ヘッダの値は，応答電文の Cookie 要素に格納されます。また，Set-Cookie ヘッダの各属性は Cookie 要素の属性値に設定されます。

複数の Set-Cookie ヘッダが存在した場合は，それぞれの Set-Cookie ヘッダについて Cookie 要素が生成されます。

HTTP レスポンスヘッダから応答電文（ヘッダ）への変換例を次に示します。

- HTTP レスポンスヘッダの Set-Cookie ヘッダ

```
Set-Cookie: AAA=1234; expires=Sat, 27-Oct -2012 10:00:00 GMT; path=/
Set-Cookie: BBB=5678; expires=Sat, 27-Oct -2012 10:00:00 GMT;
```

- 生成後の応答電文（ヘッダ）

```
<Cookies>
  <Cookie name="AAA" path="/" host="hostname:80※">AAA=1234; expires=Sat, 27-Aug-2011
10:00:00 GMT; path=/</Cookie>
  <Cookie name="BBB" host="hostname:80※">BBB=5678; expires=Sat, 27-Aug-2011 10:00:00
GMT</Cookie>
</Cookies>
```

注※

host 属性には，サービス部品への接続に使用したホスト名とポート番号が格納されます。

Set-Cookie ヘッダから path 属性を取得できなかった場合は，応答電文（ヘッダ）に path 属性を付加しないで処理されます。この場合，path 属性値は"/"として扱われます。

要求電文から HTTP リクエストヘッダへの変換処理

HTTP アダプタの要求電文として渡された Cookie 情報は，次の両方の条件を満たすものが送信対象として HTTP リクエストヘッダに設定されます。

- サービス部品の URI と Cookie 要素の host 属性に設定された値（ホスト名とポート番号）が完全一致する。
- サービス部品の URI に対して Cookie 要素の path 属性に設定されたパスが前方一致する。

例：サービス部品の URI が http://localhost:80/abc/xyz の場合

- Cookie 情報の host 属性値=localhost:80, path 属性値=/abc → 送信される
- Cookie 情報の host 属性値=somehost:80, path 属性値=/abc → 送信されない
- Cookie 情報の host 属性値=localhost:443, path 属性値=/abc → 送信されない
- Cookie 情報の host 属性値=localhost:80, path 属性値=/xyz → 送信されない
- Cookie 情報の host 属性値=localhost:80, path 属性値=/abc/xyz/def → 送信されない

Cookie 情報が複数ある場合は，次のようにひとまとめの Cookie ヘッダに変換されます。

要求電文（ヘッダ）から HTTP リクエストヘッダへの変換例を次に示します。

- 要求電文（ヘッダ）の Cookie 要素

```
<Cookies>
  <Cookie name="AAA" path="/" host="hostname:80">AAA=1234; expires=Sat, 27-Oct-2012 10
:00:00 GMT; path=/</Cookie>
  <Cookie name="BBB" host="hostname:80">BBB=5678; expires=Sat, 27-Oct-2012 10:00:00 GM
T</Cookie>
</Cookies>
```

Cookie の要素名はローカル名だけ使用します。ヘッダ応答電文からヘッダ要求電文に Cookie 情報を一括引継した場合，ヘッダ要求電文の Cookie 要素の名前空間はヘッダ応答電文になりますが，名前空間を参照しないため，送信対象として正しく扱われます。

- 生成後の HTTP リクエストヘッダ

```
Cookie: AAA=1234; BBB=5678
```

注意事項

- 異なる path 属性値を持つ複数の Cookie 要素を設定した場合，path 属性値が詳細な順に，HTTP ヘッダに Cookie 情報が設定されます。

  （例）NAME1=VALUE1 path=/, NAME2=VALUE2 path=/xyz の値を持つ Cookie 要素を設定した場合：

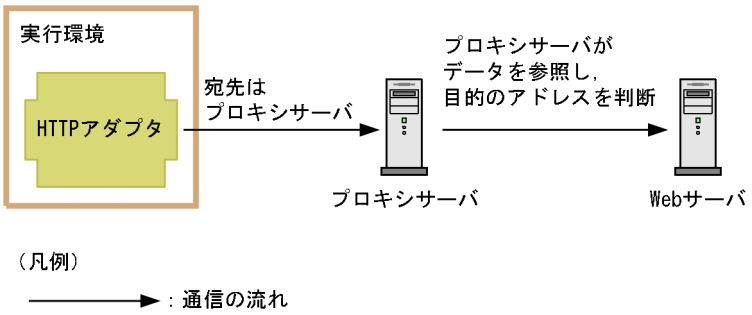
  Cookie: NAME2=VALUE2; NAME1=VALUE1
- 正常に処理できない Cookie 情報を受信した場合は，解析をスキップし，処理を続行します。
- このほかに「2.2.4(4)　注意事項」に示す注意事項も参照してください。

2.16.6　プロキシサーバを経由した通信

HTTP アダプタでは，プロキシサーバを経由して外部のネットワークにあるサービスを利用できます。

プロキシサーバを経由した通信の概要を次の図に示します。

図 2-165　プロキシサーバを利用した通信



プロキシサーバ経由で接続するには JavaVM のシステムプロパティの設定が必要です。設定が必要なプロパティを次の表に示します。

表 2-99　プロキシサーバ経由の接続に必要なシステムプロパティ

プロパティ名	指定内容
http.proxyHost	プロキシサーバのホスト名または IP アドレスを指定します。空文字を指定した場合は，プロキシサーバに接続されません。
http.proxyPort	プロキシサーバのポート番号を設定します。 http.proxyHost が適切に設定されていた場合に，http.proxyPort に空文字を指定したときは，http.proxyHost に指定されているホストの 80 番ポートにアクセスされます。

プロパティ名	指定内容
http.proxyPort	http.proxyHost を指定していない場合は，http.proxyPort を指定しても，プロキシサーバに接続されません。
http.nonProxyHosts	プロキシサーバを利用しないホスト名群を必要に応じて指定します。 このプロパティで指定したホストに接続する場合は，http.proxyHost または https.proxyHost に指定したプロキシサーバは経由されません。ホストを複数指定する場合は，「|」で区切って設定します。ホスト名とホスト名の間には，「|」以外の文字（空白など）は指定できません。
https.proxyHost	SSL プロトコルによる接続で使用するプロキシサーバのホスト名または IP アドレスを指定します。 空文字を指定した場合は，プロキシサーバに接続されません。
https.proxyPort	SSL プロトコルによる接続で使用するプロキシサーバのポート番号を設定します。 https.proxyHost が適切に設定されていた場合に，https.proxyPort に空文字を指定したときは，https.proxyHost に指定されているホストの 443 番ポートにアクセスされます。 https.proxyHost を指定していない場合は，https.proxyPort を指定しても，プロキシサーバに接続されません。

設定方法の詳細は，マニュアル「アプリケーションサーバ Web サービス開発ガイド」の「10.10　プロキシサーバ経由の接続」を参照してください。

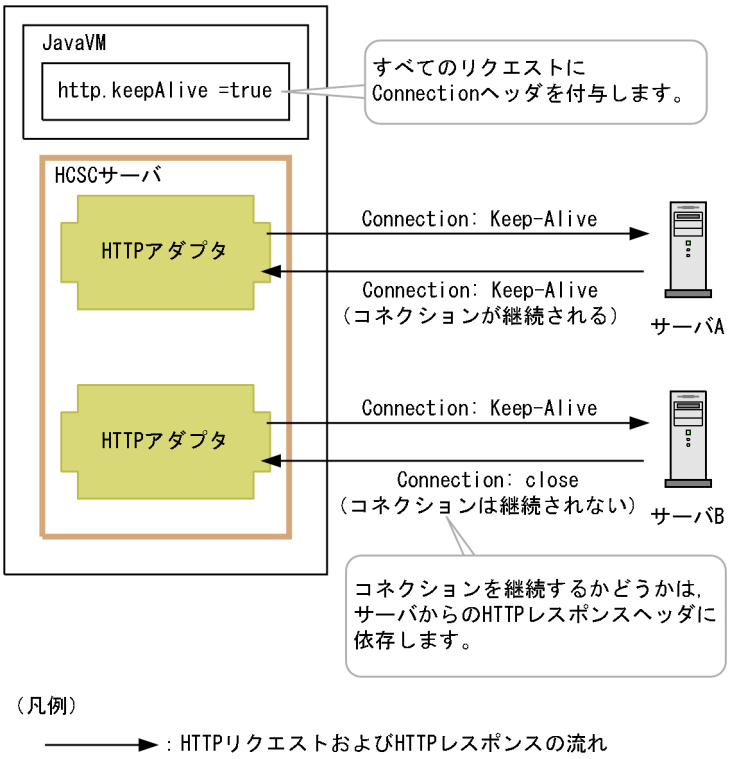
2.16.7　コネクション継続（Keep-Alive）を使用した通信

Connection ヘッダに Keep-Alive を指定することで，一度使用したコネクションを継続して利用でき，次回以降のコネクション接続コストを減らすことができます。

Keep-Alive 接続を使用するかどうかは JavaVM のシステムプロパティで設定します。なお，HTTP アダプタから Keep-Alive 接続で通信しても，サーバ側が Keep-Alive を使用しない設定の場合，サーバからの HTTP レスポンスの Connection ヘッダは close となり，コネクションは継続されません。

Keep-Alive を使用したコネクション継続の概要を次の図に示します。

図 2-166　Keep-Alive 接続の概要



Keep-Alive 接続を使用する場合に設定が必要なプロパティを次の表に示します。

表 2-100　Keep-Alive 接続に必要なシステムプロパティ

プロパティ名	指定内容
http.keepAlive	Keep-Alive 接続を有効にするかどうかを true または false で指定します。デフォルトは true です。
http.maxConnections	特定の時間内で保持される接続先ごとの最大接続数を int 型で指定します。デフォルトは 5 です。

JavaVM のシステムプロパティの設定については，JavaVM のドキュメントを参照してください。

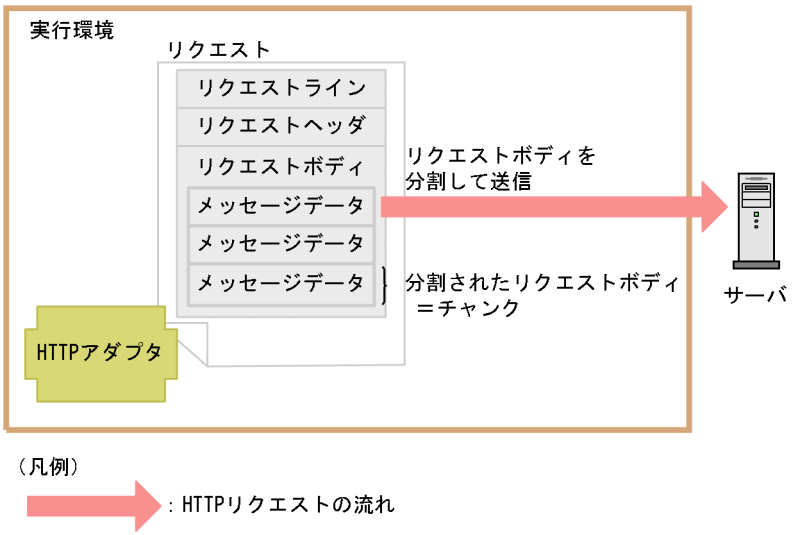
なお，Keep-Alive 接続に関する設定は，システムプロパティで設定するため，HTTP アダプタ単位ではなくサーバ全体で有効になります。

2.16.8　チャンク送信を使用した通信

HTTP アダプタでは，HTTP リクエストを送信する際，ボディデータを複数の塊（チャンク）に分割し，その塊ごとにデータを送信することができます。

チャンク送信の概要を次の図に示します。

図 2-167　チャンク送信の概要



チャンク送信の使用有無およびチャンクサイズは，HTTP アダプタ定義ファイルの adphttp.protocol.chunked-encoding-size プロパティで設定できます。

なお，チャンク送信を行う場合，HTTP リダイレクト機能は使用できません。

HTTP アダプタ定義ファイルの詳細は，マニュアル「サービスプラットフォーム リファレンス」の「3.10.1　HTTP アダプタ定義ファイル」を参照してください。

2.16.9　HTTP ボディの圧縮／解凍

通信データ量が多い場合，送受信する HTTP ボディのデータを圧縮／解凍することで，トラフィック量を減らすことができます。

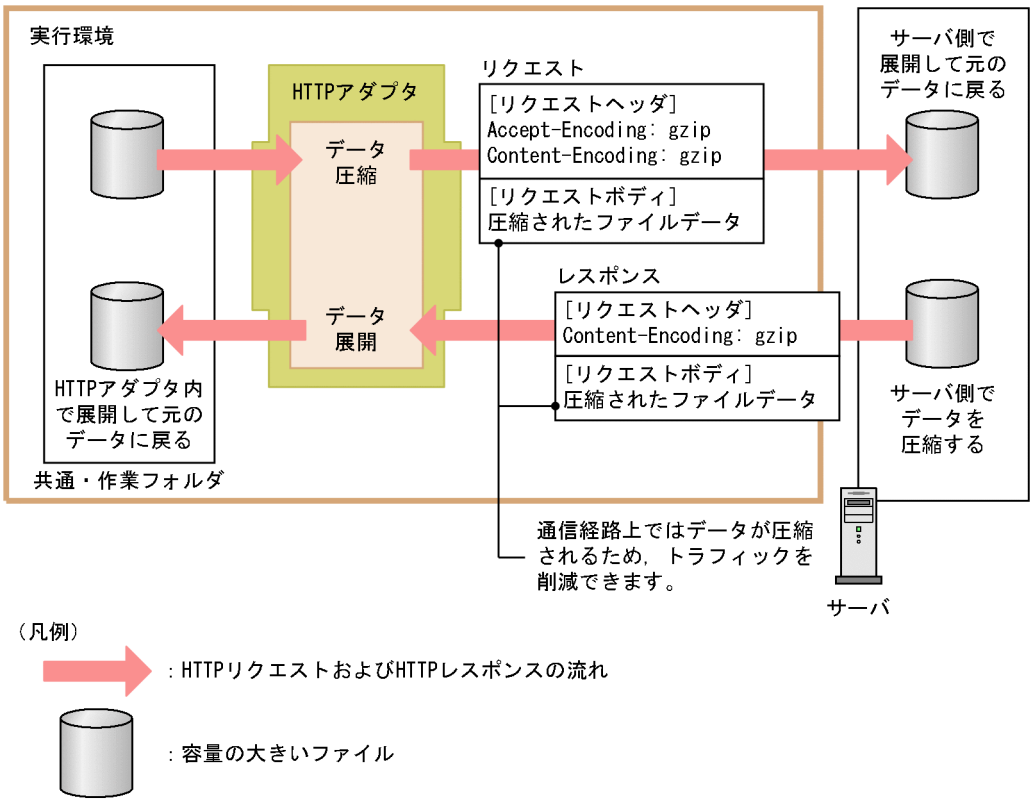
HTTP メッセージの送受信時に HTTP ボディを圧縮／解凍するかどうかは，HTTP アダプタ定義ファイルの adphttp.protocol. encoding-type プロパティで設定できます。

このプロパティに gzip を指定した場合，HTTP ボディのデータが gzip 形式でエンコード／デコードして送受信されます。

HTTP アダプタ定義ファイルの詳細は，マニュアル「サービスプラットフォーム リファレンス」の「3.10.1　HTTP アダプタ定義ファイル」を参照してください。

HTTP ボディの圧縮／解凍の例として，ファイル連携で adphttp.protocol.encoding-type プロパティに gzip を指定した場合の処理の流れを次の図に示します。

図 2-168　HTTP ボディの圧縮／解凍の例



2.16.10　HTTP アダプタのタイマ管理

HTTP アダプタでは次のタイムアウト時間を設定できます。

- 接続タイムアウト
- 読み込みタイムアウト
- 排他管理タイムアウト

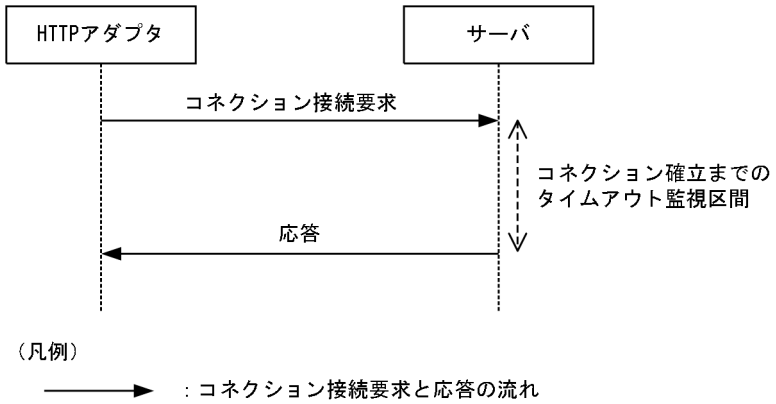
それぞれのタイムアウト値は，HTTP アダプタ定義ファイルで設定できます。HTTP アダプタ定義ファイルの詳細は，マニュアル「サービスプラットフォーム リファレンス」の「3.10.1　HTTP アダプタ定義ファイル」を参照してください。

(1)　接続タイムアウト

サーバとの接続が確立するまでの接続タイムアウトを設定できます。

接続タイムアウトの概要を次の図に示します。

図 2-169　接続タイムアウト



設定できるパラメタを次の表に示します。

表 2-101　接続タイムアウトの設定

値	キー名称	デフォルト値
接続タイムアウト（ミリ秒）	adphttp.protocol.connect-timeout	60000

(2)　読み込みタイムアウト

HTTP リクエストを送信してから HTTP レスポンスが返るまでの読み込みタイムアウトを設定できます。HTTP リクエストを送信してから数回に分けて HTTP レスポンスが返る場合は，1 回ごとにタイムアウトを監視します。

読み込みタイムアウトの概要を次の図に示します。

図 2-170　読み込みタイムアウト（1 回でレスポンスが返る場合）

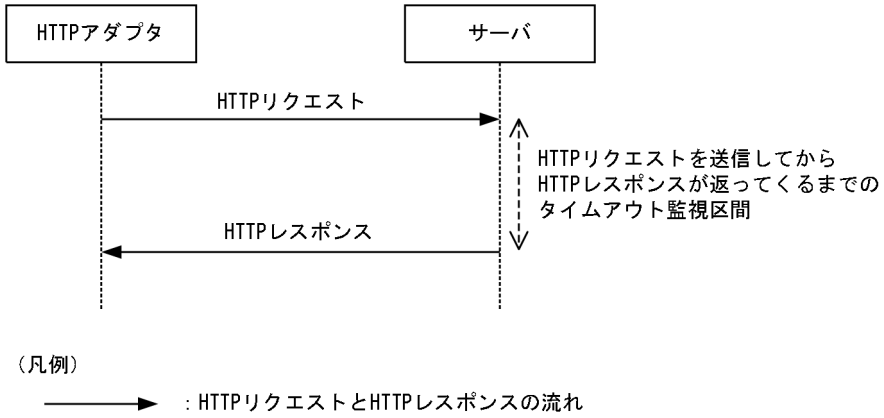
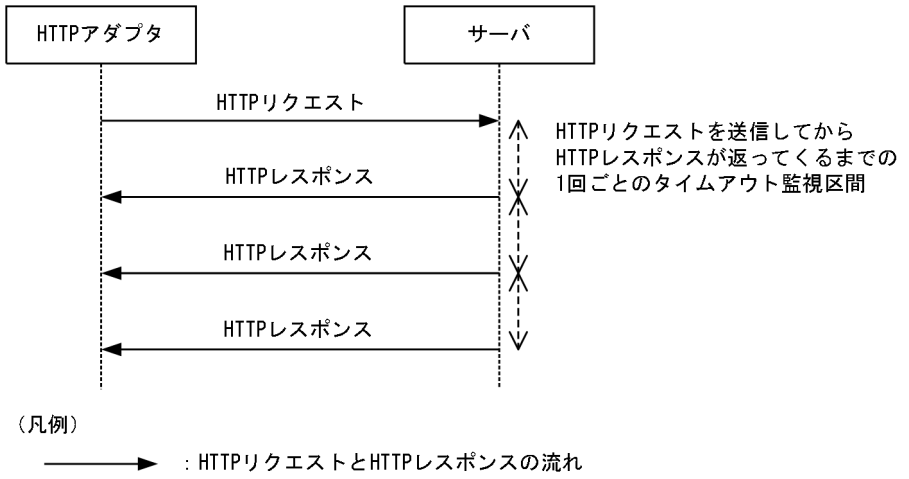


図 2-171　読み込みタイムアウト（3 回に分けてレスポンスが返る場合）



設定できるパラメタを次の表に示します。

表 2-102　読み込みタイムアウトの設定

値	キー名称	デフォルト値
読み込みタイムアウト（ミリ秒）	adphttp.protocol.read-timeout	300000

(3)　排他管理タイムアウト

HTTP アダプタから共通フォルダにアクセスする場合，共通フォルダに対して共有ロックが掛かります。HTTP アダプタでは，このときのロックリトライ回数およびロックリトライ間隔を指定できます。

設定できるパラメタを次の表に示します。

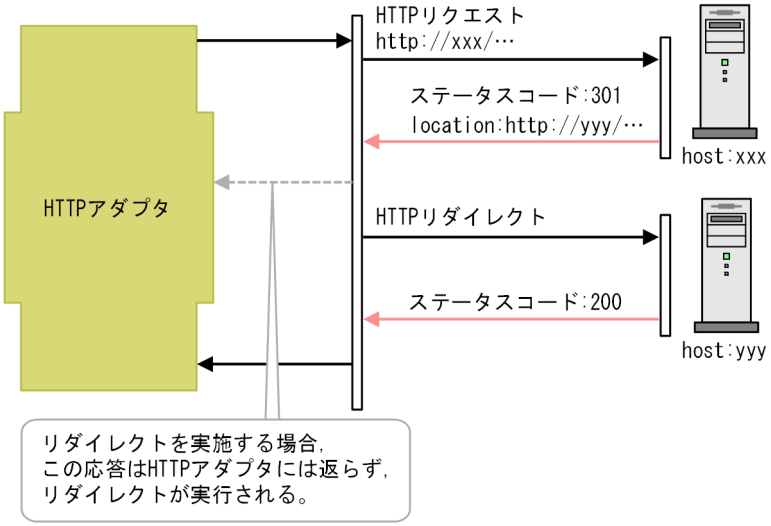
表 2-103　読み込みタイムアウトの設定

値	キー名称	デフォルト値
ロックリトライ回数（回）	adphttp.file.read-lock.retry.count	0
ロックリトライ間隔（秒）	adphttp.file.read-lock.retry.interval	1

2.16.11　HTTP アダプタのリダイレクト

HTTP アダプタでは，呼び出すサービスが一時的に呼び出せない場合に，リダイレクトを実施し，代替のサービスを呼び出せます。

図 2-172　HTTP アダプタのリダイレクトの概要



次の条件をすべて満たす場合は，自動的に HTTP リダイレクトを実施します。

- レスポンス時のステータスコードが 301，302，303，307 のとき
- リダイレクト先の URL を保持する location ヘッダが返されたとき

HTTP アダプタのリダイレクトは，HTTP アダプタ定義ファイルの adphttp.protocol.follow-redirect プロパティで設定します。

HTTP アダプタ定義ファイルの詳細については，マニュアル「サービスプラットフォーム リファレンス」の「3.10.1　HTTP アダプタ定義ファイル」を参照してください。

注意事項

- HTTP アダプタのリダイレクトは，J2EE サーバの互換モードが推奨モードの場合だけ使用できます。
- リダイレクトの動作は，JAX-RS 2.1 の仕様に従います。

2.16.12　ベーシック認証機能を使用した接続

HTTP アダプタでは，ベーシック認証を使用して対象のサーバに接続できます。

ベーシック認証機能を使用するには，WWW 認証の認証タイプおよびユーザ名の設定と，認証情報の設定が必要です。

ベーシック認証以外の WWW 認証の設定については，「2.16.16　WWW 認証を使用したセキュアな接続」を参照してください。

(1) 認証タイプおよびユーザ名の設定

認証タイプおよびユーザ名は，要求電文（ヘッダ），または HTTP アダプタ実行環境（共通）プロパティファイルで設定します。

(a) 要求電文の設定

要求電文（ヘッダ）に認証タイプおよびユーザ名を設定して，サーバにリクエストを送信します。

認証タイプおよびユーザ名で設定するリクエストヘッダを次に示します。

表 2-104 認証タイプおよびユーザ名の設定項目（要求電文）

キー名称	設定値
/http-header-request/http-header-Authorization/@type 属性	basic
/http-header-request/http-header-Authorization 要素	HTTP アダプタアカウント定義ファイル※に登録したユーザ名を指定します。

注※

HTTP アダプタアカウント定義ファイルの詳細については，マニュアル「サービスプラットフォーム リファレンス」の「6.14.1 HTTP アダプタアカウント定義ファイル」を参照してください。

設定例

認証タイプおよびユーザ名を設定する場合の要求電文（ヘッダ）の設定例を次に示します。

```
<http-header-request>
  <http-header-Authorization type="basic">user1</http-header-Authorization>
</http-header-request>
```

設定方法の詳細については，マニュアル「サービスプラットフォーム 開発ガイド 基本開発編」の「4. 電文フォーマットの作成」を参照してください。

(b) HTTP アダプタ実行環境（共通）プロパティファイルの設定

ベーシック認証での接続を HTTP アダプタ実行環境（共通）プロパティファイルに設定する場合，次に示すプロパティを設定します。

表 2-105 認証タイプおよびユーザ名の設定（HTTP アダプタ実行環境（共通）プロパティファイル）

項目	キー名称	設定値
WWW 認証の認証タイプ	adphttp.request.header.authorization.type	basic
WWW 認証の認証情報	adphttp.request.header.authorization	HTTP アダプタアカウント定義ファイル※に登録したユーザ名を指定します。

注※

HTTP アダプタアカウント定義ファイルの詳細については，マニュアル「サービスプラットフォーム リファレンス」の「6.14.1　HTTP アダプタアカウント定義ファイル」を参照してください。

プロパティの詳細については，マニュアル「サービスプラットフォーム リファレンス」の「6.14.2　HTTP アダプタ実行環境プロパティファイル」を参照してください。

(2)　認証情報の設定

HTTP アダプタのベーシック認証では，呼び出し先のサービスごとに認証情報を管理するため，ユーザ名，パスワードのほかに，接続先 URI を認証情報として管理します。異なる接続先 URI を設定することで，同一のユーザ名でも別のアカウント情報として管理することができます。

認証情報の設定には，次の作業が必要です。

- HTTP アダプタアカウント定義ファイルの作成
  アカウント情報を管理するコマンドにユーザ名，パスワード，および接続先 URI を設定して，HTTP アダプタアカウント定義ファイルを作成します。
- HTTP アダプタ実行環境（共通）プロパティファイルの設定
  HTTP アダプタアカウント定義ファイルのパスを設定します。

(a)　HTTP アダプタアカウント定義ファイルの作成

ベーシック認証の認証情報として使用するユーザ名，パスワード，および接続先 URI は，作成した HTTP アダプタアカウント定義ファイルに設定します。

HTTP アダプタアカウント定義ファイルは，csahttpuseradd コマンドで作成します。csahttpuseradd コマンドの詳細については，マニュアル「サービスプラットフォーム リファレンス」の「csahttpuseradd (HTTP アダプタで使用するアカウント情報の登録・更新)」を参照してください。

なお，HTTP アダプタアカウント定義ファイルの内容はコマンドで管理するため，編集はできません。

認証タイプが basic の場合，ユーザ名と接続先 URI をキーとして HTTP アダプタアカウント定義ファイルからパスワードを取得し，認証情報を作成します。HTTP アダプタアカウント定義ファイルでは，認証情報が適切な接続先にだけ送信されるように，認証情報を送信する接続先のスキーム，ホスト名，ポート，パスを接続先 URI として定義します。接続先 URI とのマッチングの詳細については，「(3)　接続先 URI とのマッチングルール」を参照してください。

(b)　HTTP アダプタ実行環境（共通）プロパティファイルの設定

HTTP アダプタアカウント定義ファイルを有効にするには，HTTP アダプタ実行環境（共通）プロパティファイルで，HTTP アダプタアカウント定義ファイルの絶対パスを指定する必要があります。

表 2-106　HTTP アダプタアカウント定義ファイルのパスの設定

項目	キー名称	設定値
HTTP アダプタアカウント定義ファイル※のパス	adphttp.config.account-inf-filepath	HTTP アダプタアカウント定義ファイルの保存先のパスを絶対パスで指定します。

注※

HTTP アダプタアカウント定義ファイルの詳細については，マニュアル「サービスプラットフォーム リファレンス」の「6.14.1　HTTP アダプタアカウント定義ファイル」を参照してください。

認証タイプが basic で，adphttp.config.account-inf-filepath キーが設定されていない場合は，実行時にエラーとなります。

プロパティの詳細については，マニュアル「サービスプラットフォーム リファレンス」の「6.14.2　HTTP アダプタ実行環境プロパティファイル」を参照してください。

(3)　接続先 URI とのマッチングルール

次のどれかに指定した接続先 URI，および HTTP アダプタから呼び出すサービスの URI は，設定値を正規化し，マッチングされます。

- csahttpuseradd コマンドの-uri オプションに指定した接続先 URI
- csahttpuserdel コマンドの-uri オプションに指定した接続先 URI
- -userinfofile オプションのユーザ情報指定ファイルに指定した接続先 URI

接続する HTTP アダプタから呼び出すサービスの URI に対して，もっとも長く前方一致したものを優先して採用します。

マッチングの手順と設定に応じたマッチングの例について説明します。

(a)　マッチングの手順

次の設定値を指定したときのマッチングの手順を次に示します。

設定値

[scheme]://[user-info]@[host]:[port][path]?[query]#[fragment]

手順

1. URI.normalize()で URI を正規化します。
2. URI.getScheme()で[scheme]を取得し，String.toLowerCase()で小文字に統一します。
   この値を[scheme']とします。
3. URI.getHost()で[host]を取得し，String.toLowerCase()で小文字に統一します。
   この値を[host']とします。

スキームおよびホスト名は大文字小文字を区別しないため，すべて小文字に正規化した状態でマッチングされます。

4. URI.getPort()で[port]を取得します。
   この値を[port']とします。
   [port]が省略されている場合は[scheme]によって，次のデフォルトポートを代替値として使用します。
   - 指定 URI のポートが省略されていた場合
     アダプタ内のスキームを見て適切なデフォルトポートを補完します（http→80，https→443）。
   - http，https 以外のスキームが指定されている場合
     ポートは補完しません。
5. URI.getPath()で[path]を取得します。
   この値を[path']とします。
   [path]が省略されている場合は「/」を補完します。
6. [scheme']://[host']:[port'][path']の形式で正規化した URI を生成してマッチングします。

なお，URL エンコードが必要な場合，使用できる文字コードは UTF-8 だけです。

(b) マッチングの例

HTTP アダプタアカウント定義ファイルに設定した接続先 URI と，マッチングした URL の例について説明します。HTTP アダプタアカウント定義ファイルには，次の接続先 URI を設定したものとします。

1. http://localhost:80/aaa/bbb/
2. http://localhost:80/aaa/bbb/ccc/
3. http://localhost:80/aaa/bbb/ccc
4. https://localhost:443/aaa/bbb/ccc
5. https://localhost:443/aaa/bbb/

- 接続先 URI が http://localhost/aaa/bbb/ccc/の場合
  http://localhost:80/aaa/bbb/ccc/のように，ポート番号を補完した URI で前方一致させます。
  この場合，1.と 2.と 3.が合致しますが，2.が最も長く合致しているため 2.のパスワードが採用されます。
- 接続先 URI が https://localhost/aaa/bbb/ccc の場合
  https://localhost:443/aaa/bbb/ccc のように，ポート番号を補完した URI で前方一致させます。
  この場合，合致するものは 4.だけのため，4.のパスワードが採用されます。

HTTP アダプタアカウント定義ファイルに，要求電文または HTTP アダプタ実行環境（共通）プロパティファイルで指定されたユーザ名がない，または送信 URI に対して scheme，host，port，および path 部分がマッチしないなど，送信する認証情報が取得できなかった場合はエラーとなります。

2.16.13　HTTP アダプタを利用したシステムの例

HTTP アダプタを利用したシステムの例を紹介します。

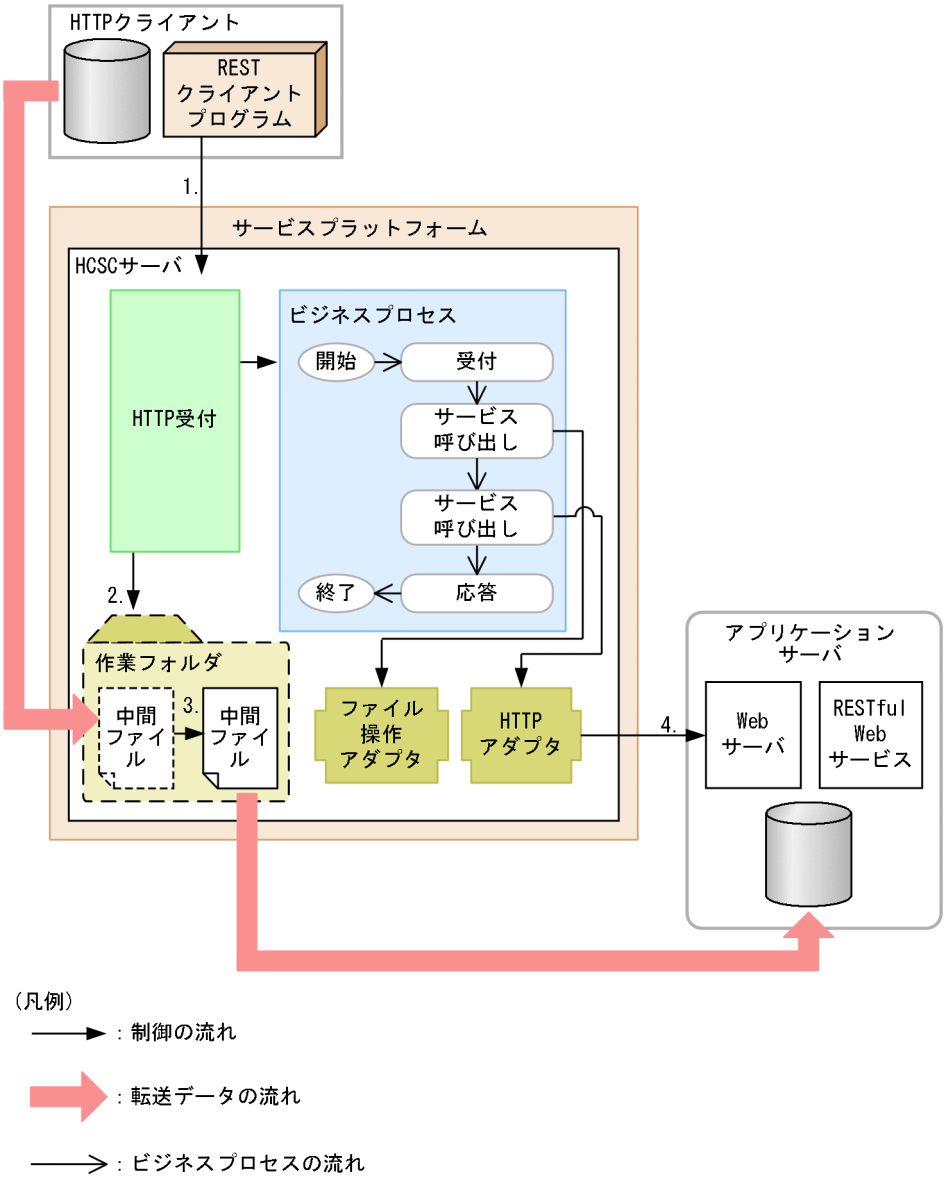
- HTTP クライアントから送信したファイルをサーバにアップロードする
- サーバに保管されたファイルを別のサーバに移行する

それぞれのシステムについて説明します。

(1)　HTTP クライアントから送信したファイルをサーバにアップロードする

この例では，HTTP クライアントから送信したファイルを，ビジネスプロセスから呼び出したファイル操作アダプタで変換したあと，目的のサーバにアップロードする処理を実行しています。

図 2-173　HTTP アダプタを利用したシステムの例（1）



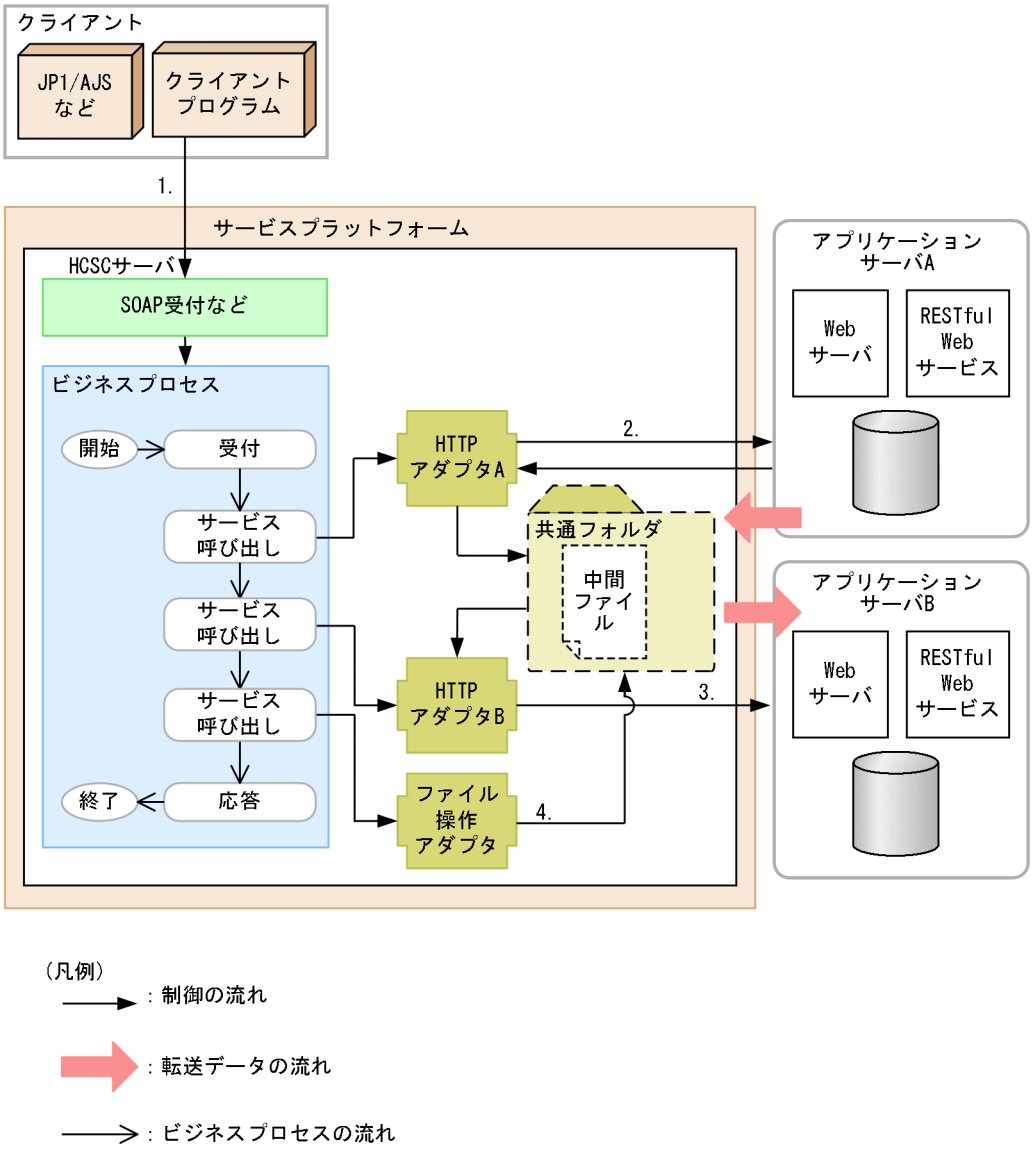
サービス部品の呼び出しおよびファイルの流れを次に示します。なお，次の番号は図中の番号と対応しています。

1. サービスリクエスタ（HTTP クライアントの REST クライアントプログラム）から HCSC サーバにファイルを送信します。
2. HTTP 受付で受信したファイルが中間ファイルとして作業フォルダに生成されます。
3. ビジネスプロセスから呼び出されたファイル操作アダプタで中間ファイルを変換します。
4. ビジネスプロセスから呼び出された HTTP アダプタで変換後のファイルを目的のサーバへ送信します。

(2)　サーバに保管されたファイルを別のサーバに移行する

この例では，サーバに保管されたファイルを HCSC サーバにダウンロードして，目的となる別のサーバに送信する処理を実行しています。

図 2-174　HTTP アダプタを利用したシステムの例（2）



サービス部品の呼び出しおよびファイルの流れを次に示します。なお，次の番号は図中の番号と対応しています。

1. サービスリクエスタ（クライアントプログラム）からビジネスプロセスを実行するリクエストを送信します。
2. ビジネスプロセスから呼び出された HTTP アダプタ A が，対象サーバ（アプリケーションサーバ A）にアクセスしてファイルを受信したあと，共通フォルダにファイルを保存します。
3. ビジネスプロセスから呼び出された HTTP アダプタ B が，共通フォルダに保存されたファイルを目的のサーバ（アプリケーションサーバ B）に送信します。
4. 受信した共通フォルダ内のファイルをファイル操作アダプタで削除します。

2.16.14 要求電文（ヘッダ）と HTTP アダプタ実行環境（共通）プロパティファイルのパラメタの対応関係

HTTP アダプタの要求電文（ヘッダ），HTTP アダプタ実行環境プロパティファイル，および HTTP アダプタ共通実行環境プロパティファイルで設定できるパラメタの対応関係を次に示します。なお，ここでは要求電文（ヘッダ）の要素名を XPath で表記しています。

同じ内容が要求電文（ヘッダ），HTTP アダプタ実行環境プロパティファイル，HTTP アダプタ実行環境共通プロパティファイルの間で複数設定された場合，次の表でより優先度の高いものが適用されます。

表 2-107 要求電文（ヘッダ）と HTTP アダプタ実行環境（共通）プロパティファイルの重複時の優先度

指定個所	重複時の優先度
要求電文（ヘッダ）	高
HTTP アダプタ実行環境プロパティファイル （HCSC サーバ単位の指定：あり）	|
HTTP アダプタ実行環境共通プロパティファイル （HCSC サーバ単位の指定：あり）	|
HTTP アダプタ実行環境プロパティファイル （HCSC サーバ単位の指定：なし）	|
HTTP アダプタ実行環境共通プロパティファイル （HCSC サーバ単位の指定：なし）	低

要求電文（ヘッダ）と各プロパティとの対応を次に示します。

(1) リクエスト ID

送信ファイルの入力元フォルダ種別属性，または受信ファイルの出力先フォルダ種別属性に作業フォルダを指定する場合は，要求電文（ヘッダ）で指定してください。作業フォルダを使用しない場合は，指定を省略できます。

表 2-108 リクエスト ID

項目	高 ← 優先度 → 低			デフォルト値	指定要否
	要求電文（ヘッダ）	HTTP アダプタ実行環境プロパティファイル	HTTP アダプタ実行環境共通プロパティファイル		
リクエスト ID	/http-header-request/request-id	指定できません。	指定できません。	－	任意

（凡例）

－：デフォルト値はありません。

(2) パート

必要に応じて要求電文（ヘッダ），またはどちらかのプロパティファイルで指定してください。単一のファイルデータを送信する場合は指定を省略できます。

表 2-109 パート

項目	高 ← 優先度 → 低			デフォルト値	指定要否
	要求電文（ヘッダ）	HTTP アダプタ実行環境プロパティファイル	HTTP アダプタ実行環境共通プロパティファイル		
パート型	/http-header-request/http-body/style	adphttp.request.body.style	adphttp.request.body.style	－	任意
パート名	/http-header-request/http-part/message/part-id	adphttp.request.part.message.part-id	adphttp.request.part.message.part-id	－	任意
	/http-header-request/http-part/files/file/part-id	adphttp.request.part.file.<追番>.part-id	adphttp.request.part.file.<追番>.part-id	－	任意

（凡例）

－：デフォルト値はありません。

(3) リクエストメソッド

要求電文（ヘッダ），またはどちらかのプロパティファイルで必ず指定してください。

表 2-110　リクエストメソッド

項目	高 ← 優先度 → 低			デフォルト値	指定要否
	要求電文（ヘッダ）	HTTP アダプタ実行環境プロパティファイル	HTTP アダプタ実行環境共通プロパティファイル		
リクエストメソッド	/http-header-request/method	adphttp.request.method	adphttp.request.method	－	必須

（凡例）

－：デフォルト値はありません。

(4)　リクエスト URI

スキーム・オーソリティは，要求電文（ヘッダ），またはどちらかのプロパティファイルで必ず指定してください。

パスは，必要に応じて要求電文（ヘッダ），またはどちらかのプロパティファイルで指定してください。ルートパス（"/"）を使用する場合は指定を省略できます。

クエリは，必要に応じて要求電文（ヘッダ），またはどちらかのプロパティファイルで指定してください。クエリを複数の場所で指定した場合，優先度が低い場所で指定したクエリは優先度が高い場所で指定したクエリですべて上書きされます。複数の場所で指定したクエリの結合はできません。

表 2-111　リクエスト URI

項目	高 ← 優先度 → 低			デフォルト値	指定要否
	要求電文（ヘッダ）	HTTP アダプタ実行環境プロパティファイル	HTTP アダプタ実行環境共通プロパティファイル		
スキーム・オーソリティ	/http-header-request/uri-scheme-authority	adphttp.request.uri-scheme-authority	adphttp.request.uri-scheme-authority	－	必須
パス	/http-header-request/uri-path	adphttp.request.uri-path	adphttp.request.uri-path	－	任意
クエリ	/http-header-request/uri-query/任意要素	adphttp.request.uri-query.<追番>	adphttp.request.uri-query.<追番>	－	

（凡例）

－：デフォルト値はありません。

(5)　認証情報

必要に応じて要求電文（ヘッダ），またはどちらかのプロパティファイルで指定してください。認証情報を使用しない場合は指定を省略（デフォルト値を使用）できます。

これらのパラメタは常にセットで指定する必要があります。1 つの場所に認証値，または認証種別属性を単独で指定することはできません。

表 2-112　認証情報

項目	高 ← 優先度 → 低			デフォルト値	指定要否
	要求電文（ヘッダ）	HTTP アダプタ実行環境プロパティファイル	HTTP アダプタ実行環境共通プロパティファイル		
認証値	/http-header-request/http-header-Authorization	adphttp.request.header.authorization	adphttp.request.header.authorization	空文字列	任意
認証種別属性	/http-header-request/http-header-Authorization/@type	adphttp.request.header.authorization.type	adphttp.request.header.authorization.type	none	

(6)　Content-Type

必要に応じて要求電文（ヘッダ），またはどちらかのプロパティファイルで指定してください。Content-Type ヘッダの設定をしない場合は指定を省略できます。

文字コード属性の設定が不要な場合は，指定を省略できます。この場合，メディアタイプだけ使用されます。

文字コード属性を単独で指定した場合は，指定が有効になりません。

表 2-113　Content-Type

項目	高 ← 優先度 → 低			デフォルト値	指定要否
	要求電文（ヘッダ）	HTTP アダプタ実行環境プロパティファイル	HTTP アダプタ実行環境共通プロパティファイル		
メディアタイプ	/http-header-request/http-header-Content-Type	adphttp.request.header.content-type	adphttp.request.header.content-type	−	任意
	/http-header-request/http-part/files/file/Content-Type	adphttp.request.part.file.<追番>.content-type	adphttp.request.part.file.<追番>.content-type	−	
文字コード属性	/http-header-request/http-header-Content-Type/@charset	adphttp.request.header.content-type.charset	adphttp.request.header.content-type.charset	−	
	/http-header-request/http-part/message/Content-Type/@charset	adphttp.request.part.message.content-type.charset	adphttp.request.part.message.content-type.charset	−	
	/http-header-request/http-part/files/file/Content-Type/@charset	adphttp.request.part.file.<追番>.content-type.charset	adphttp.request.part.file.<追番>.content-type.charset	−	

（凡例）

−：デフォルト値はありません。

(7) Cookie

必要に応じて要求電文（ヘッダ），またはどちらかのプロパティファイルで指定してください。Cookie ヘッダの設定をしない場合は指定を省略できます。

Cookie 情報，Cookie 名属性，送信先ホスト名属性は常にセットで指定する必要があります。

送信先パス属性の設定が不要な場合は，指定を省略できます。この場合，送信先パスの判定が行われないため，すべてのパスに対して Cookie が送信されます。

1 つの場所に Cookie 情報，Cookie 名属性，送信先ホスト名属性，送信先パス属性を単独で指定することはできません。

表 2-114 Cookie

項目	高 ← 優先度 → 低			デフォルト値	指定要否
	要求電文（ヘッダ）	HTTP アダプタ実行環境プロパティファイル	HTTP アダプタ実行環境共通プロパティファイル		
Cookie 情報	/http-header-request/http-header-Cookies/Cookie	指定できません。	指定できません。	－	任意
Cookie 名属性	/http-header-request/http-header-Cookies/Cookie/@name	指定できません。	指定できません。	－	
送信先ホスト名属性	/http-header-request/http-header-Cookies/Cookie/@host	指定できません。	指定できません。	－	
送信先パス属性	/http-header-request/http-header-Cookies/Cookie/@path	指定できません。	指定できません。	－	

（凡例）

－：デフォルト値はありません。

(8) リクエストヘッダ

必要に応じて要求電文（ヘッダ），またはどちらかのプロパティファイルで指定してください。

リクエストヘッダを複数の場所で指定した場合，優先度が低い場所で指定したリクエストヘッダは優先度が高い場所で指定したリクエストヘッダですべて上書きされます。複数の場所で指定したリクエストヘッダの結合はできません。

表 2-115　リクエストヘッダ

項目	高 ← 優先度 → 低			デフォルト値	指定要否
	要求電文（ヘッダ）	HTTP アダプタ実行環境プロパティファイル	HTTP アダプタ実行環境共通プロパティファイル		
リクエストヘッダ	/http-header-request/http-header/任意要素	adphttp.request.header.userdef.<追番>	adphttp.request.header.userdef.<追番>	－	任意

（凡例）

－：デフォルト値はありません。

(9)　リクエストボディ（要求電文（ボディ）のデータ送信種別）

必要に応じて要求電文（ヘッダ），またはどちらかのプロパティファイルで指定してください。リクエストボディを送信しない場合は，指定を省略（デフォルト値を使用）できます。

送信ファイルのパラメタを指定した場合は，このパラメタの指定は無視されます。

表 2-116　リクエストボディ（要求電文（ボディ）のデータ送信種別）

項目	高 ← 優先度 → 低			デフォルト値	指定要否
	要求電文（ヘッダ）	HTTP アダプタ実行環境プロパティファイル	HTTP アダプタ実行環境共通プロパティファイル		
リクエストボディ（要求電文（ボディ）のデータ送信種別）	/http-header-request/http-part/message/binding	adphttp.request.part.message.binding	adphttp.request.part.message.binding	none	任意

(10)　送信ファイル

要求電文（ボディ）の代わりにファイルをリクエストボディとして送信する場合は，要求電文（ヘッダ），またはどちらかのプロパティファイルで指定してください。

リクエストボディを送信しない場合や要求電文（ボディ）をリクエストボディとして送信する場合は，指定を省略できます。

これらのパラメタは常にセットで指定する必要があります。1 つの場所に入力元フォルダ名，入力元フォルダ種別属性，入力元ファイル名を単独で指定することはできません。

表 2-117　送信ファイル

項目	高 ← 優先度 → 低			デフォルト値	指定要否
	要求電文（ヘッダ）	HTTP アダプタ実行環境プロパティファイル	HTTP アダプタ実行環境共通プロパティファイル		
入力元フォルダ名	/http-header-request/http-part/files/file/input-folder-name	adphttp.request.part.file.<追番>.input-folder-name	adphttp.request.part.file.<追番>.input-folder-name	−	任意
入力元フォルダ種別属性	/http-header-request/http-part/files/file/input-folder-name/@common	指定できません。	指定できません。	−	
入力元ファイル名	/http-header-request/http-part/files/file/local-file-name	adphttp.request.part.file.<追番>.local-file-name	adphttp.request.part.file.<追番>.local-file-name	−	
	/http-header-request/http-part/files/file/file-name	adphttp.request.part.file.<追番>.file-name	adphttp.request.part.file.<追番>.file-name	−	

（凡例）

−：デフォルト値はありません。

(11)　受信ファイルの出力先

ファイルを受信する場合は，要求電文（ヘッダ），またはどちらかのプロパティファイルで指定してください。ファイルを受信しない場合は，指定を省略できます。

これらのパラメタは常にセットで指定する必要があります。1 つの場所に出力先フォルダ名，出力先フォルダ種別属性，または出力先サブフォルダを単独で指定することはできません。

表 2-118　受信ファイルの出力先

項目	高 ← 優先度→ 低			デフォルト値	指定要否
	要求電文（ヘッダ）	HTTP アダプタ実行環境プロパティファイル	HTTP アダプタ実行環境共通プロパティファイル		
出力先フォルダ名	/http-header-request/output-folder-name	adphttp.request.output-folder-name	adphttp.request.output-folder-name	−	任意
出力先フォルダ種別属性	/http-header-request/output-folder-name/@common	指定できません。	指定できません。	−	
出力先サブフォルダ	/http-header-request/output-sub-folder	adphttp.request.output-sub-folder	adphttp.request.output-sub-folder	−	

（凡例）

−：デフォルト値はありません。

(12) 署名付き AWS API リクエスト

署名付き AWS API リクエストで認証する場合は，要求電文（ヘッダ），またはどちらかのプロパティファイルで指定してください。署名付き AWS API リクエストで認証しない場合は，指定を省略できます。

表 2-119 署名付き AWS API リクエスト

項目	高 ← 優先度→ 低			デフォルト値	指定要否
	要求電文（ヘッダ）	HTTP アダプタ実行環境プロパティファイル	HTTP アダプタ実行環境共通プロパティファイル		
署名付き AWS API リクエストの認証方法	/http-header-request/http-header-signed-aws-api-request/@type	adphttp.request.header.signed.aws.api.request.type	adphttp.request.header.signed.aws.api.request.type	none	任意
アクセスキーペア ID	/http-header-request/http-header-signed-aws-api-request	adphttp.request.header.signed.aws.api.request	adphttp.request.header.signed.aws.api.request	－	
署名付き AWS API リクエストの署名バージョン	/http-header-request/http-header-signed-aws-api-request/@version	adphttp.request.header.signed.aws.api.request.version	adphttp.request.header.signed.aws.api.request.version	aws4_request	
接続先 AWS サービス	/http-header-request/http-header-signed-aws-api-request/@service	adphttp.request.header.signed.aws.api.request.service	adphttp.request.header.signed.aws.api.request.service	－	
接続先 AWS リージョン	/http-header-request/http-header-signed-aws-api-request/@region	adphttp.request.header.signed.aws.api.request.region	adphttp.request.header.signed.aws.api.request.region	－	

（凡例）

－：デフォルト値はありません。

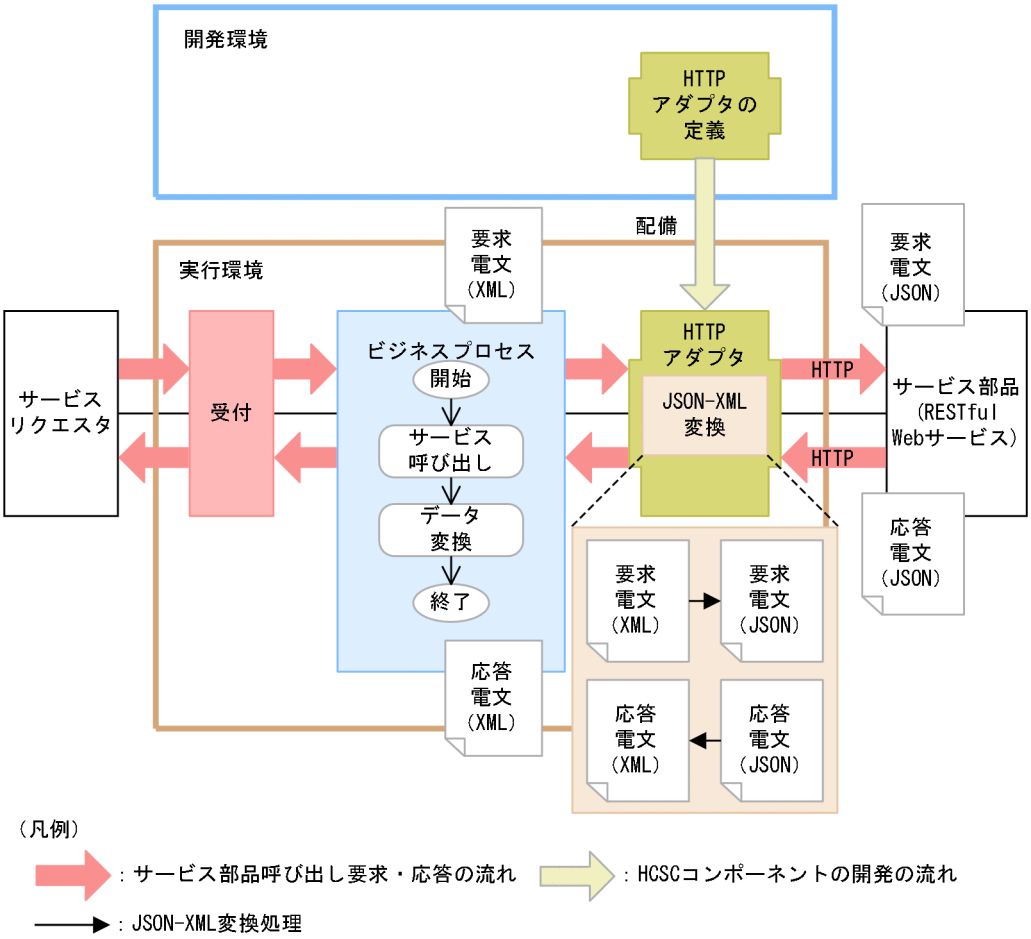
2.16.15 HTTP アダプタの JSON-XML 変換機能

HTTP アダプタの JSON-XML 変換機能とは，HTTP アダプタで XML 形式の要求電文を JSON 形式に，または JSON 形式の応答電文を XML 形式に変換する機能です。

(1) HTTP アダプタの JSON-XML 変換機能の概要

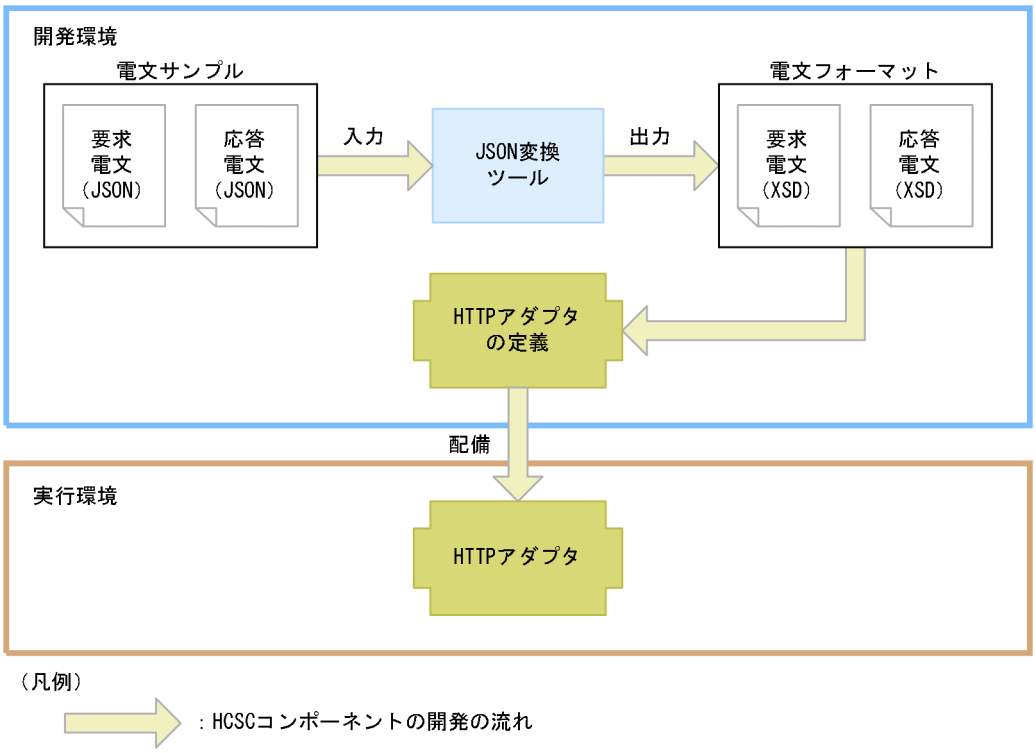
ビジネスプロセスで扱う XML 形式の要求電文を，HTTP アダプタで JSON 形式に変換して RESTful Web サービスに送信します。また，RESTful Web サービスから応答された JSON 形式の応答電文を XML 形式に変換し，ビジネスプロセスへ応答します。HTTP アダプタの JSON-XML 変換機能を使用したサービス部品呼び出し要求および応答の流れを次の図に示します。

図 2-175　JSON-XML 変換機能を使用したサービス部品呼び出し要求・応答の流れ（HTTP アダプタ）



HTTP アダプタに設定する電文フォーマット（XML スキーマ）は，JSON 変換ツールで生成します。JSON 変換ツールは，JSON 形式の電文のサンプルを受け取り，JSON-XML 変換後の XML 形式の電文に適合する電文フォーマットを生成します。HTTP アダプタの開発の流れを次の図に示します。

図 2-176　JSON-XML 変換機能を使用する HTTP アダプタの開発の流れ



HTTP アダプタの定義の詳細については，マニュアル「サービスプラットフォーム 開発ガイド 受付・アダプタ定義編」の「3.3.13　HTTP アダプタを定義する」を，JSON 変換ツールの詳細については，マニュアル「サービスプラットフォーム 開発ガイド 基本開発編」の「14.　JSON 変換ツールを使用したデータ変換」を参照してください。

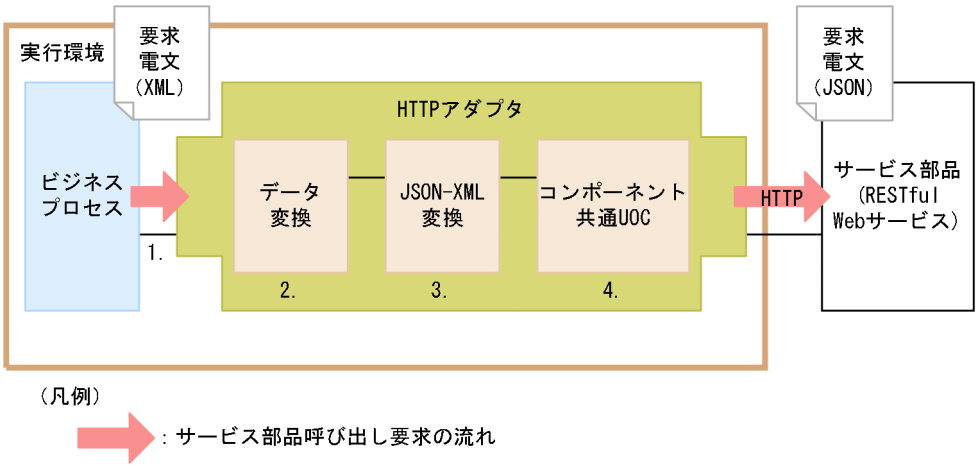
(2)　HTTP アダプタのリクエスト処理時の JSON-XML 変換機能

HTTP アダプタのリクエスト処理時の JSON-XML 変換のタイミングや設定について説明します。

(a)　HTTP アダプタのリクエスト処理時の JSON-XML 変換のタイミング

HTTP アダプタのリクエスト処理時の，JSON-XML 変換のタイミングを次の図に示します。

図 2-177　リクエスト処理時の JSON-XML 変換のタイミング（HTTP アダプタ）



図中の 1.～4.について説明します。

1. ビジネスプロセスから要求電文を受信します。
2. データ変換では，ビジネスプロセスから送信された XML 形式の要求電文を扱います。
3. HTTP アダプタでは，データ変換とコンポーネント共通 UOC の呼び出しの間で JSON-XML 変換が実行されます。
4. コンポーネント共通 UOC では，JSON-XML 変換機能で変換された JSON 形式の要求電文を扱います。

(b)　HTTP アダプタのリクエスト処理時に JSON-XML 変換をする場合の設定

HTTP アダプタ定義ファイルの adphttp.request.switchover.json-transfer.mode プロパティの値が「true」の場合は，HTTP アダプタのリクエスト処理時に JSON-XML 変換が実行されます。要求電文（ヘッダ）の json-transfer 要素の値が「false」の場合は，HTTP アダプタのリクエスト処理時に JSON-XML 変換は実行されません。HTTP アダプタ定義ファイルのプロパティおよび要求電文（ヘッダ）の設定と JSON-XML 変換の実行可否を次の表に示します。

表 2-120　リクエスト処理時の JSON-XML 変換の実行可否（HTTP アダプタ）

HTTP アダプタ定義ファイル adphttp.request.switchover.json-transfer.mode プロパティ	要求電文（ヘッダ） json-transfer 要素	JSON-XML 変換の実行
－（プロパティを設定していない）	要素なし	実行されない
	false	
	true	
false	要素なし	実行されない
	false	
	true	
true	要素なし	実行される

HTTP アダプタ定義ファイル adphttp.request.switchover.json-transfer.mode プロパティ	要求電文（ヘッダ） json-transfer 要素	JSON-XML 変換の実行
true	false	実行されない
	true	実行される

(c)　HTTP アダプタの HTTP リクエストヘッダに関連する処理

JSON-XML 変換機能に関連している HTTP アダプタの HTTP リクエストヘッダについて説明します。

HTTP アダプタの HTTP リクエストヘッダの処理の詳細については，「2.16.3　HTTP リクエストと要求電文の関係」を参照してください。

Content-Type ヘッダ

JSON-XML 変換は，要求電文（ヘッダ）の Content-Type の設定に関わらず実行されます。また，要求電文（ヘッダ）に Content-Type が設定されていない場合は，HTTP リクエストヘッダに Content-Type ヘッダが設定されません。

Content-Type ヘッダの charset 属性

要求電文（ヘッダ）の Content-Type に charset 属性が設定されていない場合は，HTTP リクエストヘッダに設定される Content-Type ヘッダに charset 属性が設定されません。

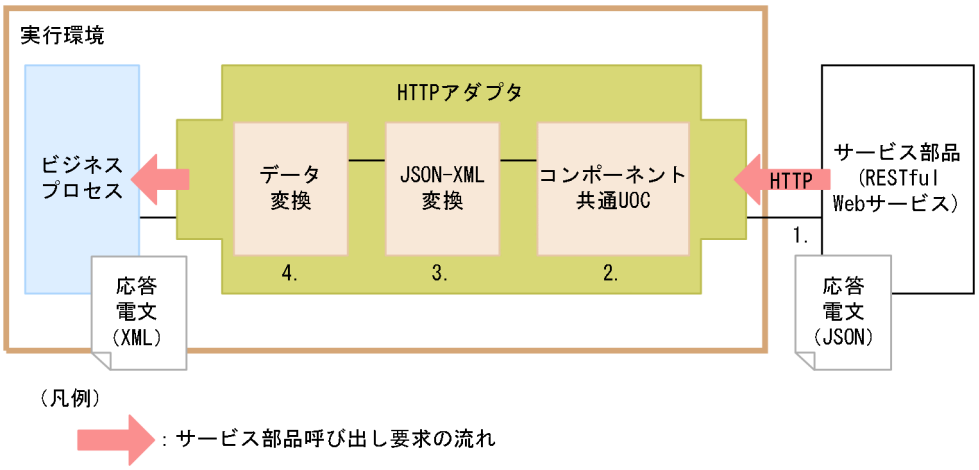
(3)　HTTP アダプタのレスポンス処理時の JSON-XML 変換機能

HTTP アダプタのレスポンス処理時の JSON-XML 変換のタイミングや設定について説明します。

(a)　HTTP アダプタのレスポンス処理時の JSON-XML 変換のタイミング

HTTP アダプタのレスポンス処理時の JSON-XML 変換のタイミングを次の図に示します。

図 2-178　レスポンス処理時の JSON-XML 変換のタイミング（HTTP アダプタ）



図中の 1.～4.について説明します。

1. HTTP アダプタが RESTful Web サービスから応答電文を受信します。
2. HTTP アダプタのコンポーネント共通 UOC では，サービス部品から返却された JSON 形式の応答電文を扱います。
3. HTTP アダプタでは，コンポーネント共通 UOC の呼び出しとデータ変換の間で JSON-XML 変換が実行されます。
4. HTTP アダプタのデータ変換では，JSON-XML 変換機能で変換された XML 形式の応答電文を扱います。

(b) HTTP アダプタのレスポンス処理時に JSON-XML 変換をする場合の設定

HTTP アダプタ定義ファイルの adphttp.response.switchover.json-transfer.mode プロパティの値が「true」の場合，HTTP アダプタのレスポンス処理時に JSON-XML 変換が実行されます。応答電文（ヘッダ）では，JSON-XML 変換を実行するかどうかの切り替えはできません。HTTP アダプタ定義ファイルのプロパティの設定と JSON-XML 変換の実行可否を次の表に示します。

表 2-121 レスポンス処理時の JSON-XML 変換の実行可否（HTTP アダプタ）

HTTP アダプタ定義ファイル adphttp.response.switchover.json-transfer.mode プロパティ	JSON-XML 変換の実行
－（プロパティを設定していない）	実行されない
false	実行されない
true	実行される

(c) HTTP アダプタの HTTP レスポンスヘッダに関連する処理

JSON-XML 変換機能に関連している HTTP アダプタの HTTP レスポンスヘッダについて説明します。

HTTP アダプタの HTTP レスポンスヘッダの処理の詳細については，マニュアル「サービスプラットフォーム 解説」の「2.16.4 HTTP レスポンスと応答電文の関係」を参照してください。

Content-Type ヘッダについて

JSON-XML 変換は，HTTP レスポンスヘッダの Content-Type ヘッダの値に関わらず実行されます。また，Content-Type ヘッダが HTTP レスポンスヘッダに設定されていない場合は，応答電文（ヘッダ）に Content-Type が設定されません。

Content-Type ヘッダの charset 属性について

HTTP レスポンスヘッダの Content-Type ヘッダに charset 属性が設定されていない場合は，応答電文（ヘッダ）の charset 属性が設定されません。

2.16.16 WWW 認証を使用したセキュアな接続

HTTP アダプタでは，WWW 認証を使用して対象のサーバと安全に接続できます。

WWW 認証を使用するには，要求電文（ヘッダ），HTTP アダプタ実行環境プロパティファイル，または HTTP アダプタ実行環境共通プロパティファイルでの設定が必要です。

WWW 認証に関する設定個所および設定内容を次に示します。

表 2-122　WWW 認証に関する設定項目

項目	設定個所	設定項目	設定する値
認証タイプ	要求電文（ヘッダ）	http-header-Authorization 要素の type 属性	次のどちらかを設定します。 • none WWW 認証を使用しません。 • raw WWW 認証を使用します。 • basic ベーシック認証を使用します。
	HTTP アダプタ実行環境プロパティファイル，HTTP アダプタ実行環境共通プロパティファイル	adphttp.request.header.authorization.type プロパティ	
認証情報	要求電文（ヘッダ）	http-header-request/http-header-Authorization 要素	WWW 認証の認証情報を指定します。
	HTTP アダプタ実行環境プロパティファイル，HTTP アダプタ実行環境共通プロパティファイル	adphttp.request.header.authorization プロパティ	

2.16.17　OAuth 2.0 に対応したサービスへのアクセス

HTTP アダプタは，OAuth 2.0 に対応したサービスへアクセスするために，OAuth 2.0 で必要な情報が容易に設定できるインターフェースを提供します。

(1)　HTTP アダプタにおける OAuth 2.0 の認可フロー対応機能の概要

OAuth 2.0 に対応した認証機能は，認可および認証の処理方式のうち，認証部分を HTTP アダプタで実現します。

前提条件

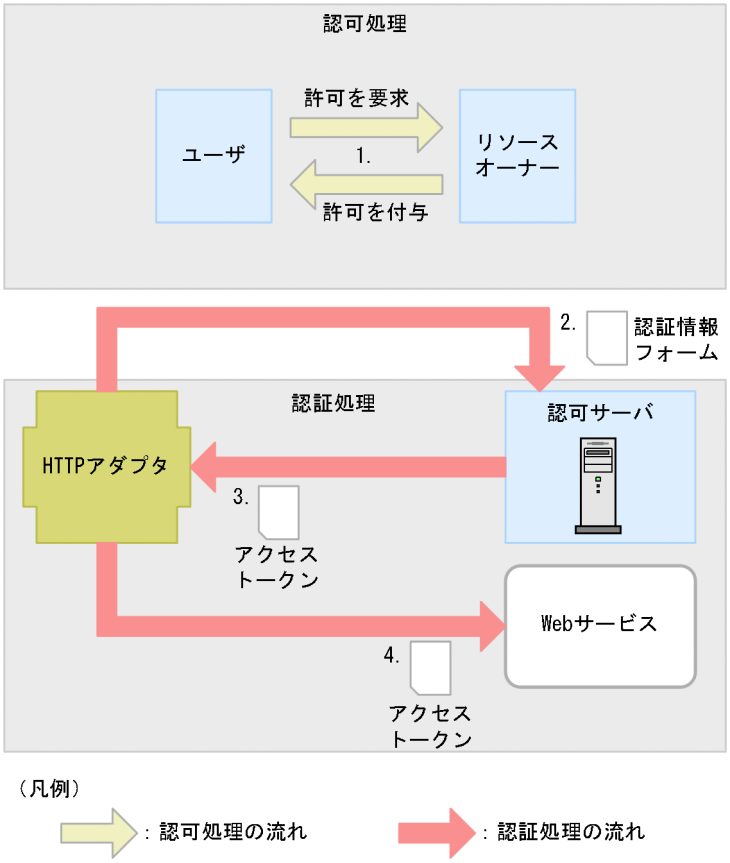
ユーザは事前に，HTTP アダプタが認証情報を作成する際に使用する情報を登録する必要があります。

認証情報の登録方法については，マニュアル「サービスプラットフォーム リファレンス」の「csauseradd (パスワード認証のユーザ情報の登録・更新)」および「csakeyadd（公開鍵認証の秘密鍵情報の登録・更新)」を参照してください。

例

処理方式の例を次の図に示します。

図 2-179　認可および認証の処理方式の例



説明

1. ユーザはリソースの使用許可を要求し，リソースオーナーから使用許可を取得します。
2. HTTP アダプタは，ユーザが要求電文に指定したフォームデータ，および認証情報を基に作成したフォームデータを OAuth 2.0 が対応する認証フローを使用して，認可サーバに送信します。
3. 認可サーバは送信された認証処理を基に認証処理を実行し，特定の Web サービス呼び出しに使用するアクセストークンを HTTP アダプタに返します。
4. HTTP アダプタは取得したアクセストークンを使用して Web サービスを呼び出します。

(2)　OAuth 2.0 に対応した HTTP 通信の例

HTTP アダプタから，OAuth 2.0 が対応する認証フローを使用して，認証処理を実施します。このとき，認可サーバからアクセストークンを取得します。取得したアクセストークンを使用して Web サービスを呼び出します。

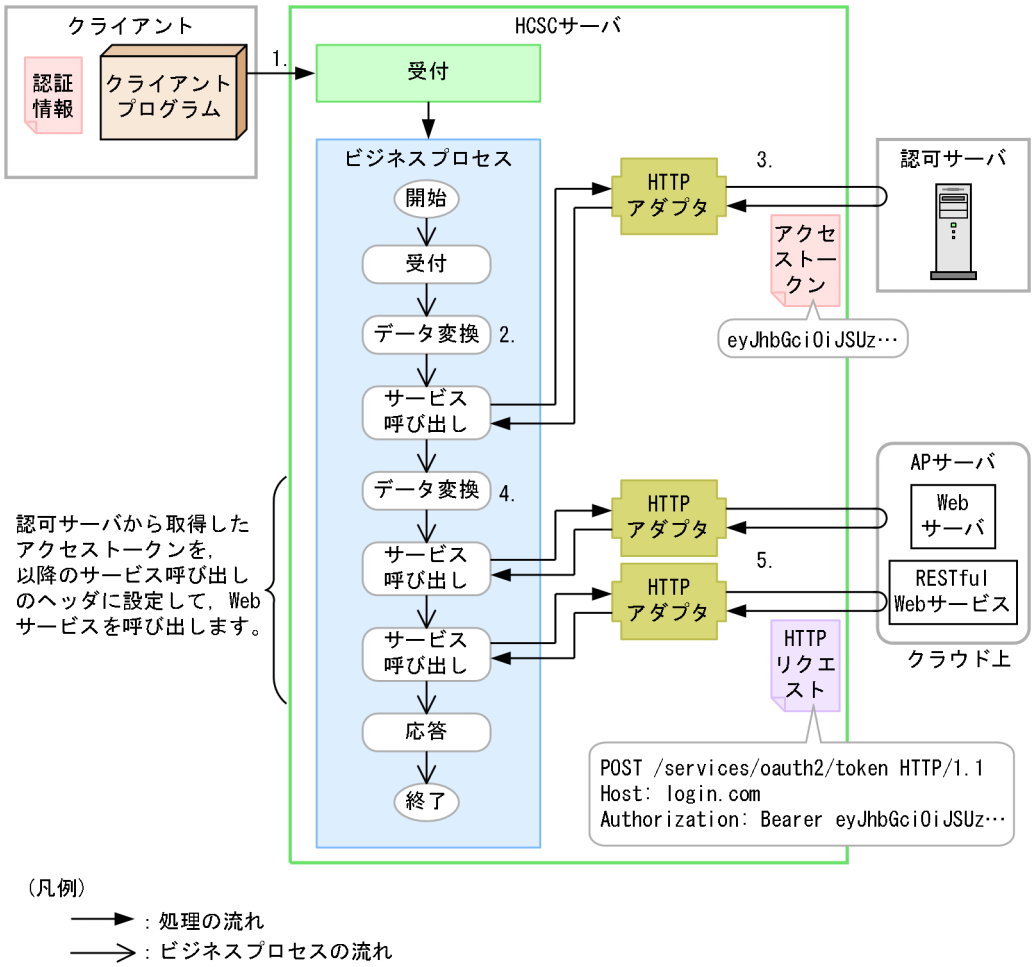
前提条件

Web サービスや認可サーバが，HTTP アダプタが対応する OAuth 2.0 の認証フローに対応している必要があります。

構成例

OAuth 2.0 に対応した認証例を次の図に示します。

図 2-180　OAuth 2.0 に対応した認証例



説明

1. クライアントのプログラムからビジネスプロセスを実行するリクエストを発行します。
2. データ変換アクティビティを使用して，認証情報を HTTP アダプタのボディ（Form）に設定します。
3. HTTP アダプタを使用して，認可サーバからアクセストークンを取得します。
4. 取得したアクセストークンを，データ変換アクティビティを使用して，Web サービスを呼び出す HTTP アダプタのヘッダに設定します。
5. アクセストークンを設定した HTTP アダプタで Web サービスを呼び出します。

設定の詳細については，「(6)　取得したアクセストークンを使用して Web サービスを呼び出す場合の HTTP アダプタの設定」を参照してください。

(3)　HTTP アダプタが対応する認証フロー

HTTP アダプタが対応する OAuth 2.0 の認証フローを次に示します。

- JWT Bearer Token Flow
- Resource Owner Password Credentials Flow

(a) JWT Bearer Token Flow

JWT Bearer Token Flow は，JWT（JSON Web Token）を使用した電子署名付きの認証フローです。電子署名によって認証情報の改ざんをチェックできます。

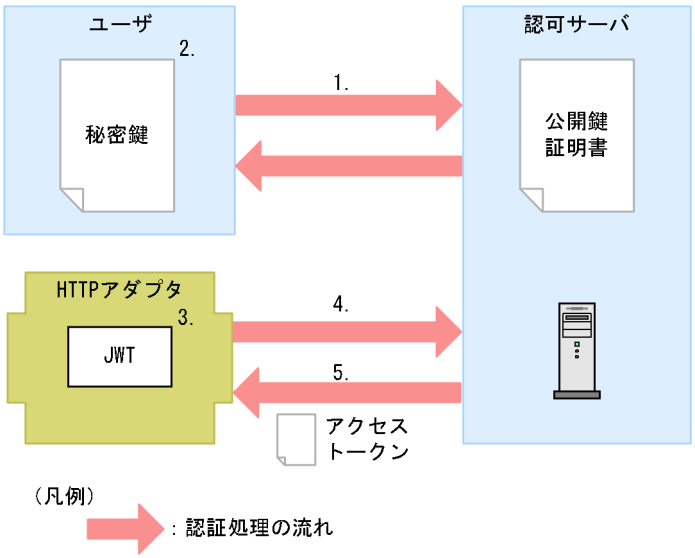
前提条件

JWT Bearer Token Flow は，信頼関係を構築できるシステムから，ユーザによる認証および認可の手順を踏まずに，直接アクセストークンを取得できる場合に使用できます。

例

JWT Bearer Token Flow の流れを次に示します。

図 2-181 JWT Bearer Token Flow の流れ



説明

1. ユーザは事前に証明書と秘密鍵を作成して，証明書を認可サーバに登録し，認可サーバ側で公開鍵を保持します。
2. JWT を格納するフォームデータ（assertion）を HTTP アダプタで作成する場合，ユーザは事前に csakeyadd コマンドを使用して，秘密鍵を登録します。
3. JWT を格納するフォームデータ（assertion）を HTTP アダプタで作成する場合，認証情報を基に電子署名をして，JWT を作成します。
4. HTTP アダプタは，JWT を認可サーバに送信します。
5. 認可サーバで JWT を検証し，検証結果が正しい場合にアクセストークンを発行します。

(b) Resource Owner Password Credentials Flow

Resource Owner Password Credentials Flow は，HTTP アダプタがリソースオーナーのユーザ名とパスワードを使用して，認可サーバにアクセストークンを要求する認証フローです。

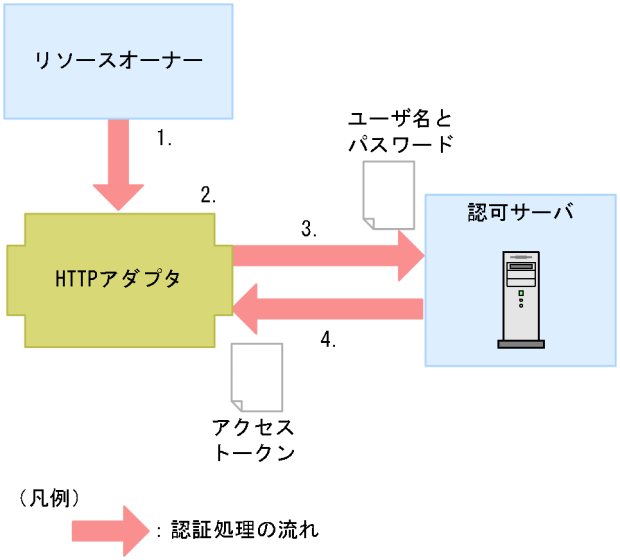
前提条件

リソースオーナーが，HTTP アダプタにユーザ名とパスワードを提供してもよい場合に使用できます。

例

Resource Owner Password Credentials Flow の流れを次に示します。

図 2-182　Resource Owner Password Credentials Flow の流れ



説明

1. リソースオーナーが，ユーザ名とパスワードを提供します。
2. ユーザは事前に csauseradd コマンドを使用して，提供されたユーザ名とパスワードを登録します。
3. HTTP アダプタは，認可サーバのトークンエンドポイントに対して，ユーザが指定したユーザ名と，csauseradd コマンドで登録したパスワードをリクエストに含め，リクエストを送信します。
4. 認可サーバはクライアントの認証を実行し，ユーザ名とパスワードの有効性を検査します。有効であれば，認可サーバはアクセストークンを発行します。

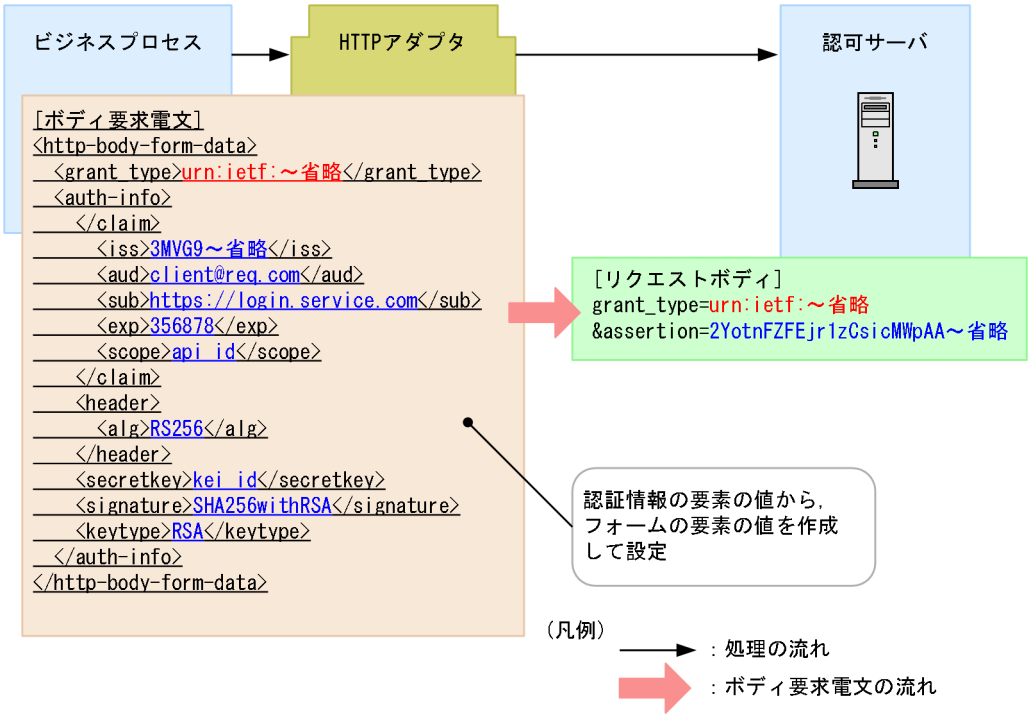
(4)　認証フローを使用する場合の HTTP リクエストおよび HTTP レスポンス

OAuth 2.0 の認証フローを使用する場合の HTTP リクエスト・レスポンスと，要求・応答電文に設定する値について説明します。

(a)　HTTP アダプタから認可サーバに送信するリクエストボディ

アクセストークンを取得する場合は，HTTP リクエストボディで送信するデータ種別としてフォームデータ（ノーマルモード）を選択します。

図 2-183　認証フローとして JWT Bearer Token Flow を使用する場合のフォームデータ設定の概要

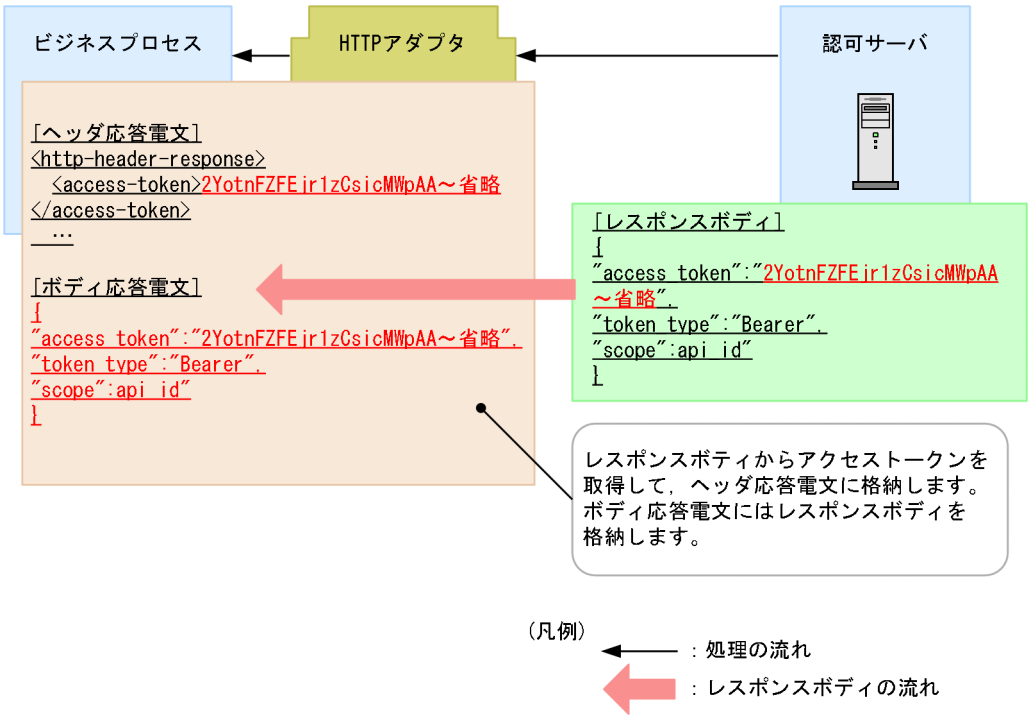


(b)　認可サーバから受信したレスポンスボディの情報を格納する応答電文

HTTP アダプタでは，OAuth 2.0 の認証フローを使用して認可サーバから認証情報を受信する場合，JSON 形式の応答だけをサポートしています。HTTP アダプタは，認証情報を受信したとき，HTTP レスポンスボディの情報を解析し，アクセストークンを応答電文（ヘッダ）に格納します。

また，レスポンスボディの内容は，ボディ応答電文へ割り当てられます。

図 2-184　認証フローとして JWT Bearer Token Flow を使用する場合のレスポンスの概要



(5)　認証フローを使用してアクセストークンを取得する場合の HTTP アダプタの設定

OAuth 2.0 が対応する認証フローを使用して，アクセストークンを取得する場合の HTTP アダプタの設定内容について説明します。

(a)　要求電文の設定

要求電文のボディにフォームデータとして，認証フローごとに対応する値を設定して，認可サーバにリクエストを送信します。共通の設定項目として，キー名称の grant_type に使用する認証フローに対応する固定の文字列を設定します。

注意事項

パスワードなどの情報を送信するため，送信時に https を使用するなどセキュリティを考慮する必要があります。

●JWT Bearer Token Flow を使用する場合

- 対応する電文フォーマット

  認証フローに対応する電文フォーマットを次に示します。

認証フロー		ボディ要求電文フォーマット
JWT Bearer Token Flow	assertion を HTTP アダプタで作成する場合	adphttp_body_form-data_jwt_assertion1.xsd

認証フロー		ボディ要求電文フォーマット
JWT Bearer Token Flow	HTTP アダプタ以外の発行者が作成した assertion を HTTP アダプタで使用する場合	adphttp_body_form-data_jwt_assertion2.xsd

- assertion 設定機能の設定

  サービスプラットフォームでは，HTTP アダプタ内で，リクエストボディで設定されたクレーム情報や，ヘッダ情報などを基に作成した値を assertion の設定値として使用する機能を提供します。

  この機能を使うために，ユーザは csakeyadd コマンドを使用して，認可サーバから取得した秘密鍵を登録する必要があります。ユーザが要求電文（ボディ）に設定した認証情報を基に，HTTP アダプタが assertion の設定値を作成します。作成した assertion の設定値を，リクエスト時に送信するフォームデータの「assertion」に設定します。

  詳細については，マニュアル「サービスプラットフォーム 開発ガイド 受付・アダプタ定義編」の「3.3.13　HTTP アダプタを定義する」の電文フォーマットの作成に関する説明を参照してください。

- JWT Bearer Token Flow で設定するリクエストボディ

  JWT Bearer Token Flow で設定するリクエストボディを次に示します。

キー名称	設定値
grant_type	urn:ietf:params:oauth:grant-type:jwt-bearer（固定文字列）
assertion	認証情報から作成した JWT bearer token

- 任意の設定項目を指定する場合の電文フォーマットの設定項目

  そのほかの任意の設定項目を指定する場合は，電文フォーマットに設定項目を追加してください。

  設定方法の詳細については，マニュアル「サービスプラットフォーム 開発ガイド 基本開発編」の「4.　電文フォーマットの作成」を参照してください。

  カスタマイズした電文フォーマットの例を次に示します。これらの例は一部を抜粋しています。

adphttp_body_form-data_jwt_assertion1.xsd の場合

ここでは，Authorization Grants として scope パラメタ，JWT のクレームとして any 要素，および JWT のヘッダとして type 要素を追加する例を示します。

＜修正前＞

```
<xsd:complexType name="http-body-form-data-type">
   <xsd:sequence>
      <xsd:element name="grant_type" type="xsd:string" minOccurs="1" maxOccurs="1"/>
      <xsd:element name="auth-info" type="tns:auth-info-type" minOccurs="1" maxOccurs=
"1"/>
   </xsd:sequence>
 </xsd:complexType>

 <xsd:complexType name="claim-type">
   <xsd:sequence>
      <xsd:element name="iss" type="xsd:string" minOccurs="0" maxOccurs="1"/>
      <xsd:element name="aud" type="xsd:string" minOccurs="0" maxOccurs="1"/>
      <xsd:element name="sub" type="xsd:string" minOccurs="0" maxOccurs="1"/>
```

```
      <xsd:element name="exp" type="xsd:string" minOccurs="0" maxOccurs="1"/>
      <xsd:element name="nbf" type="xsd:string" minOccurs="0" maxOccurs="1"/>
      <xsd:element name="iat" type="xsd:string" minOccurs="0" maxOccurs="1"/>
      <xsd:element name="jti" type="xsd:string" minOccurs="0" maxOccurs="1"/>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="header-type">
    <xsd:sequence>
      <xsd:element name="alg" type="xsd:string" minOccurs="1" maxOccurs="1"/>
    </xsd:sequence>
  </xsd:complexType>
```

＜修正後＞

```
<xsd:complexType name="http-body-form-data-type">
    <xsd:sequence>
      <xsd:element name="grant_type" type="xsd:string" minOccurs="1" maxOccurs="1"/>
      <xsd:element name="auth-info" type="tns:auth-info-type" minOccurs="1" maxOccurs=
"1"/>
      <xsd:element name="scope" type="xsd:string" minOccurs="0" maxOccurs="1"/>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="claim-type">
    <xsd:sequence>
      <xsd:element name="iss" type="xsd:string" minOccurs="0" maxOccurs="1"/>
      <xsd:element name="aud" type="xsd:string" minOccurs="0" maxOccurs="1"/>
      <xsd:element name="sub" type="xsd:string" minOccurs="0" maxOccurs="1"/>
      <xsd:element name="exp" type="xsd:string" minOccurs="0" maxOccurs="1"/>
      <xsd:element name="nbf" type="xsd:string" minOccurs="0" maxOccurs="1"/>
      <xsd:element name="iat" type="xsd:string" minOccurs="0" maxOccurs="1"/>
      <xsd:element name="jti" type="xsd:string" minOccurs="0" maxOccurs="1"/>
      <xsd:element name="any" type="xsd:string" minOccurs="0" maxOccurs="1"/>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="header-type">
    <xsd:sequence>
      <xsd:element name="alg" type="xsd:string" minOccurs="1" maxOccurs="1"/>
      <xsd:element name="typ" type="xsd:string" minOccurs="0" maxOccurs="1"/>
    </xsd:sequence>
  </xsd:complexType>
```

adphttp_body_form-data_jwt_assertion2.xsd の場合

ここでは，Authorization Grants として scope パラメタを追加する例を示します。

＜修正前＞

```
<xsd:complexType name="http-body-form-data-type">
    <xsd:sequence>
      <xsd:element name="grant_type" type="xsd:string" minOccurs="1" maxOccurs="1"/>
      <xsd:element name="assertion" type="xsd:string" minOccurs="1" maxOccurs="1"/>
    </xsd:sequence>
  </xsd:complexType>
```

＜修正後＞

```
<xsd:complexType name="http-body-form-data-type">
   <xsd:sequence>
      <xsd:element name="grant_type" type="xsd:string" minOccurs="1" maxOccurs="1"/>
      <xsd:element name="assertion" type="xsd:string" minOccurs="1" maxOccurs="1"/>
      <xsd:element name="scope" type="xsd:string" minOccurs="0" maxOccurs="1"/>
   </xsd:sequence>
 </xsd:complexType>
```

●Resource Owner Password Credentials Flow を使用する場合

- 対応する電文フォーマット

  認証フローに対応する電文フォーマットを次に示します。

認証フロー	ボディ要求電文フォーマット
Resource Owner Password Credentials Flow	adphttp_body_form-data_password.xsd

- パスワードの設定

  Resource Owner Password Credentials Flow を使用する場合，username 要素にリソースオーナーのユーザネームとして，csauseradd コマンドで登録したユーザ名を設定します。password 要素は，csauseradd コマンドで登録したパスワードを HTTP アダプタが取得して設定します。

  詳細については，マニュアル「サービスプラットフォーム 開発ガイド 受付・アダプタ定義編」の「3.3.13　HTTP アダプタを定義する」の電文フォーマットの作成に関する説明を参照してください。

- JWT Bearer Token Flow で設定するリクエストボディ

  Resource Owner Password Credentials Flow で設定するリクエストボディを次に示します。

キー名称	設定値
grant_type	password（固定文字列）
username	リソースオーナーのユーザ名
password	リソースオーナーのパスワード

- 任意の設定項目を指定する場合の電文フォーマットの設定項目

  そのほかの任意の設定項目を指定する場合は，電文フォーマットに個別に設定項目を追加してください。

  設定方法の詳細については，マニュアル「サービスプラットフォーム 開発ガイド 基本開発編」の「4.　電文フォーマットの作成」を参照してください。

  カスタマイズした電文フォーマットの例を次に示します。例は全体から一部を抜粋しています。

adphttp_body_form-data_password.xsd の場合

＜修正前＞

```
<xsd:complexType name="http-body-form-data-type">
   <xsd:sequence>
      <xsd:element name="grant_type" type="xsd:string" minOccurs="1" maxOccurs="1"/>
      <xsd:element name="username" type="xsd:string" minOccurs="1" maxOccurs="1"/>
   </xsd:sequence>
 </xsd:complexType>
```

＜修正後＞

```
<xsd:complexType name="http-body-form-data-type">
   <xsd:sequence>
      <xsd:element name="grant_type" type="xsd:string" minOccurs="1" maxOccurs="1"/>
      <xsd:element name="username" type="xsd:string" minOccurs="1" maxOccurs="1"/>
      <xsd:element name="scope" type="xsd:string" minOccurs="0" maxOccurs="1"/>
   </xsd:sequence>
 </xsd:complexType>
```

(b)　応答電文の設定

認可サーバからレスポンス受信時に応答電文（ヘッダ）に，アクセストークンが格納されます。

詳細については，マニュアル「サービスプラットフォーム 開発ガイド 受付・アダプタ定義編」の「3.3.13 HTTP アダプタを定義する」の電文フォーマットの形式に関する説明を参照してください。

応答電文にはアクセストークンが格納されているため，ユーザ電文トレース出力時にアクセストークンが，ユーザ電文トレースに出力されます。ユーザ電文トレースにアクセストークンを出力したくない場合は，ユーザ電文トレースを取得しない設定にしてください。

(c)　HTTP アダプタ定義ファイルの設定

HTTP アダプタ実行時に OAuth 2.0 の認証フローを使用する場合，次に示すプロパティを設定します。

プロパティの詳細については，マニュアル「サービスプラットフォーム リファレンス」の「3.10.1　HTTP アダプタ定義ファイル」を参照してください。

キー名称	設定値
adphttp.oauth.authentication	true

注意事項

OAuth 2.0 の認証フローを使用する場合に，adphttp.oauth.authentication プロパティに true を指定しないと，認証情報が不正になるため，HTTP リクエストがエラー応答したり，応答電文（ヘッダ）にアクセストークンが格納されなかったりすることがあります。

(d)　HTTP アダプタ実行環境プロパティファイルの設定

csakeyadd コマンド，または csauseradd コマンドを事前に実行した場合，次の表のプロパティの設定値に，コマンド実行時の引数に指定したパスワード認証情報定義ファイル，または公開鍵認証情報定義ファイルのパスを設定してください。

HTTP アダプタ実行環境プロパティファイルから設定するプロパティを次に示します。

キー名称	設定値
adphttp.config.password-auth-inf-filepath	パスワード認証情報定義ファイルのパス
adphttp.config.publickey-auth-inf-filepath	公開鍵認証情報定義ファイルのパス

OAuth 2.0 が対応する認証フローを使用する場合，認証情報をフォームデータとして送信するため，次の表のプロパティの設定値に「form-data」を設定してください。

HTTP アダプタ実行環境プロパティファイルから設定するプロパティを次に示します。

キー名称	設定値
adphttp.request.part.message.binding	form-data

各プロパティの詳細については，マニュアル「サービスプラットフォーム リファレンス」の「6.14.2 HTTP アダプタ実行環境プロパティファイル」を参照してください。

(6) 取得したアクセストークンを使用して Web サービスを呼び出す場合の HTTP アダプタの設定

取得したアクセストークンを，Web サービスを呼び出す HTTP アダプタの要求電文（ヘッダ）の Authorization に設定して Web サービスを呼び出します。

要求電文（ヘッダ）から設定する場合の設定項目を次に示します。

キー名称	設定値
/http-header-request/http-header-Authorization/@type 属性	raw
/http-header-request/http-header-Authorization 要素	Bearer△アクセストークン

（凡例）

△：半角スペース

要求電文（ヘッダ）で指定する場合の例を次に示します。

```
<http-header-request>
  <http-header-Authorization type=”raw”>Bearer eyJhbGciOiJSUz…
</http-header-Authorization>
</http-header-request>
```

(7) 認証フローを使用する場合の定義例

Salesforce が提供する API へ接続することを想定した定義例について説明します。

(a) 定義の条件

定義例の前提条件を次に示します。

- 認証処理で使用する認証情報は，受付の要求電文から取得します。
- assertion は，HTTP アダプタで作成します。
- 認可サーバ，Web サービスは，Salesforce を想定しています。

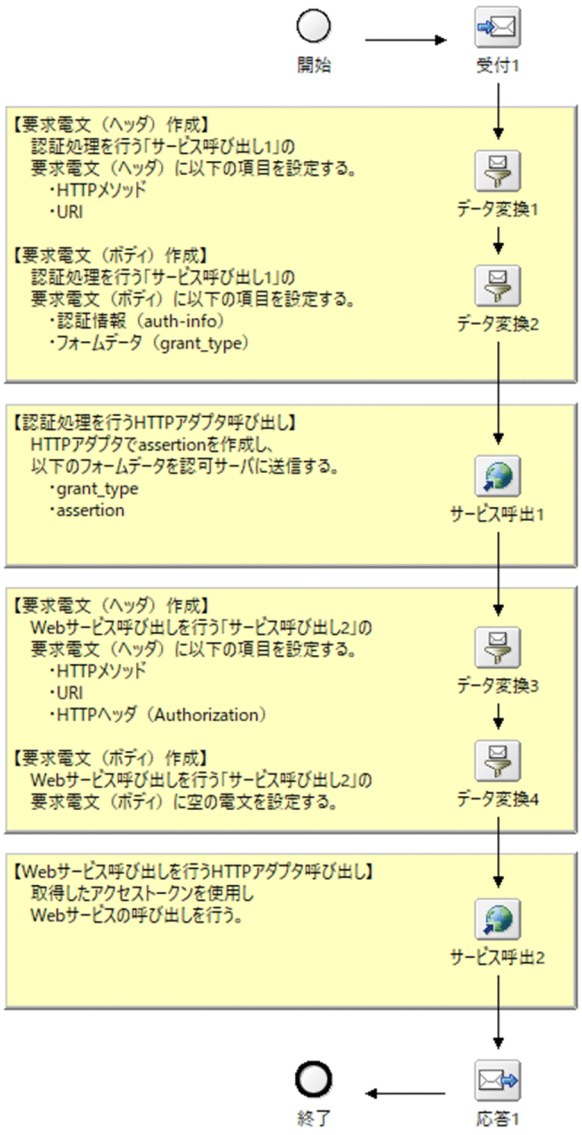
- Web サービスの呼び出しだけで，応答電文の解析は実施しません。

Salesforce の設定値の詳細については，Salesforce のドキュメントを参照してください。

(b)　ビジネスプロセスの定義例

ビジネスプロセスの設定例を次に示します。

図 2-185　JWT Bearer Token Flow を使用して Web サービスを呼び出すビジネスプロセスの例



(c)　認証処理を行う HTTP アダプタの定義例

ここでは OAuth 2.0 の認証フロー（JWT Bearer Token Flow）を使用する上で，必要となる次の作業について説明します。

HTTP アダプタの基本的な定義の流れについては，マニュアル「サービスプラットフォーム 開発ガイド 受付・アダプタ定義編」の「3.3.13　HTTP アダプタを定義する」を参照してください。

- 秘密鍵の登録
- 電文フォーマットの設定
- 定義ファイルの設定
- データ変換の定義

それぞれの定義について説明します。

●秘密鍵の登録

HTTP アダプタで assertion の設定値を作成する場合は，ユーザは事前に JWT の作成に使用する秘密鍵を，csakeyadd コマンドを実行して登録する必要があります。

●電文フォーマットの作成

認証処理を行う HTTP アダプタでは，要求電文，応答電文の電文フォーマットに次の XML スキーマを定義してください。

要求電文（ヘッダ）

- 格納先

  <サービスプラットフォームのインストールディレクトリ>¥CSC¥custom-adapter¥HTTP¥schema

- 使用する定義

  adphttp_header_request1.xsd

要求電文（ボディ）

認証情報を受付の要求電文から取得し，HTTP アダプタで assertion を作成するため次の xsd を使用します。

- 格納先

  <サービスプラットフォームのインストールディレクトリ>¥CSC¥custom-adapter¥HTTP¥schema

- 使用する定義

  adphttp_body_form-data_jwt_assertion1.xsd

応答電文（ヘッダ）

- 格納先

  <サービスプラットフォームのインストールディレクトリ>¥CSC¥custom-adapter¥HTTP¥schema

- 使用する定義

  adphttp_header_response1.xsd

応答電文（ボディ）

応答電文ボディの解析をしないため，空の HTTP メッセージボディを表す電文フォーマットを使用します。

- 格納先

  <サービスプラットフォームのインストールディレクトリ>¥CSC¥custom-adapter¥HTTP¥schema

- 使用する定義

```
adphttp_body_empty.fdx
```

●定義ファイルの作成

HTTP アダプタ実行時に OAuth 2.0 の認証フローを使用するため，HTTP アダプタ定義ファイルの adphttp.oauth.authentication プロパティの値に「true」を設定します。

認証情報をフォームデータとして送信するため，HTTP アダプタ実行環境プロパティファイルの adphttp.request.part.message.binding プロパティの値に「form-data」を設定します。

●データ変換の定義

認証処理を行う HTTP アダプタでは，データ変換を使用して，要求電文に次に示す値を設定する必要があります。

- 要求電文（ヘッダ）

キー名称	設定値
/http-header-request/method 要素	POST（固定文字列）
/http-header-request/uri-scheme-authority 要素	https://login.salesforce.com（固定文字列）
/http-header-request/uri-path 要素	/services/oauth2/token（固定文字列）

- 要求電文（ボディ）

キー名称	設定値
/http-body-form-data/grant_type 要素	urn:ietf:params:oauth:grant-type:jwt-bearer（固定文字列）
/http-body-form-data/auth-info/claim/iss 要素	Salesforce の接続アプリケーションから取得するコンシューマ鍵
/http-body-form-data/auth-info/claim/aud 要素	Salesforce のアカウント名（メールアドレス形式）
/http-body-form-data/auth-info/claim/sub 要素	https://login.salesforce.com（固定文字列）
/http-body-form-data/auth-info/claim/exp 要素	HTTP アダプタが作成する JWT の有効期限 JWT の有効期限を 5 分以内で設定する（1970-01-01T0:0:0Z からの秒数）。
/http-body-form-data/auth-info/header/alg 要素	JWT クレームセットの暗号化に使用するアルゴリズム
/http-body-form-data/auth-info/secretkey 要素	csakeyadd コマンドで登録した秘密鍵の鍵 ID
/http-body-form-data/auth-info/signature 要素	Java 暗号化アーキテクチャ標準アルゴリズム名（Signature アルゴリズム） 電子署名に使用する Signature オブジェクト作成に使用するアルゴリズムを設定する。
/http-body-form-data/auth-info/keytype 要素	Java 暗号化アーキテクチャ標準アルゴリズム名（KeyFactory アルゴリズム） 秘密鍵を変換する KeyFactory オブジェクト作成に使用するアルゴリズムを設定する。 秘密鍵が対応するアルゴリズムを設定する。

(d) Web サービスを呼び出す HTTP アダプタの定義例

ここでは Web サービスを呼び出す上で，定義が必要となる作業について説明します。

HTTP アダプタの基本的な定義の流れについては，マニュアル「サービスプラットフォーム 開発ガイド 受付・アダプタ定義編」の「3.3.13　HTTP アダプタを定義する」を参照してください。

- 電文フォーマットの設定
- データ変換の定義

それぞれの定義について説明します。

●電文フォーマットの作成

Web サービスを呼び出す HTTP アダプタでは，要求電文，応答電文の電文フォーマットに次の XML スキーマを定義してください。

要求電文（ヘッダ）

- 格納先

  <サービスプラットフォームのインストールディレクトリ>¥CSC¥custom-adapter¥HTTP¥schema

- 使用する定義

  adphttp_header_request1.xsd

要求電文（ボディ）

Salesforce の API を呼び出す場合は，要求電文ボディからデータを設定する必要はないため，空の HTTP メッセージボディを表す電文フォーマットを使用します。

- 格納先

  <サービスプラットフォームのインストールディレクトリ>¥CSC¥custom-adapter¥HTTP¥schema

- 使用する定義

  adphttp_body_empty.fdx

応答電文（ヘッダ）

- 格納先

  <サービスプラットフォームのインストールディレクトリ>¥CSC¥custom-adapter¥HTTP¥schema

- 使用する定義

  adphttp_header_response1.xsd

応答電文（ボディ）

応答電文ボディの解析をしないため，空の HTTP メッセージボディを表す電文フォーマットを使用します。

- 格納先

  <サービスプラットフォームのインストールディレクトリ>¥CSC¥custom-adapter¥HTTP¥schema

- 使用する定義

adphttp_body_empty.fdx

●データ変換の定義

Web サービスを呼び出す HTTP アダプタでは，データ変換を使用して，要求電文に呼び出す API に合わせた値を設定する必要があります。

- 要求電文（ヘッダ）

キー名称	設定値※
/http-header-request/method 要素	GET
/http-header-request/uri-scheme-authority 要素	https://API コールの送信先となる組織のインスタンス.salesforce.com
/http-header-request/uri-path 要素	/services/data/v43.0/
/http-header-request/http-header-Authorization 要素	Bearer△アクセストークン
/http-header-request/http-header-Authorization/@type 属性	raw

（凡例）△：半角スペース

注※

設定値は，リソースのリストを取得する場合の設定例を示しています。

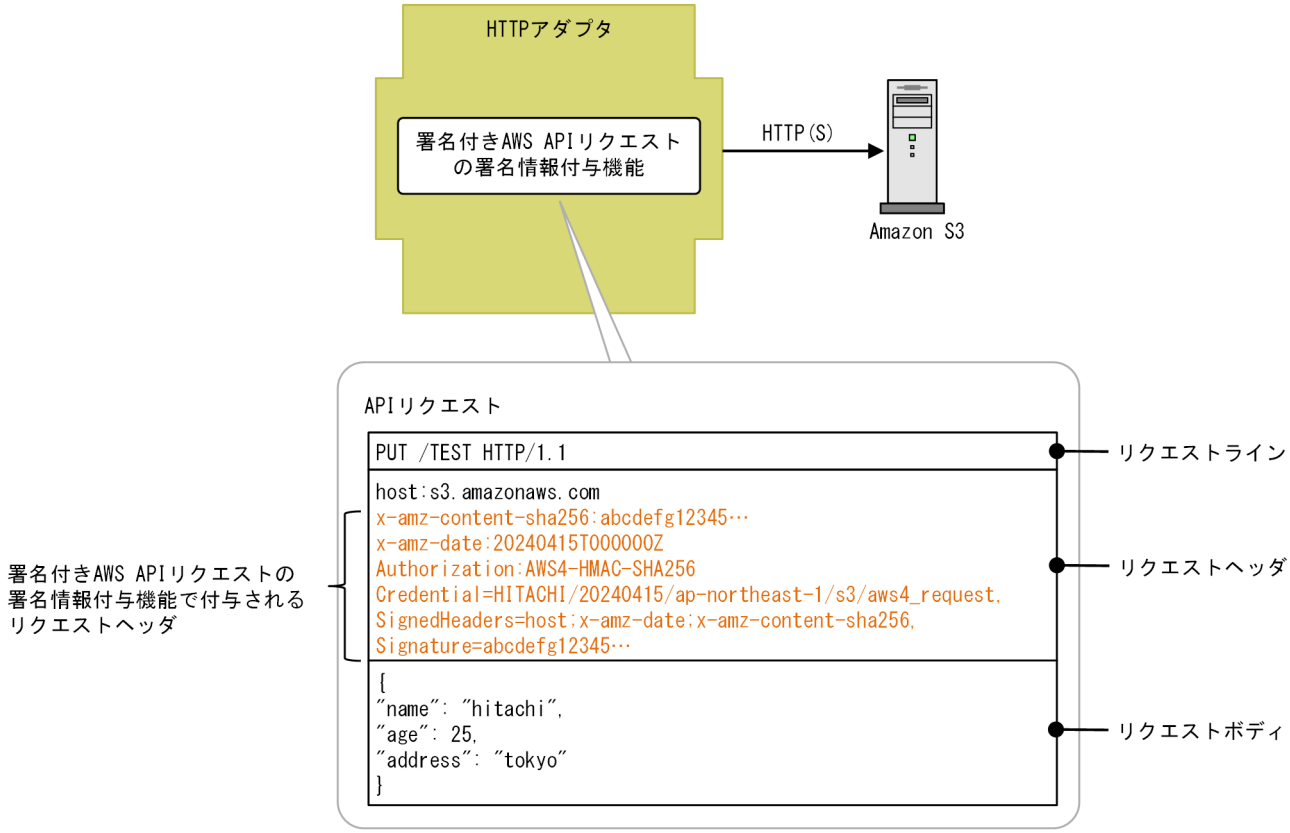
- 要求電文（ボディ）

要求電文ボディを設定しない場合でも，変数の初期化を行う必要があるため，変換先変数に要求電文ボディの変数を指定し，マッピングを行わないで，データ変換を行う必要があります。

2.16.18　署名付き AWS API リクエストの署名情報付与

HTTP アダプタのリクエスト処理時に，署名付き AWS API リクエストに必要な署名情報をリクエストヘッダに付与します。

図 2-186　署名付き AWS API リクエストの署名情報付与の概要



HTTP アダプタは，生成した署名情報をリクエストヘッダに付与します。認証情報として，アクセスキー ID，およびシークレットアクセスキーを用いて，署名付き AWS API リクエストの署名を生成します。

署名付き AWS API リクエストの署名情報付与を設定できるサポート範囲を，次に示します。

- AWS サービスの連携先：Amazon S3

  Amazon S3 のマルチパートアップロード API を使用したオブジェクトのアップロード，またはコピーが使用できます。1 回のリクエストで Amazon S3 にアップロード，またはダウンロードできるオブジェクトサイズは，JAX-RS 2.1 の制限に従います。

  ただし，Amazon S3 のチャンクアップロードを使用したオブジェクトのアップロードは使用できません。Amazon S3 の詳細については，Amazon S3 のマニュアルを参照してください。
- 署名バージョン：Signature Version 4（署名バージョン 4）

(1)　リクエストボディの送信データ種別

署名付き AWS API リクエストの署名情報付与で使用できる，リクエストボディの送信データ種別を次に示します。これ以外のデータ種別で送信した場合は，エラーを出力し，リクエストに失敗します。

表 2-123　リクエストボディの送信データ種別

送信データ種別	説明	リクエストボディの例
データなし	リクエストボディを指定しません。	－

送信データ種別		説明	リクエストボディの例
データあり	ボディ要求電文（パススルーモード）	ボディ要求電文で指定されたデータをリクエストボディとして設定します。	<body> <element>AAA</element> </body>
	ファイル	ファイルの内容をリクエストボディとして設定します。	hitachi.txt

－：該当しない。

(2)　署名付き AWS API リクエストの署名情報付与の設定

署名付き AWS API リクエストの署名情報付与は，HTTP アダプタのヘッダ要求電文，または HTTP アダプタ実行環境プロパティファイルで設定します。HTTP アダプタのヘッダ要求電文と HTTP アダプタ実行環境プロパティファイルの両方が設定されている場合は，ヘッダ要求電文の設定が優先されます。

ヘッダ要求電文の詳細については，マニュアル「サービスプラットフォーム 開発ガイド 受付・アダプタ定義編」の「3.3.13(2)　電文フォーマットの作成」を，HTTP アダプタ実行環境プロパティファイルの詳細については，マニュアル「サービスプラットフォーム リファレンス」の「6.14.2　HTTP アダプタ実行環境プロパティファイル」を参照してください。

署名付き AWS API リクエストの署名情報付与で設定する項目を次に示します。

表 2-124　署名付き AWS API リクエストの署名情報付与の設定項目

ヘッダ要求電文	HTTP アダプタ実行環境プロパティファイル	説明
/http-header-request/http-header-signed-aws-api-request/@type 属性	adphttp.request.header.signed.aws.api.request.type	署名付き AWS API リクエストの認証方法を指定します。
/http-header-request/http-header-signed-aws-api-request 要素	adphttp.request.header.signed.aws.api.request	署名付き AWS API リクエストで使用するアクセスキーペア ID を指定します。
/http-header-request/http-header-signed-aws-api-request/@version 属性	adphttp.request.header.signed.aws.api.request.version	署名バージョンを指定します。
/http-header-request/http-header-signed-aws-api-request/@service 属性	adphttp.request.header.signed.aws.api.request.service	接続先 AWS サービスを指定します。
/http-header-request/http-header-signed-aws-api-request/@region 属性	adphttp.request.header.signed.aws.api.request.region	接続先 AWS リージョンを指定します。

(3) HTTP リクエストに付与されるリクエストヘッダ情報

署名付き AWS API リクエストの署名情報付与が有効な場合，HTTP リクエスト処理時に，リクエストヘッダに次の署名付き AWS API リクエストの署名情報が付与されます。

HTTP リクエストに付与されるリクエストヘッダ情報を次に示します。

表 2-125　HTTP リクエストに付与されるリクエストヘッダ情報

ヘッダ名※1	付与する値
x-amz-date	HTTP リクエスト発行時の日時情報（ISO 8601 形式の UTC 時刻）です。 例：20240415T000000Z
x-amz-content-sha256	HTTP リクエストボディのハッシュ値です。 HTTP ボディが空の場合は，空の文字列のハッシュを指定します。 Amazon S3 に署名付き AWS API リクエストを発行する場合に付与します。
Authorization※2	認証情報です。次の形式で，値を設定します。△は空白文字を示します。 AWS4-HMAC-SHA256△Credential=<Credential の値>,△ SignedHeaders=<SignedHeaders の値>,△Signature=<Signature の値>

注※1

x-amz-date および x-amz-content-sha256 を，ユーザがヘッダ要求電文（/http-header-request/http-header の直下），または HTTP アダプタ実行環境プロパティファイル（adphttp.request.header.userdef.<追番>）に設定した場合，HTTP リクエスト処理時にユーザが設定した値が優先されます。

Authorization は，ユーザが設定した値は無視されます。

上記のヘッダ以外の AWS API リクエストで必要なリクエストヘッダを設定する場合は，ヘッダ要求電文（/http-header-request/http-header の直下），または HTTP アダプタ実行環境プロパティファイル（adphttp.request.header.userdef.<追番>）に個別に設定項目を追加してください。

注※2

Authorization ヘッダの構成要素を次に示します。

表 2-126　Authorization ヘッダの構成要素

構成要素	付与する値
Credential	次の形式で構成されたスコープ情報です。 <your-access-key-id>/<date>/<aws-region>/<aws-service>/aws4_request • <your-access-key-id>：アクセスキー ID • <date>：HTTP リクエスト発行時の日時情報（YYYYMMDD 形式） • <aws-region>：接続先 AWS リージョン名 例：ap-northeast-1 • <aws-service>：接続先 AWS サービス名 例：s3（Amazon S3 にリクエストを送信する場合）
SignedHeaders	署名の計算に使用したリクエストヘッダをセミコロンで区切ったリストです。 例：host;x-amz-date
Signature	64 個の小文字の 16 進数文字で表現される 256 ビットの署名です。

構成要素	付与する値
Signature	例：fe5f80f77d5fa3beca038a248ff027d0445342fe2855ddc963176630326f1024

(4) 認証情報の管理

署名付き AWS API リクエストの認証で使用する認証情報（アクセスキー ID，シークレットアクセスキー）は，HTTP アダプタがサポートするアカウント情報管理コマンドを使用して管理します。

アカウント情報管理コマンドを使用して設定したアクセスキー ID やシークレットアクセスキーの認証情報は，アカウント情報として HTTP アダプタアカウント定義ファイルに出力されます。

アカウント情報管理コマンドの詳細については，マニュアル「サービスプラットフォーム リファレンス」の「5.4 サービスプラットフォームの運用環境および実行環境で使用するコマンド」を参照してください。

HTTP アダプタの認証情報管理で使用するコマンドを，次に示します。

表 2-127 認証情報管理で使用するコマンド

コマンド	説明
csahttpuseradd	署名付き AWS API リクエストの署名生成を行うときに必要となるアクセスキー ID，シークレットアクセスキーを登録するためのコマンドです。
csahttpuserdel	HTTP アダプタアカウント定義ファイルに登録されているアクセスキー ID とシークレットアクセスキーを削除するコマンドです。
csahttpuserls	HTTP アダプタアカウント定義ファイルに登録されているアクセスキーペア ID を表示するコマンドです。

第三者によるファイルの閲覧を抑止するため，管理者だけが HTTP アダプタアカウント定義ファイルを閲覧できるようにアクセス権限を設定してください。

HTTP アダプタアカウント定義ファイルに，ヘッダ要求電文，または HTTP アダプタ実行環境プロパティファイルで指定されたアクセスキーペア ID が存在しないなど，送信する認証情報が取得できない場合はエラーとなります。

HTTP アダプタアカウント定義ファイルは，HTTP アダプタ実行環境プロパティファイルの「adphttp.config.account-inf-filepath」に設定します。

(5) Developer's Kit for Java の設定

AWS サービスと連携する場合，Java 環境にドメインネームシステム（DNS）の名前解決結果のキャッシュ時間を設定する必要があります。

セキュリティマネージャが有効な場合，Developer's Kit for Java に対してドメインネームシステム（DNS）の名前解決結果のキャッシュ時間（networkaddress.cache.ttl）を次の個所に設定します。

<Developer's Kit for Java インストールディレクトリ>/conf/security/java.security

(6) 署名付き AWS API リクエストの定義例

ここでは，HTTP アダプタで署名付き AWS API リクエストを使用した，Amazon S3 と連携する場合の定義例について説明します。

(a) 定義方法の前提条件

署名付き AWS API リクエストの定義方法は次の前提条件を基に説明しています。

- AWS で発行した，アクセスキー ID，およびシークレットアクセスキーを csahttpuseradd コマンドで登録済みとします。
- HTTP アダプタで共通フォルダに保存されたファイルを取得し，署名付き AWS API リクエスト（PutObject アクション）を発行して，S3 のバケットにアップロードする場合を想定しています。

(b) 定義の流れ

署名付き AWS API リクエストの署名情報付与を使用する上で，定義が必要となる次の作業について説明します。

- アクセスキー ID，およびシークレットアクセスキーの登録
- 電文フォーマットの定義
- 定義ファイルの定義
- データ変換の定義

それぞれの定義について説明します。

HTTP アダプタの基本的な定義の流れについては，マニュアル「サービスプラットフォーム 開発ガイド 受付・アダプタ定義編」の「3.3.13　HTTP アダプタを定義する」を参照してください。

●アクセスキー ID，およびシークレットアクセスキーの登録

署名付き AWS API リクエストの署名情報付与を使用する場合，アクセスキー ID，およびシークレットアクセスキーを，csahttpuseradd コマンドを実行して登録します。

●電文フォーマットの定義

次の格納先にある要求電文，および応答電文の電文フォーマットに XML スキーマを定義してください。

電文フォーマットの格納先

<サービスプラットフォームのインストールディレクトリ>¥CSC¥custom-adapter¥HTTP¥schema

XML スキーマを定義する電文フォーマットを次に示します。

要求電文（ヘッダ）

adphttp_header_request1.xsd

要求電文（ボディ）

adphttp_body_empty.fdx

HTTP アダプタでファイルデータを送信する場合，アダプタに渡されたボディ要求電文の値は無視されるため，空の HTTP メッセージボディを表す電文フォーマットを使用します。

応答電文（ヘッダ）

adphttp_header_response1.xsd

応答電文（ボディ）

adphttp_body_empty.fdx

署名付き AWS API リクエスト（PutObject アクション）の正常応答は空のレスポンスボディが返却されるため，空の HTTP メッセージボディを表す電文フォーマットを使用します。

●定義ファイルの定義

署名付き AWS API リクエストの署名情報付与を HTTP アダプタ実行環境プロパティファイルで設定する場合は，次の項目を設定します。

表 2-128　HTTP アダプタ実行環境プロパティファイルの設定項目

カテゴリ	項目名	設定値
リクエストライン	adphttp.request.method	PUT
	adphttp.request.uri-scheme-authority	https://<バケット名>.s3.<リージョン名>.amazonaws.com
	adphttp.request.uri-path	アップロード先のファイルパス 例：/sample.txt
署名付き AWS API リクエストの署名情報付与	adphttp.request.header.signed.aws.api.request.type	header
	adphttp.request.header.signed.aws.api.request	署名付き AWS API リクエストで使用するアクセスキーペア ID 例：TESTEXAMPLE
	adphttp.request.header.signed.aws.api.request.version	aws4_request
	adphttp.request.header.signed.aws.api.request.service	s3
	adphttp.request.header.signed.aws.api.request.region	接続先 AWS リージョン 例：ap-northeast-1
Amazon S3 に送信するファイル	adphttp.config.account-inf-filepath	<HTTP アダプタアカウント定義ファイルのパス>
	adphttp.request.part.file.1.input-folder-name	共通フォルダ定義名 例：test
	adphttp.request.part.file.1.local-file-name	Amazon S3 に送信するファイル名 例：sample.txt

●データ変換の定義

署名付き AWS API リクエストの署名情報付与をデータ変換を使用して設定する場合は，要求電文（ヘッダ）に次の項目を設定します。

- 要求電文（ヘッダ）

カテゴリ	項目名	設定値
リクエストライン	/http-header-request/method	PUT
	/http-header-request/uri-scheme-authority	https://<バケット名>.s3.<リージョン名>.amazonaws.com
	/http-header-request/uri-path	アップロード先のファイルパス 例：/sample.txt
署名付きAWS APIリクエストの署名情報付与	/http-header-request/http-header-signed-aws-api-request/@type	header
	/http-header-request/http-header-signed-aws-api-request	署名付きAWS APIリクエストで使用するアクセスキーペアID
	/http-header-request/http-header-signed-aws-api-request/@version	aws4_request
	/http-header-request/http-header-signed-aws-api-request/@service	s3
	/http-header-request/http-header-signed-aws-api-request/@region	接続先AWSリージョン
Amazon S3に送信するファイル	/http-header-request/http-part/files/file/input-folder-name/@common	true
	/http-header-request/http-part/files/file/input-folder-name	共通フォルダ定義名 例：test
	/http-header-request/http-part/files/file/local-file-name	Amazon S3に送信するファイル名 例：sample.txt

- 要求電文（ボディ）

  要求電文ボディを設定しない場合でも，変数の初期化を行う必要があるため，変換先変数に要求電文ボディの変数を指定し，マッピングを行わないで，データ変換を行う必要があります。

(7) HTTPアダプタのリダイレクト設定時の注意事項

HTTPアダプタのリダイレクトを有効にして，署名付きAWS APIリクエストを発行する場合，リダイレクト後の2回目のリクエストの署名情報は再計算されないため，署名認証はエラーとなります。

リダイレクト後の2回目のリクエストに，再計算した署名情報を付与したい場合は，1回目のリクエストで301，302，303，または307のレスポンスが返ってきたときに，再度別のサービス呼出アクティビティでHTTPアダプタを呼び出すようにビジネスプロセスで定義する必要があります。

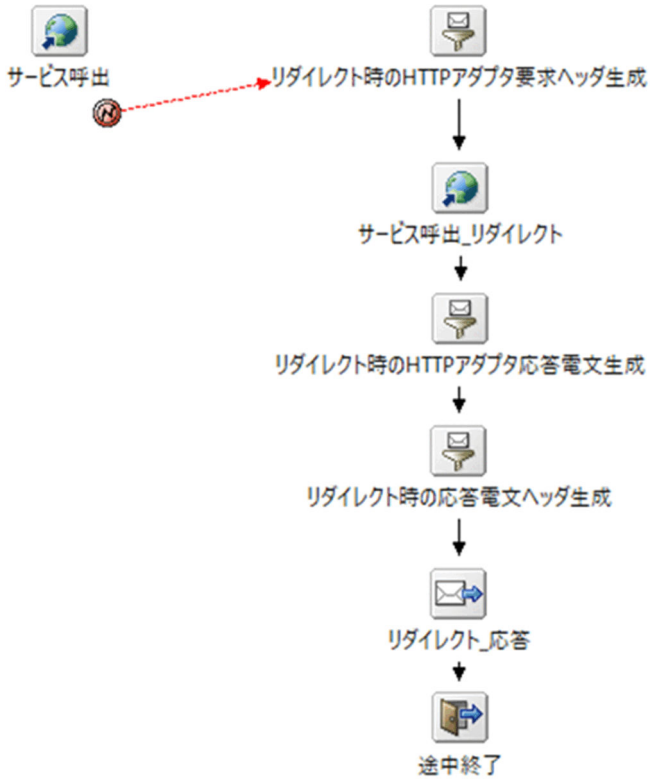
リダイレクト後の2回目のリクエストに，再計算した署名情報を付与する場合の，ビジネスプロセスおよびHTTPアダプタの設定例について説明します。

(a) ビジネスプロセスの設定

1 回目のリクエストで 301，302，303，または 307 のレスポンスが返ってきた場合，レスポンスに含まれる Location ヘッダの値をリダイレクト後の 2 回目のリクエストの接続先としてヘッダ要求電文に設定し，再度別のサービス呼出アクティビティで HTTP アダプタを呼び出すように定義します。

リダイレクトに対応した署名付き AWS API リクエストのビジネスプロセスの定義例を次に示します。

図 2-187　リダイレクトに対応した署名付き AWS API リクエストのビジネスプロセスの定義例



(b) HTTP アダプタの設定

HTTP アダプタ定義ファイルの「adphttp.response-code.fault」プロパティに，301，302，303，または 307 のレスポンスを，フォルト応答として返すように設定します。

リダイレクトに対応した署名付き AWS API リクエストの HTTP アダプタ定義ファイルの定義例を次に示します。

表 2-129　リダイレクトに対応した署名付き AWS API リクエストの HTTP アダプタ定義ファイルの定義例

プロパティ	指定内容	設定値
adphttp.response-code.fault	フォルト処理するステータスコードを正規表現で指定します。	30[1237]

2.17　外部コマンドの実行によるシステムとの接続

コマンドアダプタを利用することで，ビジネスプロセスから次に示す種類のファイルを実行できます。

- EXE ファイル（Windows）
- BAT ファイル（Windows）
- sh ファイル（Linux）

この機能によって，ほかのアダプタでサポートしている SOAP や REST のような汎用的な接続インターフェースを持たない製品や，作成済みのユーザスクリプト資産をビジネスプロセスの連携に組み込むことができます。

2.17.1　前提条件

コマンドアダプタを利用するための前提条件を次に示します。

- 指定できる外部コマンドは，サービスプラットフォームのインストールマシンに存在するコマンドだけです。
- 外部コマンドプロセスを実行できるユーザは，サービスプラットフォームの J2EE サーバを起動したユーザです。
- GUI アプリケーションは起動しないでください。
- 外部コマンドからバックグラウンドプロセスは起動しないでください。
- 1 回の呼び出しで実行できる外部コマンドは 1 つまでです。
  複数の外部コマンドによる AND または OR の実行制御も使用できません。
- 外部コマンドに対する入出力パイプは使用できません。
- 外部コマンドに対する標準入出力・標準エラー出力のファイルリダイレクトは使用できません。

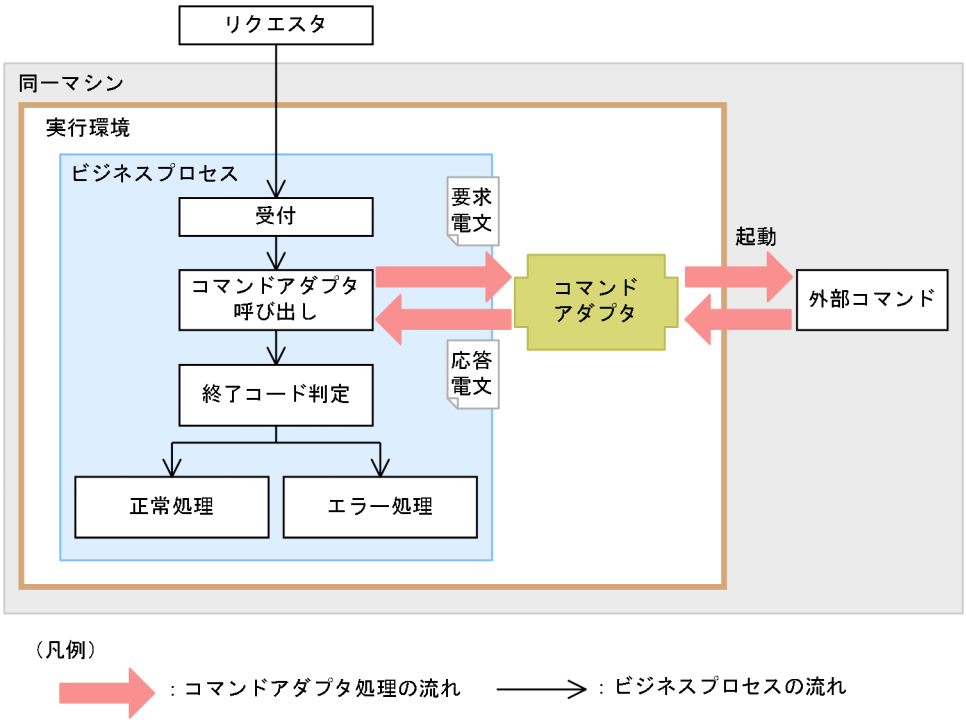
注意事項

コマンドアダプタの利用は，Service Coordinator 09-71-01 以降が前提です。

2.17.2　コマンドアダプタを使用した外部コマンドの実行

コマンドアダプタを使用した外部コマンド実行の流れを次の図に示します。

図 2-188　コマンドアダプタを使用した外部コマンド実行の流れ



（凡例）
：コマンドアダプタ処理の流れ　：ビジネスプロセスの流れ

ビジネスプロセスからコマンドアダプタへ，要求電文を利用して次に示す外部コマンド情報を渡します。

- 外部コマンドのファイルパス
- コマンドライン引数

コマンドアダプタでは要求電文を解析し，同一マシン上に存在する外部コマンドを起動します。

外部コマンドプロセス実行後，コマンドアダプタからビジネスプロセスへ，応答電文を利用して次に示す終了情報が渡されます。

- 外部コマンドプロセスの終了コード
- 外部コマンドプロセスの標準出力への出力内容

ビジネスプロセスでは応答電文の終了コードを解析し，処理を分岐することができます。

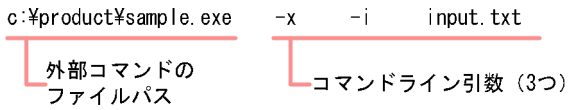
2.17.3　要求電文を利用した外部コマンド情報の指定

ビジネスプロセスからコマンドアダプタを呼び出すときの要求電文には，次に示す情報を指定します。

- 外部コマンドのファイルパス
- コマンドライン引数

外部コマンドのファイルパスとコマンドライン引数の関係を次に示します。

（例）コマンドライン文字列が「c:¥product¥sample.exe -x -i input.txt」の場合



注意事項

標準入力のファイルリダイレクトは使用できません。

また，上記例のコマンドライン文字列を生成する要求電文の作成例を次に示します。

```
<?xml version="1.0" encoding="UTF-8"?>
<request xmlns="http://www.hitachi.co.jp/soft/xml/cosminexus/csc/adapter/cmd/request"xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
        <command-line>
                <command>c:¥product¥sample.exe</command>
                <argument1>-x</argument1>
                <argument2>-i</argument2>
                <argument3>input.txt</argument3>
        </command-line>
</request>
```

コマンドアダプタの要求電文フォーマットの作成については，マニュアル「サービスプラットフォーム 開発ガイド 受付・アダプタ定義編」の「3.3.14(1)　電文フォーマットの作成」を参照してください。

(1)　外部コマンドのファイルパス

同一マシン上に存在する外部コマンドのファイルパスを指定します。

指定は必須です。絶対パスで指定することを推奨します。

注意事項

- 絶対パスで指定しない（ファイル名だけを指定，または相対パスで指定する）場合，実行するコマンドは OS のコマンドサーチ，および Java の API 仕様に依存します。そのため，動作を保証しません。
- ファイルパス中の環境変数の展開はしません。

要求電文で指定できる外部コマンドの条件を次の表に示します。

表 2-130　要求電文で指定できる外部コマンドの条件

OS	外部コマンドの条件
Windows	拡張子が次のどれかであること。 • bat • exe • com

OS	外部コマンドの条件
Windows	J2EE サーバユーザの実行権限があること。
Linux	J2EE サーバユーザの実行権限があること。

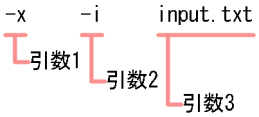
(2)　コマンドライン引数

外部コマンドに与えるコマンドライン引数を指定します。

指定は任意です。最大 100 個の引数を指定できます。

指定する際には，コマンドライン文字列の引数部分を，引数ごとに分割して指定する必要があります。

コマンドライン引数の指定例を次に示します。



2.17.4　応答電文で取得できる外部コマンドプロセスの終了情報

コマンドアダプタの応答電文から，次に示す情報を取得できます。

- 外部コマンドプロセスの終了コード
- 外部コマンドプロセスの標準出力への出力内容

注意事項

- 標準エラー出力への出力内容は，応答電文では取得できません。
- 標準エラー出力への出力内容は，製品が規定するログファイルに出力されます。詳細はマニュアル「サービスプラットフォーム システム構築・運用ガイド」の「7.7.17(6)　外部コマンドプロセスの標準エラー出力ログ」を参照してください。

コマンドアダプタの応答電文フォーマットの作成については，マニュアル「サービスプラットフォーム 開発ガイド 受付・アダプタ定義編」の「3.3.14(1)　電文フォーマットの作成」を参照してください。

2.17.5　外部コマンドプロセスに関する動作オプションの設定

(1)　動作オプションの設定方法

コマンドアダプタでは，次の方法で動作オプションを指定できます。

- 定義ファイルで指定
- 要求電文で指定（一部オプションに限定）

オプション設定値の反映タイミングと対応する設定方法を次の表に示します。なお，定義ファイルと要求電文で同じオプションを指定した場合，要求電文の指定が優先されます。

表 2-131　外部コマンドプロセスに関する動作オプションの設定方法

反映タイミング	動作オプションの設定方法
コマンドアダプタ起動時	コマンドアダプタ実行環境プロパティファイルで設定します。 コマンドアダプタ実行環境プロパティファイルで設定できる項目の詳細は，マニュアル「サービスプラットフォーム リファレンス」の「6.15.3　コマンドアダプタ実行環境プロパティファイル」を参照してください。
リクエスト時	要求電文で設定します。 要求電文フォーマットの作成方法については，マニュアル「サービスプラットフォーム 開発ガイド 受付・アダプタ定義編」の「3.3.14(1)　電文フォーマットの作成」を参照してください。

(2)　動作オプションに設定できる項目

コマンドアダプタでは，外部コマンドプロセスに関する次の項目を動作オプションに設定できます。

- 外部コマンドプロセスの環境変数
- 外部コマンドプロセスの作業ディレクトリ
- タイムアウト時間
- タイムアウト発生時の外部コマンドプロセス強制終了可否
- 応答電文で取得する標準出力の上限サイズ

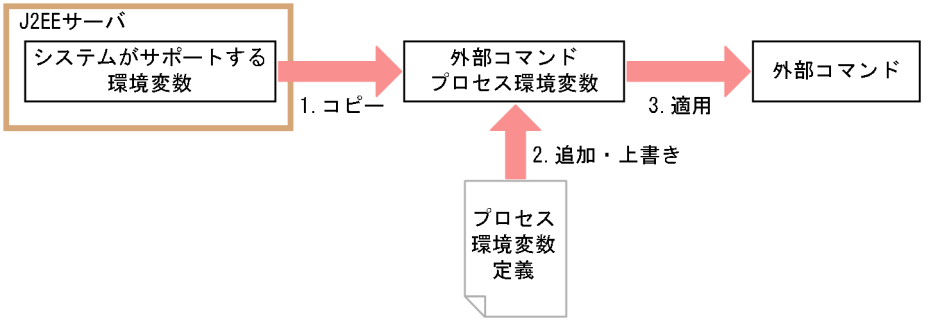
(a)　外部コマンドプロセスの環境変数

外部コマンドプロセスだけで有効となる環境変数を定義できます。

外部コマンドプロセスの環境変数は，コマンドアダプタ内で生成されます。

生成の流れを次の図に示します。

図 2-189　外部コマンドプロセスの環境変数の生成の流れ



図内の処理について説明します。なお，次の番号は，図中の番号と対応しています。

1. J2EE サーバが持つシステム環境変数をコピーします。
   システム環境変数については，System.getEnv()メソッドの仕様を参照してください。
2. ユーザが作成する「プロセス環境変数定義ファイル」が存在する場合は，1.でコピーしたシステム環境変数の内容を基に，プロセス環境変数を追加または上書きをします。
   ファイルが存在しない場合は，1.でコピーしたシステム環境変数をそのまま使用します。
   プロセス環境変数定義ファイルパスは，要求電文，またはコマンドアダプタ実行環境プロパティファイルで指定します。プロセス環境変数定義ファイルの詳細は，マニュアル「サービスプラットフォームリファレンス」の「6.15.2　プロセス環境変数定義ファイル」を参照してください。
3. 外部コマンドプロセス起動時に環境変数（システム環境変数，およびプロセス環境変数）を適用します。

外部コマンドプロセスの環境変数は，コマンドアダプタのリクエストのたびに生成されます。したがって，ほかのリクエストで起動する外部コマンドプロセスや，J2EE サーバ，OS の環境変数に影響を与えません。

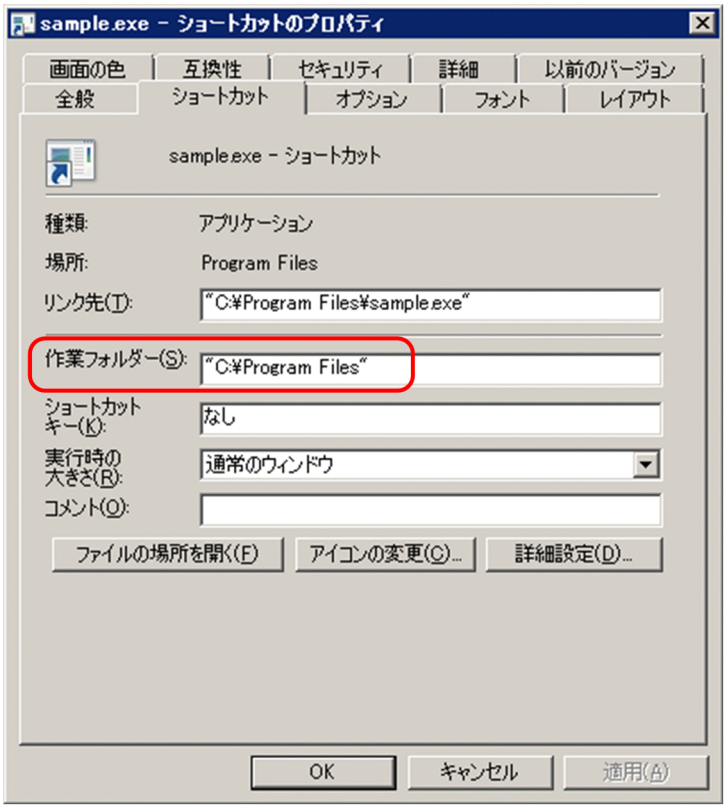
(b)　外部コマンドプロセスの作業ディレクトリ

作業ディレクトリとは，次の方法などで取得または設定できるものを指します。コマンドアダプタでは，この作業ディレクトリを変更できます。

- Windows の場合
  - コマンドプロンプトで cd コマンドを実行してカレントディレクトリを確認する



  - 外部コマンドのショートカットのプロパティで［作業フォルダー］に指定する



- Linux の場合

  シェルで pwd コマンドを実行してカレントディレクトリを確認する



(c) タイムアウト時間

外部コマンドプロセスの実行が何らかの原因で終わらない場合，ビジネスプロセスにシステム例外を通知するためのタイムアウト時間を設定できます。

タイムアウト時間を経過しても外部コマンドプロセスが終了しなかった場合は，ビジネスプロセスにシステム例外が通知されます。

(d) タイムアウト発生時の外部コマンドプロセス強制終了可否

タイムアウトが発生した際に，その時点で外部コマンドプロセスの強制終了を試みるかどうかを設定できます。

強制終了しない設定の場合，コマンドアダプタや HCSC サーバの状態に関係なく，起動した外部コマンドプロセスは処理を継続します。なお，タイムアウト発生後の外部コマンドプロセスの状態をサービスプラットフォームから調べる方法はありません。

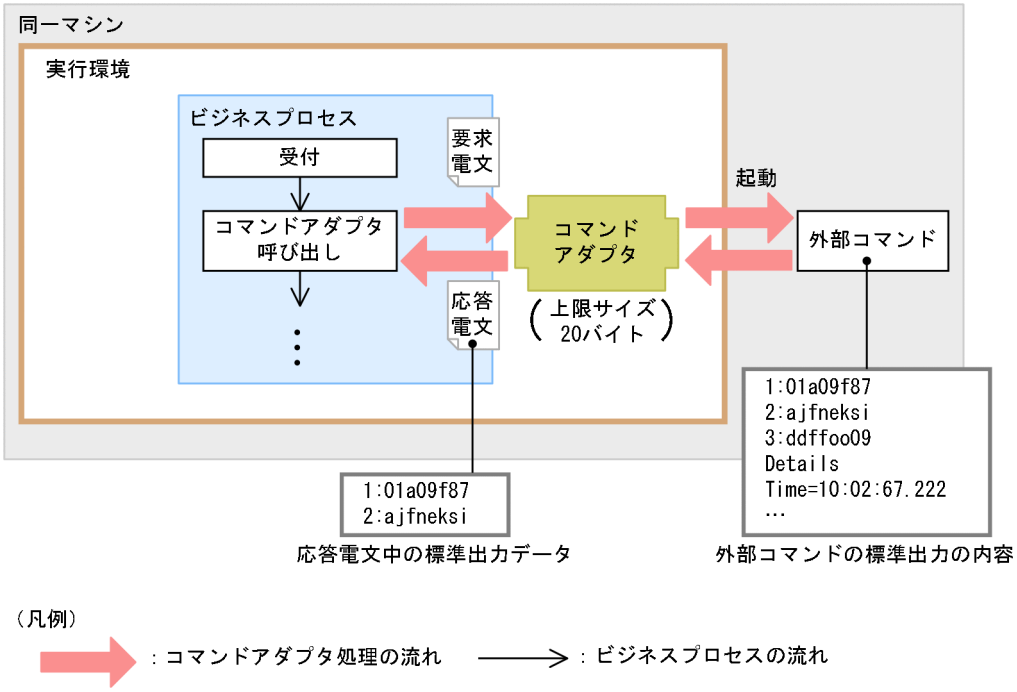
(e)　応答電文で取得する標準出力の上限サイズ

コマンドアダプタの応答電文で取得する標準出力のサイズ上限値を設定できます。

コマンドアダプタは，標準出力の出力データを，出力された分だけ内部的に保持しています。標準出力のサイズ上限値を設定すると，そのサイズまでしか保持しません。そのため，次の場合にサイズを設定することで，コマンドアダプタが保持するデータのメモリ使用量を節約できます。

- 出力された標準出力の内容をビジネスプロセスで使用しない場合
- 出力された標準出力の内容を規定の長さまでしか使用しない場合

図 2-190　標準出力のサイズ制限のイメージ



注意事項

外部コマンドプロセスの標準出力のサイズが設定値を上回った場合，メッセージログに警告メッセージが出力されます。

2.17.6　実行許可コマンド定義ファイルによる実行制限

コマンドアダプタ利用時のセキュリティ対策として，実行する外部コマンド，およびコマンドライン引数を制限します。制限には，ユーザが作成する実行許可コマンド定義ファイルを使用します。実行許可コマンド定義ファイルは，正規表現を記述したテキストファイルです。定義例を次に示します。

```
C:¥¥Program Files¥¥SampleProduct¥¥bin¥¥sample(?:Start|Stop)¥.bat
C:¥¥Program Files¥¥SampleProduct¥¥bin¥¥sampleList¥.bat(?:△-l)?
C:¥¥Program Files¥¥SampleProduct¥¥bin¥¥sampleExport¥.bat△-o△¥w{8}¥.txt
```

（凡例）

△：半角空白

定義例のとおりに実行許可コマンド定義ファイルを作成した場合に，実行可能な外部コマンドと指定可能なコマンドライン引数を次の表に示します。

表 2-132　実行可能な外部コマンドとコマンドライン引数

外部コマンド	コマンドライン引数
C:¥Program Files¥SampleProduct¥bin¥sampleStart.bat	-
C:¥Program Files¥SampleProduct¥bin¥sampleStop.bat	-
C:¥Program Files¥SampleProduct¥bin¥sampleList.bat	-
C:¥Program Files¥SampleProduct¥bin¥sampleList.bat	-l
C:¥Program Files¥SampleProduct¥bin¥sampleExport.bat	-o ＜8文字の英数字＞.txt

（凡例）

－：指定しません。

2.18　SFTP サーバとの接続

SFTP アダプタを利用することで，サービスプラットフォームと SFTP サーバとの間で SFTP プロトコルによるファイル転送ができます。

2.18.1　前提条件

SFTP アダプタを利用するための前提条件を次に示します。

- SFTP アダプタの連携先の SFTP サーバは，OpenSSH を利用した SSH サーバである必要があります。
- SFTP アダプタは，SSH プロトコルのバージョンが「2」，かつ SFTP プロトコルのバージョンが「3」の SFTP サーバをサポートしています。

2.18.2　SFTP アダプタを使用したシステム

SFTP アダプタは，SFTP サーバとサービスプラットフォームとの間でファイルを送受信するためのサービスアダプタです。

SFTP アダプタでは，ファイルに対する操作の単位をオペレーションと呼びます。

SFTP アダプタでサポートするオペレーションを次に示します。

- PUT オペレーション
  SFTP アダプタから SFTP サーバへファイルデータを送信します。
- GET オペレーション
  SFTP サーバから送信されたファイルデータを SFTP アダプタで受信します。
- GETINFO オペレーション
  SFTP サーバ側に存在するファイルの情報の一覧またはファイルの名前の一覧を取得します。

SFTP アダプタを使用したファイル送受信の例として，次のようなことができます。

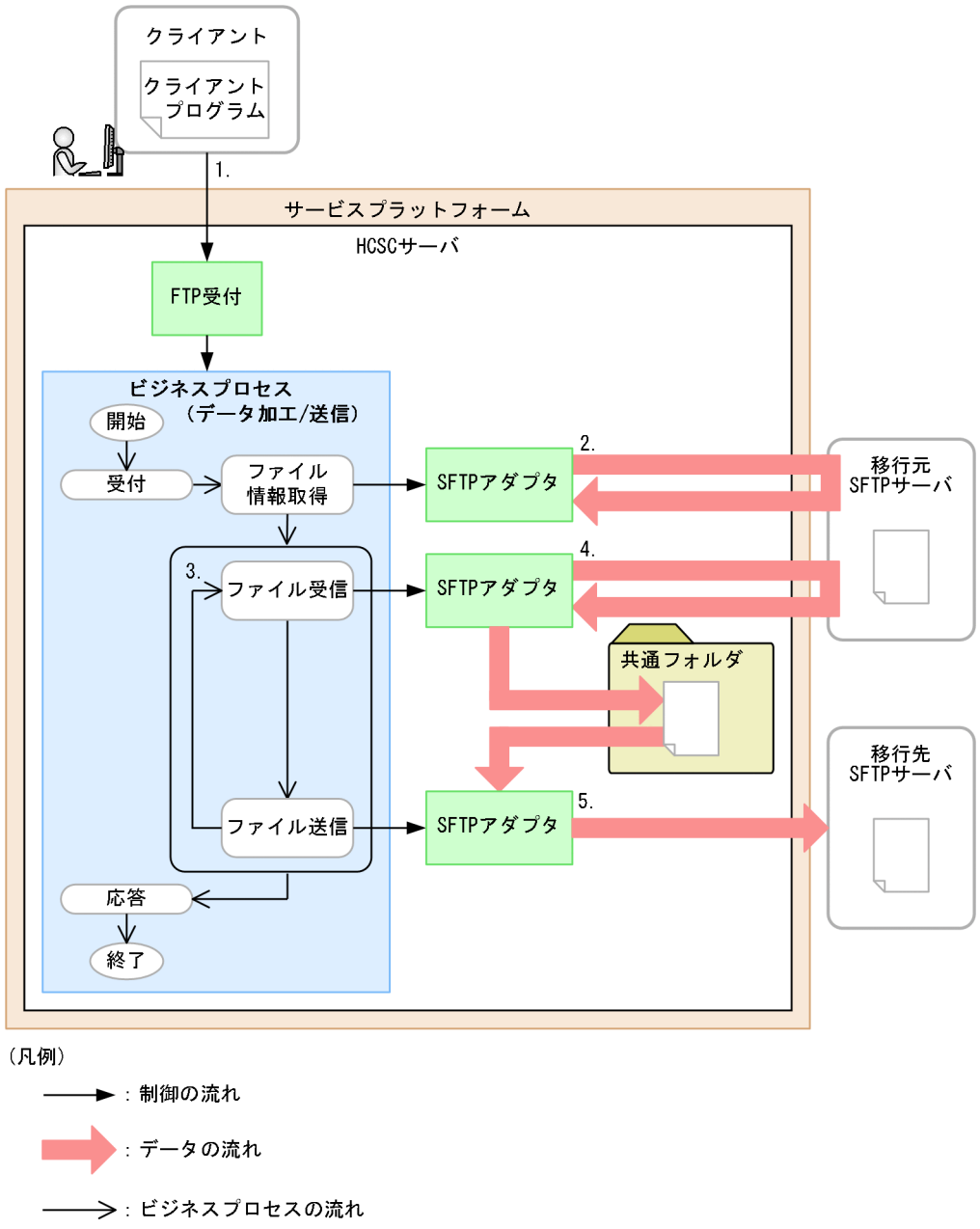
- 移行元サーバから取得したファイル情報一覧を基にファイルを取得して移行先サーバへ送信する
- クライアント側から転送されたファイルを変換して目的のサーバへ送信する

以降に，それぞれのシステムの例について説明します。

(1) 移行元サーバから取得したファイル情報一覧を基にファイルを取得して移行先サーバへ送信する

SFTP アダプタが取得した移行元サーバのファイル情報を基に，移行元サーバからファイル取得して移行先サーバへ送信する例を示します。

図 2-191 SFTP アダプタを利用したシステムの例（1）



図中の 1.～5.について説明します。

1. クライアントのプログラムからビジネスプロセスを実行するリクエストを発行します。
2. SFTP アダプタは移行元 SFTP サーバにアクセスし，ファイル名一覧を取得します。
3. 取得したファイル名一覧からファイル名を 1 つずつ取得します。

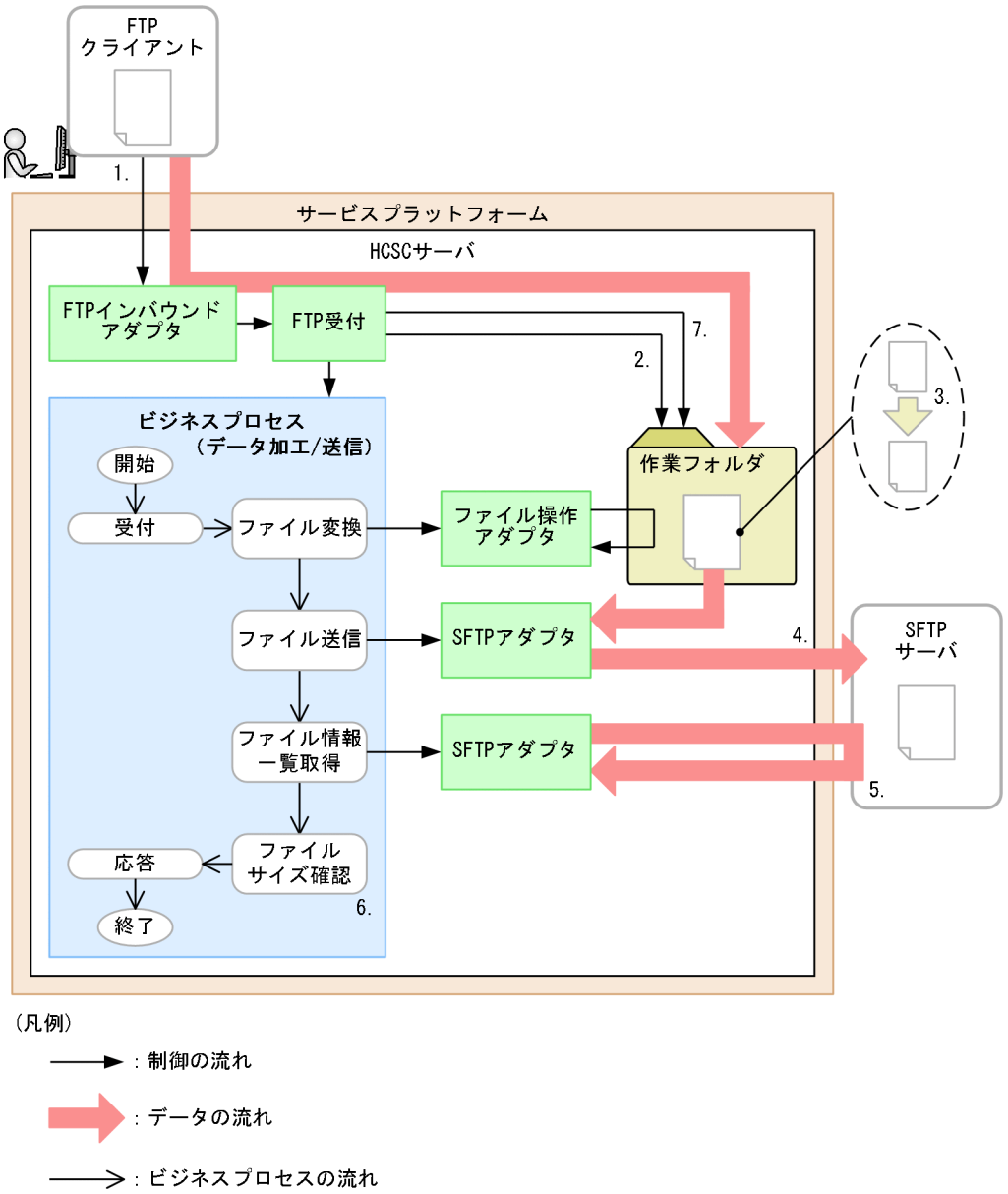
4. SFTP アダプタは取得したファイル名を使用して，移行元 SFTP サーバからファイルを受信し，共通フォルダに保存します。

5. SFTP アダプタは，共通フォルダに保存されたファイルを移行先 SFTP サーバへ送信します。

(2) クライアント側から転送されたファイルを変換して目的のサーバへ送信する

ここでは，クライアント側から送信されたファイルを HCSC サーバでファイル変換し，変換後のファイルを SFTP アダプタから SFTP サーバに送信するシステムの例を示します。

図 2-192 SFTP アダプタを利用したシステムの例（2）



図中の 1.～7.について説明します。

1. クライアントのプログラムから HCSC サーバへファイルを送信します。

2. FTP 受付は作業フォルダを作成し，受信したファイルを中間ファイルとして生成します。
3. ファイル操作アダプタは，FTP 受付で生成された中間ファイルをファイル変換オペレーションによって変換します。
4. SFTP アダプタは，変換後のファイルを目的の SFTP サーバへ送信します。
5. 送信後に SFTP アダプタは SFTP サーバのファイル情報一覧を取得します。
6. Java 呼出アクティビティでファイル情報一覧から該当するファイルのファイルサイズを取得し，ファイル変換オペレーションの応答電文に設定されているファイルサイズと比較します。
7. ビジネスプロセス終了後に，FTP 受付は作業フォルダを削除します。

2.18.3 SFTP アダプタで設定できるファイル転送の機能

SFTP アダプタで設定できるファイル転送機能に関する情報を次に示します。

表 2-133 設定できるファイル転送機能に関する情報

大分類	中分類	小分類	説明
SFTP 通信	SFTP コマンド		SFTP アダプタが実行するコマンドは，次の 2 つに大別できます。 • SFTP アダプタのオペレーションに対応する SFTP コマンド • ユーザが要求電文に指定するユーザ指定の SFTP コマンド SFTP コマンドの詳細は，「2.18.5(3) SFTP アダプタが実行するコマンド」を参照してください。
	文字セット		UTF-8 固定です。
	データ圧縮		ファイル転送時にデータを圧縮します。 データ圧縮時の圧縮レベルは変更できません。
	ファイルの書き込み形式	上書き	転送先にファイルが存在する場合にファイルを上書きします。 ファイルが存在しない場合は新規に作成します。
		追記	転送先にファイルが存在する場合にファイルに追記します。 ファイルが存在しない場合は新規に作成します。
	複数ファイルの転送		複数ファイルの転送およびディレクトリ単位の転送はできません。
	タイムアウト	セッション接続時	指定した時間を経過しても SFTP サーバへのセッション接続要求が終了しなかった場合は例外を発生させます。
		データ読み込み	指定した時間を経過しても SFTP サーバから応答データが読み込めなかった場合に例外を発生させます。
	IP アドレスの固定		SFTP アダプタが使用する IP アドレスを固定にします。
	ポート番号の固定		SFTP アダプタが使用するポート番号を固定にします。
	IPv6 サポート		SFTP アダプタが接続する SFTP サーバおよび SFTP アダプタが使用する IP アドレスを，IPv6 形式で指定できます。

大分類	中分類	小分類	説明
セキュリティ	ホスト認証	厳格なホスト鍵チェック	サーバの正当性を確保するために，ホスト公開鍵が記載されたファイルを使用してホスト認証をします。
	ユーザ認証	パスワード認証	SFTP サーバが存在するマシンのユーザ名とパスワードを使用して認証をします。
		公開鍵認証	公開鍵・秘密鍵を使用した公開鍵暗号方式で認証をします。
	暗号		SFTP アダプタで使用できる暗号の優先順位を変更できます。 ただし，SFTP アダプタで使用できる暗号が決まっているため，暗号の追加はできません。
排他制御	占有ロック		共有フォルダ内のファイルの更新処理時に，排他管理をします。 他プロセスまたは他スレッドとの競合によって排他管理に失敗したときは，指定間隔おきに指定回数まで排他取得処理をリトライします。
	共有ロック		共有フォルダ内のファイルの参照処理時に，排他管理をします。 他プロセスまたは他スレッドとの競合によって排他管理に失敗したときは，指定間隔おきに指定回数まで排他取得処理をリトライします。
ユーザ認証情報管理コマンド	パスワード認証情報管理	アカウント登録	パスワード認証に使用するユーザおよびパスワードを登録します。
		アカウント一覧表示	登録されているユーザを表示します。
		アカウント削除	登録されているユーザおよびパスワードを削除します。
		コマンド実行のログ	コマンド実行時の例外情報をログとして出力します。
	公開鍵認証情報管理	秘密鍵の登録	公開鍵認証に使用する秘密鍵およびパスフレーズ（任意）を登録します。 鍵ファイルの中身を管理するのではなく，鍵ファイルのファイルパスを管理します。
		秘密鍵の一覧表示	登録されている秘密鍵のファイルパスを表示します。
		秘密鍵の削除	登録されている秘密鍵およびパスフレーズを削除します。
		コマンド実行のログ	コマンド実行時の例外情報をログとして出力します。
ログ・トレース	メッセージログ		ログおよびトレース情報を出力します。
	保守ログ（メソッドトレース）		
	性能解析トレース		
	例外ログ		
	SFTP プロトコルトレース		

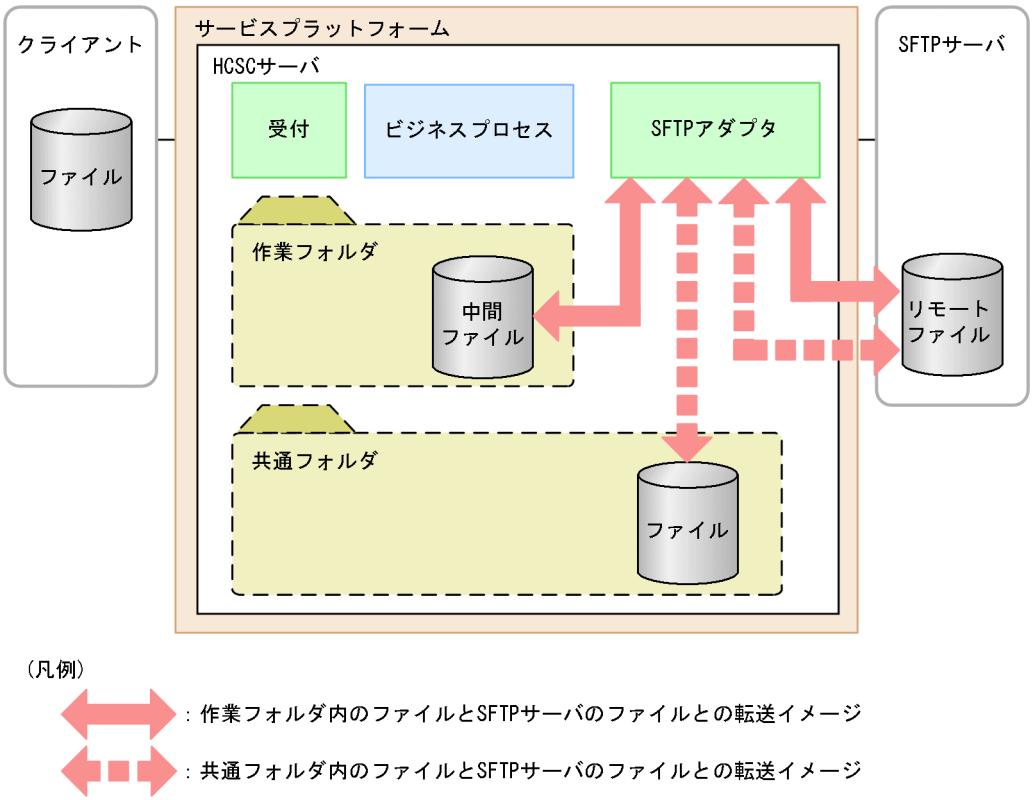
2.18.4　SFTP アダプタのファイルアクセス方式

SFTP アダプタのファイルへのアクセス方式は次の 2 つです。

- 作業フォルダ内の中間ファイルにアクセスする方式
- クラスタ構成内で共通に利用する共通フォルダのファイルにアクセスする方式

SFTP アダプタからフォルダ内のファイルにアクセスするときのデータの流れを次の図に示します。

図 2-193　フォルダ内のファイルにアクセスするときのデータの流れ



ファイル操作が PUT オペレーションまたは GET オペレーションの場合，作業フォルダを利用するか共通フォルダを利用するかの判定は，要求電文のローカル側フォルダ（local-folder）の common 属性の設定によって異なります。ローカル側にフォルダが設定されていないときは作業フォルダを使用し，設定されているときは共通フォルダを使用します。

なお，作業フォルダや共通フォルダを使用する場合，フォルダへのパスは，HCSC サーバランタイム定義ファイルに設定する必要があります。

GETINFO オペレーションは，ファイル一覧情報の取得結果を応答電文にだけ格納します。ファイル一覧情報の取得結果は，共通フォルダおよび作業フォルダには出力されません。

PUT オペレーションおよび GET オペレーションで共通フォルダを使用するときに指定するパスを次に示します。

表 2-134　共通フォルダを使用するときに指定するパス

種別	指定するパス
ローカルパス	<要求電文のローカル側フォルダから取得した共通フォルダ定義名に対応する共通フォルダ>/<要求電文のローカル側ファイル名>
リモートパス	<要求電文のリモートパス名>

SFTP アダプタでは，共通フォルダルートの配下にサブフォルダを作成できます。この作成したサブフォルダの相対パスを指定すると，ファイルを転送できます。

共通フォルダルートの配下に作成したサブフォルダの相対パスは，各オペレーションの要求電文または応答電文のローカル側ファイル名（local-file-name）に指定してください。指定例を次に示します。

```
<request>
   :
  <local-folder common="true">default</local-folder>
  <local-file-name>/folder1/folder2/a.txt</local-file-name>
   :
</request>
```

注意事項

- 作業フォルダについては，相対パスの指定はできません。
- 共通フォルダルートより上位のパスを指定した場合，エラーになります。
- 相対パスの先頭のファイル区切り文字（"/"）は省略できます。

2.18.5　SFTP アダプタと SFTP サーバ間の通信

SFTP アダプタと SFTP サーバ間の通信について説明します。

(1)　SFTP サーバとの接続

SFTP サーバとの接続では，SFTP アダプタは SFTP サーバに対してセッション接続を要求します。セッション接続では次の処理が実行されます。

- アルゴリズムネゴシエーション
- ホスト認証
- ユーザ認証

セッション接続は，リクエストごとに接続または切断をし，ほかのリクエストで再利用をしません。

以降で，セッション接続時の処理について説明します。

(a) アルゴリズムネゴシエーション

暗号化通信路の確立のために，次の情報のリスト（以降，アルゴリズムのリストと呼びます）を SFTP アダプタと SFTP サーバ間で交換します。

- 鍵交換方式
- 共通鍵暗号方式
- メッセージ認証コード
- ホスト認証の方式
- 圧縮アルゴリズム
- 公開鍵認証で許可する署名アルゴリズム

SFTP アダプタがサポートするアルゴリズムのリストを次に示します。

表 2-135　アルゴリズムのリスト

種別	アルゴリズムのリスト
鍵交換方式	• ecdh-sha2-nistp256 • ecdh-sha2-nistp384 • ecdh-sha2-nistp521 • diffie-hellman-group-exchange-sha256 • diffie-hellman-group16-sha512 • diffie-hellman-group18-sha512 • diffie-hellman-group14-sha256
共通鍵暗号方式	• aes128-ctr • aes192-ctr • aes256-ctr • aes128-gcm@openssh.com • aes256-gcm@openssh.com
メッセージ認証コード	• hmac-sha2-256-etm@openssh.com • hmac-sha2-512-etm@openssh.com • hmac-sha1-etm@openssh.com • hmac-sha2-256 • hmac-sha2-512 • hmac-sha1
ホスト認証の方式	• ecdsa-sha2-nistp256 • ecdsa-sha2-nistp384 • ecdsa-sha2-nistp521 • rsa-sha2-512 • rsa-sha2-256
圧縮アルゴリズム	• zlib@openssh.com • zlib

種別	アルゴリズムのリスト
公開鍵認証で許可する署名アルゴリズム	• ecdsa-sha2-nistp256 • ecdsa-sha2-nistp384 • ecdsa-sha2-nistp521 • rsa-sha2-512 • rsa-sha2-256

使用されるアルゴリズムは，クライアントから送信されたリストを先頭から順に参照し，サーバのリストに存在するものが選ばれます。

暗号の優先順位は，上記の表の記載順となります（上位にあるほど優先順位が高い）。

アルゴリズムの優先順位は，SFTP アダプタ実行環境プロパティファイルで入れ替えることができます。

(b) ホスト認証

ホスト認証とは，サーバのなりすましを判断するための認証です（クライアント側でサーバの正当性を確保します）。

ホスト認証では厳格なホスト公開鍵チェックをすることによって，接続先が目的のサーバであるかどうかを検証できます。

●厳格なホスト公開鍵チェックをする場合

SFTP アダプタのホスト認証は，ホスト公開鍵が記載されたファイル（以降，ホスト公開鍵ファイルと呼びます）を使用します。

SFTP サーバから受け取ったホスト公開鍵と SFTP アダプタ実行環境プロパティファイルで設定したホスト公開鍵を比較し，ホスト公開鍵が存在しない場合や，ホスト公開鍵が変更されている場合はエラーになります。

ホスト認証の定義手順については，「2.18.9(1) ホスト認証の認証情報の定義手順」を参照してください。

●厳格なホスト公開鍵チェックをしない場合

ホスト公開鍵のチェックをしません。

(c) ユーザ認証

ユーザ認証とは，クライアントが指定したユーザがサーバにログインできるかをサーバ側が確認するための認証です。SFTP アダプタでは，ユーザ認証を省略することはできません。

SFTP アダプタでサポートするユーザ認証を次に示します。

●パスワード認証

SFTP サーバが存在するマシンのユーザ名とパスワードを使用して認証をします。

パスワード認証の定義手順については，「2.18.9(2) パスワード認証の認証情報の定義手順」を参照してください。

●公開鍵認証

公開鍵・秘密鍵を使用した公開鍵暗号方式で認証をします。

公開鍵認証の定義手順については，「2.18.9(3)　公開鍵認証の認証情報の定義手順」を参照してください。

使用できる公開鍵・秘密鍵の条件を次に示します。

・公開鍵・秘密鍵の種類

次のどちらかである必要があります。

- RSA
- ECDSA

・公開鍵・秘密鍵の鍵長

次の条件を満たしていない場合，動作は保証されません。

- RSA の場合の鍵長は，2048 バイトまたは 4096 バイトのどちらか
- ECDSA の場合の鍵長は，256 バイト，384 バイト，または 521 バイトのどれか

・パスフレーズを設定した秘密鍵の暗号化

次に示すどちらかの公開鍵・秘密鍵で暗号化します。

- RSA
- ECDSA

・公開鍵・秘密鍵の形式

次のどれかである必要があります。

- OpenSSL's PEM format
  鍵ファイルのヘッダが，「-----BEGIN RSA」，「-----BEGIN EC」のどちらかで始まる形式
- PKCS#8 形式
  秘密鍵にパスフレーズを設定しない場合：「-----BEGIN PRIVATE」で始まる形式
  秘密鍵にパスフレーズを設定する場合：「-----BEGIN ENCRYPTED PRIVATE KEY」で始まる形式
- OpenSSH Private Key Format 形式
  「-----BEGIN OPENSSH PRIVATE KEY」で始まる形式

(2)　ファイル転送のパターン

SFTP アダプタで実行可能なファイル転送のパターンを次に示します。

- SFTP アダプタから SFTP サーバへのファイル転送
- SFTP サーバから SFTP アダプタへのファイル転送
- SFTP サーバのファイルリストの取得

以降で，それぞれのパターンについて説明します。

(a) SFTP アダプタから SFTP サーバへのファイル転送

SFTP アダプタから SFTP サーバへファイルを転送します。

1 リクエストで転送できるファイルは 1 つだけで，ディレクトリ単位での転送や複数のファイルの転送はできません。

また，転送先にすでにファイルがある場合は上書き，または追記をします。

要求電文の overwrite-data 要素の値が true の場合は上書きを，要素の値が false の場合は追記をします。

ファイル転送に失敗した場合，SFTP アダプタは後処理として転送先のファイルは削除しません。そのため，転送先のファイルが削除されるかどうかは SFTP サーバに依存します。

(b) SFTP サーバから SFTP アダプタへのファイル転送

SFTP サーバから SFTP アダプタへファイルを転送します。

1 リクエストで転送できるファイルは 1 つだけで，ディレクトリ単位での転送や複数のファイルの転送はできません。

また，転送先にすでにファイルがある場合は上書きをします。

ファイル転送に失敗した場合，後処理としてローカル側のファイルの削除を試みます。

(c) SFTP サーバのファイルリストの取得

SFTP サーバのファイル情報の一覧，またはファイル名一覧を取得します。

ファイル情報とファイル名のどちらの一覧を取得するかは，要求電文で指定します。

また，取得対象は要求電文で指定し，ディレクトリを指定した場合はディレクトリ配下のファイルの情報を，ファイルを指定した場合は該当するファイルの情報を取得します。

ファイル情報の一覧，またはファイル名一覧の取得結果は応答電文に格納し，ファイルへは出力しません。

ファイル情報の一覧取得では，取得したファイルごとに応答電文に格納されます。

応答電文に格納されるときのフォーマットを次に示します。

```
パーミッション ユーザ識別子 グループ識別子 ファイルサイズ(バイト) 更新時刻 ファイル名
```

ファイルへは，各項目を空白区切りで出力します。

ファイルへの出力例を次に示します。

```
-rw-r--r-- 0 0 9936 Thu Oct 16 05:28:47 JST 2018 data1
```

(3) SFTP アダプタが実行するコマンド

SFTP アダプタが実行するコマンドは，次の 2 つに大別できます。

- SFTP アダプタのオペレーションに対応する SFTP コマンド
- ユーザが要求電文に指定するユーザ指定の SFTP コマンド

次に，それぞれの SFTP コマンドについて説明します。

(a) SFTP アダプタのオペレーションに対応する SFTP コマンド

オペレーションに対応する SFTP コマンドは，SFTP アダプタが要求電文から実行するオペレーションを自動的に判別して実行します。

SFTP アダプタが自動的に SFTP コマンドを実行するため，ユーザはコマンドの書式を考慮する必要はありません。

SFTP アダプタのオペレーションに対応する SFTP コマンドの一覧を次に示します。

表 2-136 SFTP アダプタのオペレーションに対応する SFTP コマンドの一覧

コマンド名	説明	対応するオペレーション
get	リモートマシンのファイルをローカルマシンにダウンロードします。	GET オペレーション
ls	リモートマシン上のファイル一覧またはファイル名一覧を取得します。	GETINFO オペレーション
put	ローカルマシンのファイルをリモートマシンにアップロードします。	PUT オペレーション

(b) ユーザが要求電文に指定するユーザ指定の SFTP コマンド

ユーザ指定の SFTP コマンドは，要求電文の sftp-commands-before および sftp-commands-after に実行指示がある場合だけコマンドが実行されます。

ユーザは，コマンドの機能および書式を考慮した上で，要求電文に SFTP コマンドを指定する必要があります。

ユーザ指定の SFTP コマンドの一覧を次に示します。

表 2-137 ユーザ指定の SFTP コマンドの一覧

コマンド名	コマンドの形式	説明
cd	cd 〈変更先のディレクトリのパス〉※	リモート側のカレントディレクトリを変更します。

コマンド名	コマンドの形式	説明
cd	cd 〈変更先のディレクトリのパス〉※	cd コマンドで変更したカレントディレクトリは，再度 cd コマンドを実行するか，リクエストが終了するまで有効です。
mkdir	mkdir 〈作成するディレクトリのパス〉	リモートマシン上にディレクトリを作成します。
rename	rename 〈変更前のファイルのパス〉〈変更後のファイルのパス〉	リモートマシン上のファイルの名前を変更します。
rm	rm 〈削除するファイル名〉	リモートマシン上のファイルを削除します。
rmdir	rmdir 〈削除するディレクトリ名〉	リモートマシン上のディレクトリを削除します。 ディレクトリ配下が空の場合だけ削除できます。

注※

接続先が Windows 上の SFTP サーバの場合は，マルチバイト文字を含むディレクトリ名を変更先のディレクトリのパスに指定しないでください。指定した場合の動作は保証されません。

注意事項

- コマンドの形式に誤りがある場合，SFTP アダプタは例外を通知します。
- 複数のコマンドを指定する場合，半角のセミコロン（;）で区切ってください。
- 複数のコマンドを指定した場合，先頭から順にコマンドを実行します。
- コマンド引数は，半角の空白区切りで指定してください。
- コマンド引数の区切りに使用する空白と，コマンドの区切りに使用するセミコロン（;）は，コマンド引数に指定できません。指定した場合の動作は保証されません。
- コマンド引数の内容については，SFTP アダプタでチェックされません。

(4)　転送データの圧縮

SFTP アダプタは，データを圧縮して転送できます。

転送データの圧縮は，要求電文または SFTP アダプタ実行環境プロパティファイルのどちらかで指定できます。要求電文フォーマットの作成方法については，マニュアル「サービスプラットフォーム 開発ガイド 受付・アダプタ定義編」の「3.3.15　SFTP アダプタを定義する」の電文フォーマットの作成に関する説明を参照してください。SFTP アダプタ実行環境プロパティファイルで設定できる項目の詳細は，マニュアル「サービスプラットフォーム リファレンス」の「6.16.1　SFTP アダプタ実行環境プロパティファイル」を参照してください。

注意事項

データを圧縮して転送する場合，転送速度が低下することがあります。

(5)　SFTP サーバとの接続の切断

リクエストの正常終了，異常終了に関係なく，SFTP アダプタはリクエスト終了時に，SFTP サーバとのセッションを切断します。

(6)　厳密な鍵交換

SFTP サーバとの接続時に，厳密な鍵交換（strict key exchange）を実施します。

厳密な鍵交換は，SFTP アダプタ実行環境プロパティファイルの「sftpadp.strict.key.exchange.enabled」プロパティで設定できます。「sftpadp.strict.key.exchange.enabled」プロパティを設定していない場合は，デフォルトで厳密な鍵交換を実施します。

SFTP アダプタ実行環境プロパティファイルの詳細は，マニュアル「サービスプラットフォーム リファレンス」の「6.16.1　SFTP アダプタ実行環境プロパティファイル」を参照してください。

ただし，接続先の OpenSSH サーバが厳密な鍵交換をサポートしていない場合は，厳密な鍵交換は行われません。

厳密な鍵交換をサポートしていない SFTP サーバとの接続を許可するかどうかは，SFTP アダプタ実行環境プロパティファイルの「sftpadp.required.strict.key.exchange」プロパティで設定できます。「sftpadp.required.strict.key.exchange」プロパティを設定していない場合は，デフォルトで厳密な鍵交換をサポートしていない SFTP サーバとの接続を許可します。

(7)　公開鍵認証で許可する署名アルゴリズムの設定

公開鍵認証を行う場合，公開鍵認証で許可する署名アルゴリズムを設定できます。公開鍵認証で許可する署名アルゴリズムは，SFTP アダプタ実行環境プロパティファイルの「sftpadp.pubkey.accepted.algorithms」プロパティで設定します。

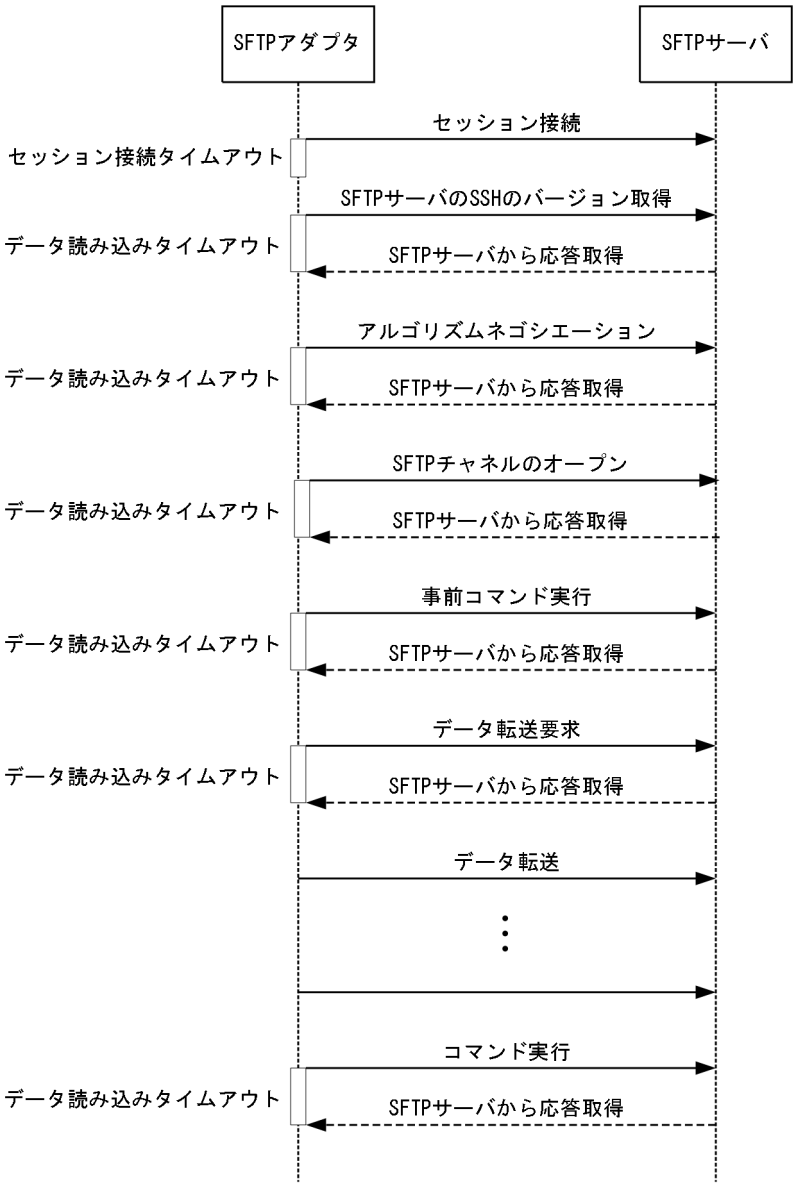
SFTP アダプタ実行環境プロパティファイルの詳細は，マニュアル「サービスプラットフォーム リファレンス」の「6.16.1　SFTP アダプタ実行環境プロパティファイル」を参照してください。

「sftpadp.pubkey.accepted.algorithms」プロパティに設定した値が，接続先 SFTP サーバで許可されていない署名アルゴリズムの場合は，エラーを出力し，リクエスト処理を中断します。

2.18.6　SFTP アダプタのタイムアウトを監視するポイント

SFTP アダプタでタイムアウトを監視するポイントを次に示します。

図 2-194　SFTP アダプタでタイムアウトを監視するポイント



SFTP アダプタでは，セッション接続タイムアウトおよびデータ読み込みタイムアウトを設定できます。各タイムアウト値は，SFTP アダプタ実行環境プロパティファイルで設定できます。SFTP アダプタ実行環境プロパティファイルの詳細は，マニュアル「サービスプラットフォーム リファレンス」の「6.16.1 SFTP アダプタ実行環境プロパティファイル」を参照してください。

(1)　セッション接続タイムアウト

SFTP サーバへのセッション接続要求時に，セッション接続タイムアウトに指定した時間を経過しても接続要求が終了しない場合，SFTP アダプタは例外を通知します。

設定できるパラメタを次の表に示します。

表 2-138　セッション接続タイムアウトの設定

値	キー名称	デフォルト値
セッション接続タイムアウト（秒）	sftpadp.session.timeout	10

また，セッション接続要求時に相手先ビジーなどで接続を確立できなかった場合，セッション接続タイムアウトに指定されたタイムアウト値まで SFTP サーバへの接続をリトライします。

セッション接続タイムアウトのタイムアウト値には，リトライ回数とリトライ間隔を指定できます。リトライ回数に達してもセッション接続を確立できなかった場合，SFTP アダプタは例外を通知します。

設定できるパラメタを次の表に示します。

表 2-139　セッション接続タイムアウトのタイムアウト値の設定

値	キー名称	デフォルト値
セッションを確立できなかったときのリトライ回数※1	sftpadp.session-con.retry.count	5
セッションを確立できなかったときのリトライ間隔※2（秒）	sftpadp.session-con.retry.interval	2

注※1

値に 0 を指定した場合，リトライしません。

注※2

値に 0 を指定した場合，間隔を空けないでリトライします。

(2)　データ読み込みタイムアウト

SFTP サーバからの応答データの読み込み（アルゴリズムネゴシエーションの結果取得，GET オペレーション実行時のファイルデータ取得など）で，データ読み込みタイムアウトに指定した時間を経過しても応答データが SFTP サーバから読み込めない場合，SFTP アダプタは例外を通知します。

設定できるパラメタを次の表に示します。

表 2-140　データの読み込みタイムアウトの設定

値	キー名称	デフォルト値
データ読み込みタイムアウト（秒）	sftpadp.data-read.timeout	60

2.18.7　SFTP アダプタの排他管理

共通フォルダ内のファイルを PUT オペレーションで SFTP サーバに転送するとき，共通フォルダ内のファイルに対して共有ロックが設定されます。

また，GET オペレーションで共通フォルダにファイルを取得するとき，共通フォルダのファイルに対して占有ロックが設定されます。

共有ロックまたは占有ロックの設定時に，ほかのプロセス，またはほかのスレッドとの競合によって，排他取得処理に失敗することがあります。排他取得タイムアウトでは，SFTP アダプタが共有ロックまたは占有ロックの取得に失敗したときのリトライ回数およびリトライ間隔を設定できます。

設定できるパラメタを次の表に示します。

表 2-141　排他取得タイムアウトの設定

値	キー名称	デフォルト値
共有ロック失敗時のリトライ回数	sftpadp.read-lock.retry.count	0
共有ロック失敗時のリトライ間隔（秒）	sftpadp.read-lock.retry.interval	1
占有ロック失敗時のリトライ回数	sftpadp.write-lock.retry.count	0
占有ロック失敗時のリトライ間隔（秒）	sftpadp.write-lock.retry.interval	1

タイムアウト時間は，これらのパラメタに指定した値を使った次の計算式で求められます。

- 共有ロックの場合

  共有ロック失敗時のリトライ回数×共有ロック失敗時のリトライ間隔＋排他取得時のファイルアクセス時間

- 占有ロックの場合

  占有ロック失敗時のリトライ回数×占有ロック失敗時のリトライ間隔＋排他取得時のファイルアクセス時間

リトライ回数およびリトライ間隔は，SFTP アダプタ実行環境プロパティファイルで設定できます。SFTP アダプタ実行環境プロパティファイルの詳細は，マニュアル「サービスプラットフォーム リファレンス」の「6.16.1　SFTP アダプタ実行環境プロパティファイル」を参照してください。

2.18.8　SFTP アダプタが使用する IP アドレスおよびポート番号の指定

SFTP アダプタが使用する IP アドレスおよびポート番号を指定できます。

(1)　SFTP クライアントで使用する IP アドレス

SFTP クライアントの IP アドレスは，SFTP アダプタ実行環境プロパティファイルの sftpadp.client-ipaddr プロパティで指定します。指定を省略した場合は，システムによってローカルマシンの IP アドレスが自動的に割り当てられます。ローカルマシンの IP アドレスは，java.net.NetworkInterface クラスでまとめて取得し，そのうち最初に見つかった実 IP アドレスを優先して割り当てます。実 IP アドレスが存在しない場合は，ループバックアドレスを割り当てます。

(2) SFTP クライアントで使用するポート番号

SFTP アダプタは指定されたポート番号の最小値，または前回使用したポート番号の次※を待ち受けポート番号候補とします。待ち受けポート番号候補を使用できる場合，このポート番号を採用します。待ち受けポート番号候補を使用できない場合は，次のポート番号※を候補とし，使用可否を確認します。範囲内のすべてのポートが使用できない場合は，SFTP アダプタが例外を返します。

注※

前回使用したポート番号が最大値の場合，最小値が次のポート番号となります。

なお，ポート番号を設定するためのプロパティの指定を省略した場合は，システムによって任意のポート番号が割り当てられます。

2.18.9 SFTP アダプタからファイルを送信するときの認証

ここでは，SFTP アダプタからファイルを送信するときのホスト認証の認証情報の定義手順，パスワード認証の認証情報の定義手順，および公開鍵認証の認証情報の定義手順について説明します。

(1) ホスト認証の認証情報の定義手順

ホスト認証を実施する場合の認証情報の設定手順を次に示します。

1. ホスト公開鍵ファイルの作成

SFTP アダプタにはホスト公開鍵ファイルを作成する機能がないため，ホスト公開鍵ファイルをユーザが作成する必要があります。

ホスト公開鍵ファイルを作成するには，リモートマシン上の SFTP サーバからホスト公開鍵を取得し，ローカルマシン（実行環境）で作成したファイル（ホスト公開鍵ファイル）に鍵情報を記載します。

SFTP アダプタで使用できるホスト公開鍵ファイルの形式を次に示します。

なお，次に示す形式に従っていない場合は，動作は保証されません。

```
ホスト△鍵の種類△base64エンコードされた鍵
```

（凡例）

△：半角スペースを示します。

2. ホスト公開鍵ファイルの設定

SFTP アダプタ実行環境プロパティファイルの sftpadp.host-auth-inf-filepath プロパティに，手順 1. で作成したホスト公開鍵ファイルのファイルパスを設定します。

3. ホスト認証の実施の設定

要求電文の request/host-auth/strict-hostkey-checking 要素，または SFTP アダプタ実行環境プロパティファイルの sftpadp.strict-hostkey-checking プロパティに「true」を設定します。

(2) パスワード認証の認証情報の定義手順

パスワード認証を実施する場合の認証情報の設定手順を次に示します。

1. ユーザ名およびパスワードの登録

   csauseradd コマンドを実行し，パスワード認証で使用するユーザ名およびパスワードを登録します。コマンドで登録した情報は，コマンド実行時に指定したパスワード認証情報定義ファイルに設定されます。

   csauseradd コマンドの詳細については，マニュアル「サービスプラットフォーム リファレンス」の「csauseradd（パスワード認証のユーザ情報の登録・更新)」を参照してください。

2. パスワード認証の実施の設定

   要求電文の request/user-auth/user-auth-type 要素，または SFTP アダプタ実行環境プロパティファイルの sftpadp.user-auth-type プロパティに「password」を設定します。

3. パスワード認証情報定義ファイルの設定

   SFTP アダプタ実行環境プロパティファイルの sftpadp.password-auth-inf-filepath プロパティにパスワード認証情報定義ファイルのファイルパスを設定します。

4. パスワード認証で使用するユーザの指定

   要求電文の request/user-auth/user-auth-info 要素，または SFTP アダプタ実行環境プロパティファイルの sftpadp.password-auth-info プロパティにユーザ名を設定します。

(3) 公開鍵認証の認証情報の定義手順

公開鍵認証を実施する場合の認証情報の設定手順を次に示します。

1. 公開鍵・秘密鍵の作成

   SFTP アダプタには公開鍵・秘密鍵を作成する機能がないため，公開鍵・秘密鍵をユーザが作成する必要があります。

   公開鍵・秘密鍵は，ローカルマシン（実行環境）で作成します。作成した公開鍵は公開鍵ファイルとして，秘密鍵は秘密鍵ファイルとして，それぞれローカルマシン上に保存します。

2. 公開鍵ファイルの格納

   公開鍵ファイルはリモートマシン上に格納します（公開鍵ファイルの格納場所は SFTP サーバに依存します)。

3. 秘密鍵ファイルの登録

   csakeyadd コマンドを実行し，秘密鍵ファイルに関する情報を登録します。コマンドで登録した情報は，コマンド実行時に指定した公開鍵認証情報定義ファイルに設定されます。

   csakeyadd コマンドの詳細については，マニュアル「サービスプラットフォーム リファレンス」の「csakeyadd（公開鍵認証の秘密鍵情報の登録・更新)」を参照してください。

4. 公開鍵認証情報定義ファイルの設定

SFTP アダプタ実行環境プロパティファイルの sftpadp.publickey-auth-inf-filepath プロパティに，手順 3.で登録した公開鍵認証情報定義ファイルのファイルパスを設定します。

5. 公開鍵認証の実施の設定

   要求電文の request/user-auth/user-auth-type 要素，または SFTP アダプタ実行環境プロパティファイルの sftpadp.user-auth-type プロパティに「publickey」を設定します。

6. 公開鍵認証で使用する秘密鍵の指定

   要求電文の request/user-auth/user-auth-info 要素，または SFTP アダプタ実行環境プロパティファイルの sftpadp.publickey-auth-info プロパティに csakeyadd コマンド実行時に設定した鍵 ID を設定します。

2.18.10　SFTP アダプタの要求電文に指定できるリモートのファイルパス

SFTP アダプタの要求電文では，リモートのファイルパスに絶対パス，または相対パスを指定できます。

(1)　リモートのファイルパスを指定できる要素

リモートのファイルパスを指定できる要素を次に示します。

remote-path 要素

- PUT オペレーション実行時のファイル転送先
- GET オペレーション実行時のファイル転送元

getinfo-path 要素

GETINFO オペレーション実行時のファイル情報の取得対象

sftp-commands-before 要素

コマンド引数に指定するファイルパス

sftp-commands-after 要素

コマンド引数に指定するファイルパス

(2)　相対パスの基準

相対パスの基準となるディレクトリは次のどちらかです。

- ユーザのホームディレクトリ
- cd コマンドで移動したディレクトリ

(3)　絶対パスの指定方法

リモートのファイルパスを絶対パスで指定する場合は，ファイルパスの先頭に「/」を付与してください。

注意事項

ファイルパスの先頭の文字が「/」でない場合は，相対パスが指定されたと見なされます。

2.19　ファイルイベント連携を使用したファイルの監視

ファイルイベント連携とは，特定のフォルダを監視し，ファイルの作成をファイルイベントとして検知することで，サービスプラットフォームにファイルイベントを通知する機能（インターフェース）です。

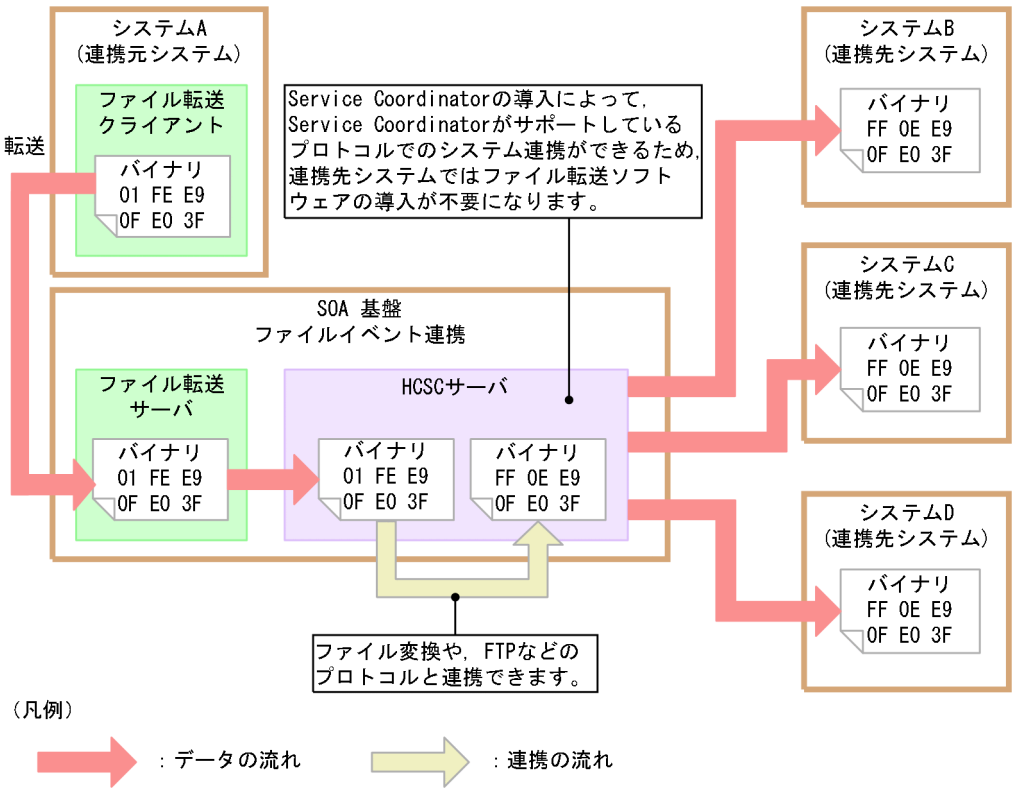
2.19.1　ファイルイベント連携を使用したシステム間連携

ファイルイベント連携をすることで，ファイルの作成を契機として，HCSC サーバ上でビジネスプロセスを利用した処理を実行できます。例えば，HCSC サーバが別途提供する機能を使って，ファイル変換機能や FTP などのプロトコルと連携することによって，システム間の連携もできるようになります。

FTP 連携機能の詳細については，「8.　FTP と連携してファイルを転送する機能（FTP 連携）」を参照してください。

ファイルイベント連携を利用したシステム間連携のイメージを次の図に示します。

図 2-195　システム間の連携イメージ



2.19.2　ファイルイベント連携で監視するフォルダおよびファイル

ファイルイベント連携で使用するフォルダ，およびファイルについて説明します。

監視対象のフォルダは次のどちらかが選択できます。選択したフォルダによって監視対象のファイルが異なります。

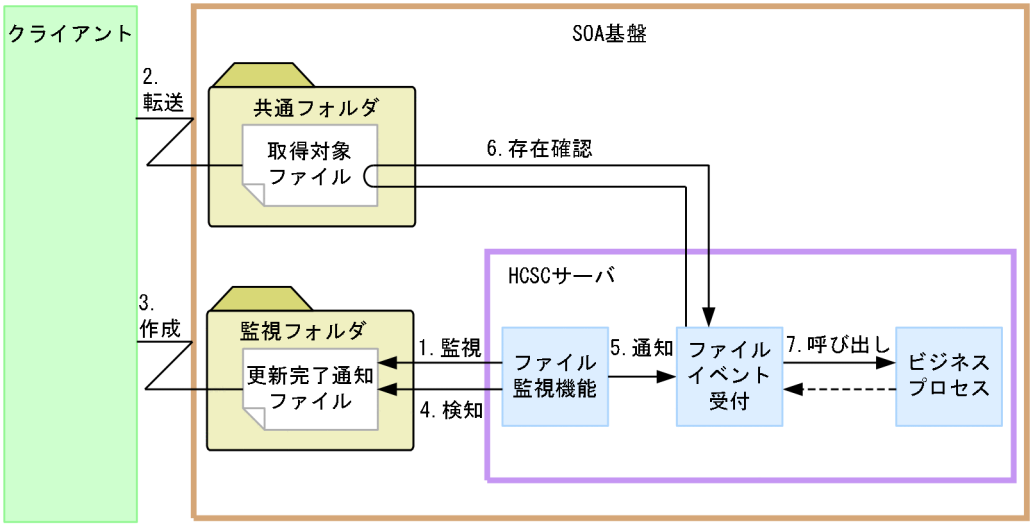
- 監視フォルダ
  監視フォルダ内の更新完了通知ファイルを監視します。
- 共通フォルダ
  共通フォルダ内の取得対象ファイルを監視します。

監視対象のフォルダ内にファイルが作成された場合のイベントを処理するには，ファイルイベント受付を使用します。次に，ファイルイベント連携の処理の流れを示します。

(1)　監視フォルダを監視する場合の処理の流れ

監視フォルダを監視する場合の，ファイルイベント受付とファイルイベント連携の処理の流れを次の図に示します。

図 2-196　ファイルイベント連携機能の処理の流れ（監視フォルダを監視する場合）



（凡例）

取得対象ファイル	：共通フォルダにファイル転送・データ連携ソフトウェアで転送されたファイル。
更新完了通知ファイル	：ユーザの取得対象ファイルの転送が完了したことを，ファイルイベント受付に通知するために作成するファイル。
——→	：制御の流れ
- - - - - - - - - - →	：応答の流れ

図中の 1.～7.について説明します。

1. ファイル監視機能が監視フォルダを監視します。
2. クライアントから共通フォルダ内に取得対象ファイルを転送します。
3. 取得対象ファイルの転送完了後に，監視フォルダ内に取得対象ファイルと同名の更新完了通知ファイルを作成します。
4. 更新完了通知ファイルの作成を検知します。

5. ファイルイベント受付にファイルイベントの検知を通知します。

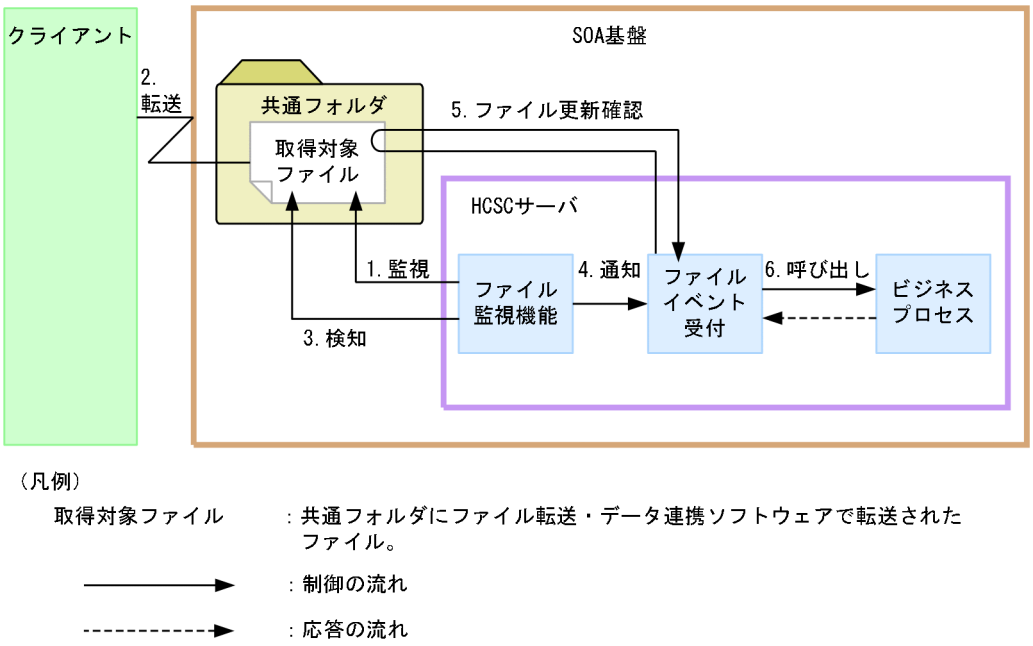
6. 取得対象ファイルの存在を確認します。

7. ビジネスプロセスを呼び出します。

(2)　共通フォルダを監視する場合の処理の流れ

共通フォルダを監視する場合の，ファイルイベント受付とファイルイベント連携機能の処理の流れを次の図に示します。

図 2-197　ファイルイベント連携機能の処理の流れ（共通フォルダを監視する場合）



図中の 1.～6.について説明します。

1. ファイル監視機能が共通フォルダを監視します。

2. クライアントから共通フォルダ内に取得対象ファイルを転送します。

3. 取得対象ファイルの作成を検知します。

4. ファイルイベント受付にファイルイベントの検知を通知します。

5. 取得対象ファイルが更新中かどうか確認します。

6. ビジネスプロセスを呼び出します。

2.20　カスタム受付を使用した場合のシステムとの接続

サービスプラットフォームでは，ユーザが任意の受付を作成できるようにカスタム受付フレームワークを提供しています。このカスタム受付フレームワーク上でユーザが作成した J2EE アプリケーションを動作させることで，任意のプロトコルに対応した受付（カスタム受付）を開発できます。

2.20.1　カスタム受付の構成

カスタム受付は，開発者が作成する受付処理と，カスタム受付フレームワークから構成される J2EE アプリケーションです。

開発者が作成する受付処理の内容と，カスタム受付フレームワークで実行される処理の内容を次に示します。

開発者が作成する受付処理の内容

- サービスリクエスタからのサービス部品呼び出し要求を受け付ける処理
- カスタム受付フレームワークに要求電文を渡す処理
- カスタム受付フレームワークから返ってきた応答電文を，サービスリクエスタに返す処理

カスタム受付フレームワークで実行される処理

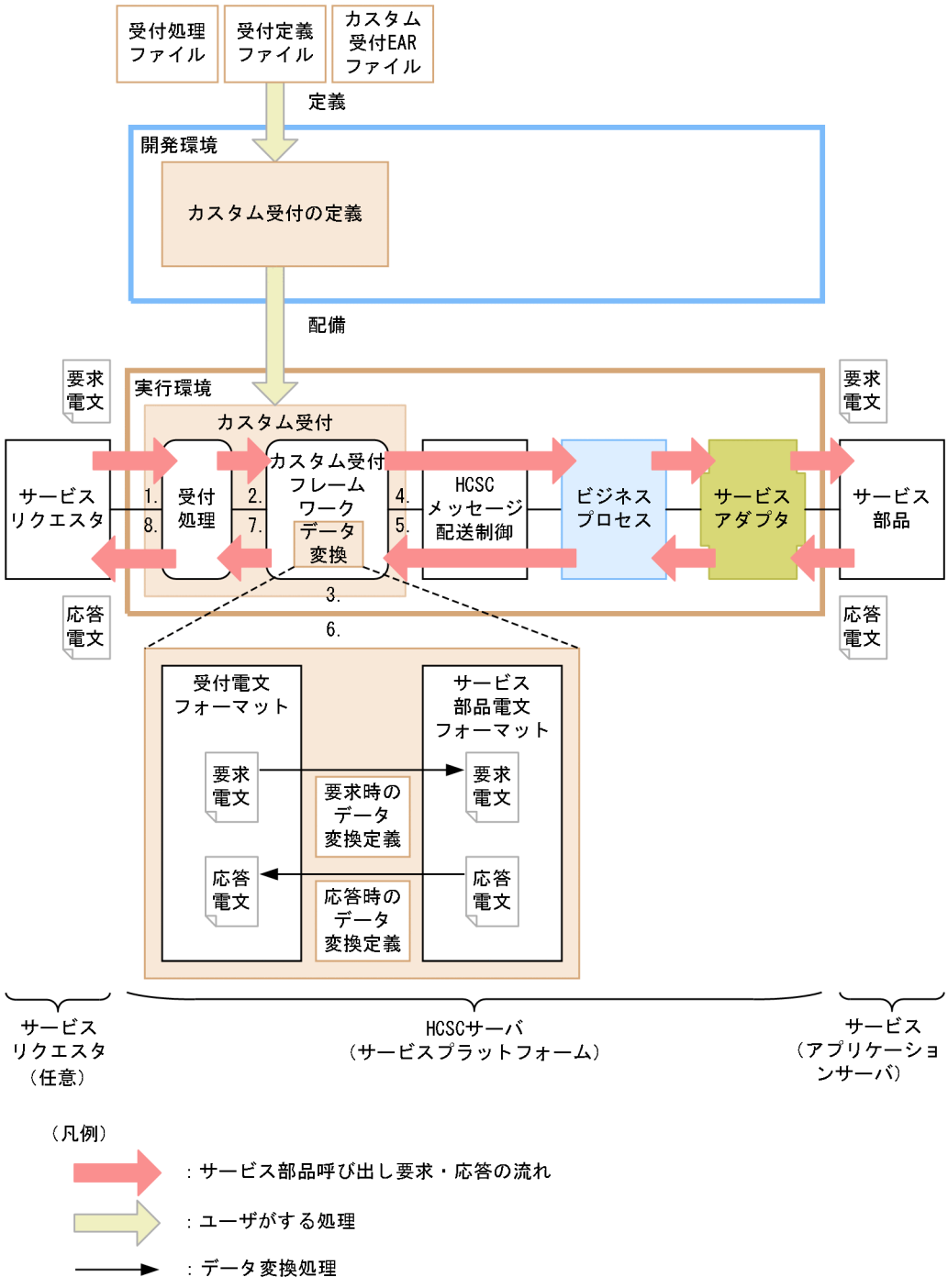
- 受付処理から渡された要求電文を，HCSC メッセージ配送制御に渡す処理
- HCSC メッセージ配送制御から返された応答電文を，受付処理に返す処理
- データ変換定義に従って，要求電文または応答電文のデータ変換を行う処理
- HTTP 受付ルーティング定義ファイルに設定された情報を基に，呼び出すサービスを選択する処理

2.20.2　カスタム受付フレームワークの概要

カスタム受付フレームワークは，カスタム受付を開発および動作させるためのフレームワークです。カスタム受付フレームワークは，開発者が作成したカスタム受付を，サービスプラットフォームの実行環境で受付として動作させるための仕組みを提供しています。

カスタム受付を使用したサービス部品の呼び出しを次の図に示します。

図 2-198　カスタム受付を利用したサービス部品の呼び出し



図中の 1.～8.について説明します。

説明

1. 受付処理は，サービスリクエスタからサービス部品呼び出し要求を受け付けます。
2. 受付処理は，サービスリクエスタから受け取った要求電文を，カスタム受付フレームワークに渡します。
3. カスタム受付フレームワークは，要求電文のデータ変換をします。

4. カスタム受付フレームワークは，HCSC メッセージ配送制御を呼び出します。
5. カスタム受付フレームワークは，HCSC メッセージ配送制御から，応答電文を受け取ります。
6. カスタム受付フレームワークは，応答電文のデータ変換をします。
7. 受付処理は，カスタム受付フレームワークから応答電文を受け取ります。
8. 受付処理は，サービスリクエスタに応答電文を返します。

2.21　汎用カスタムアダプタを使用した場合のシステムとの接続

サービスプラットフォームが提供している次のアダプタでは対応できないプロトコルを持つシステムのサービス部品を呼び出して実行するには，汎用カスタムアダプタの開発が必要です。

- SOAP アダプタ
- SessionBean アダプタ
- MDB（WS-R）アダプタ
- MDB（DB キュー）アダプタ
- DB アダプタ
- TP1 アダプタ
- ファイルアダプタ
- Object Access アダプタ
- Message Queue アダプタ
- FTP アダプタ
- ファイル操作アダプタ
- メールアダプタ
- HTTP アダプタ
- コマンドアダプタ
- SFTP アダプタ
- Kafka アダプタ
- gRPC アダプタ

サービスプラットフォームでは，汎用カスタムアダプタの開発を容易にするためのフレームワークであるカスタムアダプタ開発フレームワークを提供しています。

汎用カスタムアダプタの概要とカスタムアダプタ開発フレームワークの概要を次に示します。

2.21.1　汎用カスタムアダプタの概要

汎用カスタムアダプタは，サービスリクエスタからの要求を受けサービス部品を呼び出します。そのとき，次の 2 つの役割を果たします。

1. データ変換
   汎用カスタムアダプタにデータ変換定義が含まれる場合，事前に定義された情報に基づいて電文のデータ変換をします。
2. プロトコル変換

サービス部品のプロトコルにあわせてプロトコル変換し，サービス部品を呼び出します。

2.21.2　カスタムアダプタ開発フレームワークの概要

汎用カスタムアダプタが行う処理には，データ変換とプロトコル変換があります。このうち，すべての汎用カスタムアダプタに共通の処理であるデータ変換処理をカスタムアダプタ開発フレームワークで提供しています。このため，ユーザはプロトコル変換処理を実装することで汎用カスタムアダプタを開発できます。

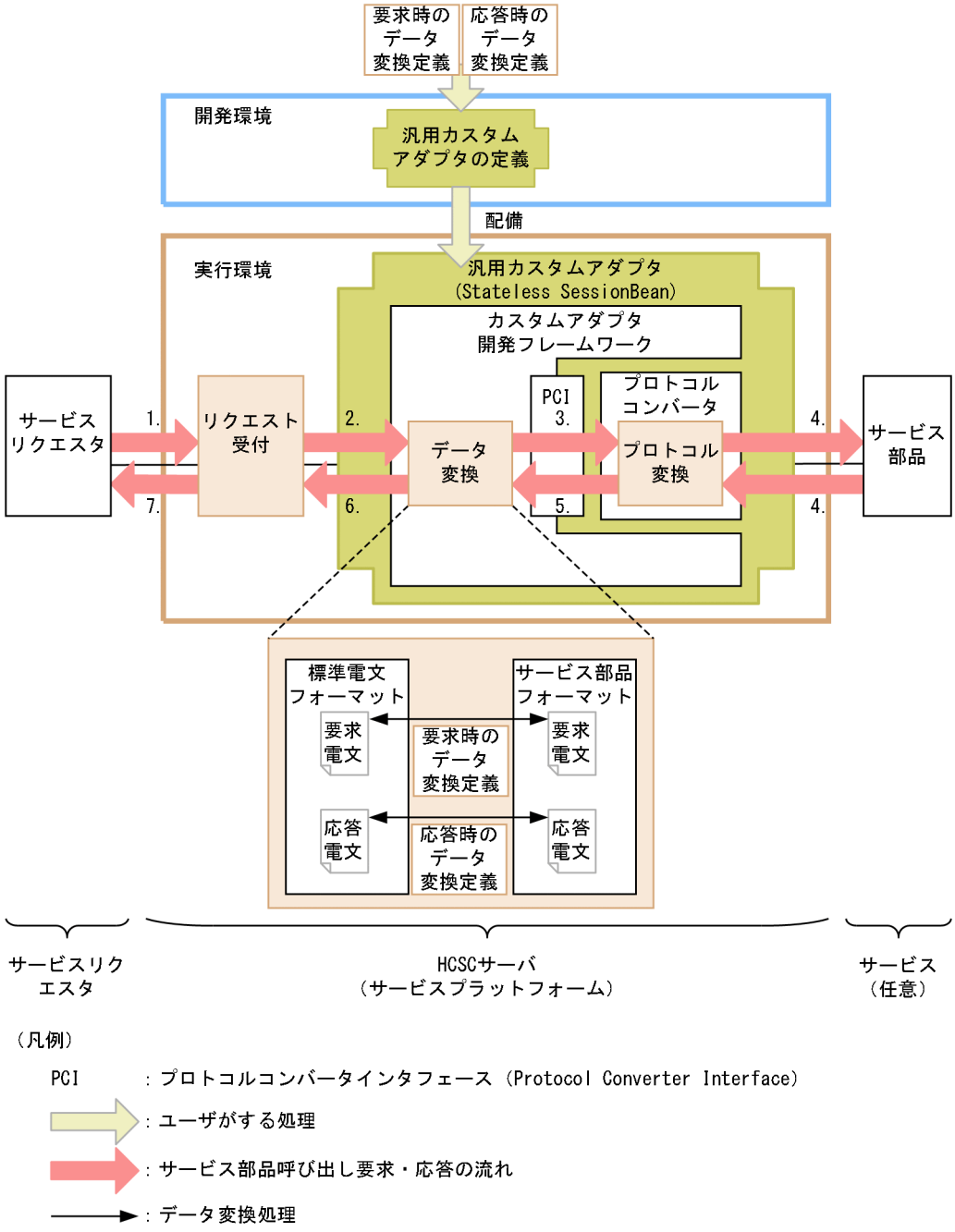
注意事項

データ変換の定義は，別途必要となります。

汎用カスタムアダプタは，カスタムアダプタ開発フレームワークの実装である Stateless SessionBean と，プロトコルコンバータで構成されます。プロトコルコンバータとは，プロトコル変換処理を実装する Java クラスのことです。

汎用カスタムアダプタを使用したサービス部品の呼び出しを次の図に示します。

図 2-199　汎用カスタムアダプタを使用したサービス部品の呼び出し（すべて同期通信の場合）



図中の 1.～7.について説明します。

説明

1. HCSC サーバは，汎用カスタムアダプタ経由で呼び出すサービス部品へのリクエストを受け取ります。
2. HCSC サーバは，EJB リモートインターフェースを使用して，カスタムアダプタ開発フレームワークで実装している汎用カスタムアダプタを呼び出します。
3. カスタムアダプタ開発フレームワークは，事前に定義された情報に基づいてデータ変換処理を実行し，プロトコルコンバータインターフェースを使用してプロトコルコンバータを呼び出します。

4. プロトコルコンバータは，サービス部品のプロトコルに応じた送受信処理を行います。
5. サービス部品の呼び出し結果をカスタムアダプタ開発フレームワークに返します。
6. カスタムアダプタ開発フレームワークは，事前に定義された情報に基づいてデータ変換処理を実行し，HCSC サーバに応答を返します。
7. サービスリクエスタに応答を返します。

カスタムアダプタ開発フレームワークによる汎用カスタムアダプタの開発については，マニュアル「サービスプラットフォーム 開発ガイド 受付・アダプタ定義編」の「付録 B　カスタムアダプタ開発フレームワーク」を参照してください。

2.22　TP1 アダプタおよび Message Queue アダプタのオペレーション動的変更機能

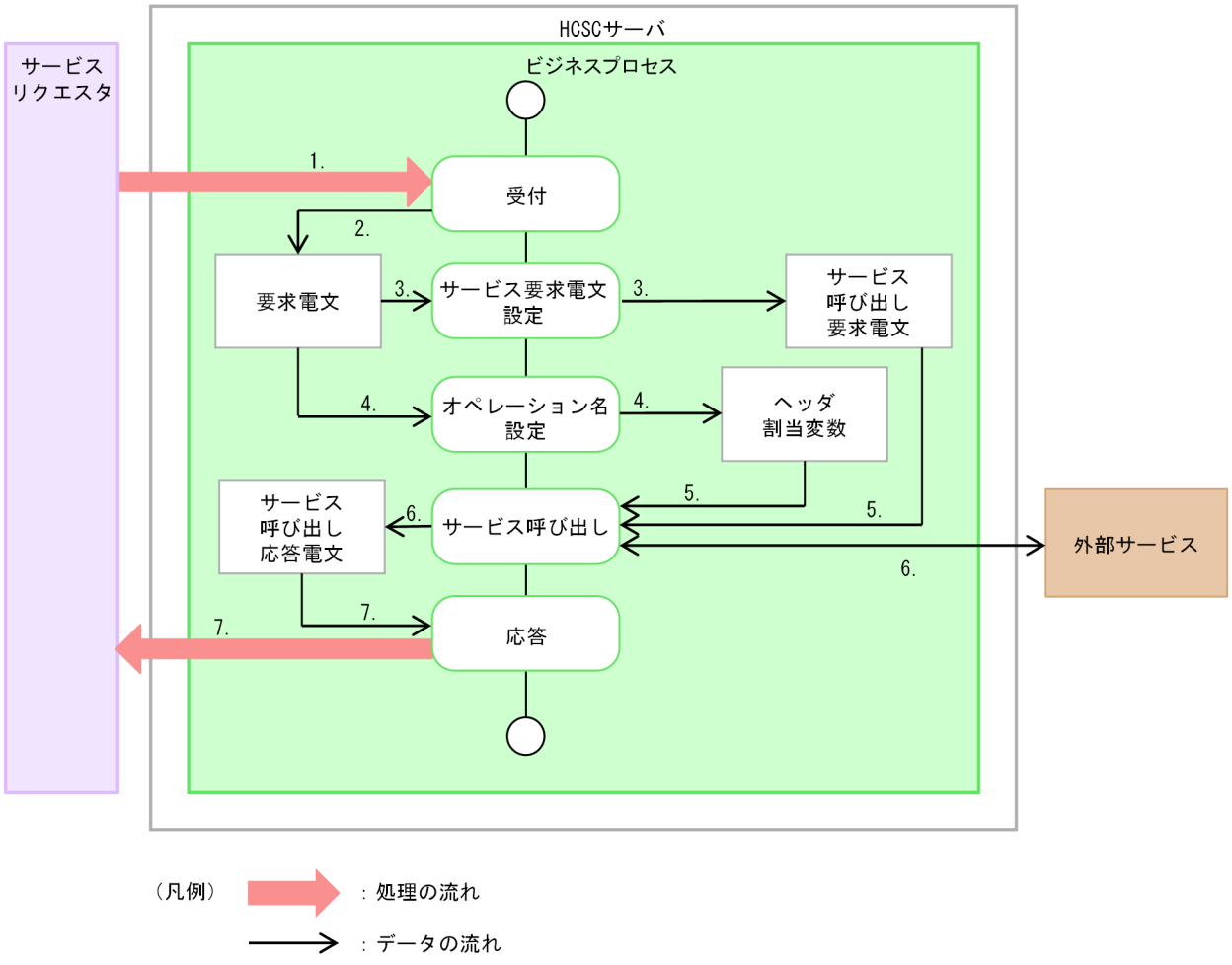
TP1 アダプタ，または Message Queue アダプタの実行オペレーションをビジネスプロセス内で動的に変更できます。

実行オペレーションを動的変更することで，ビジネスプロセス内に配置するサービスアダプタ数や，ビジネスプロセス数を削減できます。

2.22.1　オペレーション動的変更機能を利用した処理の流れ

オペレーション動的変更機能を利用した処理の流れを次に示します。

図 2-200　オペレーション動的変更機能を利用した処理の流れ



1. サービスリクエスタがビジネスプロセスにサービス部品の実行を要求します。
2. 受付アクティビティは，受け取ったサービスリクエスタからの要求電文を要求電文の変数に格納します。

3. 要求電文変数から，サービスアダプタ呼び出しに使用するサービス呼び出し要求電文（ボディ割当変数）を作成します。
4. 要求電文変数から，オペレーション名を決定し，データ変換アクティビティなどで，ヘッダ割当変数に格納します。
5. サービスアダプタを呼び出します。
   サービス呼び出しでは，要求電文のヘッダ割当変数として，4.で定義したヘッダ割当変数を指定します。
6. サービスアダプタは，ヘッダ割当変数に指定されたオペレーション名に従い，サービスを実行します。また，応答をサービス呼び出し応答電文の変数に格納します。
7. 応答アクティビティは，サービス呼び出し応答電文をサービスリクエスタに返します。

2.22.2　ヘッダ割当変数の定義

ヘッダ変数のスキーマを使用して，ヘッダ割当変数を定義します。ヘッダ変数のスキーマは「<サービスプラットフォームのインストールディレクトリ>¥CSC¥schema¥common¥dynamic_operation_header.xsd」に格納されています。

(1)　ヘッダ変数のスキーマの記述形式

ヘッダ変数のスキーマの記述形式を次に示します。

ヘッダ変数のスキーマの記述形式

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- All Rights Reserved. Copyright (C) 2024. Hitachi, Ltd. -->
<xsd:schema elementFormDefault="qualified" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:dyop="http://www.hitachi.co.jp/soft/xml/cosminexus/csc/dynamicoperation/header"
  targetNamespace="http://www.hitachi.co.jp/soft/xml/cosminexus/csc/dynamicoperation/header"
>

  <xsd:element name="DynamicOperationHeader" type="dyop:DynamicOperation"/>

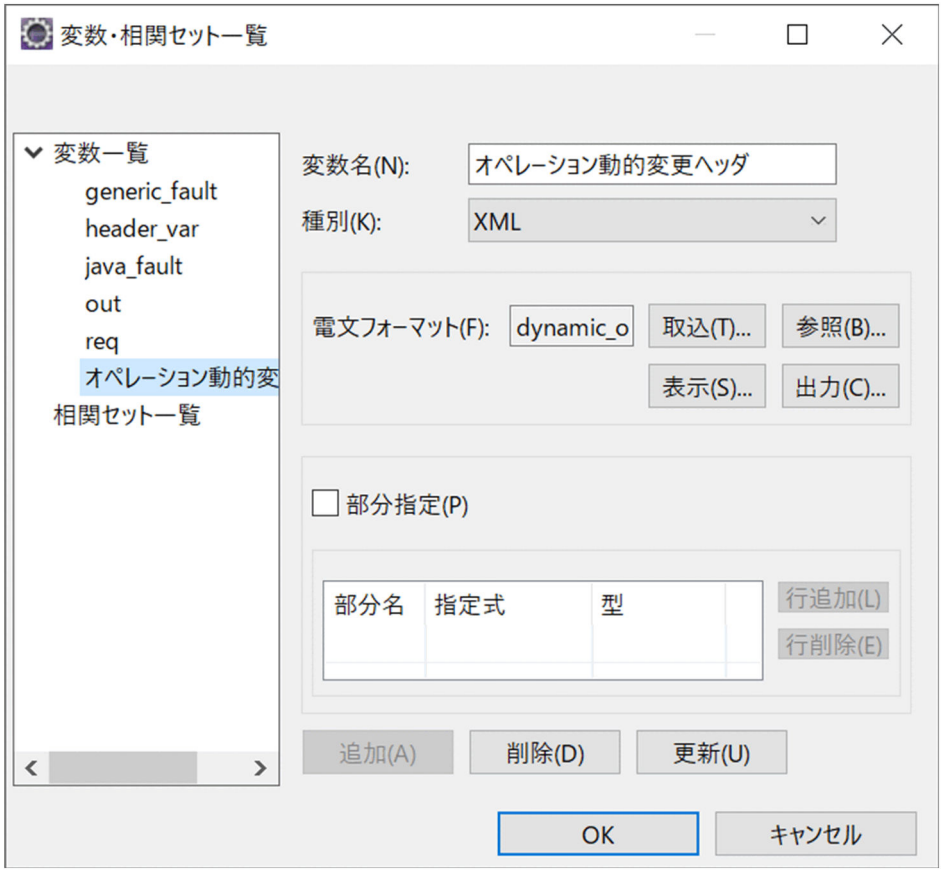
  <xsd:complexType name="DynamicOperation">
    <xsd:sequence>
      <xsd:element name="cscDynamicOperationName" minOccurs="0" maxOccurs="1" type="xsd:stri
ng"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```

(2)　ヘッダ割当変数の設定

「(1) ヘッダ変数のスキーマの記述形式」で作成したヘッダ変数のスキーマ（dynamic_operation_header.xsd）を使用して，ヘッダ割当変数を定義します。

ヘッダ割当変数の定義例を次に示します。

図 2-201　ヘッダ割当変数の定義例



定義したヘッダ割当変数の cscDynamicOperationName 要素に値をセットします。

cscDynamicOperationName 要素に要求電文の location 要素をセットした例を次の図に示します。

図 2-202　cscDynamicOperationName 要素に要求電文の location 要素をセットした例



(3)　ヘッダ割当変数の制限事項

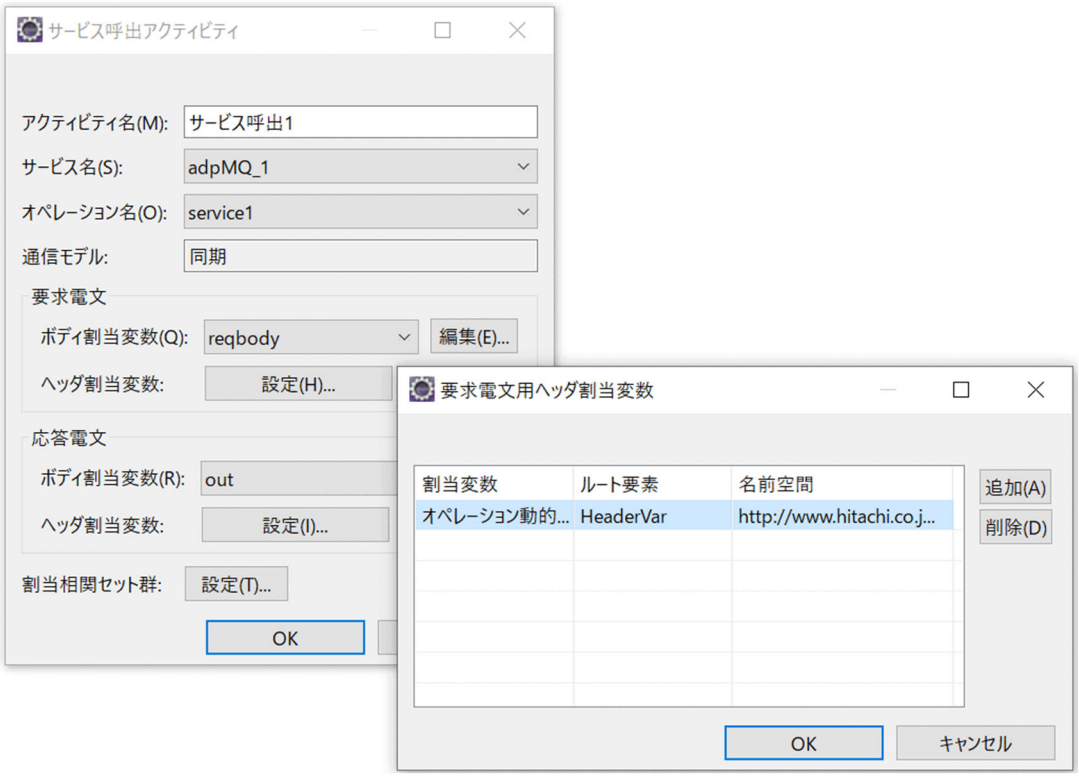
ヘッダ割当変数に指定する，ヘッダ変数のスキーマ（dynamic_operation_header.xsd）から作成した変数は，1 つだけです。ヘッダ変数のスキーマ（dynamic_operation_header.xsd）から作成した変数を複数指定した場合の動作は保証されません。

2.22.3　オペレーション動的変更機能を使ったサービス呼び出し

オペレーション動的変更機能を使ったサービス呼び出しでは，要求電文のヘッダ割当変数として「2.22.2 ヘッダ割当変数の定義」で定義したヘッダ割当変数を，サービス呼出アクティビティに指定します。

サービス呼出アクティビティのヘッダ割当変数の設定例を次に示します。

図 2-203　サービス呼出アクティビティのヘッダ割当変数の設定例



3 ビジネスプロセスの機能

この章では，ビジネスプロセスの概要，ビジネスプロセスの仕組み，ビジネスプロセスの永続化・非永続化，ビジネスプロセスのトランザクション，ビジネスプロセスの再実行，およびビジネスプロセスの待機処理について説明します。また，ビジネスプロセスと関係が深いプロセスインスタンス，アクティビティ，変数，および相関セットについても説明します。

3.1　ビジネスプロセスの概要

ビジネスプロセスとは，複数のサービス部品を実行する処理の順番や条件などを定義して，一連の業務の流れとして定義したものです。

ビジネスプロセスは，サービスプラットフォームの開発環境で処理フローを定義します。処理フローは，ビジネスプロセス定義画面で定義します。画面上にアクティビティと呼ばれる要素を配置し，コネクションで連結して定義します。

ビジネスプロセス定義画面の詳細については，マニュアル「サービスプラットフォーム リファレンス」の「1.2.3　ビジネスプロセス定義画面」を参照してください。

ここでは，ビジネスプロセスを構成する主な次の 9 つの要素について説明します。

- 受付・応答
- サービス呼び出し
- Java 呼び出し
- データ変換
- 代入
- 繰り返し
- 分岐
- 並列処理
- 待機処理

3.1.1　受付・応答

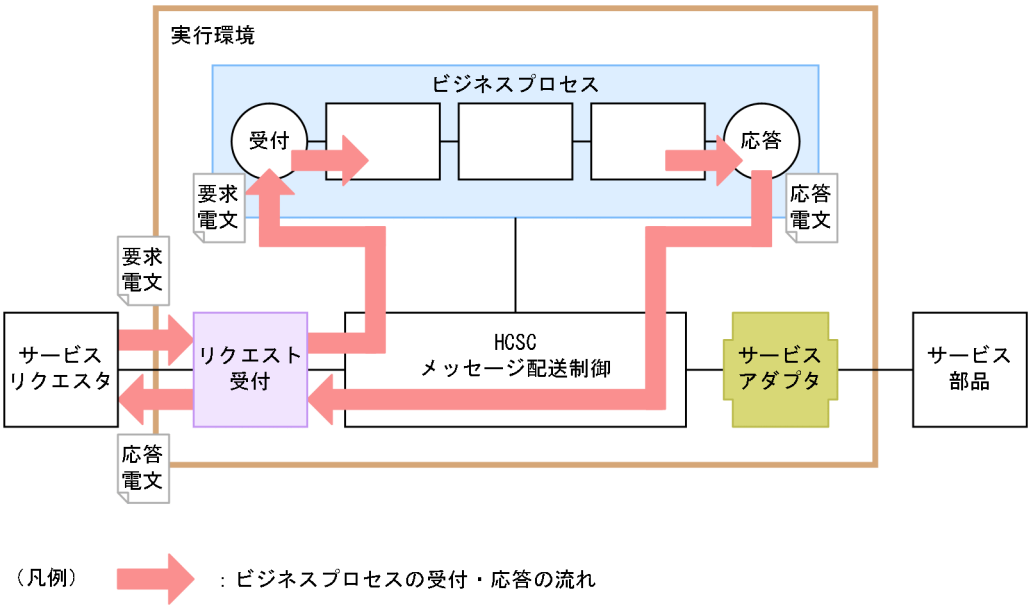
(1)　機能

ビジネスプロセスの受付でサービスリクエスタからの要求を受け付け，応答でビジネスプロセスからの応答をサービスリクエスタに返します。

サービスリクエスタからの要求を受け付けるリクエスト受付には，標準受付とユーザ定義受付の両方を使用できます。

ビジネスプロセスの受付と応答の流れを次の図に示します。

図 3-1　ビジネスプロセスの受付と応答の流れ



(2)　使用するアクティビティ

受付には受付アクティビティを使用し，応答には応答アクティビティを使用します。

受付には，次のどちらかの通信モデルが定義できます。

- 同期で応答電文が返るビジネスプロセス
- 非同期で応答電文が返らないビジネスプロセス

3.1.2　サービス呼び出し

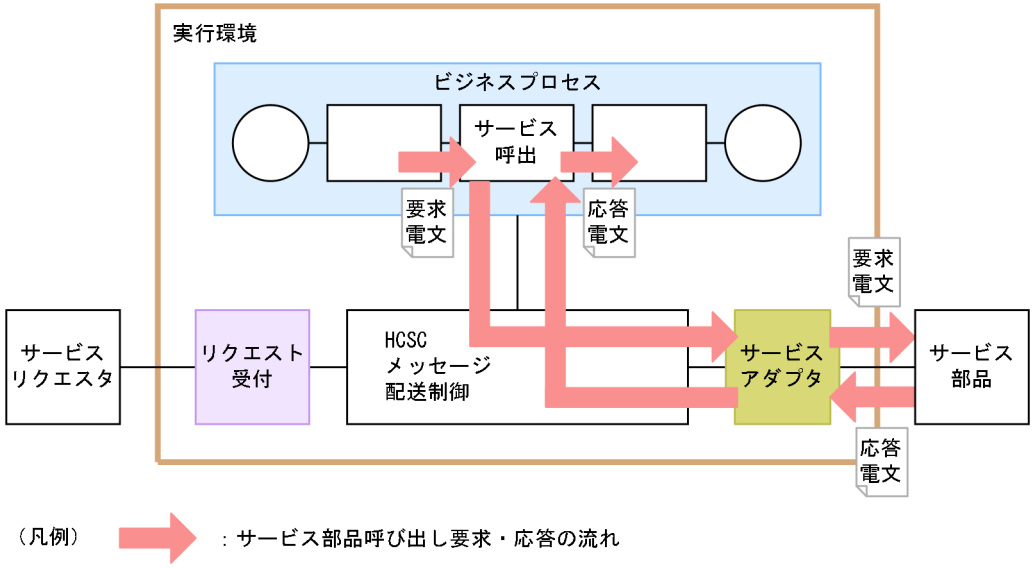
(1)　機能

サービス呼び出しでは，サービスプラットフォームの開発環境で定義した各種サービスアダプタやビジネスプロセスなどのサービス部品を実行することができます。

サービスアダプタやビジネスプロセスの通信モデルによって，同期で応答電文が返る場合と，非同期で応答電文が返らない場合のそれぞれのサービス部品を呼ぶことができます。

サービス呼び出しでサービスアダプタを実行する場合の流れを次の図に示します。

図 3-2　サービス呼び出しでサービスアダプタを実行する場合の流れ



(2)　使用するアクティビティ

サービス呼び出しには，サービス呼出アクティビティを使用します。

3.1.3　Java 呼び出し

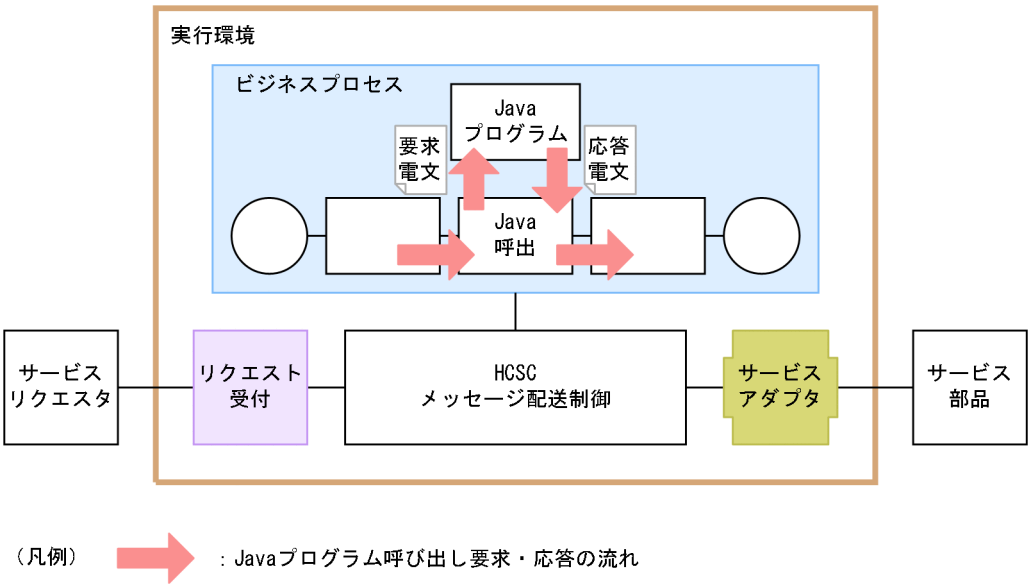
(1)　機能

Java 呼び出しでは，サービスプラットフォームが提供する専用のインターフェースを実装した Java クラスを実行することができます。

サービスプラットフォームが提供する機能だけでビジネスプロセスが定義できないような場合に，ユーザが作成する Java プログラムを使用して，XML 電文の変換や判定などの処理を実行できます。

Java 呼び出しで Java プログラムを実行する場合の流れを次の図に示します。

図 3-3　Java 呼び出しで Java プログラムを実行する場合の流れ



(2)　使用するアクティビティ

Java 呼び出しには，Java 呼出アクティビティを使用します。

Java 呼出アクティビティの引数に定義した電文やビジネスプロセス上の変数で Java プログラムを実行します。

Java プログラムの戻り値は，Java 呼出アクティビティの戻り値に定義した電文やビジネスプロセス上の変数になります。

なお，実行する Java プログラムは，サービスプラットフォームの開発環境の Java 呼出アクティビティを使用し，Java エディタ上で作成する必要があります。

3.1.4　データ変換

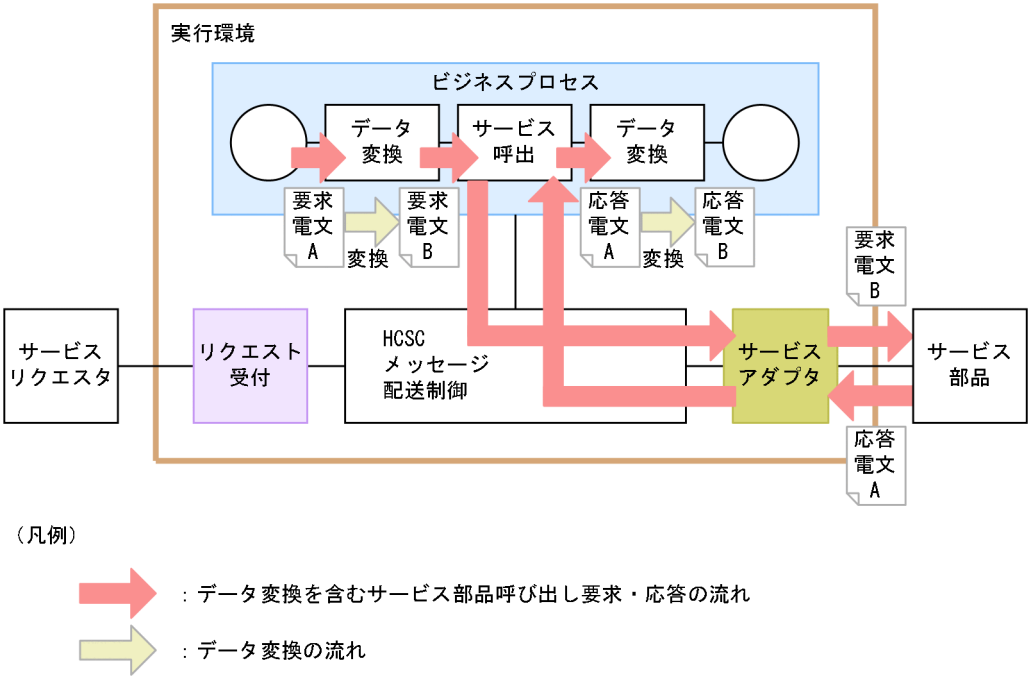
(1)　機能

データ変換では，ビジネスプロセス上を流れる電文形式を異なる電文形式に変換したり，値を加工したりすることができます。また，複数の電文を組み合わせて 1 つの電文を作ることもできます。

データ変換は，サービス部品を呼び出すときに，サービス部品の要求電文形式に合わせた電文を作成する必要がある場合や，サービス部品からの応答電文を異なる電文に変換したい場合などに使用します。

ビジネスプロセスでのデータ変換の流れを次の図に示します。

図 3-4　ビジネスプロセスでのデータ変換の流れ

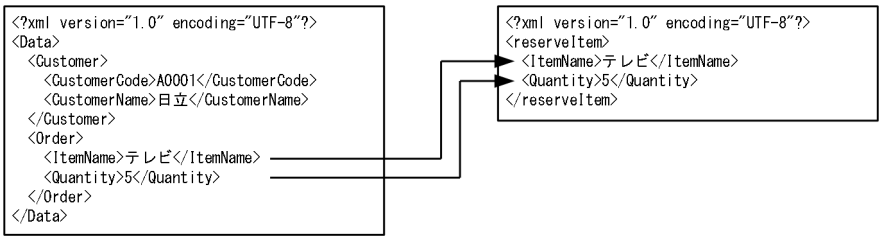


(2)　使用するアクティビティ

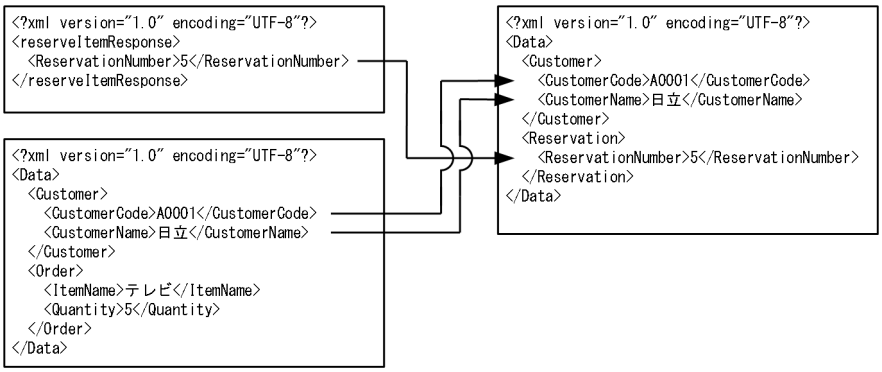
データ変換には，データ変換アクティビティを使用します。

データ変換の例を次に示します。

(例 1) 1 対 1 でのデータ変換



(例 2) N 対 1 でのデータ変換



データ変換でできる処理を次の表に示します。

表 3-1　データ変換でできる処理一覧

処理	説明
文字列連結	複数の文字列を連結します。
部分文字列取得	1 つの文字列から一部分の文字列を取り出します。
文字数取得	文字列の文字数をマッピング先に代入します。
文字列調査	文字列に指定した文字列が含まれているか，または指定した文字列で開始されているかを調査し，マッピング先に論理値を代入します。
空白除去	文字列の先頭と末尾の空白を除去します。また，文字列間の連続した空白を 1 つの半角スペースに置換します。
数値フォーマット変換	数値をフォーマット変換します。小数点文字および桁区切り文字を変更することができます。
数値演算	数値を演算（+，-，*，div，mod）します。
端数処理	小数点以下の端数を処理（四捨五入，切り捨て，および切り上げ）します。
総和	複数のノードセットの，ノードの数値を合計します。
否定演算	論理否定演算を実行します。
論理演算	論理演算（AND，OR，XOR）を実行します。
シフト演算	シフト演算を実行します。
ノード数取得	ノード数をマッピング先に代入します。
ノード名取得	ノード名をマッピング先に代入します。
ノード調査	指定した変換元ノードが存在するかを調査し，マッピング先にその論理値を代入します。
繰り返し	繰り返しを対応づけます。複数回出現する変換先ノードのインスタンスをソートすることもできます。
選択	条件によって異なる値を出力します。
値置換え	変換テーブルの指定に基づいて，マッピング元の値を変換し，マッピング先に代入します。
基数変換	基数変換を実行します。
カスタム	ユーザが作成した Java プログラムで値の加工およびノードの作成ができます。
定数	指定した値をマッピング先に代入します。
XPath	XPath 式を指定します。

3.1.5　代入

(1)　機能

代入では，ビジネスプロセス上を流れる電文形式を異なる電文形式に変換することができます。

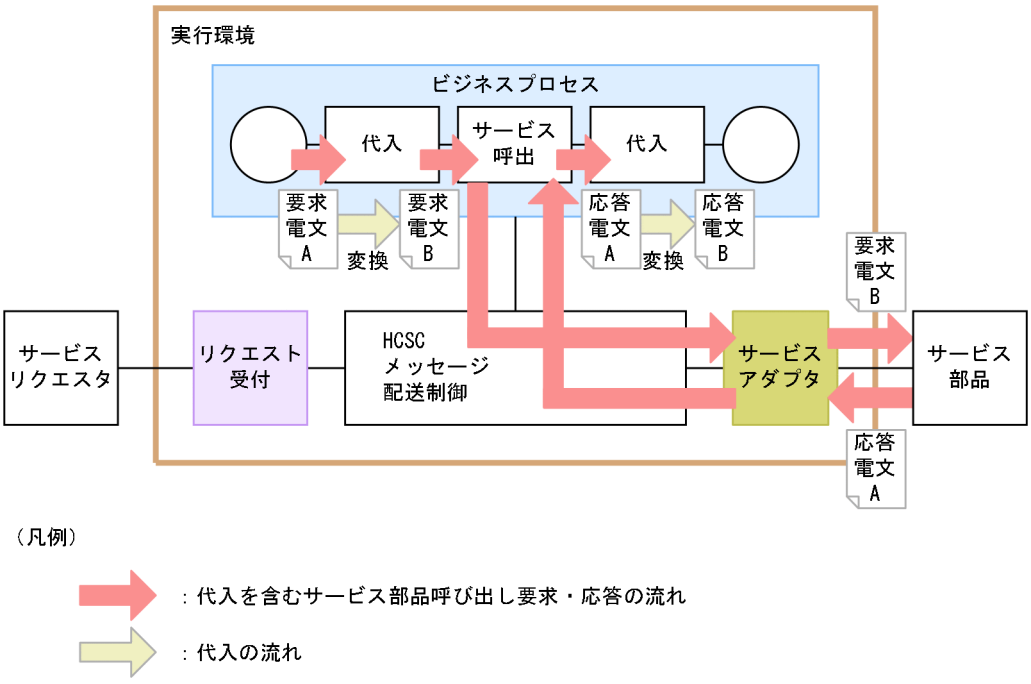
代入の主な機能を次に示します。

- XML 電文を異なる名称の XML 電文にコピーする。
- XML 電文の一部分(部分木)をほかの XML 電文にコピー（挿入）する。
- XML 以外の boolean，numeric，string 等の変数をほかの変数にコピーする。

代入は，データ変換と同じような機能を持ちますが，XML 電文中の個々の値を対象とするのではなく，XML 全体または一部をそのままコピーしたい場合に適しています。

ビジネスプロセスでの代入の流れを次の図に示します。

図 3-5　ビジネスプロセスでの代入の流れ

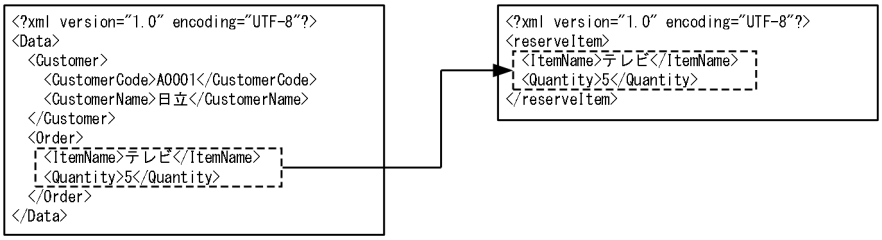


(2)　使用するアクティビティ

代入には，代入アクティビティを使用します。

代入の例を次に示します。

（例）部分木のコピー



3.1.6 繰り返し

(1) 機能

繰り返しでは，ビジネスプロセスの中で，指定した条件で処理を繰り返すこと（ループ処理）ができます。

繰り返し処理の中には，1 つまたは複数のアクティビティを配置します。

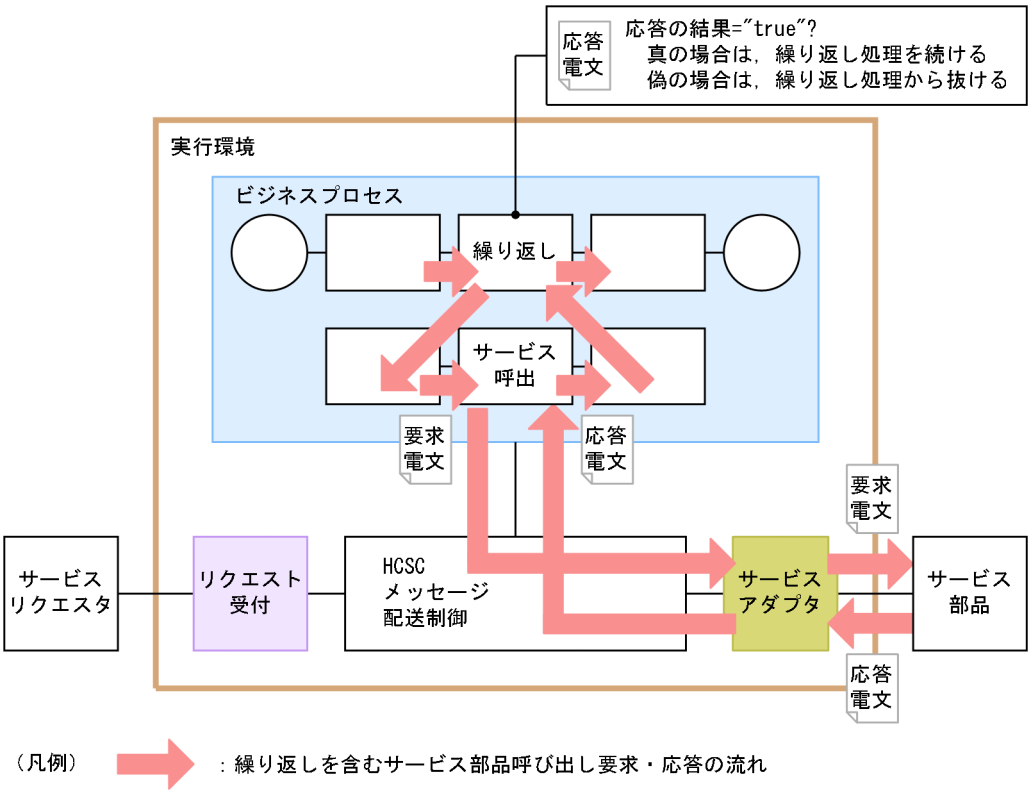
繰り返し処理の方式には次の 2 つがあります。

- 繰り返し条件設定方式
  繰り返しの条件を設定し，繰り返し条件を満たしている間は処理を繰り返します。
- 繰り返しリスト指定方式
  繰り返し要素のリストを抽出し，リストに要素がある間は処理を繰り返します。

繰り返し条件設定方式の場合，繰り返しの判定は，前判定になります。判定結果が真の場合は繰り返しを続け，偽の場合は繰り返しから抜けて次の処理へ遷移します。

繰り返し条件設定方式の場合の，ビジネスプロセスでの繰り返しの流れを次の図に示します。

図 3-6　ビジネスプロセスでの繰り返し（繰り返し条件設定方式の場合）



繰り返しリスト指定方式の場合，繰り返し要素を定義した変数を基に繰り返しリストが生成されます。繰り返しリストに含まれるノードの数の分だけ，処理が繰り返し実行されます。

繰り返しリスト指定方式の場合の，ビジネスプロセスでの繰り返しの流れについては，マニュアル「サービスプラットフォーム 開発ガイド 基本開発編」の「5.6.16　繰り返しアクティビティ」を参照してください。

(2)　使用するアクティビティ

繰り返しアクティビティを使用します。

繰り返しの判定に用いる条件式，または繰り返し要素を定義した変数の要素のパスは，XML 電文などの変数に対して XPath で指定します。

繰り返し条件設定方式と繰り返しリスト指定方式のどちらの方式を使用するかは，繰り返しアクティビティダイアログで設定します。

繰り返し処理で無限ループに陥ることを避けるために，繰り返しの回数の最大値を指定することができます。ただし，繰り返し回数の最大値に達した場合，ビジネスプロセスはエラーになります。

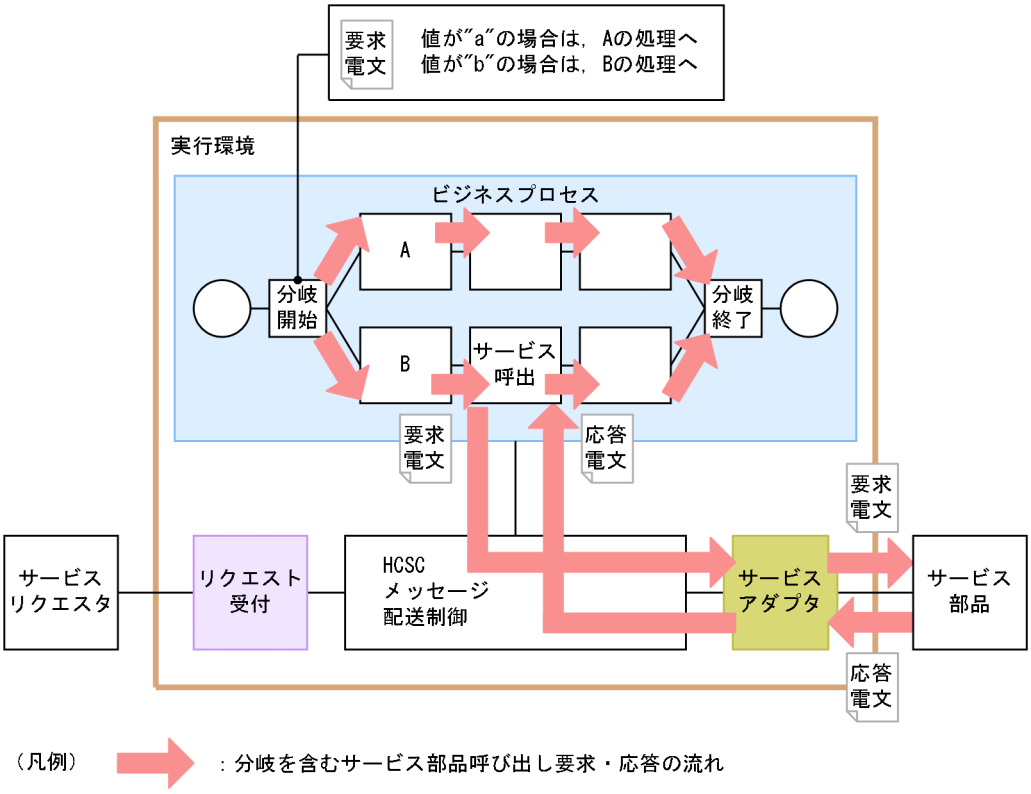
3.1.7　分岐

(1)　機能

分岐では，ビジネスプロセスの中で，条件の判定結果によって処理を分岐して実行することができます。分岐先の処理の中には，1 つまたは複数のアクティビティを配置します。

ビジネスプロセスでの分岐の流れを次の図に示します。

図 3-7　ビジネスプロセスでの分岐



(2)　使用するアクティビティ

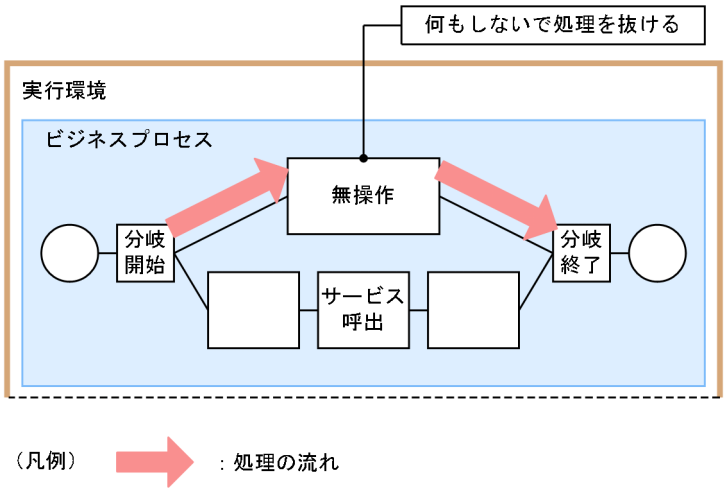
分岐の開始個所には分岐開始アクティビティを使用し，終了個所には分岐終了アクティビティを使用します。分岐先は，複数（2 つ以上）設定できます。

分岐の判定に用いる条件式は，XML 電文などの変数に対して XPath で指定します。この条件式を基に，それぞれの遷移先のアクティビティを指定します。

なお，分岐開始アクティビティと，対応する分岐終了アクティビティの間には，1 つ以上のアクティビティを設定する必要があります。そのため，片方の分岐先で処理を何もしないで次へ遷移するようにしたい場合は，無操作アクティビティを配置します。

片方の分岐先で処理を何もしないで次へ遷移するようにしたい場合のアクティビティの配置例を次の図に示します。

図 3-8　片方の分岐先で処理を何もしないで次へ遷移するようしたい場合のアクティビティの配置例



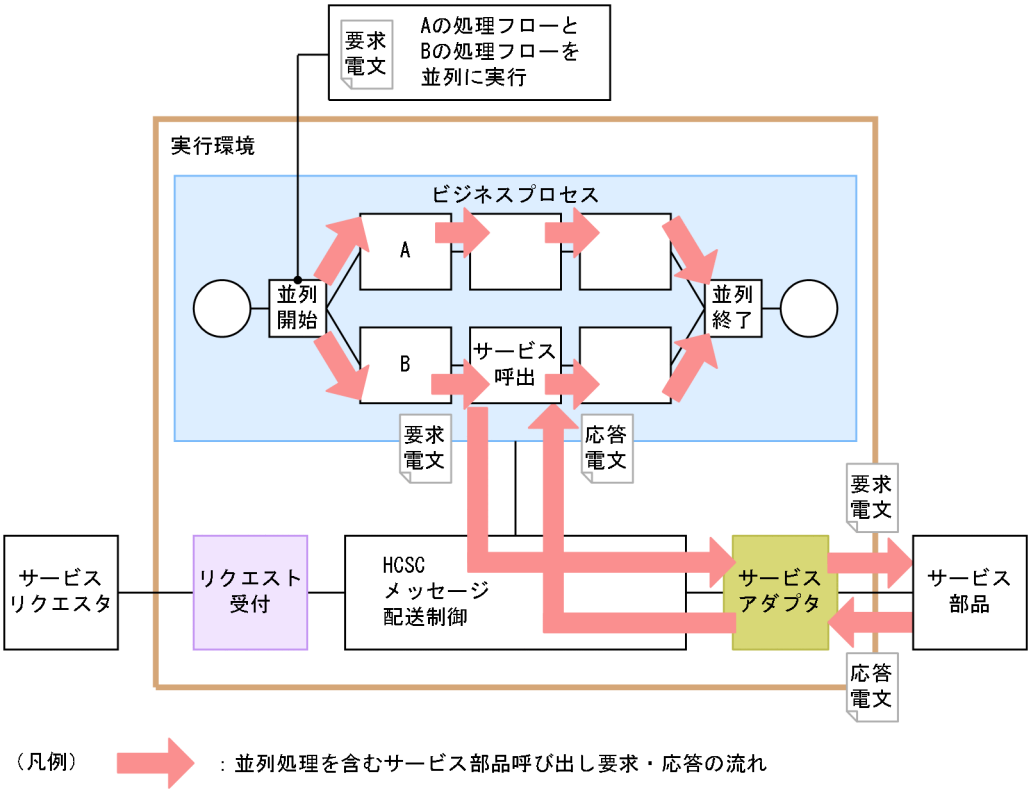
3.1.8　並列処理

(1)　機能

並列処理では，ビジネスプロセスの中で，処理フローを 2 つ以上に分け，フローごとに順次実行することができます。

ビジネスプロセスでの並列実行の流れを次の図に示します。

図 3-9　ビジネスプロセスでの並列実行の流れ



(2)　使用するアクティビティ

並列処理の開始個所には並列処理開始アクティビティを，終了個所には並列処理終了アクティビティを使用します。

また，リンクコネクションを用いることで，個々の並行に実行する処理フローの順番を制御することができます。

なお，並列処理は個々の処理フローを順次実行します。複数の処理フローを同時に実行することはできません。

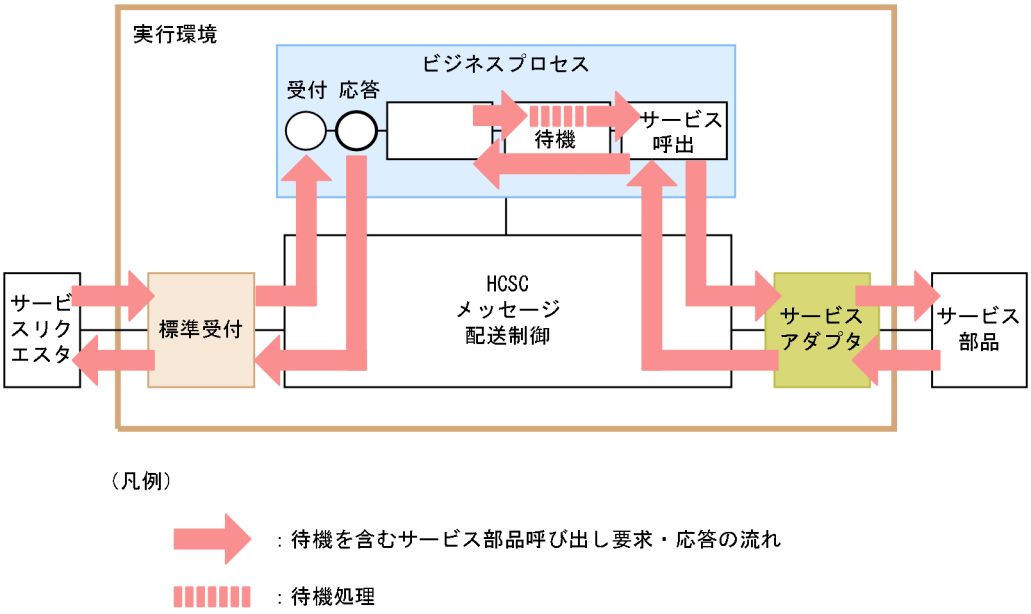
3.1.9　待機処理

(1)　機能

待機処理では，指定した期間の間，または指定した時刻まで，ビジネスプロセスの処理を待機することができます。

ビジネスプロセスでの待機処理の流れを次の図に示します。

図 3-10　ビジネスプロセスでの待機処理の流れ



(2)　使用するアクティビティ

待機処理には，待機アクティビティを使用します。

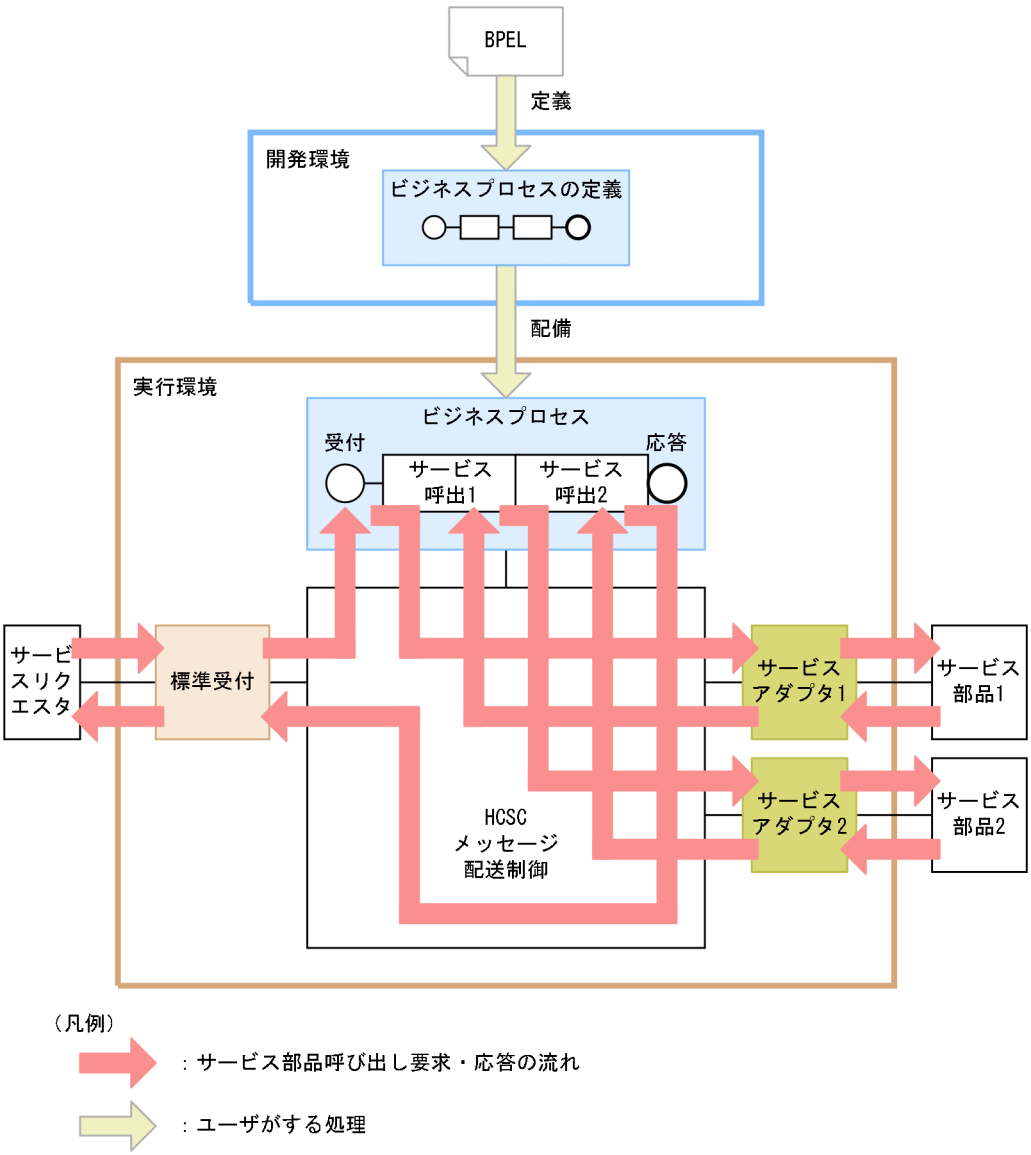
待機する時間の間隔または期限を XPath 式で指定します。ビジネスプロセスの処理の中で XPath 式を評価し，その結果で得た文字列を待機時間として使用します。

3.2　ビジネスプロセスの仕組み

3.2.1　ビジネスプロセスの処理の流れ

サービスリクエスタからビジネスプロセスを呼び出すと，呼び出されたビジネスプロセスで定義された要素やアクティビティを一つ一つ実行することで，プロセスを実行します。ビジネスプロセスの定義は，開発環境で定義します。ビジネスプロセスの処理の流れを次の図に示します。

図 3-11　ビジネスプロセスの処理の流れ



3.2.2　ビジネスプロセスの通信モデル

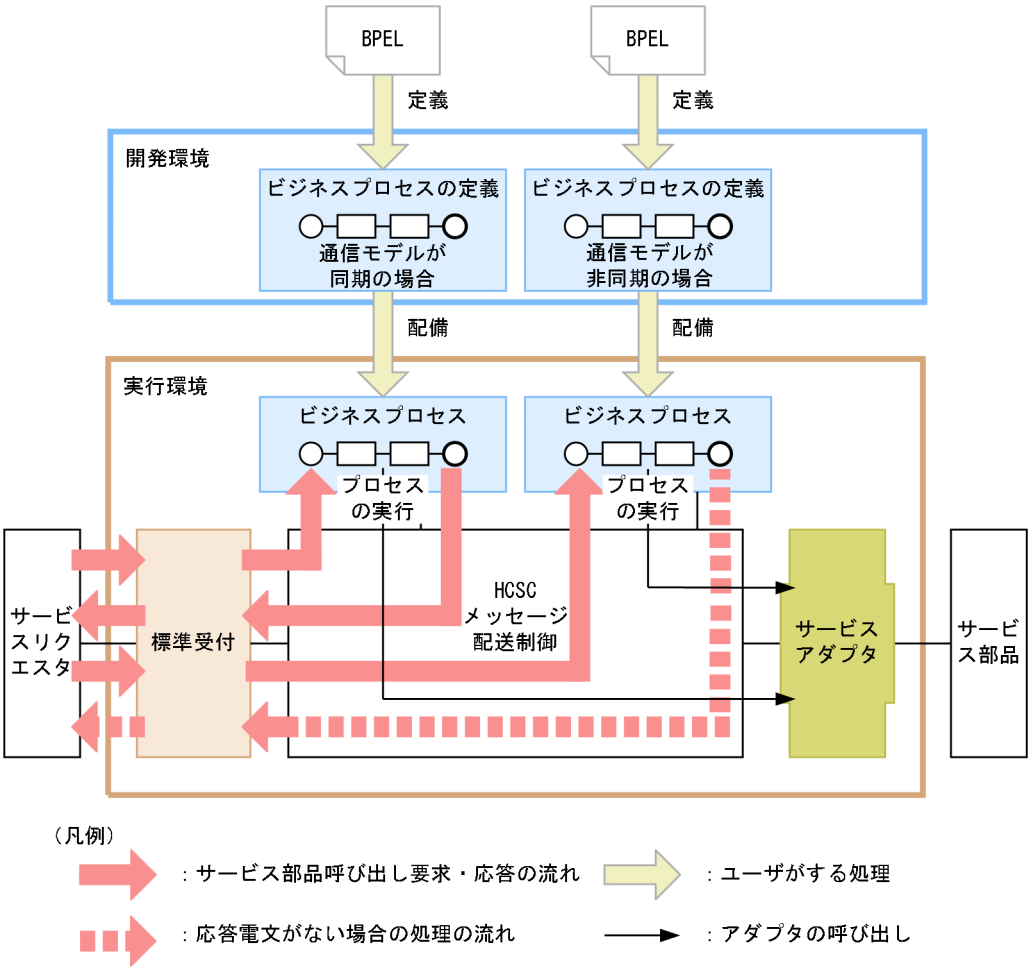
ビジネスプロセスは，同期と非同期の 2 種類の通信モデルがあります。

通信モデルが同期の場合，ビジネスプロセスを呼び出したサービスリクエスタに対して，応答電文が返ります（応答電文は応答アクティビティに定義した電文フォーマットです）。

通信モデルが非同期の場合，ビジネスプロセスを呼び出したサービスリクエスタに応答電文が返りません（処理だけがリターンします）。

通信モデルは，開発環境でビジネスプロセス定義するときに，受付アクティビティのダイアログで定義します。ビジネスプロセスでの通信モデルの違いを次の図に示します。

図 3-12　ビジネスプロセスでの通信モデルの違い



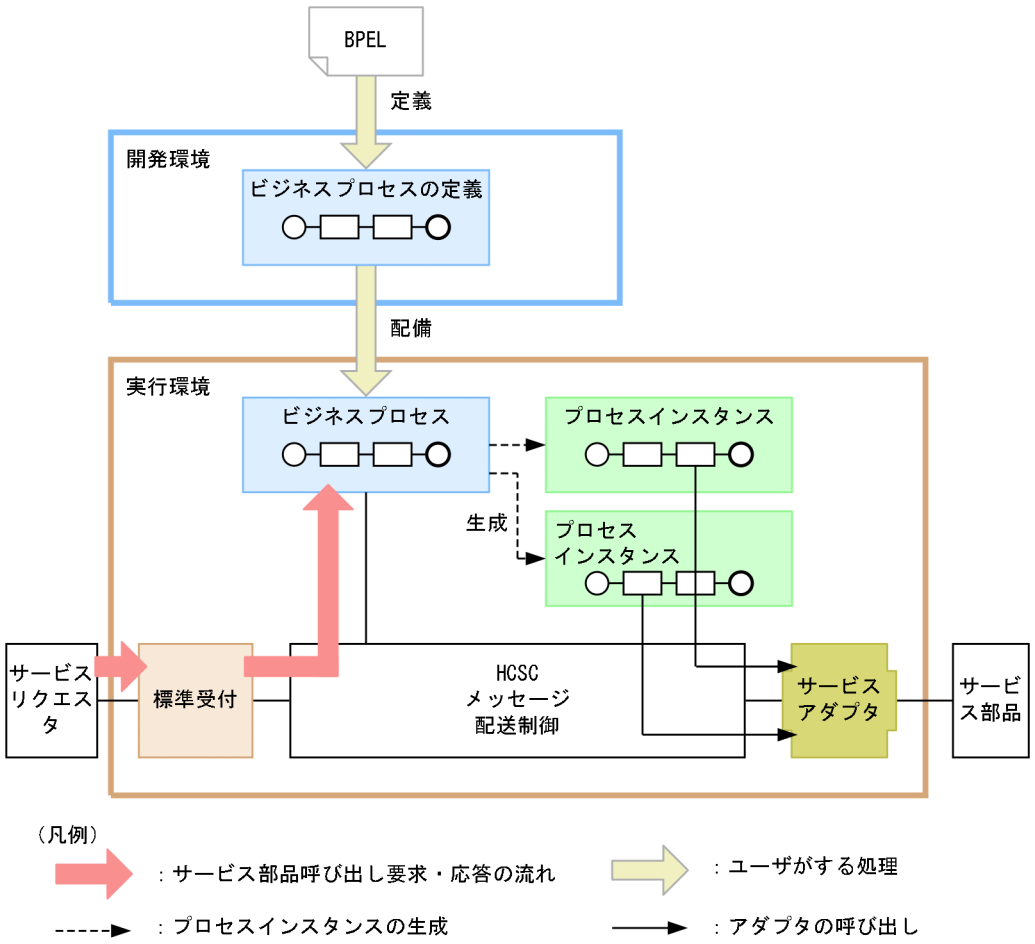
受付アクティビティダイアログの詳細については，マニュアル「サービスプラットフォーム リファレンス」の「1.4.7　受付アクティビティダイアログ」を参照してください。

3.2.3　プロセスインスタンス

ビジネスプロセスが最初に要求を受け付けるときには，プロセスインスタンスを生成します。プロセスインスタンスを生成するには，開発環境でビジネスプロセスを定義するとき，最初に要求を受け付ける受付アクティビティを定義する受付アクティビティダイアログでインスタンス生成を「yes」に設定します。

プロセスインスタンスを生成するビジネスプロセスの場合，ビジネスプロセスが要求を受け付けるたびにプロセスインスタンスを生成します。そのため，複数の要求が同時に実行された場合でも，個々に独立してプロセスを実行します。ビジネスプロセスでのプロセスインスタンスの仕組みについて次の図に示します。

図 3-13　ビジネスプロセスでのプロセスインスタンスの仕組み



受付アクティビティダイアログの詳細については，マニュアル「サービスプラットフォーム リファレンス」の「1.4.7　受付アクティビティダイアログ」を参照してください。

プロセスインスタンスの属性情報を次の表に示します。

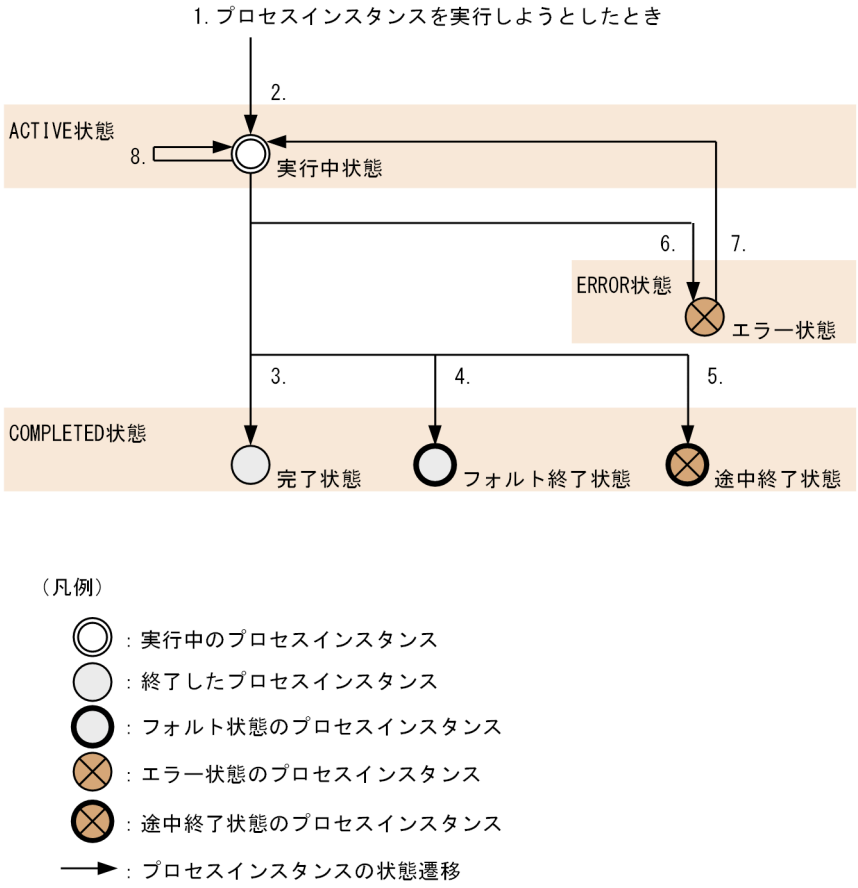
表 3-2　プロセスインスタンスの属性情報

属性	説明
識別子	個々のプロセスインスタンスを特定するための識別子です。同一のビジネスプロセス定義（バージョンは異なる場合もある）内で一意な識別子です。
ビジネスプロセス定義名	対象のプロセスインスタンスが所属するビジネスプロセスの定義名称です。サービスリクエスタから呼び出すときのサービス名に相当します。
ビジネスプロセス定義バージョン	対象のプロセスインスタンスが所属するビジネスプロセス定義バージョンです。
状態	対象のプロセスインスタンスの状態です。

属性	説明
開始時刻	対象のプロセスインスタンスの開始時刻（GMT）です。 注　ACTIVE 状態に遷移した時刻
終了時刻	対象のプロセスインスタンスの終了時刻（GMT）です。 注　COMPLETED 状態に遷移した時刻
HCSC サーバ名	対象のプロセスインスタンスが最後に実行された HCSC サーバ名です。

プロセスインスタンスの状態遷移を次の図に示します。

図 3-14　プロセスインスタンスの状態遷移



1. プロセスインスタンスを実行しようとしたとき。
2. プロセスインスタンスに所属するアクティビティを実行しようとしたとき。
3. プロセスインスタンスに所属するグローバルスコープが完了し，プロセスインスタンスが正常に終了したとき。
4. プロセス実行中にフォルトが発生し，グローバルスコープでもキャッチできないでプロセスインスタンスがフォルト終了したとき。
5. 途中終了アクティビティを実行し，プロセスインスタンスが途中終了したとき。

6. サービス呼出アクティビティまたはスコープアクティビティで例外が発生し，プロセスインスタンスがエラーで中断したとき。
7. エラー状態のプロセスインスタンスに対して再実行を実施したとき。
8. 実行中状態のプロセスインスタンスに対して再実行を実施したとき。

「図 3-14　プロセスインスタンスの状態遷移」で示すプロセスインスタンスの状態を次の表に示します。

表 3-3　プロセスインスタンスの状態

状態		説明
ACTIVE 状態	実行中状態（Executing）	プロセスインスタンスが実行中の状態です。
ERROR 状態	エラー状態（Error）	サービス呼出アクティビティまたはスコープアクティビティで例外が発生し，プロセスインスタンスがエラーで中断した状態です。
COMPLETED 状態	完了状態（Completed）	プロセスのグローバルスコープが完了して，プロセスインスタンスが完了した状態です。
	フォルト終了状態（Faulted）	フォルトが発生し，グローバルスコープでもキャッチできないでプロセスインスタンスがフォルト終了した状態です。
	途中終了状態 （Exited）	途中終了アクティビティによって，プロセスインスタンスが終了した状態です。

3.2.4　アクティビティ

アクティビティは，ビジネスプロセスの構成要素となる部品で，処理の構造を表現したものです。複数のアクティビティを連結して，ビジネスプロセスの処理フローを定義します。

開発環境のビジネスプロセス定義画面でアクティビティを配置，連結して定義します。各アクティビティの属性情報を次の表に示します。

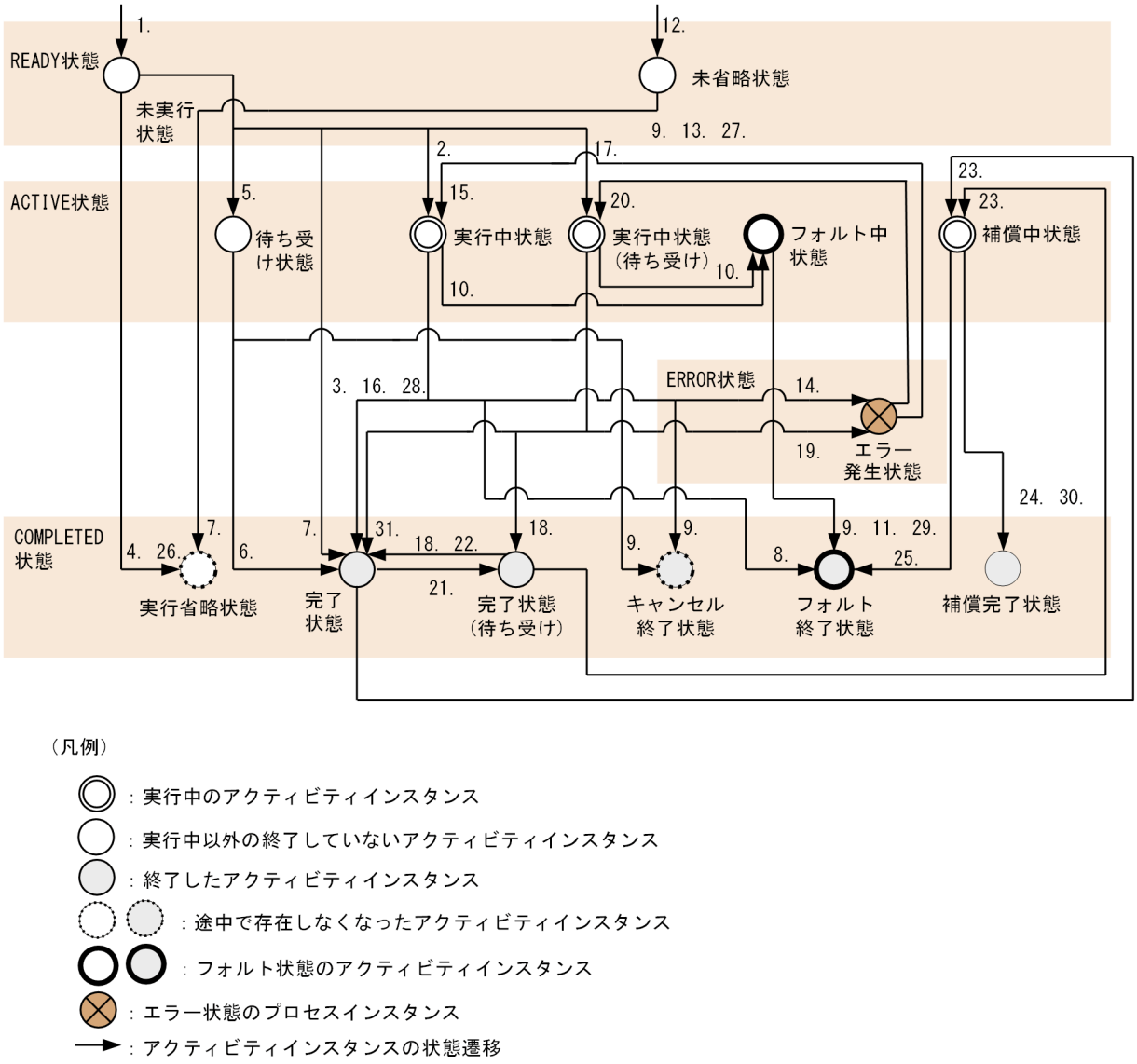
表 3-4　アクティビティの属性情報

属性	説明
プロセスインスタンス識別子	対象のアクティビティが所属するプロセスインスタンスの識別子です。
アクティビティ番号	アクティビティを識別するための番号です。アクティビティリストのインデクスに対応します。
ビジネスプロセス定義名	対象のアクティビティが所属するビジネスプロセスの定義名称です。サービスリクエスタから呼び出すときのサービス名に相当します。
ビジネスプロセス定義バージョン	対象のアクティビティが所属するビジネスプロセス定義バージョンです。
アクティビティ定義名	対象のアクティビティのアクティビティ定義名称です。

属性	説明
アクティビティ種別	対象のアクティビティのアクティビティ種別です。
状態	対象のアクティビティの状態です。
開始時刻	対象のアクティビティの開始時刻（GMT）です。 注　ACTIVE 状態に遷移した時刻
終了時刻	対象のアクティビティの終了時刻（GMT）です。 注　COMPLETED 状態に遷移した時刻
待機解除時刻	対象のアクティビティの待機解除時刻（GMT）です。待機アクティビティのときに使用します。

アクティビティの状態遷移を次の図に示します。

図 3-15　アクティビティの状態遷移



（凡例）

：実行中のアクティビティインスタンス

：実行中以外の終了していないアクティビティインスタンス

：終了したアクティビティインスタンス

：途中で存在しなくなったアクティビティインスタンス

：フォルト状態のアクティビティインスタンス

：エラー状態のプロセスインスタンス

：アクティビティインスタンスの状態遷移

1. 新たなアクティビティのインスタンスが生成されたとき。

2. アクティビティインスタンスを実行しようとしたとき。
3. アクティビティインスタンスの実行が完了したとき。
4. アクティビティインスタンスを実行したが，実行中にならないで実行が完了したとき。
5. サービスリクエスタからの要求を受信した受付以外の受付アクティビティインスタンスを実行しようとしたとき。
6. 待ち受け状態の受付アクティビティインスタンスがサービスリクエスタからの要求を受けたとき。
7. target link が false で活性化されたとき。
8. アクティビティインスタンスの実行時にフォルトが発生したとき。
9. フォルトが発生し，アクティビティの実行がキャンセル処理を依頼されたとき。
10. フォルトが発生し，フォルトハンドラでキャッチして対応するアクティビティインスタンスを実行中のとき。
11. フォルトが発生し，キャッチに対応するアクティビティインスタンスの実行が完了したとき。
12. 構造化アクティビティが実行省略された際に，内包するアクティビティを実行省略にするとき。
13. 未省略状態のアクティビティの実行を省略するとき。
14. サービス呼出アクティビティの処理で例外が発生し，アクティビティインスタンスの処理がエラーで中断したとき。
15. エラー状態のサービス呼出アクティビティのアクティビティインスタンスに対して再実行を実施したとき。
16. 実行中状態の待機アクティビティインスタンスに対して再実行を実施したとき。
17. トランザクションを制御するスコープアクティビティインスタンスを実行しようとしたとき。
18. トランザクションを制御するスコープアクティビティインスタンスの実行が完了したとき（「完了状態(待ち受け)」に遷移し，その後「完了状態」に遷移)。
19. トランザクションを制御するスコープアクティビティの処理で例外が発生し，アクティビティインスタンスの処理がエラーで中断したとき。
20. エラー状態のトランザクションを制御するスコープアクティビティインスタンスに対して再実行を実施したとき。
21. 非同期 EJB に応答アクティビティ実行後の呼び出しを実施したとき。
22. 非同期 EJB から応答アクティビティ実行後の呼び出しが実施されたとき。
23. 補償アクティビティから補償ハンドラ配下のアクティビティを実行しようとしたとき。
24. 補償ハンドラ配下のアクティビティの実行が完了したとき。
25. 補償ハンドラ配下のアクティビティでフォルトが発生したとき。
26. 途中終了アクティビティを実行してアクティビティを実行しなかったとき。
27. 未省略状態のアクティビティが存在する状態で，途中終了アクティビティを実行してプロセスを終了したとき。

28. 実行中状態のアクティビティが存在する状態で，途中終了アクティビティでプロセスを終了したとき。
29. フォルト処理実行中に途中終了アクティビティでプロセスを終了したとき。
30. 補償処理実行中に途中終了アクティビティでプロセスを終了したとき。
31. トランザクション制御を行うスコープアクティビティ実行中に途中終了アクティビティでプロセスを終了したとき。

「図 3-15　アクティビティの状態遷移」で示すアクティビティの状態を次の表に示します。

表 3-5　アクティビティの状態

状態		説明
READY 状態	未実行状態（Unexecution）	アクティビティインスタンスが未実行で，今後実行される可能性のある状態です。
	未省略状態（Unomission）	アクティビティインスタンスが未実行で，今後実行省略される状態です。
ACTIVE 状態	実行中状態（Executing）	アクティビティインスタンスが実行中の状態です。
	実行中状態（待ち受け）（Executingwait）	ロールバックが行われ，スコープアクティビティのインスタンスが再実行可能な状態です。
	待ち受け状態（Awaiting）	受付アクティビティのインスタンスが受け付けられる状態です。
	フォルト中状態（Faulting）	アクティビティインスタンスが faultHandler でキャッチしたアクティビティを実行している状態です。
	補償中状態（Conpensating）	補償処理が実行中の状態です。
ERROR 状態	エラー状態（Error）	サービス呼出アクティビティまたはスコープアクティビティの処理で例外が発生し，アクティビティインスタンスの処理がエラーで中断した状態です。
COMPLETED 状態	完了状態（Completed）	アクティビティインスタンスが正常に実行され終了した状態です。
	完了状態（待ち受け）（Completedwait）	アクティビティインスタンスが正常に実行され終了した状態です。 アクティビティのインスタンスが再実行可能な状態です。
	補償完了状態（Conpensated）	補償処理が正常に実行され終了した状態です。
	キャンセル終了状態（Canceled）	フォルトが発生しアクティビティインスタンスの実行がキャンセルされた状態です。
	フォルト終了状態（Faulted）	アクティビティインスタンスの実行中にフォルトが発生して終了した状態です。
	実行省略状態（Omitted）	アクティビティインスタンスの実行が省略された状態です。

各アクティビティの詳細については，マニュアル「サービスプラットフォーム 開発ガイド 基本開発編」の「5.6　アクティビティの定義」を参照してください。

3.2.5　変数の種類と役割

ビジネスプロセスでは，メッセージに含まれる値や業務で使用する値を変数で保持できます。変数にはグローバル変数とローカル変数の 2 種類があり，開発環境のビジネスプロセス定義画面で定義します。変数の役割を次の表に示します。

表 3-6　変数の役割

変数	説明
グローバル変数	グローバル変数は，1 つのプロセスインスタンス内であれば，どこからでも参照できます。また，プロセスが完了しても変数の値はデータベースに記録されているため，ビジネスプロセス全体の進行状況を分析できます。変数名は，プロセス内で一意にします。
ローカル変数	ローカル変数は，宣言したスコープ内で参照できます。宣言しているスコープが終了すると，変数の値もデータベースから削除されるため，プロセス実行中の条件判定など一時的に値を参照するために利用できます。変数名は，スコープ内で一意にします。スコープが異なれば同じ変数名を宣言できます。ただし，変数を参照する場合に，参照個所で最も内側に近いスコープで宣言された変数を参照します。

変数の定義については，マニュアル「サービスプラットフォーム 開発ガイド 基本開発編」の「5.5.1　変数の定義」を参照してください。

3.2.6　相関セットの役割

相関セットはサービスリクエスタが送信するメッセージを一意に識別するための文字列です。相関セットで使用できるメッセージ種別は，XML 型だけです。

1 つのビジネスプロセスに対して，サービスリクエスタからメッセージを受け取れるプロセスインスタンスが HCSC サーバ上に複数ある場合，HCSC サーバはメッセージに含まれる相関セットの値を利用してプロセスインスタンスを特定します。

サービスリクエスタは相関セットを含めた要求電文を作成して，HCSC サーバに対してサービス部品の呼び出し（ビジネスプロセスの呼び出し）を要求します。ビジネスプロセスからの応答にも相関セットが含まれ，どの送信メッセージに対する結果かを識別できます。相関セットは開発環境のビジネスプロセス定義画面で定義します。

なお，相関セットは，ビジネスプロセスがサービス部品を呼び出すとき，およびビジネスプロセスがサービス部品から応答を受けたときにも，新たに相関セットを指定してメッセージを送信できます。

相関セットによるプロセスインスタンスの識別については，マニュアル「サービスプラットフォーム 開発ガイド 基本開発編」の「5.5.2　相関セットの定義」を参照してください。

3.3　永続化するビジネスプロセスと永続化しないビジネスプロセス

ビジネスプロセスには，データベースにプロセスの実行状態および実行履歴を永続化するビジネスプロセスと，永続化しないビジネスプロセスの 2 種類があります。永続化とは，プロセスの実行状態および実行履歴をデータベースに記録することです。

それぞれのビジネスプロセスの違いについて，次に説明します。

3.3.1　永続化するビジネスプロセス

永続化するビジネスプロセスは，プロセスの実行状態と実行履歴の記録があるため，障害が発生した場合に，プロセスの実行の進捗度がわかり，プロセスを再実行できます。開発環境でビジネスプロセスを追加するときに，永続化する設定にできます。

次に示す内容を実施する場合には，ビジネスプロセスを永続化に設定します。

プロセスインスタンスの実行履歴の管理

- コマンドによるプロセスインスタンスの実行履歴の取得
- 運用環境の画面によるプロセスインスタンスの実行履歴の取得

プロセスインスタンスの再実行

- コマンドによるプロセスインスタンスの再実行
- 運用環境の画面によるプロセスインスタンスの再実行
- サービスリクエスタ（SOAP 通信/SessionBean）によるプロセスインスタンスの再実行

ビジネスプロセスを新規に追加する方法については，マニュアル「サービスプラットフォーム 開発ガイド 基本開発編」の「5.2.1　ビジネスプロセスを新規に追加する」を参照してください。

3.3.2　永続化しないビジネスプロセス

永続化しないビジネスプロセスは，プロセスの実行状態と実行履歴の記録を採らないため，高性能を実現する必要があるビジネスプロセスの場合に設定します。開発環境でビジネスプロセスを追加するときに，永続化しない設定にできます。

ただし，永続化するビジネスプロセスではできることが，永続化しないビジネスプロセスではできないといった制約事項があります。永続化しないビジネスプロセスは，次の条件を満たす場合に定義できます。

1. 非同期のサービスアダプタに対して，サービスを呼び出すサービス呼出アクティビティを含まない。
2. 待機アクティビティを含まない。
3. 受付アクティビティを複数含まない。

4. プロセスインスタンスの再実行を使用しない（障害発生時は，サービスリクエスタからサービス呼び出し要求（ビジネスプロセスの呼び出し）を再度実行する）。

ビジネスプロセスを新規に追加する方法については，マニュアル「サービスプラットフォーム 開発ガイド 基本開発編」の「5.2.1　ビジネスプロセスを新規に追加する」を参照してください。

3.4 ビジネスプロセスのトランザクション

永続化するビジネスプロセスの場合，ビジネスプロセスを実行する過程でアクティビティの状態をデータベースに出力します。

ここでは，アクティビティの状態を確定するタイミング（トランザクションの開始とコミット）について説明します。

トランザクションの開始とコミットのタイミングを次の表に示します。

表 3-7 トランザクションの開始とコミットのタイミング

処理	タイミング
トランザクションの開始	受付アクティビティでメッセージを受信したとき。
	通信モデルが同期の場合のサービス呼出アクティビティで，メッセージを送信する前のトランザクションをコミットした直後。 また，ビジネスプロセスの状態の互換性（HCSC サーバセットアップ定義の bp-status-compatible）に OFF を指定した場合は，サービス呼出アクティビティの状態の互換性（HCSC サーバセットアップ定義の bp-invoke-status-compatible）の設定にかかわらず，サービス呼出アクティビティでメッセージを送信する前のトランザクションをコミットした直後（通信モデルが同期，非同期のどちらでも同じタイミングになります）。
	ビジネスプロセスの状態の互換性（HCSC サーバセットアップ定義の bp-status-compatible）に OFF を指定，かつサービス呼出アクティビティの状態の互換性（HCSC サーバセットアップ定義の bp-invoke-status-compatible）に ON を指定した場合，サービス呼出アクティビティの処理が完了したときのトランザクションをコミットした直後。 また，ビジネスプロセスの状態の互換性に OFF を指定，かつサービス呼出アクティビティの状態の互換性に OFF を指定した場合，サービス呼出アクティビティの処理で例外が発生したときのトランザクションをコミットした直後。
	応答アクティビティの処理のあとに存在するアクティビティを実行する直前。
	プロセス再実行ユティリティによって，プロセスを再実行するとき。
	待機アクティビティに指定された間隔または期限でタイマが解除された直後。
	待機アクティビティの処理が完了したときのトランザクションをコミットした直後。
	トランザクション制御を行うスコープアクティビティの開始で，トランザクションをコミットした直後。
	トランザクション制御を行うスコープアクティビティの終了で，トランザクションをコミットした直後。
トランザクションのコミット	応答アクティビティでメッセージを応答する直前。
	通信モデルが同期の場合のサービス呼出アクティビティでメッセージを送信する直前。 また，ビジネスプロセスの状態の互換性（HCSC サーバセットアップ定義の bp-status-compatible）に OFF を指定した場合は，サービス呼出アクティビティの状態の互換性（HCSC サーバセットアップ定義の bp-invoke-status-compatible）の設定にかかわらず，サービス呼出アクティビティでメッセージを送信する直前（通信モデルが同期，非同期のどちらでも同じタイミングになります）。

処理	タイミング
トランザクションのコミット	ビジネスプロセスの状態の互換性（HCSC サーバセットアップ定義の bp-status-compatible）に OFF を指定，かつサービス呼出アクティビティの状態の互換性（HCSC サーバセットアップ定義の bp-invoke-status-compatible）に ON を指定した場合，サービス呼出アクティビティの処理が完了したとき。 また，ビジネスプロセスの状態の互換性に OFF を指定，かつサービス呼出アクティビティの状態の互換性に OFF を指定した場合，サービス呼出アクティビティの処理で例外が発生したとき。
	実行可能なすべてのアクティビティを実行し，ビジネスプロセスの処理を終了するとき。
	待機アクティビティに指定された間隔または期限の J2EE サーバの TimerService にタイマを登録する直前。
	待機アクティビティの処理が完了したとき。
	トランザクション制御を行うスコープアクティビティが開始したとき。
	トランザクション制御を行うスコープアクティビティが終了したとき。

トランザクションの開始とコミットのタイミングは，ビジネスプロセスの状態の互換性（HCSC サーバセットアップ定義の bp-status-compatible）の指定が ON または OFF の場合，またはサービス呼出アクティビティの状態の互換性（HCSC サーバセットアップ定義の bp-invoke-status-compatible）の指定が ON または OFF の場合で動作が異なります。

サービス呼出アクティビティの状態の互換性は，ビジネスプロセスの状態の互換性の指定が OFF のときに，有効となります。

なお，サービス呼出アクティビティの状態の互換性に ON を指定した場合は，サービスプラットフォームのバージョン 08-70 以前と同じ動作となります。

それぞれのトランザクションの開始とコミットのタイミングの例を次に示します。

3.4.1 ビジネスプロセスの状態の互換性の指定を ON にした場合のトランザクション

ビジネスプロセスの状態の互換性（HCSC サーバセットアップ定義の bp-status-compatible）に ON を指定した場合のトランザクションについて説明します。

なお，ビジネスプロセスの状態の互換性に ON を指定した場合は，サービスプラットフォームのバージョン 08-00 以前と同じ動作となります。

また，サービス呼出アクティビティの状態の互換性に ON，または OFF のどちらを指定しても同じ動作となります。

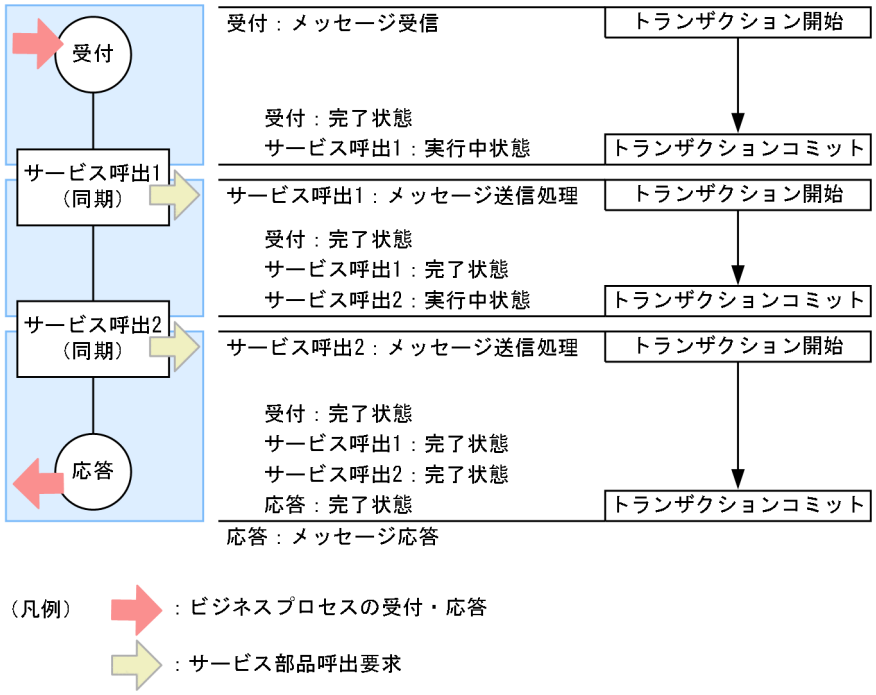
(1)　同期のサービス呼出アクティビティが 2 つ並んだ場合

正常に処理が進行した場合

それぞれのサービス呼出アクティビティでのメッセージ送信処理直前でトランザクションをコミットし，新たにトランザクションを開始します。

同期のサービス呼出で正常に処理が進行した場合のトランザクションの範囲を次の図に示します。

図 3-16　正常に処理が進行した場合のトランザクションの範囲（同期のサービス呼出）

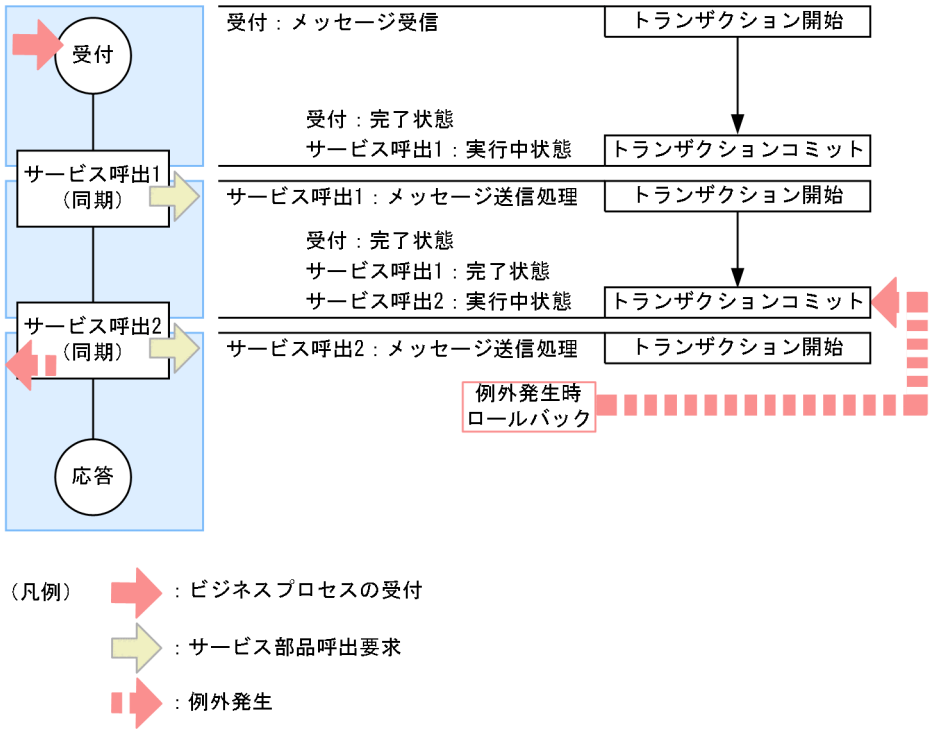


例外が発生した場合

サービス呼出 2 の処理が例外でエラーとなった場合は，トランザクションがコミットした時点（受付：完了状態，サービス呼出 1：完了状態，サービス呼出 2：実行中状態）までロールバックします。

同期のサービス呼出で例外が発生した場合のトランザクションの範囲を次の図に示します。

図 3-17　例外が発生した場合のトランザクションの範囲（同期のサービス呼出）



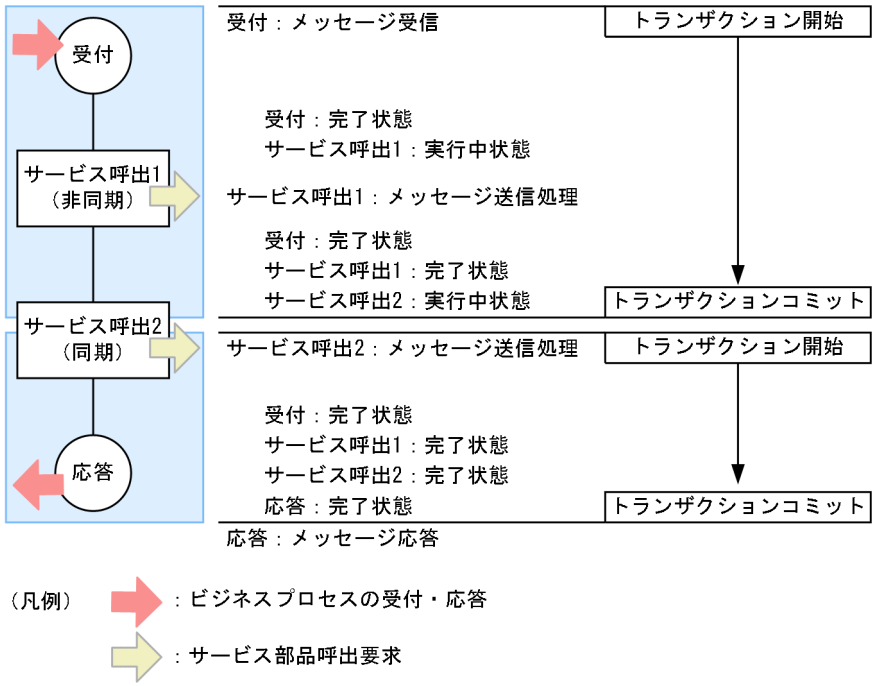
(2)　非同期と同期のサービス呼出アクティビティが 2 つ並んだ場合

正常に処理が進行した場合

非同期のサービス呼出アクティビティでのメッセージ送信処理ではトランザクションはコミットしないで，同期のサービス呼出アクティビティでのメッセージ送信処理直前でトランザクションをコミットし，新たにトランザクションを開始します。

非同期と同期のサービス呼出で正常に処理が進行した場合のトランザクションの範囲を次の図に示します。

図 3-18　正常に処理が進行した場合のトランザクションの範囲（非同期と同期のサービス呼出）

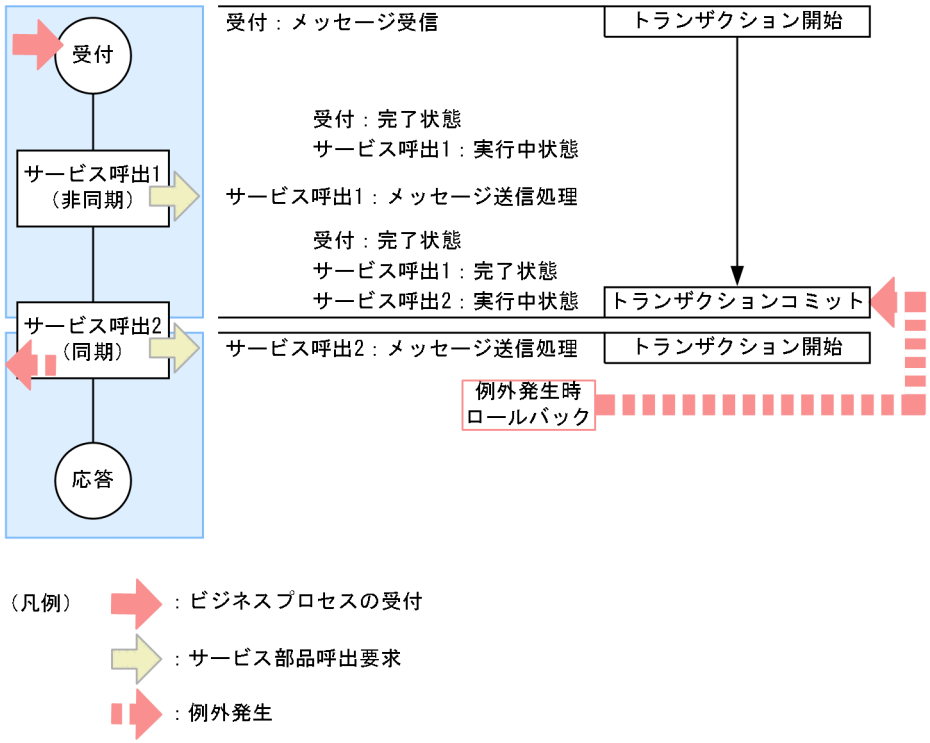


例外が発生した場合（非同期と同期のサービス呼出）

サービス呼出 1 が非同期，サービス呼出 2 が同期の構成で，サービス呼出 2 の処理が例外でエラーとなった場合は，トランザクションがコミットした時点（受付：完了状態，サービス呼出 1：完了状態，サービス呼出 2：実行中状態）までロールバックします。

非同期と同期のサービス呼出で例外が発生した場合のトランザクションの範囲を次の図に示します。

図 3-19　例外が発生した場合のトランザクションの範囲（非同期と同期のサービス呼出）

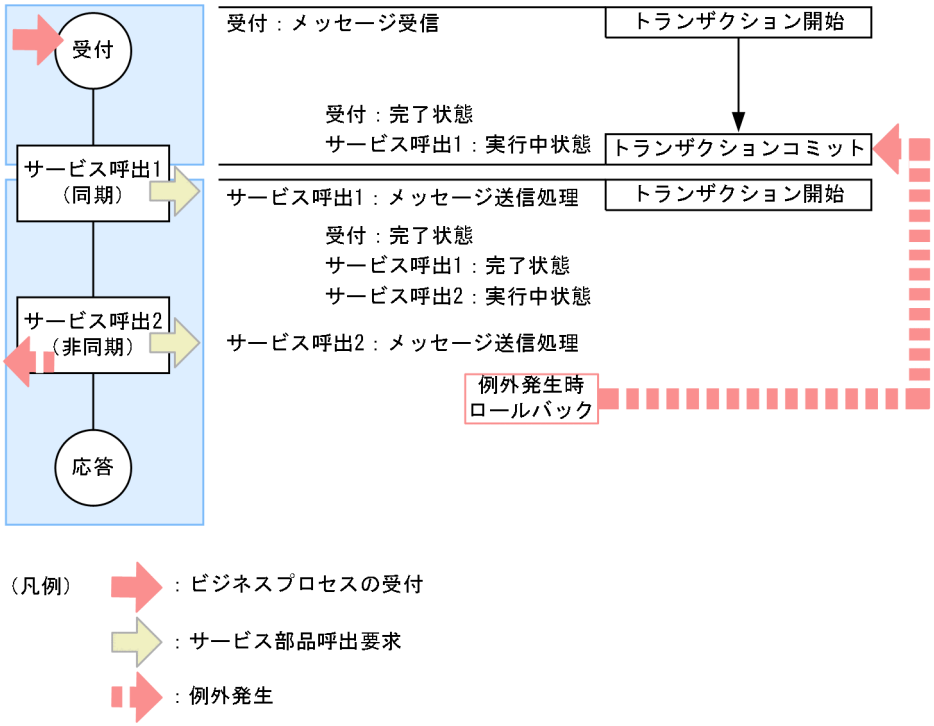


例外が発生した場合（同期と非同期のサービス呼出）

サービス呼出 1 が同期，サービス呼出 2 が非同期の構成で，サービス呼出 2 の処理が例外でエラーとなった場合は，トランザクションがコミットした時点（受付：完了状態，サービス呼出 1：実行中状態）までロールバックします。

同期と非同期のサービス呼出で例外が発生した場合のトランザクションの範囲を次の図に示します。

図 3-20　例外が発生した場合のトランザクションの範囲（同期と非同期のサービス呼出）



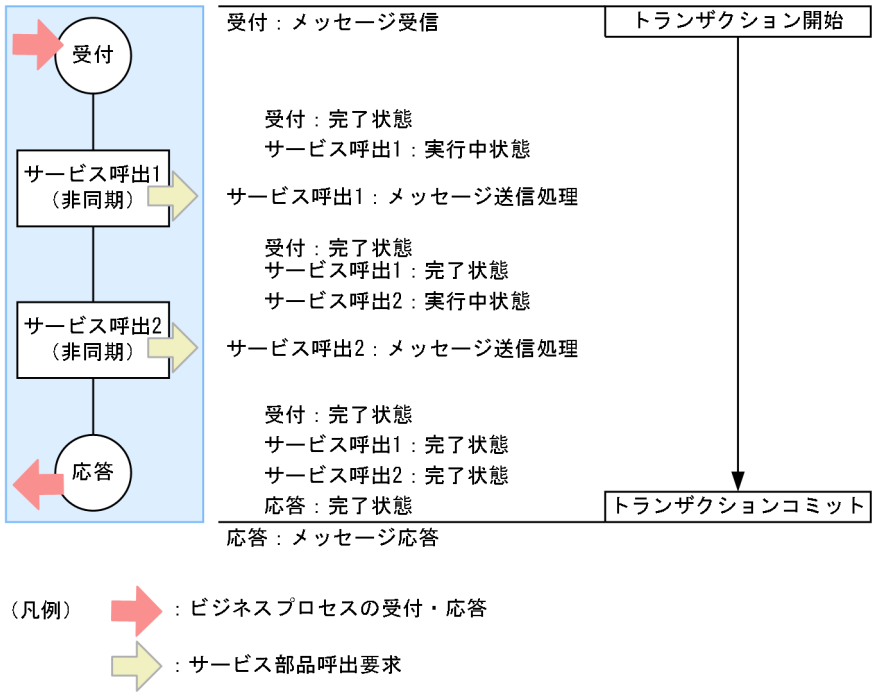
(3)　非同期のサービス呼出アクティビティが 2 つ並んだ場合

正常に処理が進行した場合

非同期のサービス呼出アクティビティでのメッセージ送信処理ではトランザクションはコミットしません。そのため，ビジネスプロセスが応答するときにトランザクションがコミットします。

非同期のサービス呼出で正常に処理が進行した場合のトランザクションの範囲を次の図に示します。

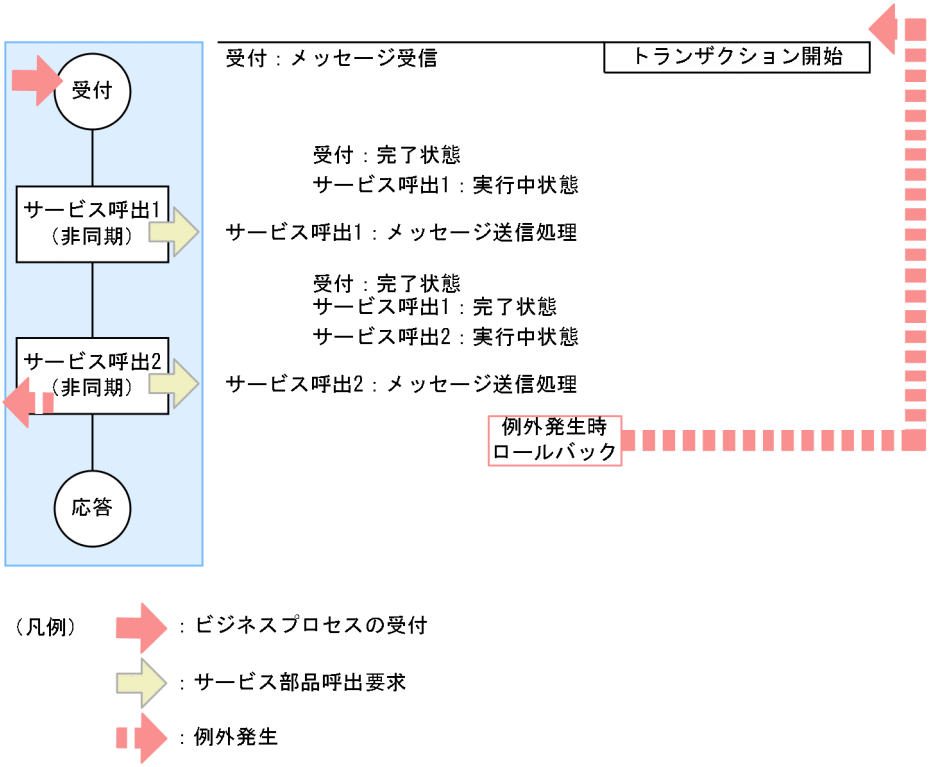
図 3-21　正常に処理が進行した場合のトランザクションの範囲（非同期のサービス呼出）



例外が発生した場合

サービス呼出 2 の処理が例外でエラーとなった場合は，トランザクションがコミットした時点（受付：未実行状態，サービス呼出 1：未実行状態，サービス呼出 2：未実行状態）までロールバックします。非同期のサービス呼出で例外が発生した場合のトランザクションの範囲を次の図に示します。

図 3-22　例外が発生した場合のトランザクションの範囲（非同期のサービス呼出）



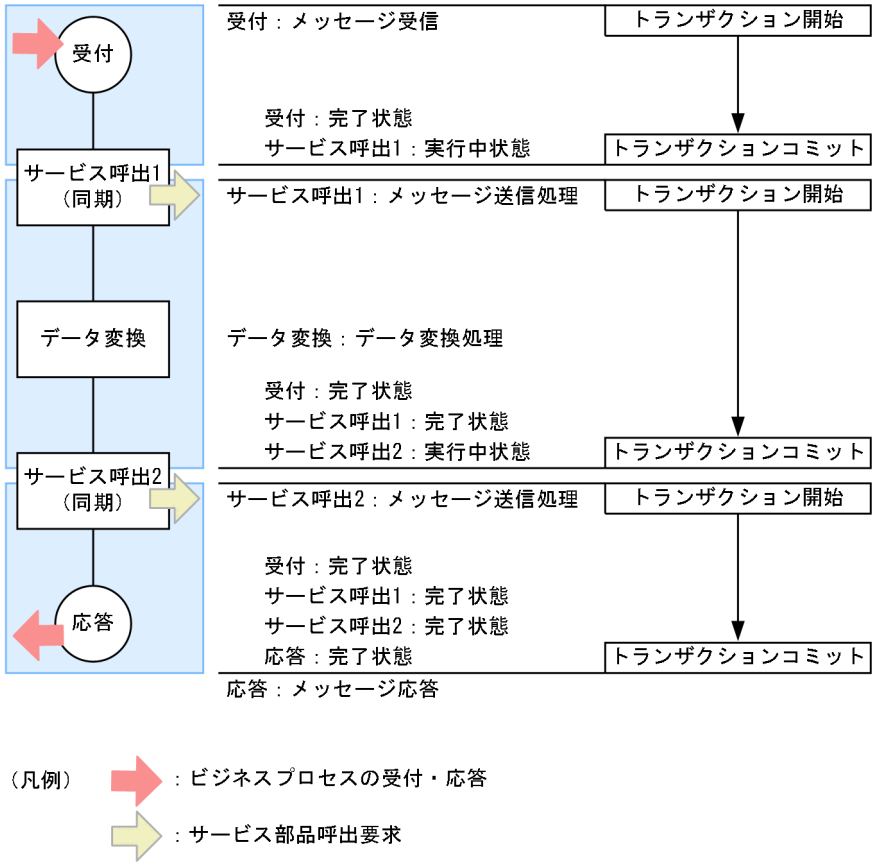
(4) 同期のサービス呼出アクティビティのあとにデータ変換アクティビティがある場合

正常に処理が進行した場合

それぞれのサービス呼出アクティビティでのメッセージ送信処理直前でトランザクションをコミットしますが，データ変換アクティビティではトランザクションがコミットしません。これは，「表 3-7　トランザクションの開始とコミットのタイミング」で示した以外のアクティビティも同様です。

データ変換などのアクティビティがある場合で正常に処理が進行したときのトランザクションの範囲を次の図に示します。

図 3-23　正常に処理が進行した場合のトランザクションの範囲（データ変換などのアクティビティがある場合）

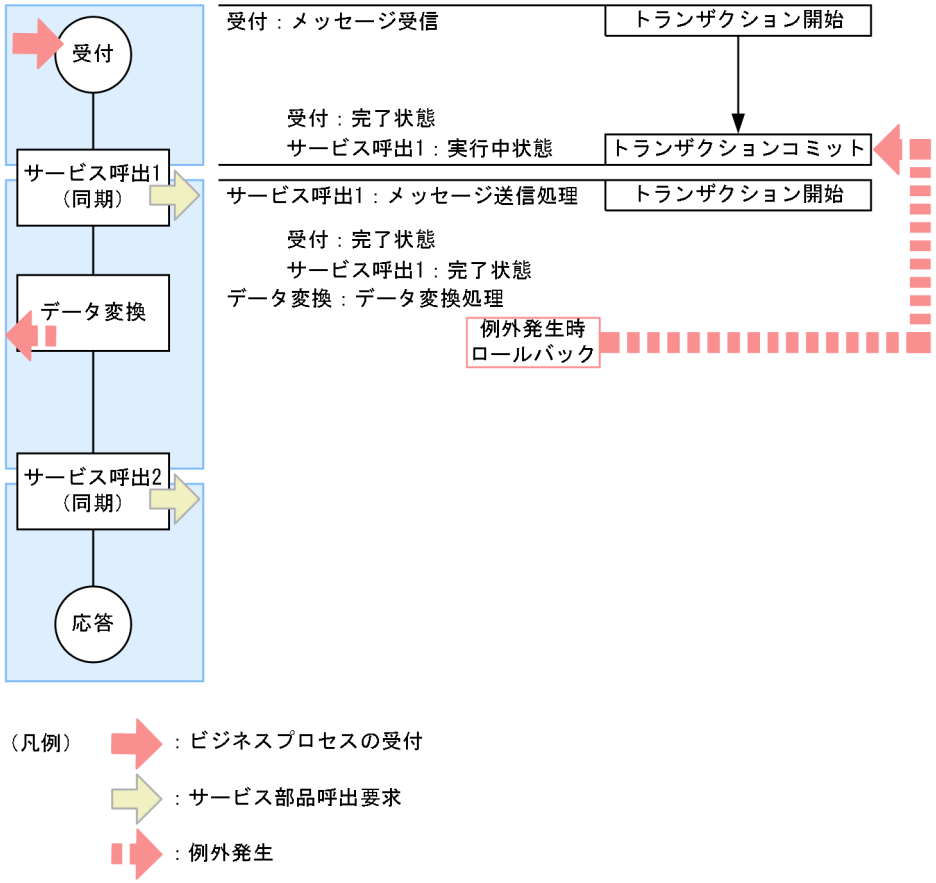


例外が発生した場合（同期サービス呼出のあとにデータ変換などのアクティビティがある場合）

サービス呼出 1 が同期の構成で，サービス呼出 1 のあとにデータ変換処理でエラーとなった場合は，トランザクションがコミットした時点（受付：完了状態，サービス呼出 1：実行中状態）までロールバックします。

同期サービス呼出のあとにデータ変換などのアクティビティがある場合で例外が発生したときのトランザクションの範囲を次の図に示します。

図 3-24　例外が発生した場合のトランザクションの範囲（同期サービス呼出のあとにデータ変換などのアクティビティがある場合）

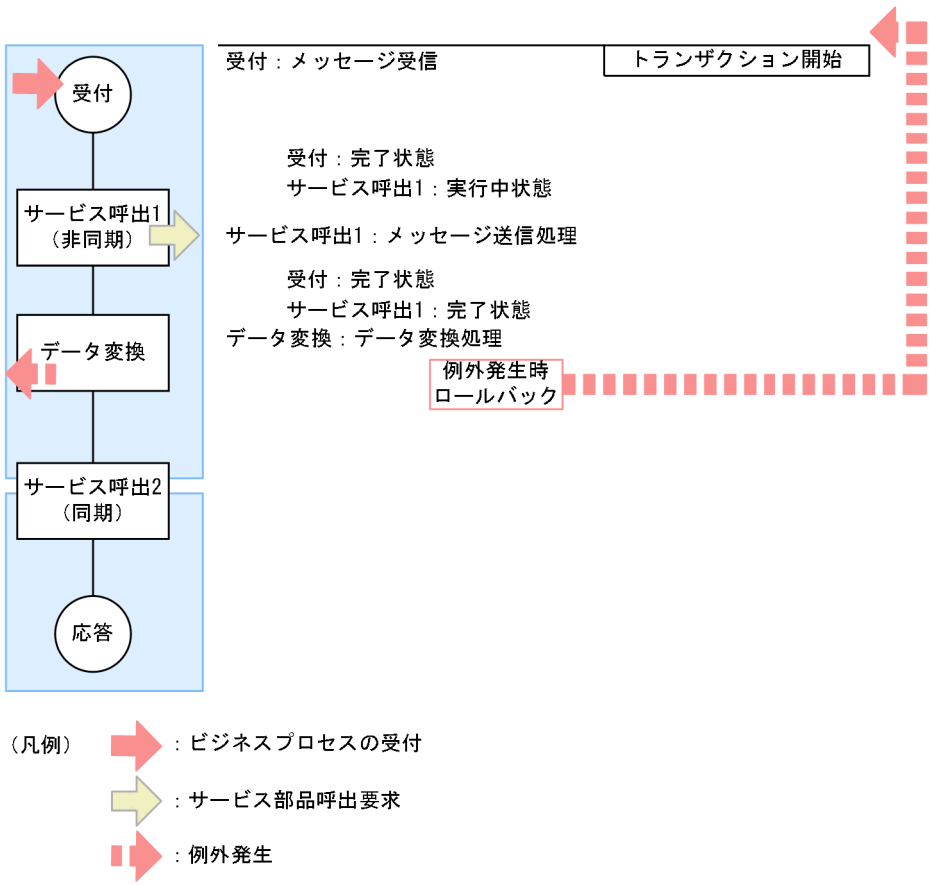


例外が発生した場合（非同期サービス呼出のあとにデータ変換などのアクティビティがある場合）

サービス呼出 1 が非同期の構成で，サービス呼出 1 のあとにデータ変換処理でエラーとなった場合は，トランザクションがコミットした時点（受付：未実行状態，サービス呼出 1：未実行状態）までロールバックします。

非同期サービス呼出のあとにデータ変換などのアクティビティがある場合で例外が発生したときのトランザクションの範囲を次の図に示します。

図 3-25　例外が発生した場合のトランザクションの範囲（非同期サービス呼出のあとにデータ変換などのアクティビティがある場合）



3.4.2　ビジネスプロセスの状態の互換性の指定を OFF，サービス呼出アクティビティの状態の互換性の指定を ON にした場合のトランザクション

ビジネスプロセスの状態の互換性（HCSC サーバセットアップ定義の bp-status-compatible）に OFF，サービス呼出アクティビティの状態の互換性（HCSC サーバセットアップ定義の bp-invoke-status-compatible）に ON を指定した場合のトランザクションについて説明します。

ビジネスプロセスの状態の互換性に OFF，サービス呼出アクティビティの状態の互換性に ON を指定した場合は，サービス呼出アクティビティの処理が，開始または完了したタイミングでトランザクションを終了します。ビジネスプロセスのサービス呼出アクティビティ実行中に例外が発生した場合，プロセスインスタンスおよびサービス呼出アクティビティの状態にエラー発生（Error）を設定します。

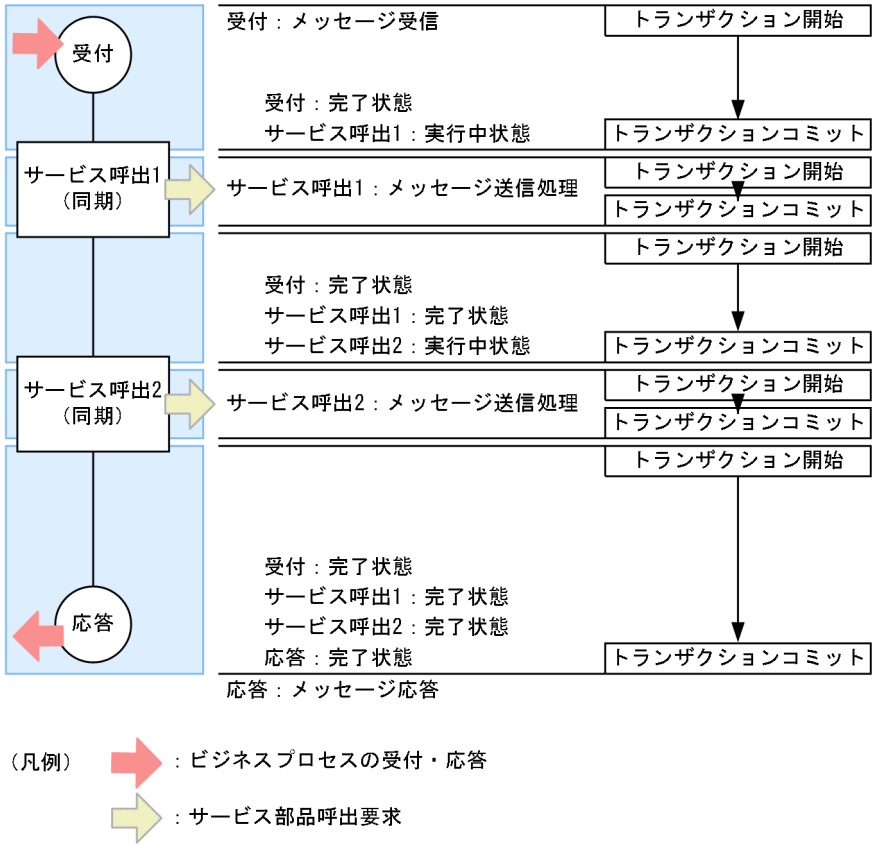
(1)　同期のサービス呼出アクティビティが 2 つ並んだ場合

正常に処理が進行した場合

それぞれのサービス呼出アクティビティでのメッセージ送信処理直前でトランザクションをコミットし，新たにトランザクションを開始します。また，メッセージ送信処理が完了したあとにもトランザクションをコミットし，新たにトランザクションを開始します。

同期のサービス呼出で正常に処理が進行した場合のトランザクションの範囲を次の図に示します。

図 3-26　正常に処理が進行した場合のトランザクションの範囲（同期のサービス呼出）

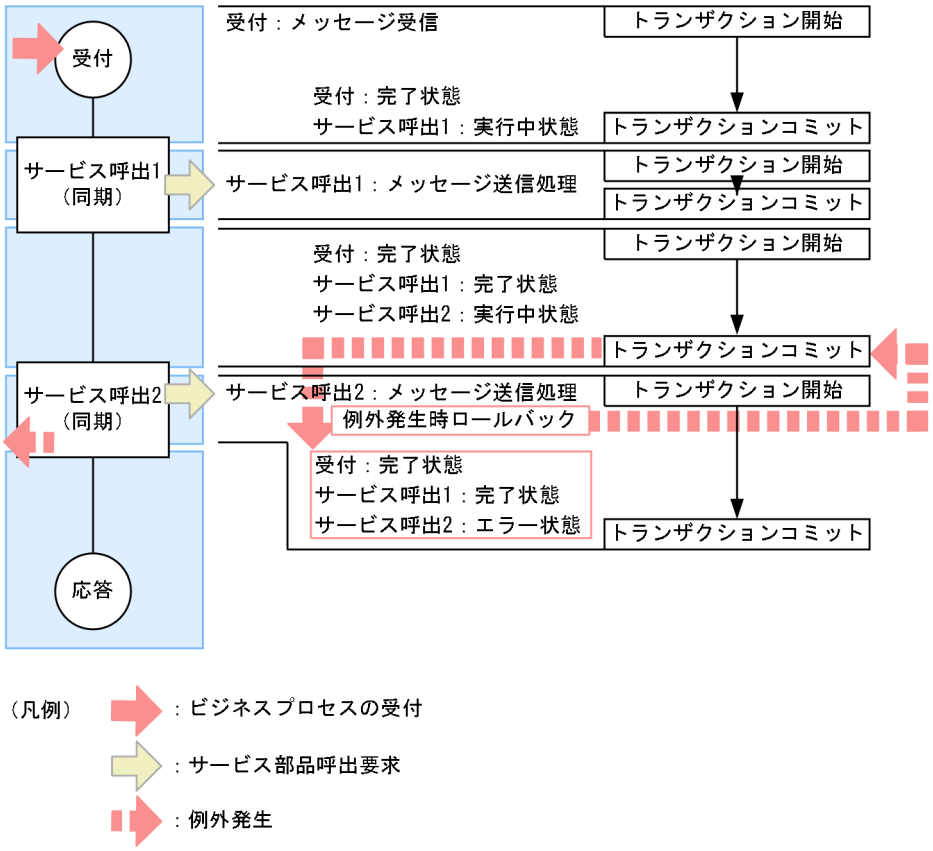


例外が発生した場合

サービス呼出 2 の処理が例外でエラーとなった場合は，トランザクションがコミットした時点（受付：完了状態，サービス呼出 1：完了状態，サービス呼出 2：実行中状態）にロールバックしたあと，プロセスインスタンスおよびサービス呼出アクティビティの状態にエラー発生（Error）を設定して，トランザクションをコミットします。

同期のサービス呼出で例外が発生した場合のトランザクションの範囲を次の図に示します。

図 3-27　例外が発生した場合のトランザクションの範囲（同期のサービス呼出）



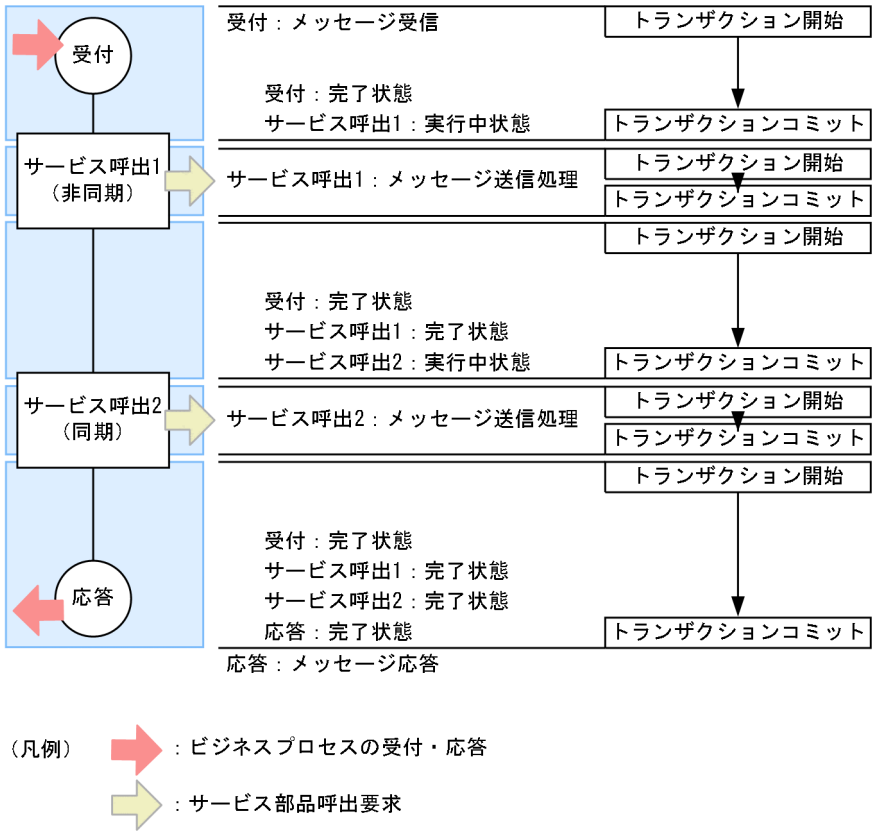
(2)　非同期と同期のサービス呼出アクティビティが 2 つ並んだ場合

正常に処理が進行した場合

それぞれのサービス呼出アクティビティでのメッセージ送信処理直前でトランザクションをコミットし，新たにトランザクションを開始します。また，メッセージ送信処理が完了したあとにもトランザクションをコミットし，新たにトランザクションを開始します。

非同期と同期のサービス呼出で正常に処理が進行した場合のトランザクションの範囲を次の図に示します。

図 3-28　正常に処理が進行した場合のトランザクションの範囲（非同期と同期のサービス呼出）

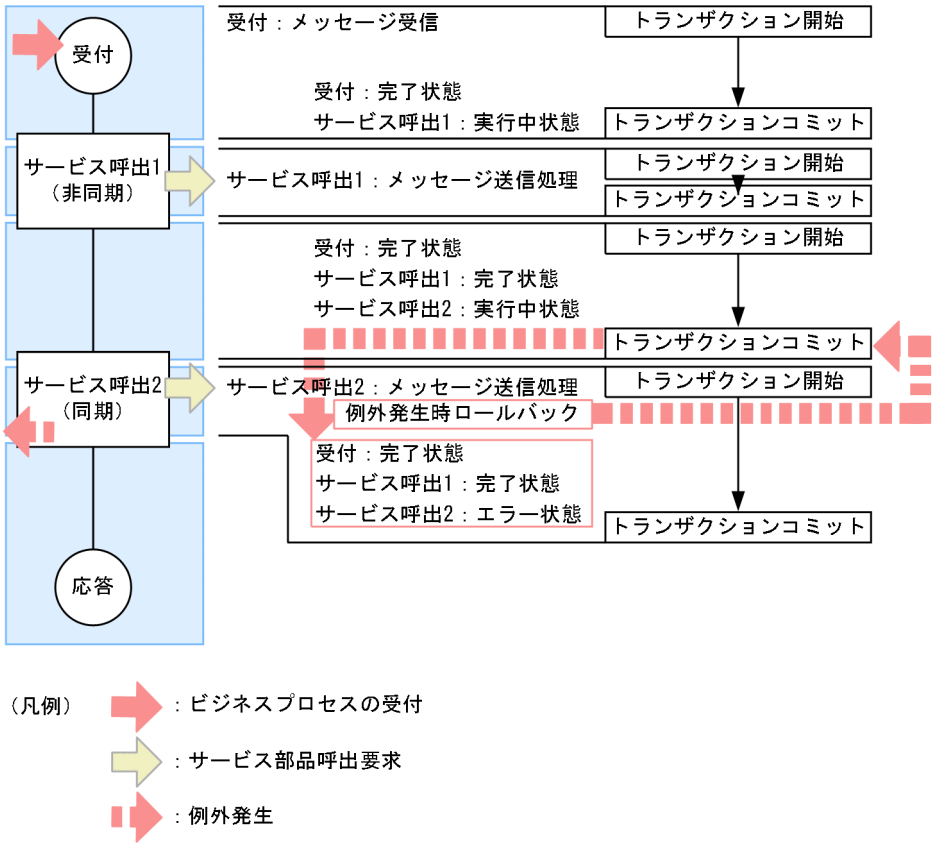


例外が発生した場合（非同期と同期のサービス呼出）

サービス呼出 2 の処理が例外でエラーとなった場合は，トランザクションがコミットした時点（受付：完了状態，サービス呼出 1：完了状態，サービス呼出 2：実行中状態）にロールバックしたあと，プロセスインスタンスおよびサービス呼出アクティビティの状態にエラー発生（Error）を設定して，トランザクションをコミットします。

非同期と同期のサービス呼出で例外が発生した場合のトランザクションの範囲を次の図に示します。

図 3-29　例外が発生した場合のトランザクションの範囲（非同期と同期のサービス呼出）



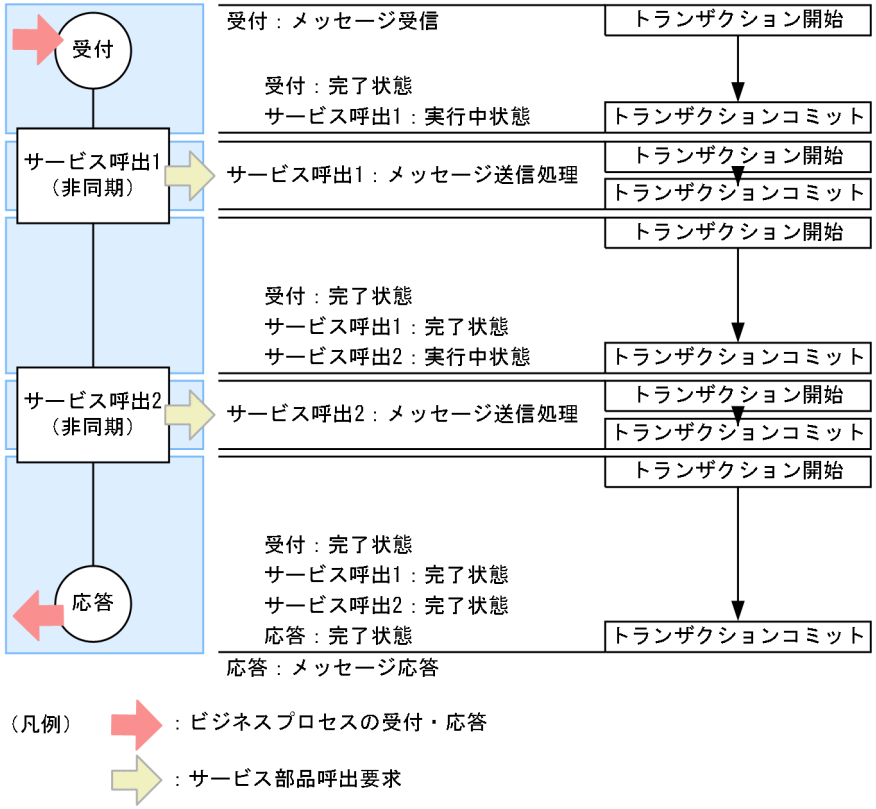
(3)　非同期のサービス呼出アクティビティが 2 つ並んだ場合

正常に処理が進行した場合

それぞれのサービス呼出アクティビティでのメッセージ送信処理直前でトランザクションをコミットし，新たにトランザクションを開始します。また，メッセージ送信処理が完了したあとにもトランザクションをコミットし，新たにトランザクションを開始します。

非同期のサービス呼出で正常に処理が進行した場合のトランザクションの範囲を次の図に示します。

図 3-30　正常に処理が進行した場合のトランザクションの範囲（非同期のサービス呼出）

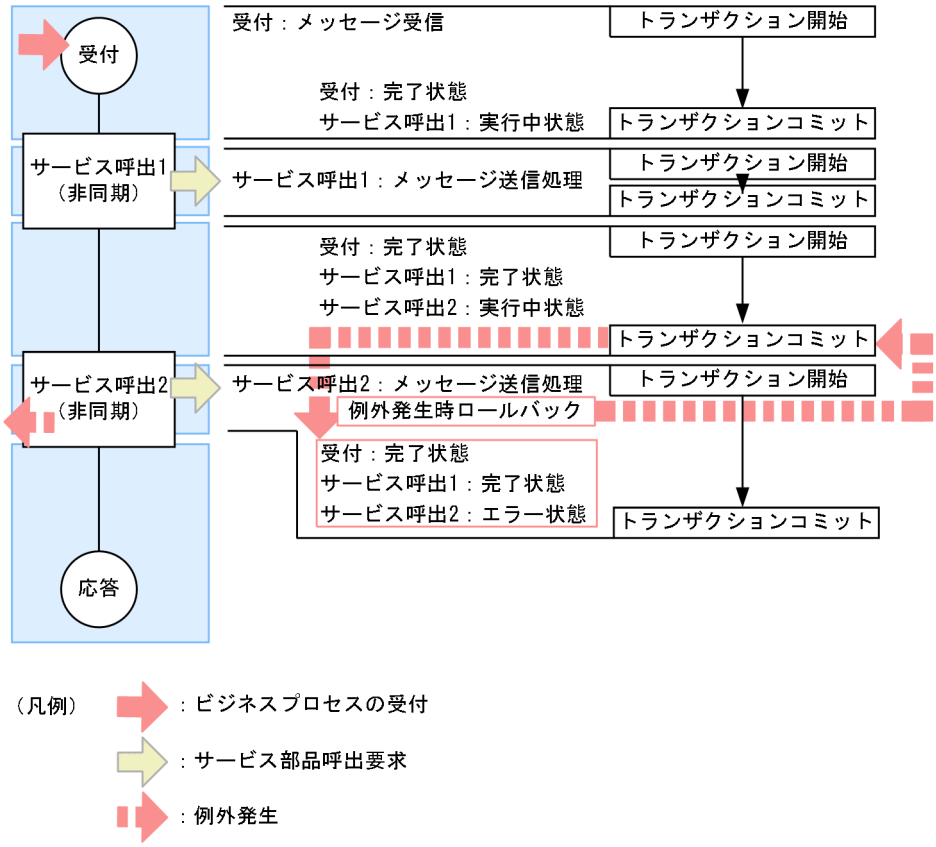


例外が発生した場合

サービス呼出 2 の処理が例外でエラーとなった場合は，トランザクションがコミットした時点（受付：完了状態，サービス呼出 1：完了状態，サービス呼出 2：実行中状態）にロールバックしたあと，プロセスインスタンスおよびサービス呼出アクティビティの状態にエラー発生（Error）を設定して，トランザクションをコミットします。

非同期のサービス呼出で例外が発生した場合のトランザクションの範囲を次の図に示します。

図 3-31　例外が発生した場合のトランザクションの範囲（非同期のサービス呼出）



(4)　同期のサービス呼出アクティビティのあとにデータ変換アクティビティがある場合

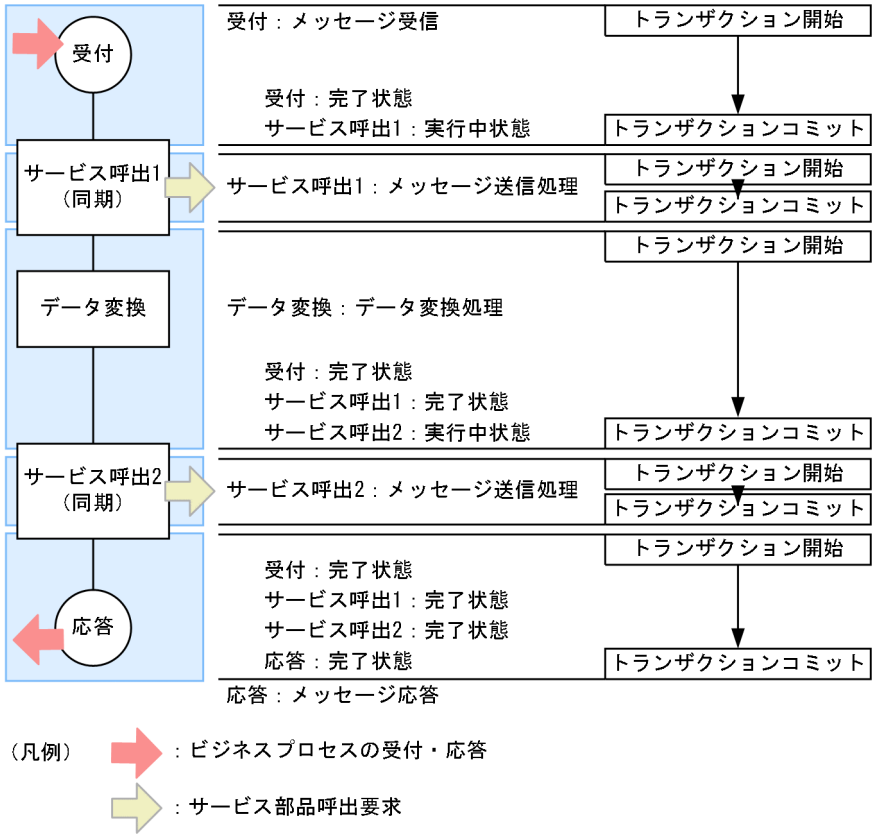
正常に処理が進行した場合

それぞれのサービス呼出アクティビティでのメッセージ送信処理直前でトランザクションをコミットし，新たにトランザクションを開始します。また，メッセージ送信処理が完了したあとにもトランザクションをコミットし，新たにトランザクションを開始します。

ただし，データ変換アクティビティではトランザクションがコミットしません。これは，「表 3-7　トランザクションの開始とコミットのタイミング」で示した以外のアクティビティも同様です。

データ変換などのアクティビティがある場合で正常に処理が進行したときのトランザクションの範囲を次の図に示します。

図 3-32　正常に処理が進行した場合のトランザクションの範囲（データ変換などのアクティビティがある場合）

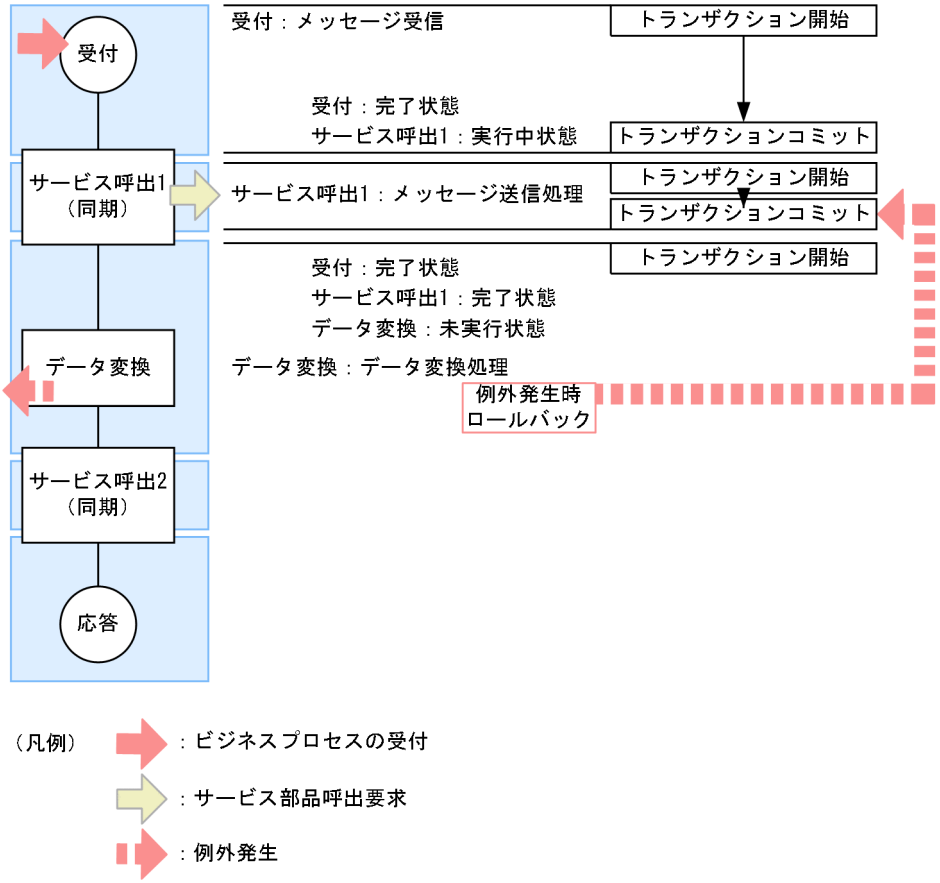


例外が発生した場合（データ変換などのアクティビティがある場合）

サービス呼出 1 のあとのデータ変換処理でエラーとなった場合は，トランザクションがコミットした時点（受付：完了状態，サービス呼出 1：完了状態，データ変換：未実行状態）までロールバックします。

データ変換などのアクティビティがある場合で例外が発生したときのトランザクションの範囲を次の図に示します。

図 3-33　例外が発生した場合のトランザクションの範囲（データ変換などのアクティビティがある場合）



3.4.3　ビジネスプロセスの状態の互換性の指定を OFF，サービス呼出アクティビティの状態の互換性の指定を OFF にした場合のトランザクション

ビジネスプロセスの状態の互換性（HCSC サーバセットアップ定義の bp-status-compatible）に OFF，サービス呼出アクティビティの状態の互換性（HCSC サーバセットアップ定義の bp-invoke-status-compatible）に OFF を指定した場合のトランザクションについて説明します。

ビジネスプロセスの状態の互換性に OFF，サービス呼出アクティビティの状態の互換性に OFF を指定した場合の動作を次に示します。

- サービス呼出アクティビティの処理が開始したタイミングでは，アクティビティの状態に実行中状態（Executing）を設定し，トランザクションを終了します。
- サービス呼出アクティビティの処理が完了したタイミングでは，アクティビティの状態に完了状態（Completed）を設定し，トランザクションを終了しません。

- ビジネスプロセスのサービス呼出アクティビティ実行中に例外が発生した場合は，プロセスインスタンスおよびサービス呼出アクティビティの状態にエラー発生（Error）を設定し，トランザクションを終了します。

なお，通信モデルが同期，非同期のどちらでも同じ動作となります。

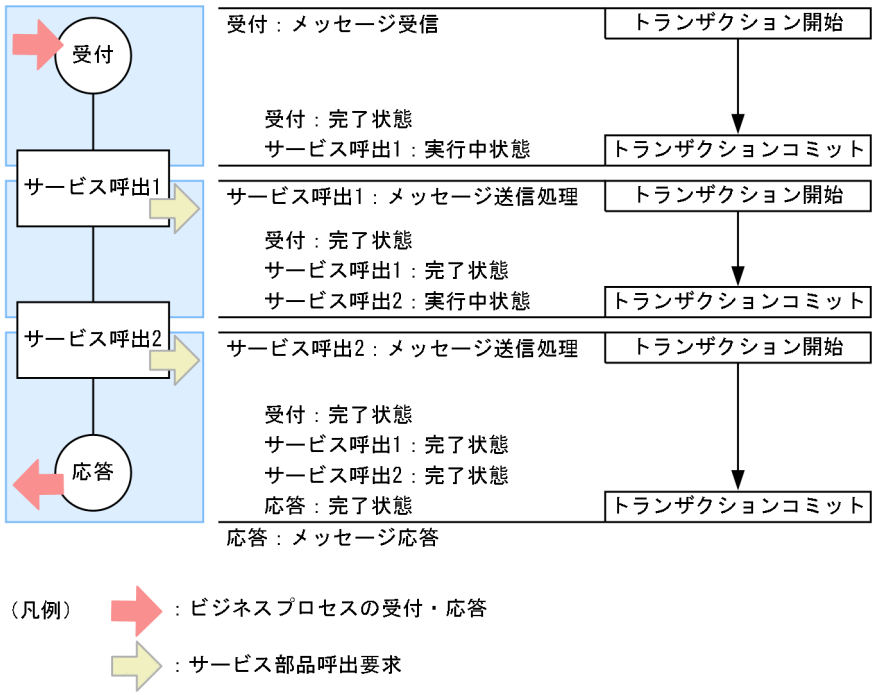
(1)　サービス呼出アクティビティが 2 つ並んだ場合

正常に処理が進行した場合

それぞれのサービス呼出アクティビティでのメッセージ送信処理直前でトランザクションをコミットし，新たにトランザクションを開始します。

サービス呼出で正常に処理が進行した場合のトランザクションの範囲を次の図に示します。

図 3-34　正常に処理が進行した場合のトランザクションの範囲

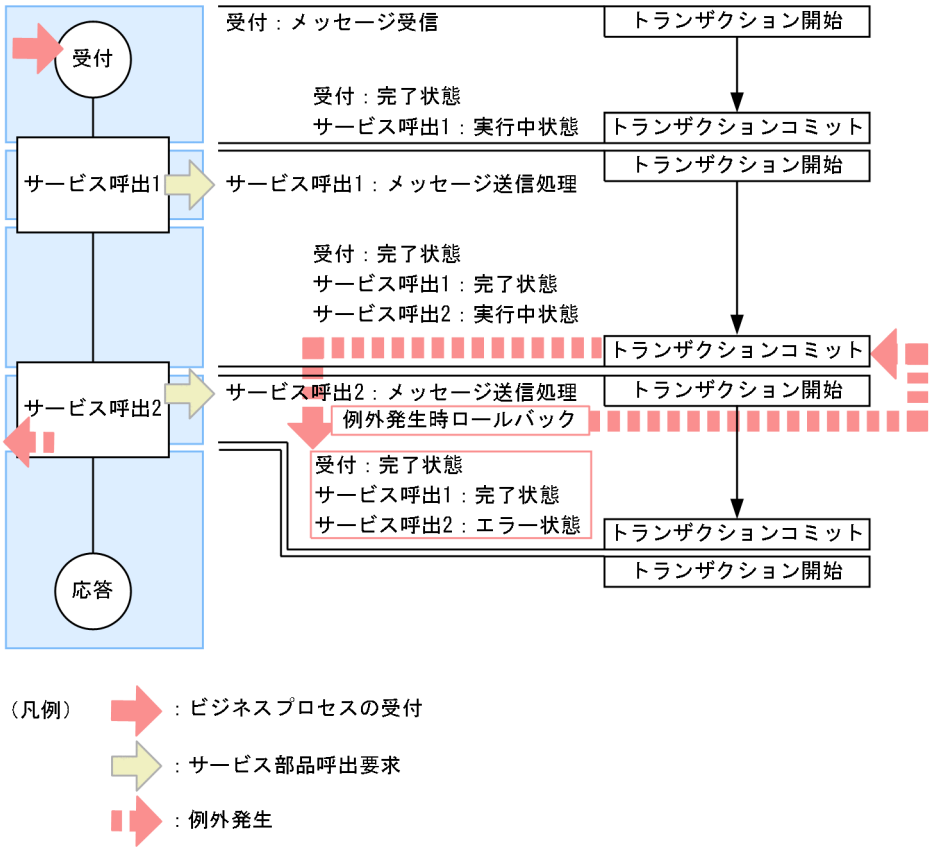


例外が発生した場合

サービス呼出 2 の処理が例外でエラーとなった場合は，トランザクションがコミットした時点（受付：完了状態，サービス呼出 1：完了状態，サービス呼出 2：実行中状態）にロールバックしたあと，プロセスインスタンスおよびサービス呼出アクティビティの状態にエラー発生（Error）を設定して，トランザクションをコミットします。

サービス呼出で例外が発生した場合のトランザクションの範囲を次の図に示します。

図 3-35　例外が発生した場合のトランザクションの範囲



(2)　サービス呼出アクティビティのあとにデータ変換アクティビティがある場合

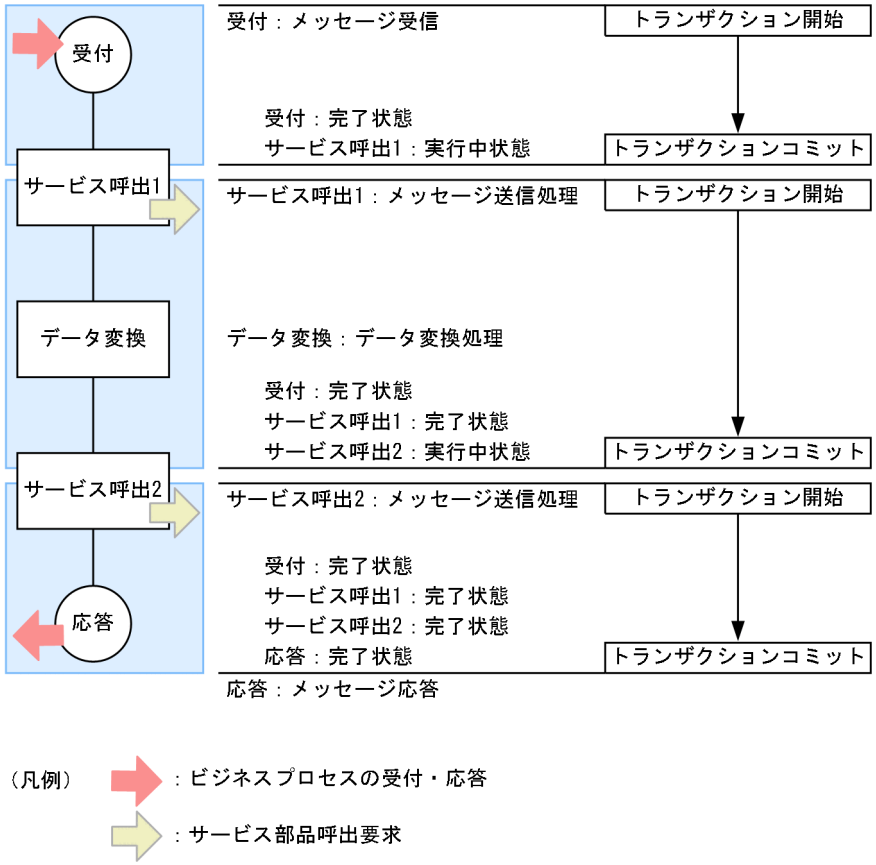
正常に処理が進行した場合

それぞれのサービス呼出アクティビティでのメッセージ送信処理直前でトランザクションをコミットし，新たにトランザクションを開始します。

ただし，データ変換アクティビティではトランザクションがコミットしません。これは，「表 3-7　トランザクションの開始とコミットのタイミング」で示した以外のアクティビティも同様です。

データ変換などのアクティビティがある場合で正常に処理が進行したときのトランザクションの範囲を次の図に示します。

図 3-36　正常に処理が進行した場合のトランザクションの範囲（サービス呼出のあとにデータ変換などのアクティビティがある場合）

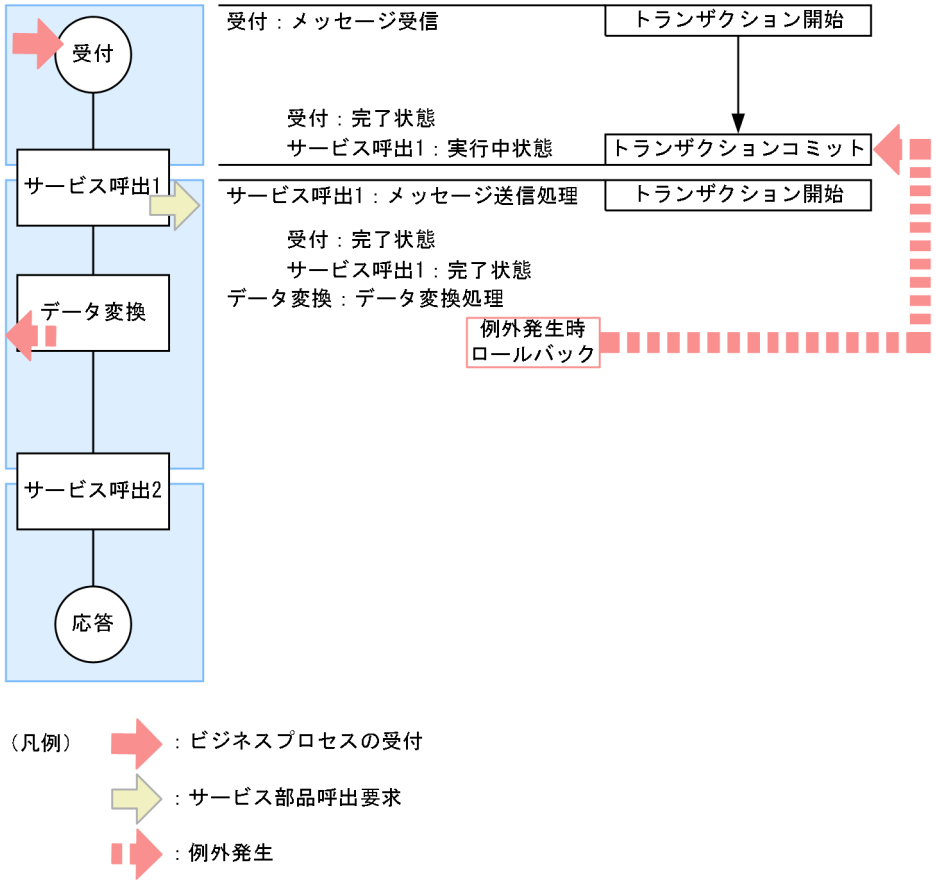


例外が発生した場合（サービス呼出のあとにデータ変換などのアクティビティがある場合）

サービス呼出 1 のあとにデータ変換処理でエラーとなった場合は，トランザクションがコミットした時点（受付：完了状態，サービス呼出 1：実行中状態）までロールバックします。

サービス呼出のあとにデータ変換などのアクティビティがある場合で例外が発生したときのトランザクションの範囲を次の図に示します。

図 3-37　例外が発生した場合のトランザクションの範囲（サービス呼出のあとにデータ変換などのアクティビティがある場合）



3.4.4　サービス呼出アクティビティのトランザクション

サービス呼出アクティビティの定義ごとのトランザクション，およびステータスの遷移を次に示します。

(1)　ビジネスプロセスの状態の互換性（HCSC サーバセットアップ定義ファイルの bp-status-compatible）に ON を指定した場合

ビジネスプロセスの状態の互換性に ON を指定した場合のサービス呼出アクティビティのトランザクション，およびステータスの遷移を次の表に示します。サービス呼出アクティビティの状態の互換性（HCSC サーバセットアップ定義ファイルの bp-invoke-status-compatible）の設定にかかわらず，同じ遷移となります。

表 3-8　ビジネスプロセスの状態の互換性（HCSC サーバセットアップ定義ファイルの bp-status-compatible）に ON を指定した場合の，トランザクション，およびステータスの遷移に関するプロパティ

サービス呼出アクティビティの動作	サービス呼出アクティビティの状態の互換性に ON を指定		サービス呼出アクティビティの状態の互換性に OFF を指定（デフォルト）	
	非同期	同期	非同期	同期
要求電文の送信	×	○（実行中）※	×	○（実行中）※
応答電文の受信（正常終了）	×	×	×	×
応答電文の受信（システム例外発生）	×	×	×	×

（凡例）

○：トランザクション制御を行います。

×：トランザクション制御を行いません。

注※

（）内はコミットされるステータスです。

HCSC サーバセットアップ定義ファイルの詳細は，マニュアル「サービスプラットフォーム リファレンス」の「6.4.3　HCSC サーバセットアップ定義ファイル」を参照してください。

(2)　ビジネスプロセスの状態の互換性（HCSC サーバセットアップ定義ファイルの bp-status-compatible）に OFF を指定した場合

ビジネスプロセスの状態の互換性に OFF を指定した場合のサービス呼出アクティビティのトランザクション，およびステータスの遷移を次の表に示します。サービス呼出アクティビティの状態の互換性（HCSC サーバセットアップ定義ファイルの bp-invoke-status-compatible）の設定によって，遷移が異なります。

表 3-9　ビジネスプロセスの状態の互換性（HCSC サーバセットアップ定義ファイルの bp-status-compatible）に OFF を指定した場合の，トランザクション，およびステータスの遷移に関するプロパティ

サービス呼出アクティビティの動作	サービス呼出アクティビティの状態の互換性に ON を指定		サービス呼出アクティビティの状態の互換性に OFF を指定（デフォルト）	
	非同期	同期	非同期	同期
要求電文の送信	○（実行中）※	○（実行中）※	○（実行中）※	○（実行中）※
応答電文の受信（正常終了）	○（完了）※	○（完了）※	×	×
応答電文の受信（システム例外発生）	○（エラー状態）※	○（エラー状態）※	○（エラー状態）※	○（エラー状態）※

（凡例）

○：トランザクション制御を行います。

×：トランザクション制御を行いません。

注※

（）内はコミットされるステータスです。

なお，サービス呼出アクティビティの実行が正常に完了したあとにアクティビティでエラーが発生した場合，サービス呼出アクティビティでの非同期サービスのキューへのメッセージ送信，およびDBアダプタを呼び出すデータベースへの更新処理は有効となります。

HCSCサーバセットアップ定義ファイルの詳細は，マニュアル「サービスプラットフォーム リファレンス」の「6.4.3　HCSCサーバセットアップ定義ファイル」を参照してください。

3.4.5　スコープの開始時および終了時にコミットする設定を選択した場合のトランザクション

スコープアクティビティは，［スコープアクティビティ］ダイアログで，［このスコープの開始と終了時のみコミットする］を選択した場合，スコープの開始時および終了時にトランザクションを制御します。

トランザクションを制御するスコープアクティビティのトランザクションについて説明します。

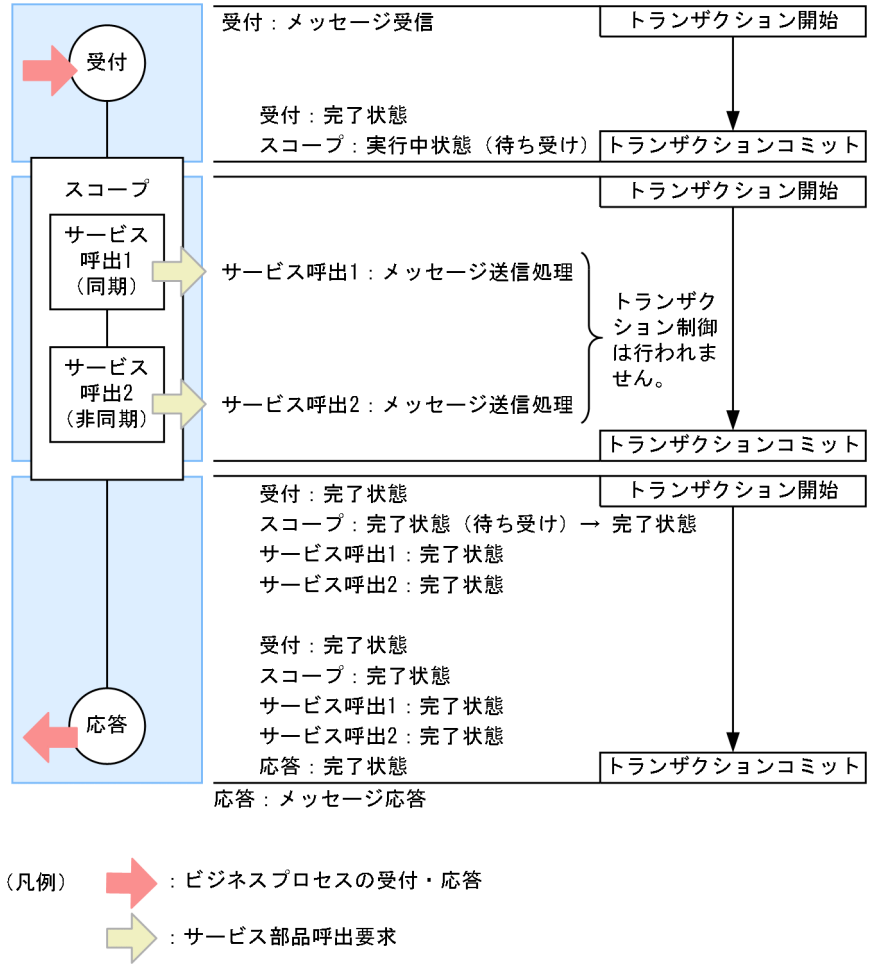
(1)　スコープアクティビティが正常終了した場合

スコープアクティビティの開始処理で，スコープアクティビティの状態を実行中状態（待ち受け）にし，トランザクションをコミットしたあと，新たなトランザクションを開始します。そのあと，スコープアクティビティ内部に定義されたアクティビティを実行します。このとき，スコープアクティビティ内に定義されたアクティビティではトランザクションの制御は行われません。

スコープアクティビティの終了処理で，スコープアクティビティの状態を完了状態（待ち受け）にし，トランザクションをコミットしたあと，新たなトランザクションを開始します。そのあと，スコープアクティビティの状態を完了状態にします。このとき，状態は遷移しますが，トランザクションはコミットしません。

スコープアクティビティが正常終了した場合のトランザクションの範囲を次の図に示します。

図 3-38　スコープアクティビティが正常終了した場合のトランザクションの範囲



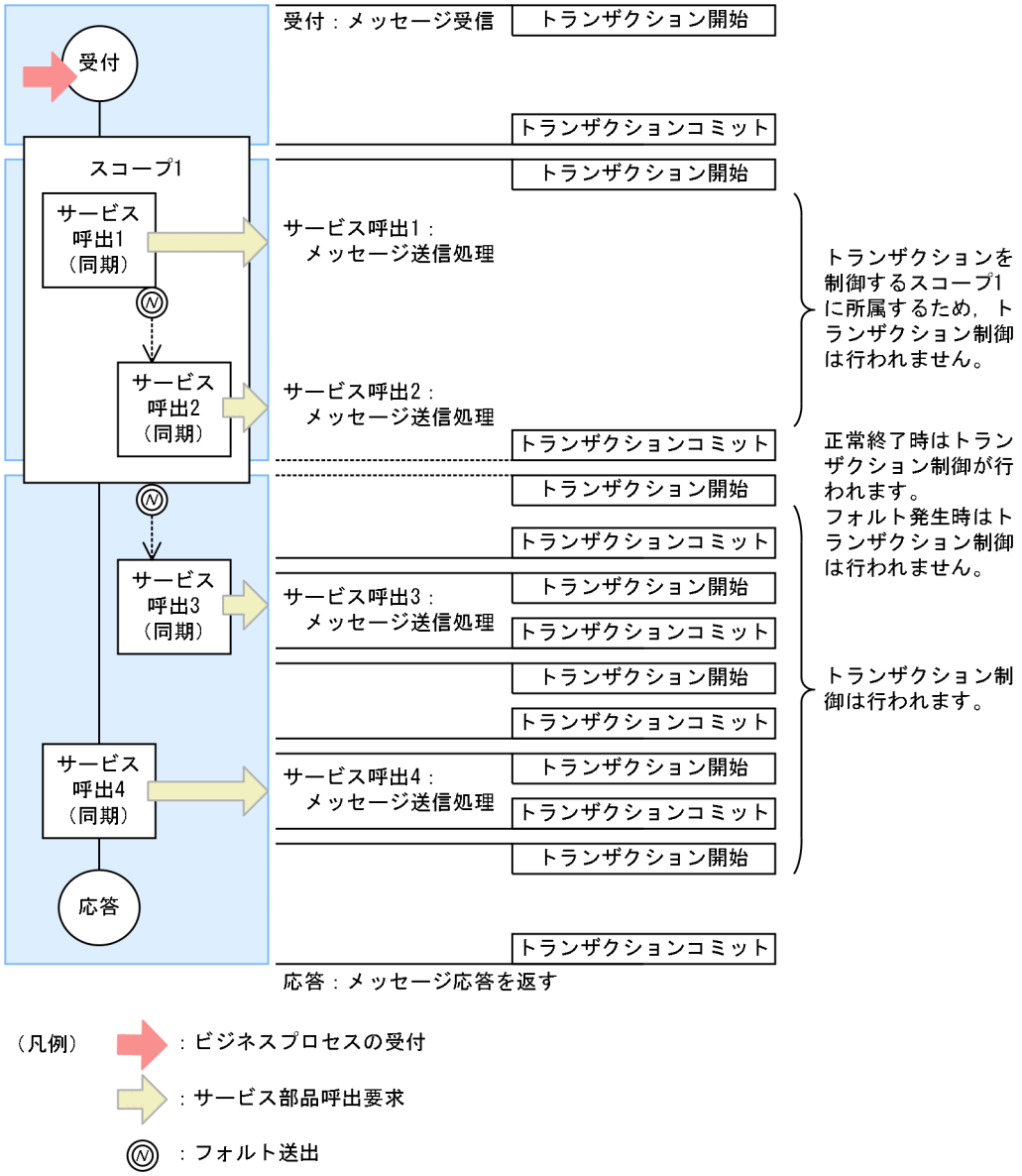
参考

トランザクションを制御するスコープアクティビティのトランザクション制御の範囲について説明します。

- トランザクションを制御するスコープアクティビティ内部に定義されているアクティビティでは，トランザクションの制御は行われません。
- トランザクションを制御するスコープアクティビティのフォルト処理を行うアクティビティでは，トランザクションの制御は行われます。
- トランザクションを制御するスコープアクティビティ以降に定義されたアクティビティでは，トランザクションの制御は行われます。

トランザクションを制御するスコープアクティビティのトランザクション制御の範囲を次の図に示します。

図 3-39　トランザクションを制御するスコープアクティビティのトランザクション制御の範囲

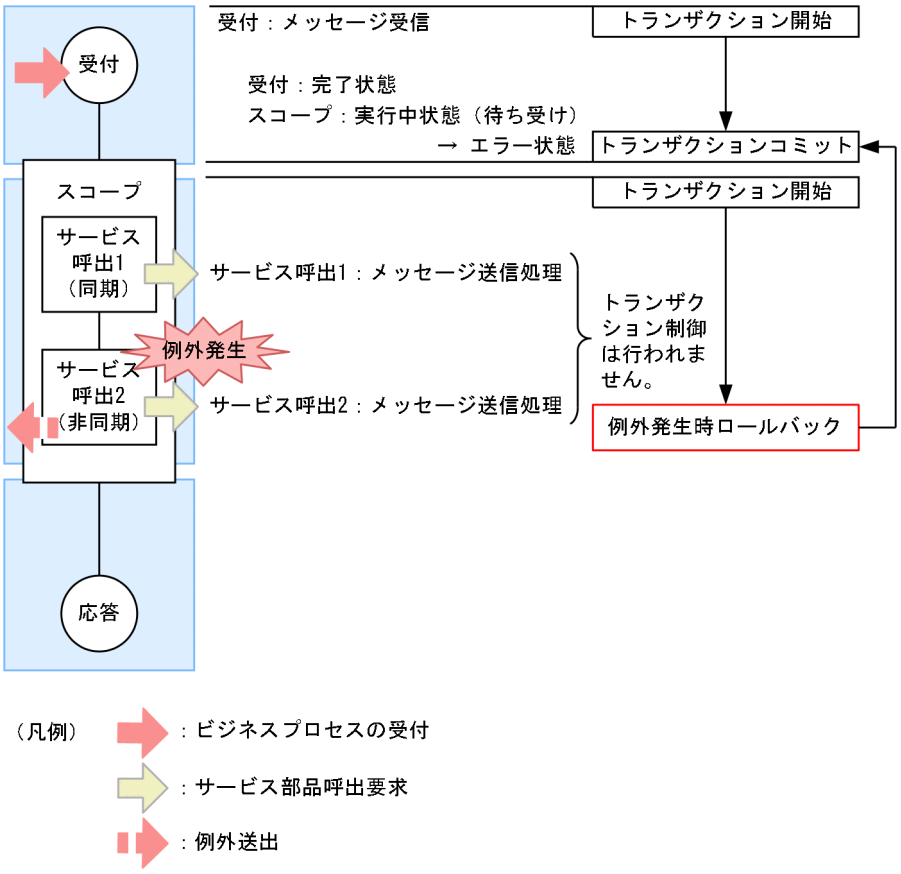


(2)　スコープアクティビティ内部でシステム例外が発生した場合

スコープアクティビティ内部でシステム例外が発生した場合，スコープアクティビティは，トランザクションがコミットした時点（受付：完了状態，スコープ：実行中状態（待ち受け）までロールバックしたあと，スコープの状態にエラー発生（Error）を設定して，トランザクションをコミットします。

スコープアクティビティ内部でシステム例外が発生した場合のトランザクションの範囲を次の図に示します。

図 3-40　スコープアクティビティ内部でシステム例外が発生した場合のトランザクションの範囲

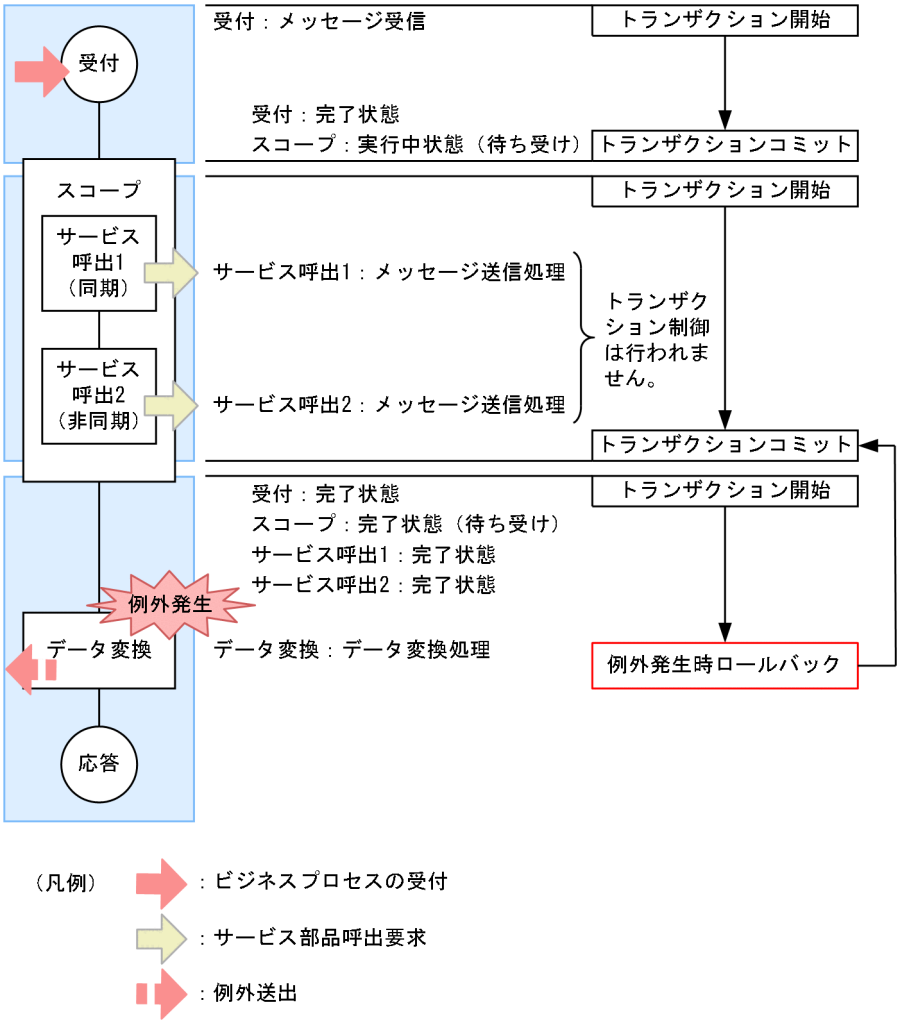


(3)　スコープアクティビティ以降のアクティビティでシステム例外が発生した場合

スコープアクティビティ以降に定義されたアクティビティでシステム例外が発生した場合，トランザクションがコミットした時点（スコープアクティビティ：完了状態（待ち受け））までロールバックします。

スコープアクティビティ以降に定義されたアクティビティでシステム例外が発生した場合のトランザクションの範囲を次の図に示します。

図 3-41　スコープアクティビティ以降に定義されたアクティビティでシステム例外が発生した場合のトランザクションの範囲



(4)　スコープアクティビティ内部でフォルトが発生した場合

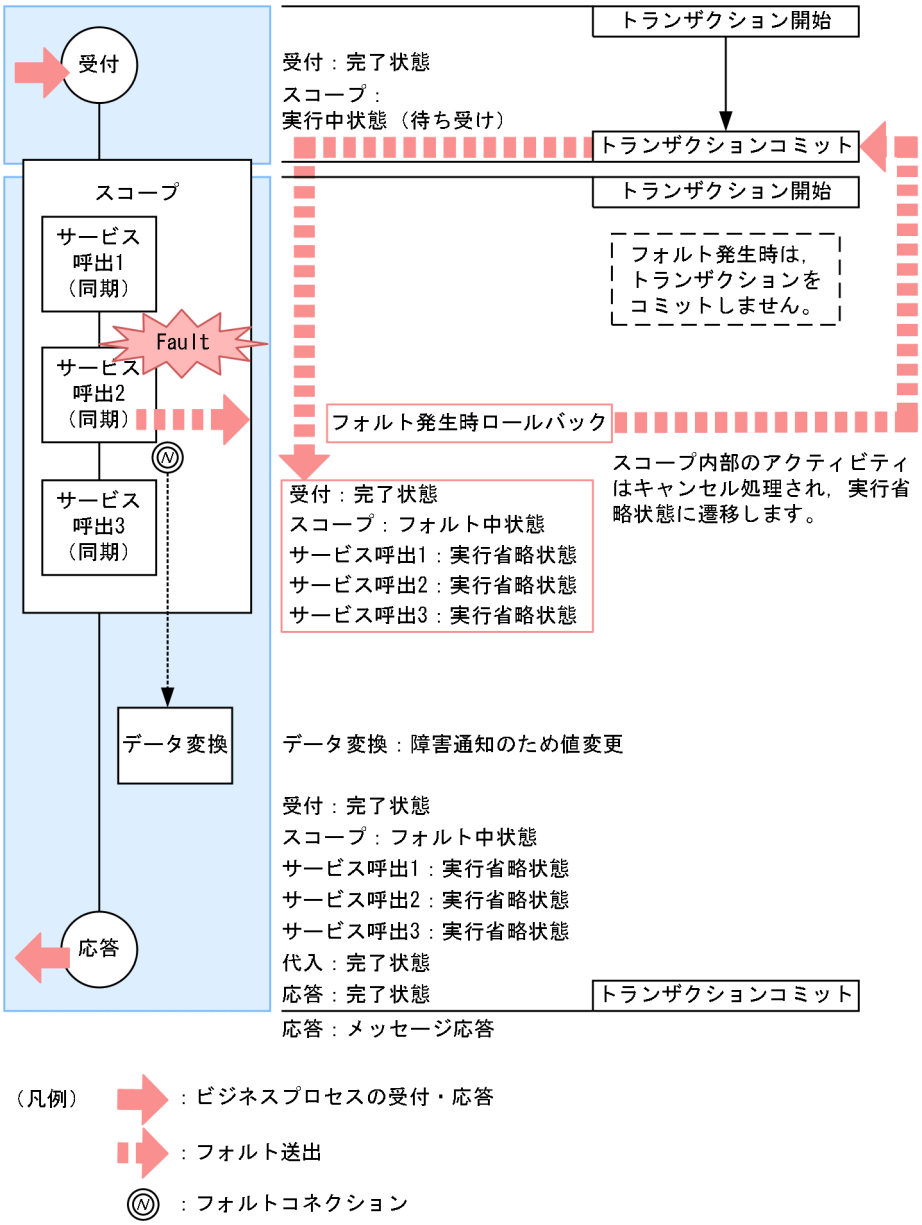
トランザクション制御を行うスコープアクティビティの内部に定義されたアクティビティでフォルトが発生し，トランザクション制御を行うスコープアクティビティにフォルトが通知された場合，トランザクションがコミットした時点（受付：完了状態，スコープ：実行中状態（待ち受け）までロールバックしたあと，スコープアクティビティ内部に定義されたアクティビティの処理はキャンセルされます。

フォルトハンドラの定義を基に，対応するアクティビティを実行する場合は，スコープアクティビティの状態をフォルト中状態（Faulting）にします。このとき，状態は遷移しますが，トランザクションはコミットしません。

上位のスコープアクティビティにフォルトを通知する場合は，スコープアクティビティの状態をフォルト終了状態（Faulted）にします。この場合も状態は遷移しますが，トランザクションはコミットしません。上位のスコープアクティビティにフォルトを通知したあとの動作は，トランザクションを制御しないスコープアクティビティの動作と同じです。

スコープアクティビティ内部でフォルトが発生した場合のトランザクションの範囲を次の図に示します。

図 3-42　スコープアクティビティ内部でフォルトが発生した場合のトランザクションの範囲

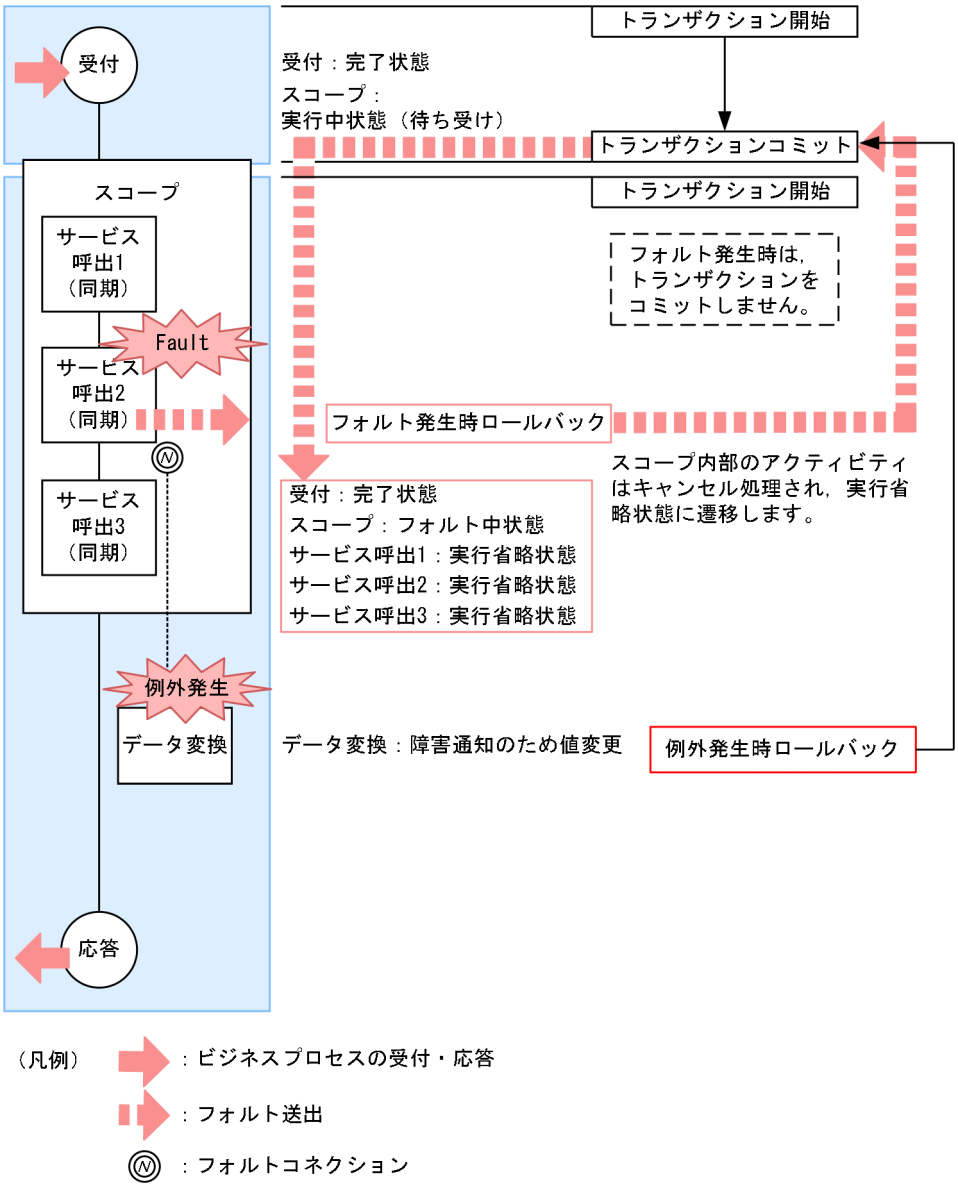


(5)　フォルト発生後にシステム例外が発生した場合

フォルトハンドラの定義を基に対応するアクティビティを実行し，システム例外が発生した場合，スコープアクティビティは実行中状態（待ち受け）までロールバックします。

フォルト発生後にシステム例外が発生した場合のトランザクションの範囲を次の図に示します。

図 3-43　フォルト発生後にシステム例外が発生した場合のトランザクションの範囲



3.4.6　スコープアクティビティのトランザクション

スコープアクティビティのトランザクションおよびステータスの遷移を次の表に示します。

表 3-10　スコープアクティビティのトランザクションおよびステータスの遷移

スコープアクティビティの動作	トランザクションを制御しない		トランザクションを制御する	
	遷移するステータス	トランザクション制御	遷移するステータス	トランザクション制御
開始	実行中	×	実行中（待ち受け）	○
終了	完了	×	完了（待ち受け）	○

スコープアクティビティの動作	トランザクションを制御しない		トランザクションを制御する	
	遷移するステータス	トランザクション制御	遷移するステータス	トランザクション制御
終了	完了	×	完了	×
システム例外発生	−	×	エラー	○
フォルト発生（コネクション処理中）	フォルト中	×	フォルト中	×
フォルト発生（コネクション処理完了）	フォルト終了	×	フォルト終了	×
フォルトを上位に通知	フォルト終了	×	フォルト終了	×
省略	省略	×	省略	×
未省略	未省略	×	未省略	×

（凡例）

○：トランザクションをコミットします。

×：トランザクションをコミットしません。

−：ロールバックするためステータスは遷移しません。

3.4.7　応答アクティビティのトランザクション

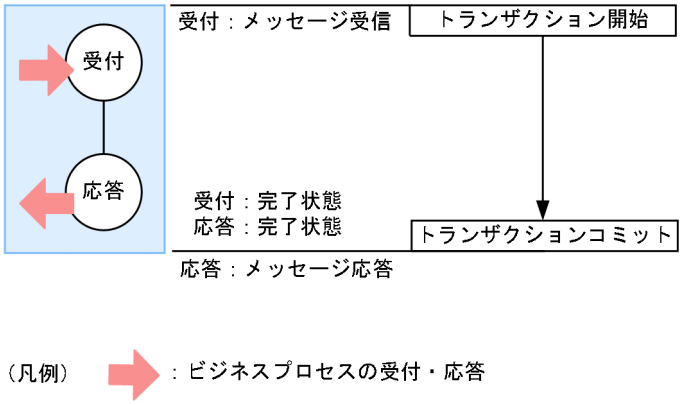
この項では，応答アクティビティのトランザクション制御について説明します。

応答アクティビティのあとに定義されたアクティビティを非同期に実行する方法については，「3.8　応答アクティビティのあとに定義されたアクティビティの非同期実行」を参照してください。

(1)　応答アクティビティのあとにアクティビティが定義されていない場合のトランザクション制御

応答アクティビティのあとにアクティビティが定義されていない場合，応答アクティビティの実行時に応答アクティビティの状態を完了状態（Completed）に遷移し，メッセージを応答する直前にトランザクションをコミットします。

図 3-44　応答アクティビティのあとにアクティビティが定義されていない場合のトランザクション制御



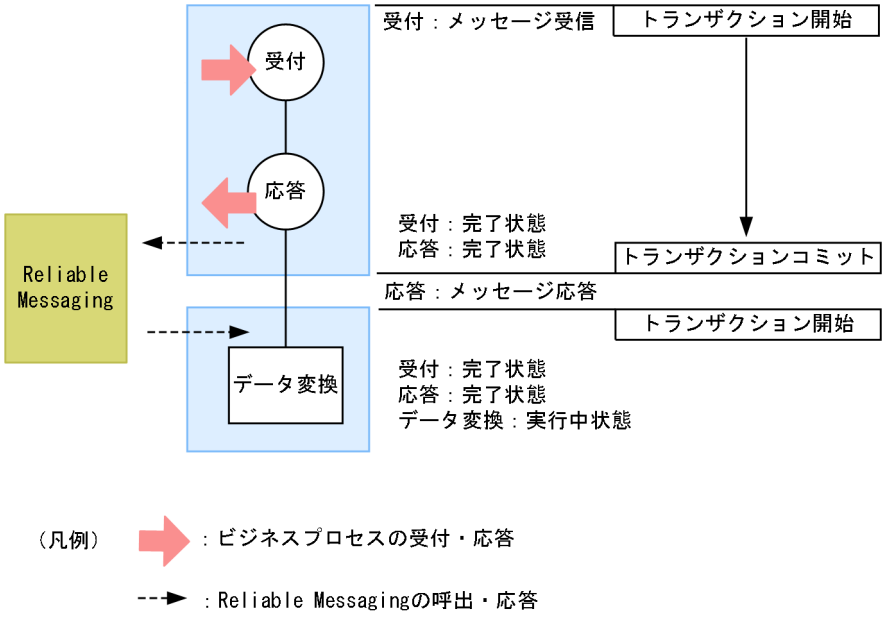
(2)　応答アクティビティのあとにアクティビティが定義されている場合のトランザクション制御

(a)　応答後のアクティビティ実行に Reliable Messaging を使用する場合

応答アクティビティのあとにアクティビティが定義されていて，応答後のアクティビティ実行に Reliable Messaging を使用する場合の，トランザクション制御を次に示します。

1. 応答アクティビティのあとにアクティビティが定義されていて，応答後のアクティビティ実行に Reliable Messaging を使用する場合，応答アクティビティ実行時に応答アクティビティの状態を完了状態（Completed）に遷移します。
2. 応答のあとに定義されたアクティビティの非同期実行の要求処理を Reliable Messaging に対して実行し，メッセージを応答する直前にトランザクションをコミットします。
3. Reliable Messaging から，応答アクティビティのあとに定義されたアクティビティの非同期実行を実行します。
4. 応答アクティビティのあとに定義されたアクティビティを実行する前に，トランザクションが開始されます。

図 3-45　応答アクティビティのあとにアクティビティが定義されていて，応答後のアクティビティ実行に Reliable Messaging を使用する場合のトランザクション制御

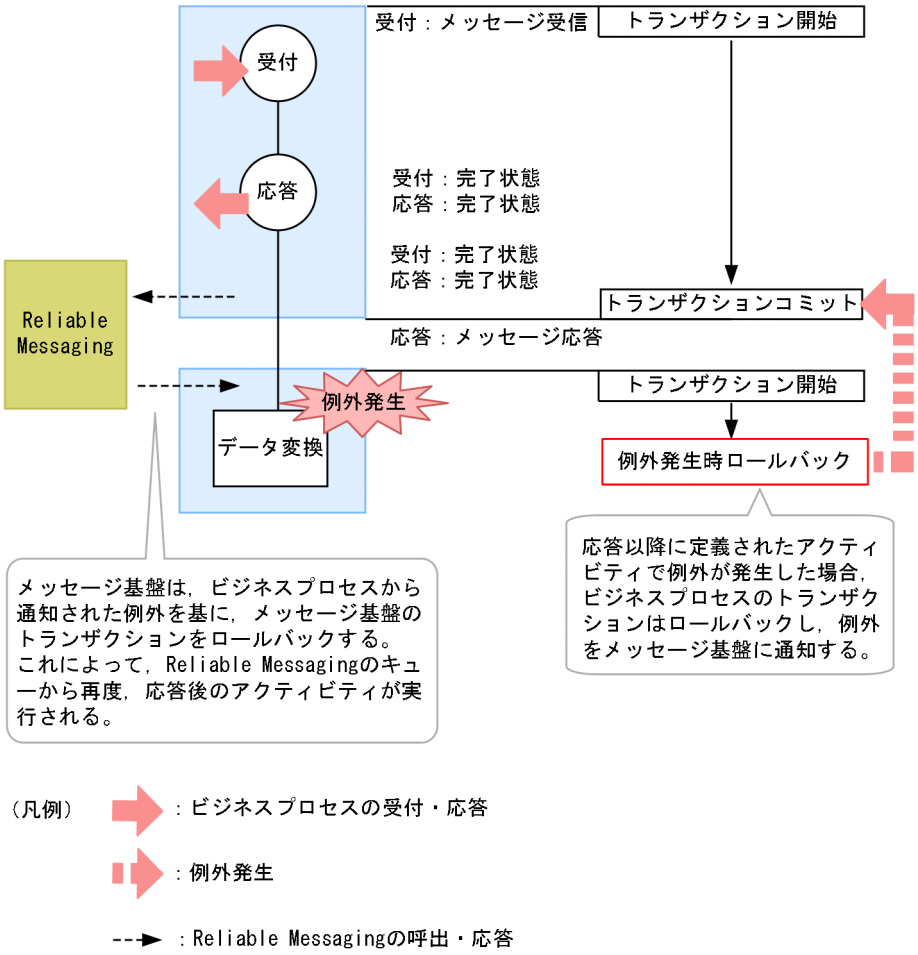


- 応答アクティビティのあとに定義されたアクティビティで例外が発生した場合

  応答後のアクティビティ実行に Reliable Messaging を使用し，応答アクティビティのあとに定義されたアクティビティで例外が発生した場合のトランザクション制御を次に示します。

  1. 応答アクティビティは完了状態（Completed）までロールバックします。
  2. ロールバックによって，Reliable Messaging から応答アクティビティのあとに定義されたアクティビティが再度実行されます。

図 3-46　応答後のアクティビティ実行に Reliable Messaging を使用し，応答アクティビティのあとに定義されたアクティビティで例外が発生した場合のトランザクション制御



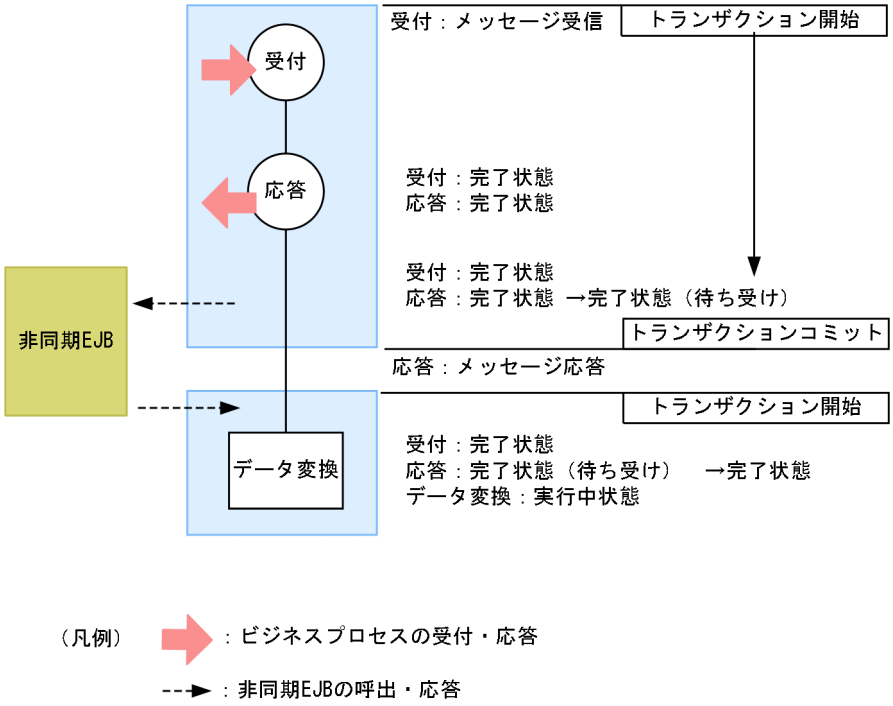
Reliable Messaging から再送されるかどうかは，配送回数の最大値（RMMaxDeliveryNum）の設定値に依存します。詳細については，マニュアル「Reliable Messaging」を参照してください。

(b)　応答後のアクティビティ実行に非同期 EJB を使用する場合

応答アクティビティのあとにアクティビティが定義されていて，応答後のアクティビティ実行に非同期 EJB を使用する場合の，トランザクション制御を次に示します。

1. 応答アクティビティの実行時に，応答アクティビティの状態を完了状態（Completed）に遷移します。
2. 応答のあとに定義されたアクティビティの非同期実行の要求処理を非同期 EJB に対して実行し，応答アクティビティの状態を完了状態（待ち受け）（Completedwait）に遷移します。
3. メッセージを応答する直前に，トランザクションをコミットします。
4. 応答のあとに定義されたアクティビティの非同期実行を非同期 EJB から実行し，応答アクティビティの状態を完了状態（Completed）に遷移します。
5. 応答のあとに定義されたアクティビティを実行する前に，トランザクションが開始されます。

図 3-47　応答アクティビティのあとにアクティビティが定義されていて，応答後のアクティビティ実行に非同期 EJB を使用する場合のトランザクション制御

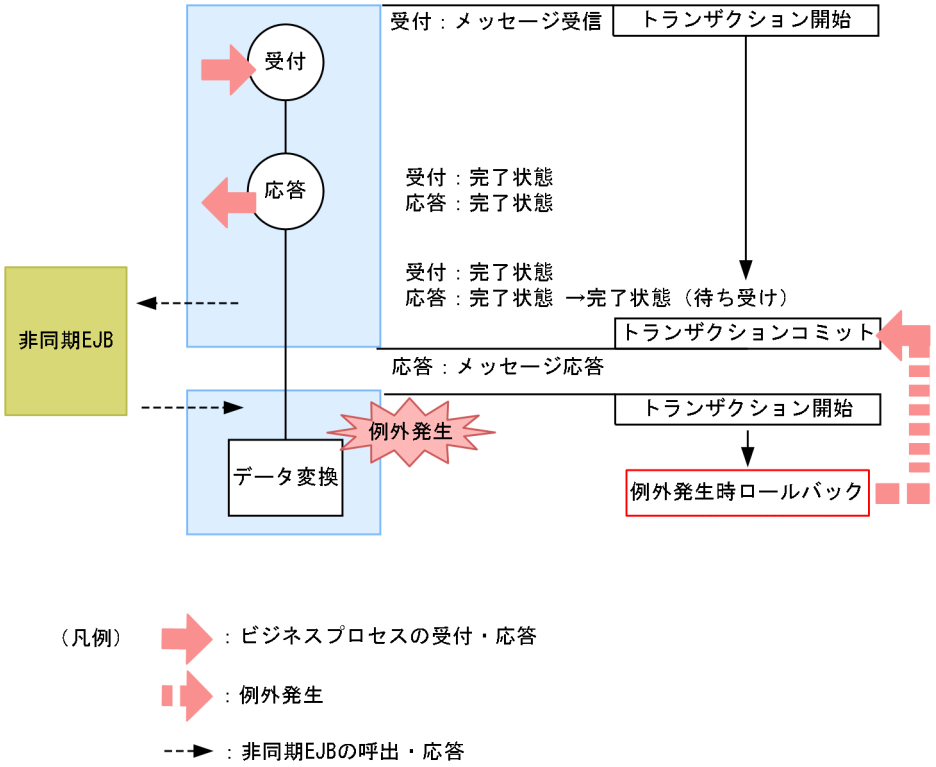


- 応答アクティビティのあとに定義されたアクティビティで例外が発生した場合

  応答後のアクティビティ実行に非同期 EJB を使用し，応答アクティビティのあとに定義されたアクティビティで例外が発生した場合，応答アクティビティは完了状態（待ち受け）（Completedwait）までロールバックします。

  応答アクティビティのあとに定義されたアクティビティは，再度実行されません。

図 3-48　応答後のアクティビティ実行に非同期 EJB を使用し，応答アクティビティのあとに定義されたアクティビティで例外が発生した場合のトランザクション制御

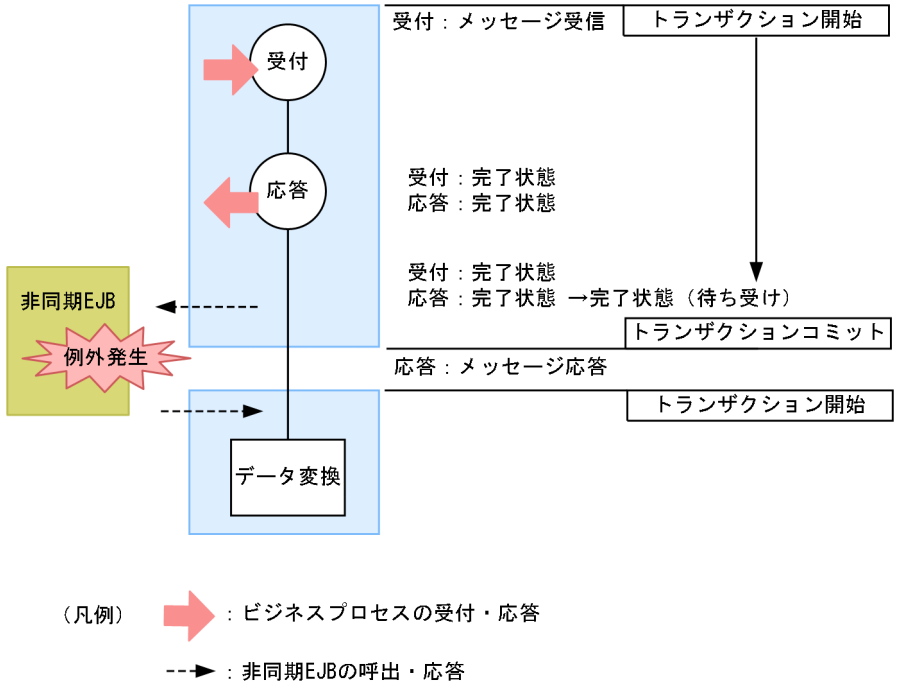


- 非同期 EJB 実行時に例外が発生した場合のトランザクション制御

  応答後のアクティビティ実行に非同期 EJB を使用し，非同期 EJB 実行時に例外が発生した場合，応答アクティビティのあとに定義されたアクティビティは実行されません。

  応答アクティビティは完了状態（待ち受け）（Completedwait）となります。

図 3-49　応答後のアクティビティ実行に非同期 EJB を使用し，非同期 EJB 実行時に例外が発生した場合のトランザクション制御



(3)　応答アクティビティのトランザクションおよびステータスの遷移

応答アクティビティのトランザクションおよびステータスの遷移を次の表に示します。

表 3-11　応答アクティビティのトランザクションおよびステータスの遷移

応答アクティビティの動作		応答後のアクティビティ実行に Reliable Messaging を使用する場合	応答後のアクティビティ実行に非同期 EJB を使用する場合
開始	遷移するステータス	完了	完了
	トランザクション制御	×※	×※
Reliable Messaging への非同期実行の要求	遷移するステータス	−	（実行不可）
	トランザクション制御	×	
Reliable Messaging からの非同期実行の応答	遷移するステータス	−	
	トランザクション制御	×	
非同期 EJB への非同期実行の要求	遷移するステータス	（実行不可）	完了（待ち受け）
	トランザクション制御		×※
非同期 EJB からの非同期実行の応答	遷移するステータス		完了
	トランザクション制御		×※
省略	遷移するステータス	省略	省略

応答アクティビティの動作		応答後のアクティビティ実行に Reliable Messaging を使用する場合	応答後のアクティビティ実行に非同期 EJB を使用する場合
省略	トランザクション制御	×	×
未省略	遷移するステータス	未省略	未省略
	トランザクション制御	×	×

（凡例）

×：トランザクションをコミットしません。

－：遷移しません。

注※

この時点ではコミットされませんが，それ以降のトランザクション制御のタイミングでコミットされます。

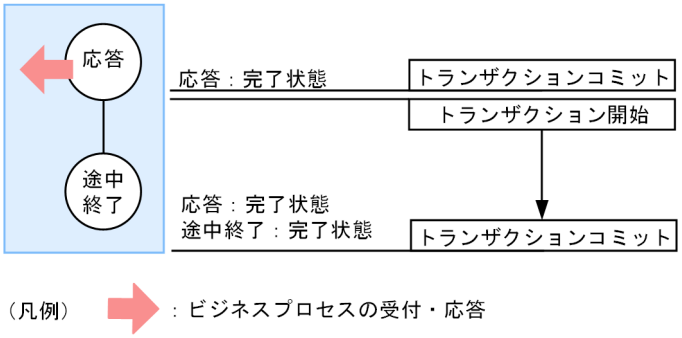
3.4.8　途中終了アクティビティのトランザクション

この項では，途中終了アクティビティのトランザクション制御について説明します。

(1)　途中終了アクティビティのトランザクション制御

途中終了アクティビティ実行時に，途中終了アクティビティの状態を完了状態（Completed）に遷移し，プロセス内の COMPLETED 状態以外のアクティビティを COMPLETED 状態に遷移して，トランザクションをコミットします。

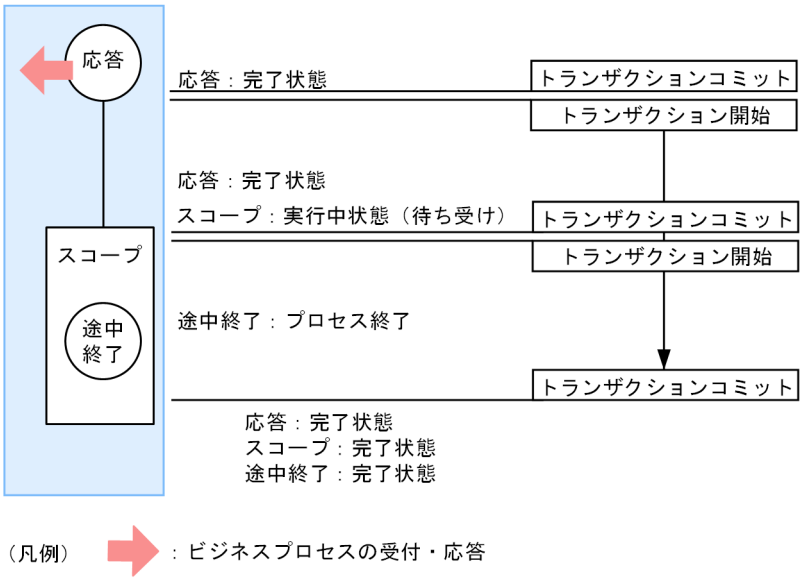
図 3-50　途中終了アクティビティのトランザクション制御



(2)　トランザクションの制御を行うスコープアクティビティ内部に途中終了アクティビティを定義した場合のトランザクション制御

トランザクション制御を行うスコープアクティビティ内部に途中終了アクティビティを定義した場合，途中終了アクティビティの終了で，途中終了アクティビティの状態を完了状態（Completed）に遷移し，プロセス内の COMPLETED 状態以外のアクティビティを COMPLETED 状態に遷移して，トランザクションをコミットします。

図 3-51　トランザクションの制御を行うスコープアクティビティ内部に途中終了アクティビティを定義した場合のトランザクション制御



3.5　ビジネスプロセスの再実行

ビジネスプロセス実行中に，実行基盤のあるマシンが停止したり，サービス部品の呼び出しに失敗したりした場合，障害が発生して中断したプロセスインスタンスを再実行できます。プロセスインスタンスを再実行することで，ビジネスプロセスの内容が処理できます。

プロセスインスタンスを再実行できる条件を次に示します。

- HCSC サーバが起動している場合
- ビジネスプロセスが起動している場合
- 同期（SessionBean）の標準受付が起動されている場合
- Reliable Messaging を使用する環境で，非同期（MDB（WS-R））の標準受付が起動されている場合
- フォルト送出機能を使用していない場合
  フォルト送出機能の詳細については，「4.7.1　システム例外をフォルトに変換する汎用フォルトの概要」を参照してください。

再実行対象となるステータスを次に示します。

プロセスのステータス	アクティビティ	
	種別	ステータス
実行中	サービス呼出	実行中
エラー状態	サービス呼出	エラー状態
実行中	待機	実行中
実行中	スコープ	実行中状態（待ち受け）
エラー状態	スコープ	エラー状態
実行中	スコープ	完了状態（待ち受け）
実行中	応答	完了状態（待ち受け）

3.5.1　運用環境の画面からの再実行

運用環境の画面から，中断したプロセスインスタンスを一件ずつ再実行できます。

画面によるプロセスインスタンスの再実行の手順については，マニュアル「サービスプラットフォーム システム構築・運用ガイド」の「6.1.5　プロセスインスタンスを再実行する」を参照してください。

3.5.2　コマンドからの再実行

コマンドを使用して，中断したプロセスインスタンスを 1 件ずつ再実行できます。

また，複数のプロセスインスタンスを一括して再実行することもできます。1 つの HCSC サーバ上にある複数のプロセスインスタンスを一括して再実行したり，HCSC サーバ名を指定して HCSC サーバ単位で再実行したりできます。プロセスインスタンスの実行履歴の一括再実行は，コマンドでだけ実行できます。

コマンドによるプロセスインスタンスの再実行の手順については，マニュアル「サービスプラットフォーム システム構築・運用ガイド」の「6.1.5　プロセスインスタンスを再実行する」を参照してください。

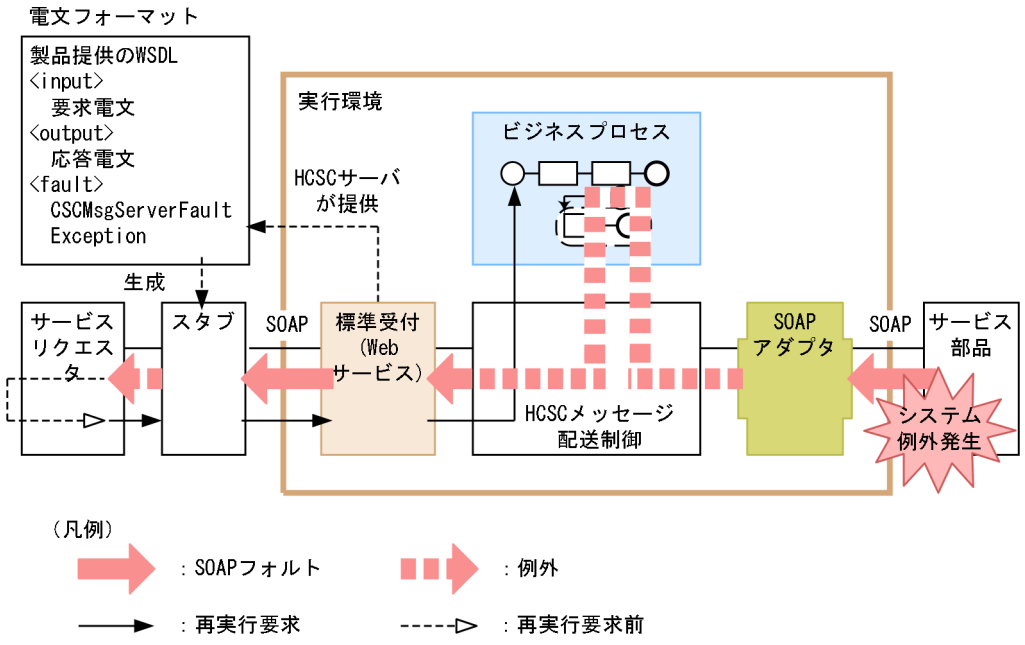
3.5.3　サービスリクエスタからの再実行（SOAP 通信）

サービス部品の呼び出し処理で例外になって中断したプロセスインスタンスは，システム運用管理者だけでなく，サービスリクエスタからも再実行できます。

サービスリクエスタから再実行するには，同期（Web サービス）の標準受付に再実行要求を出すサービスリクエスタを作成します。

サービス部品の呼び出しがエラーになり，ビジネスプロセスインスタンス ID がエラー情報として返ってきた場合は，そのビジネスプロセスインスタンス ID を指定して再実行します。SOAP 通信でのサービスリクエスタからの要求の再実行を次の図に示します。

図 3-52　SOAP 通信でのサービスリクエスタからの要求の再実行



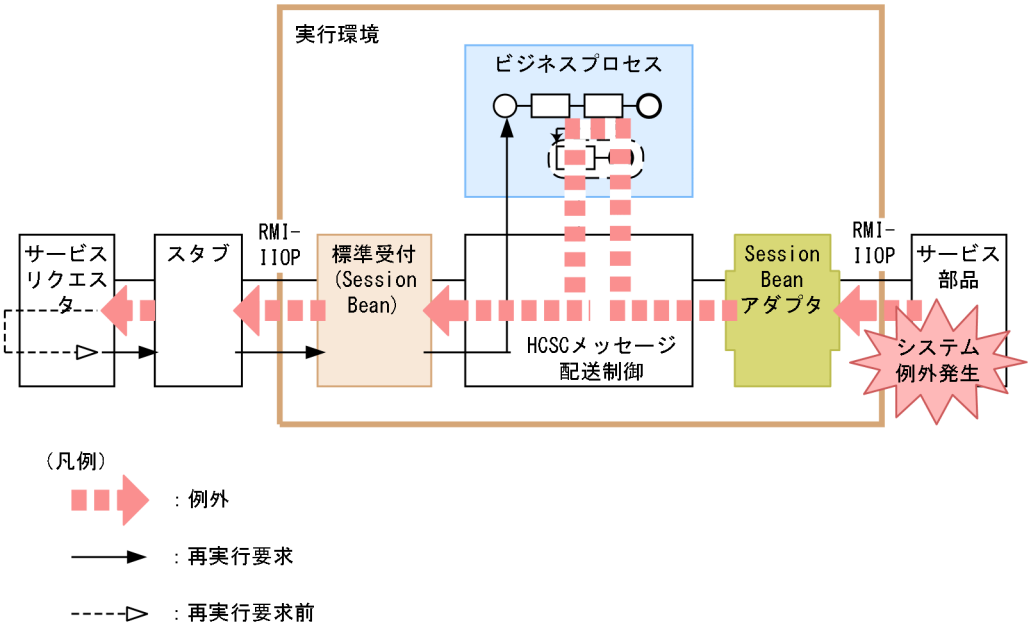
ビジネスプロセスの再実行を要求するサービスリクエスタの作成については，マニュアル「サービスプラットフォーム 開発ガイド 基本開発編」の「9.2.9　ビジネスプロセスの再実行を要求するサービスリクエスタの作成（Web サービス・SOAP 通信基盤）」を参照してください。

3.5.4　サービスリクエスタからの再実行（SessionBean）

同期（SessionBean）の標準受付に要求を出すサービスリクエスタからも，ビジネスプロセスの再実行要求ができます。サービスリクエスタから再実行するには，同期（SessionBean）の標準受付に要求を出すサービスリクエスタを作成します。

サービス部品呼び出しがエラーになり，ビジネスプロセスインスタンス ID が例外（エラー情報）として返ってきた場合は，そのビジネスプロセスインスタンス ID を指定して再実行します。SessionBean でのサービスリクエスタからのビジネスプロセスの再実行を次の図に示します。

図 3-53　SessionBean でのサービスリクエスタからのビジネスプロセスの再実行



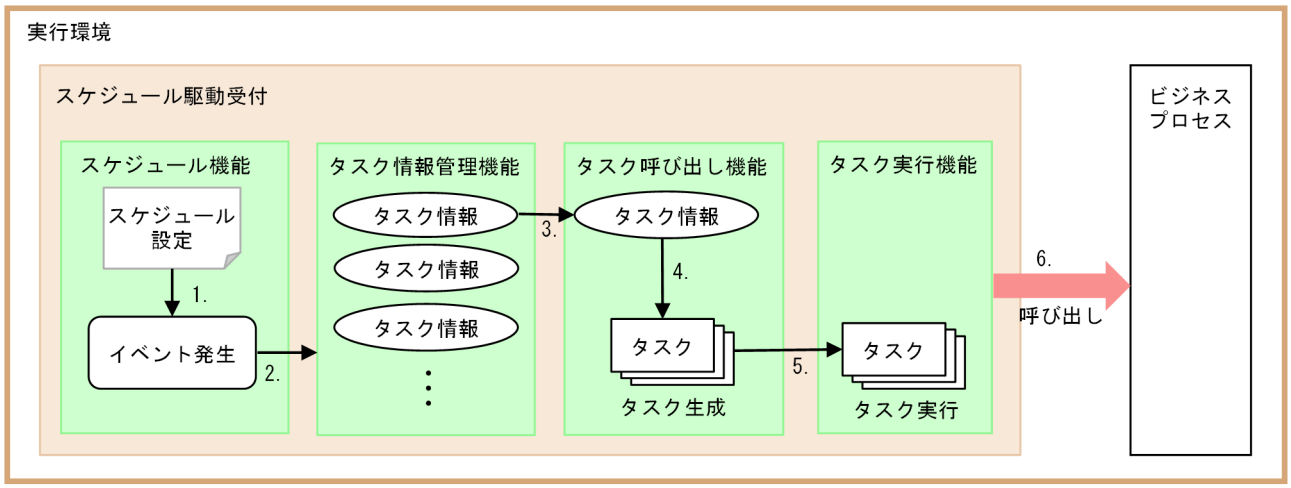
ビジネスプロセスの再実行を要求するサービスリクエスタの作成については，マニュアル「サービスプラットフォーム 開発ガイド 基本開発編」の「9.4.8　ビジネスプロセスの再実行を要求するサービスリクエスタの作成（SessionBean）」を参照してください。

3.6　ビジネスプロセスのスケジュール設定

スケジュール駆動受付を使用して，スケジュール設定に定義した間隔でタスクを実行し，ビジネスプロセスを呼び出すことができます。

ビジネスプロセスのスケジュール設定処理の流れを次の図に示します。

図 3-54　ビジネスプロセスのスケジュール設定処理の流れ



1. スケジュール駆動受付のスケジュール設定に定義したスケジュールでイベントが発生します。
2. タスク情報管理機能に，タスクの情報が登録されます。
3. タスクの呼び出し機能が，タスク管理情報機能からタスク情報を取得します。
4. タスクの呼び出し機能が，取得したタスク情報からタスクを生成します。
5. タスクの呼び出し機能が，タスク実行機能を呼び出します。
6. タスク実行機能が生成されたタスクを実行します。また，タスクの実行にあわせて，ビジネスプロセスが呼び出されます。

なお，1 つのスケジュール駆動受付から 1 つのビジネスプロセスに接続できます。複数のスケジュール駆動受付から 1 つのビジネスプロセスに接続することもできます。

スケジュール駆動受付で使用できる通信モデルは，非同期通信です。

3.6.1　スケジュール駆動受付でのスケジュールの設定

スケジュール駆動受付を使用して，設定されたスケジュールに従ってイベントを発生させます。

スケジュールの設定には，次の 2 つがあります。

- コンポーネント開始時にスケジュール処理を開始する設定
  スケジュールの処理が開始された時刻から，一定間隔でイベントを発生させます。

- 開始時刻を指定してスケジュール処理を開始する設定
  指定したスケジュールの開始時刻から，一定間隔でイベントを発生させます。

スケジュール設定は，スケジュール駆動受付定義ファイルで設定します。スケジュール設定に関して設定できる項目については，マニュアル「サービスプラットフォーム リファレンス」の「3.15.1　スケジュール駆動受付定義ファイル」を参照してください。

(1)　コンポーネント開始時にスケジュール処理を開始する設定

スケジュールの処理が呼び出されたあと，すぐにスケジュールが開始されます。スケジュールの開始後，一定間隔でイベントを発生させます。

スケジュール駆動受付で開始時刻を指定しない場合，デフォルトで周期実行が設定されます。

(2)　開始時刻を指定してスケジュール処理を開始する設定

指定したスケジュールの開始時刻から，一定間隔でイベントを発生させます。

スケジュール開始処理を呼び出した時刻以降，最初のスケジュールの開始時刻に，スケジュールが開始されます。

スケジュールの開始時刻は，スケジュール駆動受付コンフィグファイルで設定します。スケジュール駆動受付コンフィグファイルでの設定については，マニュアル「サービスプラットフォーム リファレンス」の「6.18.1　スケジュール駆動受付コンフィグファイル」を参照してください。

なお，スケジュール開始処理を呼び出した時刻は，メッセージ（KDEC84256-I）で確認できます。

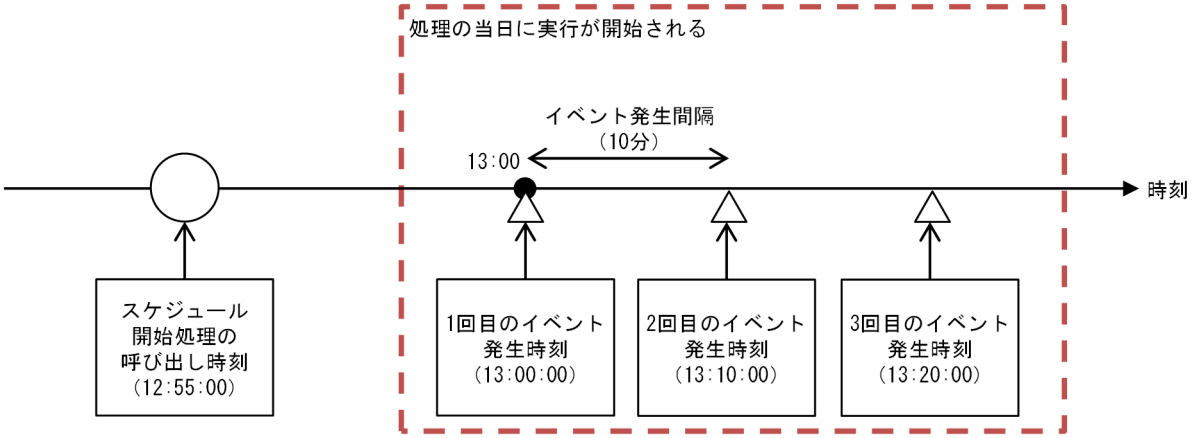
スケジュール開始処理を呼び出した時刻によって，スケジュールが開始される日付が異なります。

(a)　スケジュール開始処理を呼び出した時刻がスケジュールの開始時刻より前の場合

処理開始の当日にスケジュールが開始し，イベントが発生します。

例えば，スケジュールの開始時刻に「13:00:00」を設定し，イベントの発生間隔に「10 分」を指定した処理を呼び出した時刻が「12:55:00」の場合，1 回目のイベントはスケジュール開始処理を呼び出した当日の「13:00:00」に発生します。また 2 回目以降のイベントは，前回のイベント発生時刻からイベントの発生間隔を経て実行されます。

図 3-55　スケジュール開始処理を呼び出した時刻がスケジュールの開始時刻より前の場合



(b)　スケジュール開始処理を呼び出した時刻がスケジュールの開始時刻よりあとの場合

設定した開始時刻の形式によって，スケジュールの開始時刻が異なります。設定した開始時刻の形式とスケジュールの開始時刻の対応を次に示します。

表 3-12　設定した開始時刻の形式とスケジュールの開始時刻の対応

設定した開始時刻の形式	スケジュールの開始時刻
HH:mm:ss	スケジュール開始処理を呼び出した日の翌日の，指定した時刻に開始されます。
mm	スケジュール開始処理を呼び出した 1 時間後の，指定した分（mm）の時刻に開始されます。

（凡例）

HH：時間を表します（00～23）。

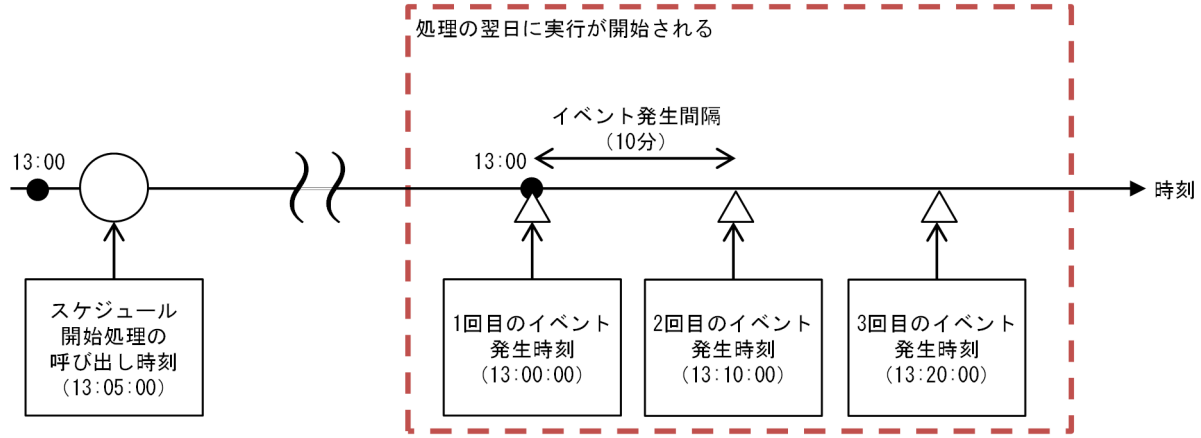
mm：分を表します（00～59）。

ss：秒を表します（00～59）。

スケジュールの開始時刻に，時:分:秒（HH:mm:ss）を設定した場合

スケジュールの開始時刻に「13:00:00」を設定し，イベントの発生間隔に「10 分」を指定した処理を呼び出した時刻が「13:05:00」の場合，1 回目のイベントはスケジュール開始処理を呼び出した翌日の「13:00:00」に発生します。また 2 回目以降のイベントは，前回のイベント発生時刻からイベントの発生間隔を経て実行されます。

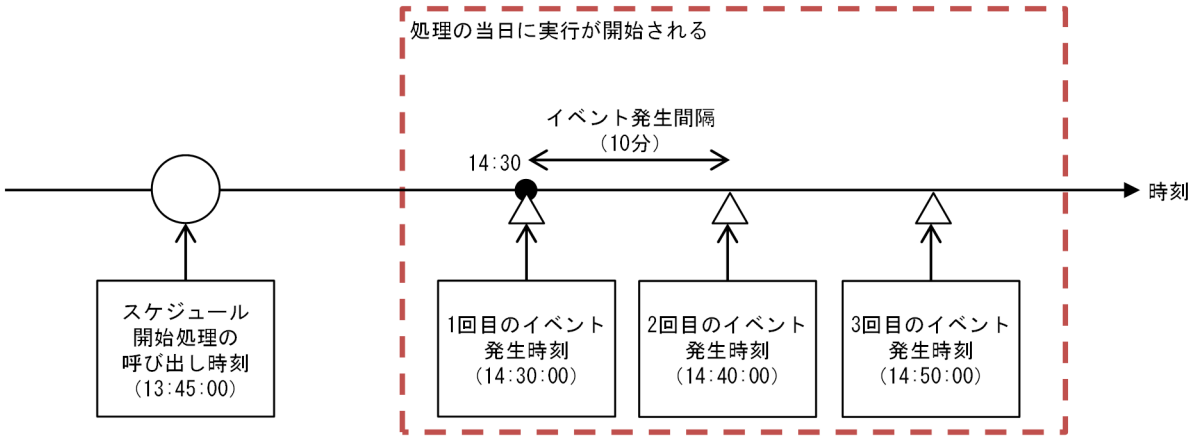
図 3-56　スケジュール開始処理を呼び出した時刻がスケジュールの開始時刻より前の場合



スケジュールの開始時刻に，分（mm）を設定した場合

スケジュールの開始時刻に「30」を設定し，イベントの発生間隔に「10 分」を指定した処理を呼び出した時刻が「13:45:00」の場合，1 回目のイベントは「14:30:00」に発生します。また 2 回目以降のイベントは，前回のイベント発生時刻からイベントの発生間隔を経て実行されます。

図 3-57　スケジュールの開始時刻に，分（mm）を設定した場合



(3)　スケジュール駆動受付のイベント処理

イベント発生時には，イベント処理が行われます。

イベント処理では，イベント発生時にスケジュール駆動受付の状態を取得して，イベント処理が続行できるかどうかを判断します。イベント処理は，スケジュール駆動受付の状態によって処理が異なります。スケジュール駆動受付の状態とイベント処理の対応を次に示します。

表 3-13　スケジュール駆動受付の状態とイベント処理の対応

スケジュール駆動受付の状態	イベント処理
active（開始状態）	イベント処理が続行できると判断されます。タスク情報をタスク情報管理機能に登録します。 なお，現在実行を待機しているタスクの数が，待機できるタスク数の最大数を超えようとした場合については，マニュアル「サービスプラットフォーム システム構築・運用ガイド」の「7.7.21(7)(b)　ビジネスプロセスの処理時に発生する障害」を参照してください。

スケジュール駆動受付の状態	イベント処理
上記以外（stopping（停止処理中）など）	イベント処理が続行できないと判断されます。 タスク情報をタスク情報管理機能に登録しないで，イベント処理を終了します。

3.6.2　スケジュール駆動受付でのタスク情報管理機能

実行するタスクの情報を管理する機能のことを，タスク情報管理機能といいます。タスク情報管理機能が実行する処理を，次に示します。

(1)　タスク情報の管理

タスク情報管理機能は，タスクの情報として「イベントの発生時刻」を管理します。イベント発生時に，スケジュール機能によって，タスクの情報がタスク情報管理機能に登録されます。登録されたタスクの情報をイベント発生順に管理して，タスク実行機能に提供します。

(2)　登録されているタスク情報の返却

タスク情報管理機能は，タスク情報管理機能に登録されているタスクの情報のうち，条件を満たすタスクの情報の返却を行います。返却するタスクの情報は，登録されているタスクの情報の中で，最も古いものになります。最も古いタスクの情報から順に返却します。

なお，タスクの情報が登録されていない場合は，タスクの情報が登録されるまでタスクの情報の返却を待ちます。

(3)　返却されたタスク情報の削除

タスク情報管理機能が返却したタスクの情報は，タスク情報管理機能の管理対象から削除されます。

次のどちらかに該当する場合で，かつ登録されているタスク情報が存在するときは，管理されているすべてのタスク情報が削除されます。

- スケジュール駆動受付の停止時
- HCSC サーバの停止時

(4)　待機できるタスク数の制御

タスク情報管理機能では，実行を待機できるタスク数を制御できます。

実行を待機できるタスク数の最大数を超えて，実行を待機するタスクが登録されようとした場合は，「3.6.1 スケジュール駆動受付でのスケジュールの設定」の「表 3-13 スケジュール駆動受付の状態とイベント処理の対応」を参照してください。

また，実行を待機できるタスクの最大数は，スケジュール駆動受付定義ファイルの scdrecp.waiting-execution-tasks.maximum プロパティで設定できます。scdrecp.waiting-execution-tasks.maximum プロパティの詳細については，マニュアル「サービスプラットフォーム リファレンス」の「3.15.1　スケジュール駆動受付定義ファイル」を参照してください。

なお，タスクの処理順序が逐次処理で，かつ新たにタスクを生成しない場合，実行を待機するタスクは存在しません。そのため，実行を待機するタスクの数を制御することはできません。タスクの処理順序が逐次処理で，かつ新たにタスクを生成しない場合の詳細については，「3.6.3(2)(b)　新たにタスクを生成しない」を参照してください。

3.6.3　スケジュール駆動受付でのタスクの呼び出し機能

タスクの呼び出し機能は，タスク情報管理機能からタスクの情報を取得し，スケジュール駆動受付を使用してタスク実行機能を呼び出して，イベント発生時に作成されたタスクを実行します。

タスクの呼び出し機能は，スケジュール駆動受付の状態が active（開始状態）の場合に開始します。また，スケジュールの実行が終了した場合に停止します。実行中のタスクが存在する場合は，タスクの処理が完了するまで停止しません。

(1)　タスクの処理順序

通常の動作では，タスクはイベントが発生した順に処理されます。ただし，イベントの発生時にすでに実行中のタスクがある場合や，実行を待機しているタスクがある場合，イベントが発生した順にタスクが処理されないことがあります。

イベントの発生時にすでに実行中のタスクがある場合は，タスクを「逐次」または「並列」で処理できます。なお，実行中のタスク処理をキャンセルすることはできません。

タスクの処理順序の詳細については，「(2) タスクの逐次処理」および「(3) タスクの並列処理」を参照してください。

スケジュール駆動受付の呼び出し先のビジネスプロセスを同時実行するかどうかによって，タスクの処理順序を選択する必要があります。ビジネスプロセスの同時実行の有無とタスクの処理順序の関係について，次に示します。

表 3-14　ビジネスプロセスの同時実行の有無とタスクの処理順序の関係

ビジネスプロセスの同時実行の有無	タスクの処理順序
同時実行できない	逐次
同時実行できる	逐次または並列

逐次処理，および並列処理の設定は，スケジュール駆動受付定義ファイルの「urecp-scd.task.processing-max.parallel.count」で設定します。

スケジュール駆動受付定義ファイルの詳細については，マニュアル「サービスプラットフォーム リファレンス」の「3.15.1　スケジュール駆動受付定義ファイル」を参照してください。

(2)　タスクの逐次処理

逐次処理では，イベントの発生順に逐次タスクを生成して実行します。

イベントの発生時にすでに実行中のタスクがある場合は，新たにタスクを生成して実行するかどうかを設定できます。新たにタスクを生成して実行するかどうかの設定は，スケジュール駆動受付定義ファイルの「scdrecp.allow.new-task.executing」で設定します。

ただし，次の場合はタスクが実行中であるかを判断できないため，実行中のタスクがあるかどうかに関係なく，新たにタスクを生成する動作となります。

- HCSC サーバやスケジュール駆動受付を再起動した場合
- HCSC サーバがクラスタ構成の場合

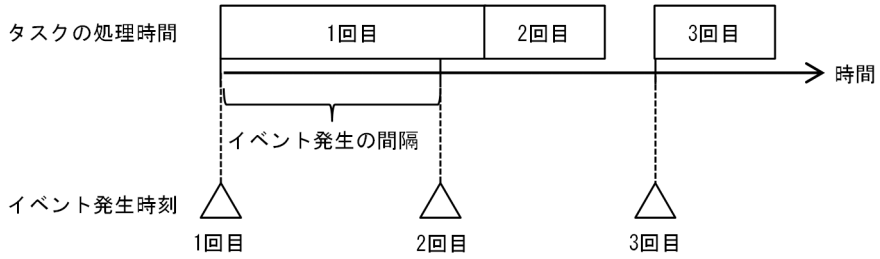
(a)　新たにタスクを実行する

イベントの発生時にすでに実行中のタスクがあった場合，新たにタスクを生成して実行を試みることができます。スケジュール駆動受付定義ファイルの「scdrecp.allow.new-task.executing」に true を設定します。

ただし，実行中のタスクの処理が終了するまで，新たなタスクの実行は，待機状態となります。実行中のタスクが終了した場合，イベントが発生した順に，待機しているタスクが実行されます。

また，タスクの実行の待機時間は，無制限となります。

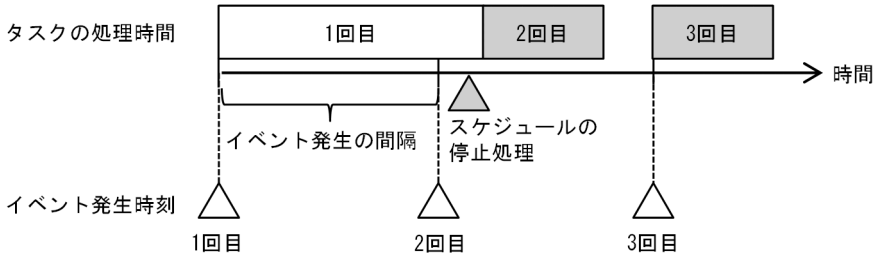
図 3-58　新たにタスクを実行するときの処理



この例では，1 回目のタスクの処理時間が，2 回目のイベント発生時刻に重なっています。この場合，2 回目のイベント発生時刻にタスクが生成されますが，1 回目のタスクの処理が終了するまで 2 回目のイベント処理で生成したタスクは実行されません。3 回目のイベントでは，3 回目のイベント発生時刻に重なっていたタスクの処理がないため，3 回目のイベント発生時刻に，タスクが実行されます。

なお，実行を待機しているタスクがある状態でスケジュール駆動受付が停止した場合，待機しているタスクの処理がすべて終了する前にスケジュールが停止してしまい，すべてのタスクが実行されないことがあります。

図 3-59　タスクの実行中にスケジュールの停止処理を実行した場合の動作（待機中のタスクを実行しない）

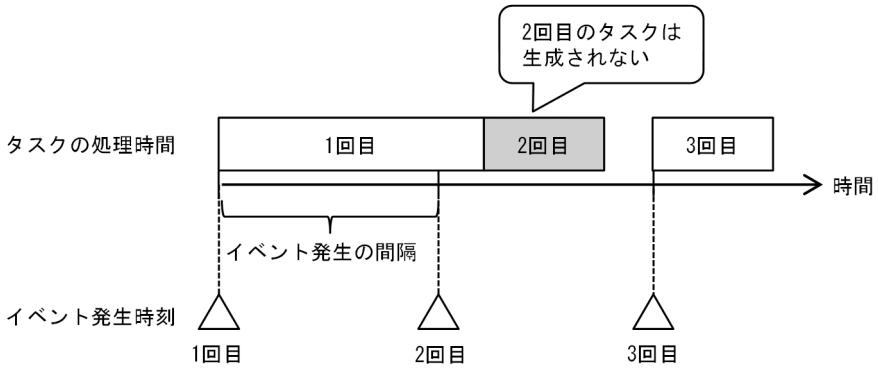


この例では，1 回目のタスクの処理時間が 2 回目のイベント発生時刻に重なっているため，2 回目のイベントで実行するタスク処理の実行を待機します。また，2 回目のイベント発生時刻より，あとにスケジュールの停止処理を実行しているため，待機している 2 回目のタスクの処理は実行されません。さらに 3 回目のイベント発生時刻までに，スケジュールが停止しているため，3 回目のイベントは発生しません。

(b)　新たにタスクを生成しない

イベントの発生時にすでに実行中のタスクがあった場合，新たにタスクを生成しないときは，スケジュール駆動受付定義ファイルの「scdrecp.allow.new-task.executing」に false を設定します。新たにタスクを生成しないときは，実行中のタスクの処理完了後の，次のイベント発生時にタスクが生成され，実行されます。

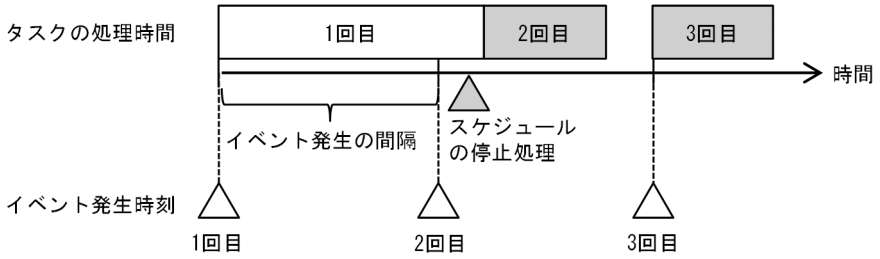
図 3-60　新たにタスクを生成しないときの処理



この例では，1 回目のタスクの処理時間が 2 回目のイベント発生時刻に重なっています。2 回目のイベント発生時刻にタスクが実行中のため，2 回目のイベントではタスクは生成されません。3 回目のイベントは，3 回目のイベント発生時刻に重なっているタスクがないため，タスクが生成・実行されます。

また，タスクの実行中にスケジュールの停止処理を実行した場合，実行中のタスクの処理が終了したあと，タスクは実行されません。

図 3-61　タスクの実行中にスケジュールの停止処理を実行した場合の動作（新たにタスクを実行しない）



この例では，1 回目のタスクの処理時間が 2 回目のイベント発生時刻に重なっているため，2 回目のイベントではタスクを生成されません。また，2 回目のイベント発生時刻よりあとにスケジュールの停止処理を実行しているため，スケジュールの停止の実行後はイベントが発生しません。

(c)　イベント発生の間隔を超えてもタスクの処理が終了しない場合

タスクの処理順序が「逐次」で，かつイベント発生の間隔を超えてもタスクの処理が終了しない場合，実行中のタスクがあることを警告メッセージ（KDEC84208-W，KDEC84209-W）としてメッセージログに出力します。なお，スケジュールの実行は継続されます。

また，エラー発生時のタスクの実行の抑止を行う場合で，現在のイベント発生時にエラー発生回数がしきい値を超えているときは，実行中のタスクは存在しません。そのため，警告メッセージ（KDEC84208-W，KDEC84209-W）は出力されません。

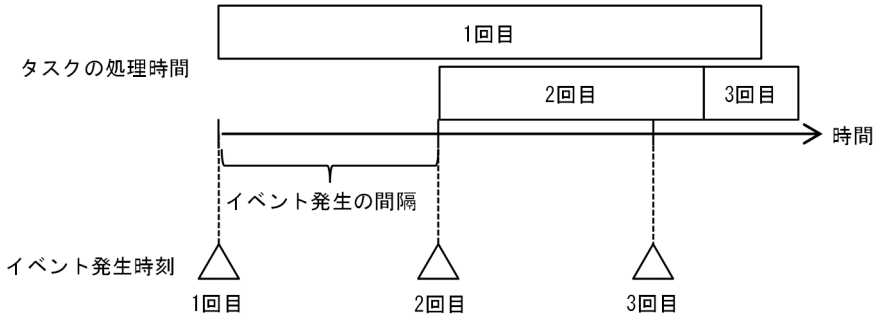
(3)　タスクの並列処理

並列処理では，イベントの発生時にすでに実行中のタスクがある場合，実行中のタスクの完了を待たないで，次のイベントを発生させます。実行中のタスクがある場合でも，並列でタスクを実行できます。

並列して処理できるタスク数は，スケジュール駆動受付定義ファイルの「urecp-scd.task.processing-max.parallel.count」で設定します。なお，並列して処理できるタスク数を超えた場合，新たなタスクは実行中のタスク処理の完了を待ってから実行されます。

タスクの実行の待機時間は，無制限となります。

図 3-62　タスクの並列処理



注　この例では，並列して処理できるタスク数を2とします。

この例では，1 回目のタスクの処理時間が，2 回目と 3 回目のイベント発生時刻に重なっています。この場合，1 回目のタスクの処理が終了するのを待たないで，2 回目のイベントでタスクが生成され，実行されます。また，3 回目のイベント発生時刻にすでにタスクが 2 つ実行されているため，3 回目のイベントではタスクの実行が待機され，2 回目のイベントのタスク完了後に，3 回目のイベントのタスクが実行されます。

(4)　エラー発生時のタスクの実行の抑止

タスクの実行時にエラーが発生した場合，エラー発生後のイベント発生時にタスクの実行を抑止することができます。この場合は，待機しているタスクの処理も実行しません。

通常は，スケジュール駆動受付内でエラーが発生した場合（スケジュール機能が異常停止した場合を除く），またはスケジュール駆動受付に例外が返却された場合でも，イベント発生時にはタスクを実行します（タスクの実行は抑止されません）。

エラー発生時にタスクの実行を抑止したい場合は，スケジュール駆動受付定義ファイルの scdrecp.suppress-execution-task.error-occures-threshold プロパティで設定できます。scdrecp.suppress-execution-task.error-occures-threshold プロパティの詳細については，マニュアル「サービスプラットフォーム リファレンス」の「3.15.1　スケジュール駆動受付定義ファイル」を参照してください。

また，待機しているタスクがある状態で，タスクの実行が抑止された場合，待機しているタスクに関するイベント履歴ログは出力されません。

(a)　タスクの実行が抑止される条件

次に示す条件をすべて満たす場合に，タスクの実行が抑止されます。

1. タスクの処理順序が「逐次」であること
2. スケジュール駆動受付定義ファイルの scdrecp.suppress-execution-task.error-occures-threshold プロパティで，エラー発生時にタスクの実行を抑止する設定を行っていること
3. タスクの実行時にエラーが発生すること

4. scdrecp.suppress-execution-task.error-occures-threshold プロパティで設定したエラーが連続で発生した回数のしきい値を超えること

   なお，エラーが連続で発生した回数には，ビジネスプロセスの呼び出しでタイムアウトが発生したケースも含みます。

(b) エラーが連続で発生した回数のしきい値の決め方

スケジュール駆動受付定義ファイルの scdrecp.suppress-execution-task.error-occures-threshold プロパティを設定する場合，一時的な障害が発生した場合を考慮した上で，タスクの実行時にエラーが連続して発生した回数をしきい値として決めてください。それにより，エラー発生時のタスクの実行を抑止することを適切に設定できます。

なお，タスク実行時に複数のエラーが発生した場合であっても，対象のタスク実行でエラーが発生した回数は 1 回としてカウントされます。例えば，タスクの実行でタイムアウトが発生して，かつビジネスプロセスがエラーとなった場合は，対象のタスク実行でエラーが発生した回数は 1 回とカウントされます。

しきい値に達するまでに，タスクの実行が正常終了した場合は，タスクの実行時にエラーが連続して発生した回数は 0 回にリセットされます。また，スケジュール駆動受付や HCSC サーバを再起動した場合も，タスクの実行時にエラーが連続して発生した回数はリセットされます。

(c) 抑止されたタスクの実行を再開する方法

抑止されたタスクの実行を再開する場合は，スケジュール駆動受付を再起動する必要があります。スケジュール駆動受付は，抑止されたタスクの実行を，自動で再開しません。

3.6.4 スケジュール駆動受付の作業フォルダ

スケジュール駆動受付では，タスク単位で作業フォルダを作成できます。作成した作業フォルダは，スケジュール駆動受付から呼び出すビジネスプロセスで使用できます。

作業フォルダの作成は，スケジュール駆動受付定義ファイルの「scdrecp.create.work-dir」で設定します。「scdrecp.create.work-dir」に true を設定した場合，タスクの実行時にリクエスト ID を生成して，作業フォルダが作成されます。生成したリクエスト ID は，要求電文に設定します。

また，ビジネスプロセスの処理が正常に終了した場合，スケジュール駆動受付で作成した作業フォルダの自動削除を設定できます。次のすべての条件に当てはまる作業フォルダが，自動的に削除されます。

- スケジュール駆動受付定義ファイルの「scdrecp.work-dir.auto-delete」に true を設定した。
- タスク処理が正常に完了した。

3.6.5　スケジュール駆動受付のタイマ監視

スケジュール駆動受付コンフィグファイルの「urecp-scd.request-timeout」の設定に従って，ビジネスプロセスに要求電文を渡してから，ビジネスプロセスの応答が返ってくるまでの処理時間を監視します。デフォルトでは，タイマ監視を行いません。

監視時間が経過しても応答が返ってこない場合は，メッセージを出力します。メッセージ出力後は，ビジネスプロセスの処理が完了してから，ビジネスプロセスの応答をスケジュール駆動受付に返却します。

タイマ監視で使われるスレッドのうち，スケジュール駆動受付の起動時に生成して常駐させるスレッドの数は，スケジュール駆動受付コンフィグファイルの「urecp-scd.resident-thread.count」で設定できます。

タイマ監視に使われるスレッドの最大数は，スケジュール駆動受付定義ファイルの「urecp-scd.task.processing-max.parallel.count」で設定します。

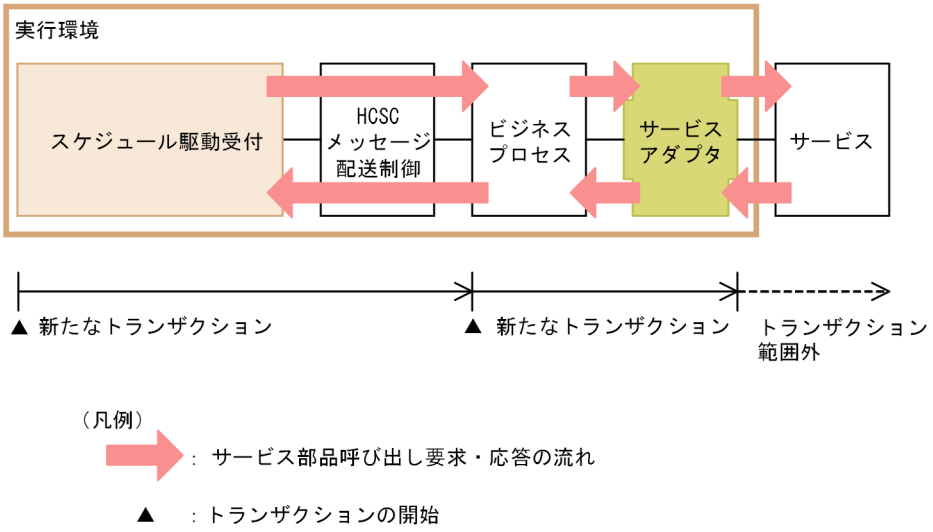
3.6.6　スケジュール駆動受付のトランザクション

ビジネスプロセスを呼び出すときのスケジュール駆動受付のトランザクションは，タイマ監視を行うかどうかで異なります。

(1)　タイマ監視を行わない場合

タイマ監視を行わない場合のトランザクションを次に示します。

図 3-63　タイマ監視を行わない場合のトランザクション

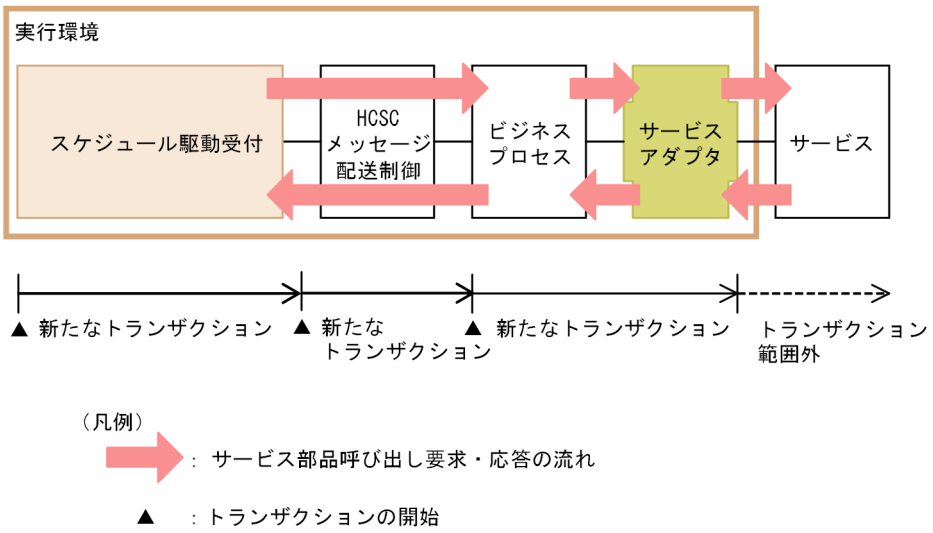


(2)　タイマ監視を行う場合

タイマ監視を行う場合は，別スレッドでビジネスプロセスの呼び出しが行われるため，新たなトランザクションを開始します。

タイマ監視に設定するタイムアウト値は，J2EE サーバのトランザクションタイムアウトの値よりも大きな値を設定することを推奨します。

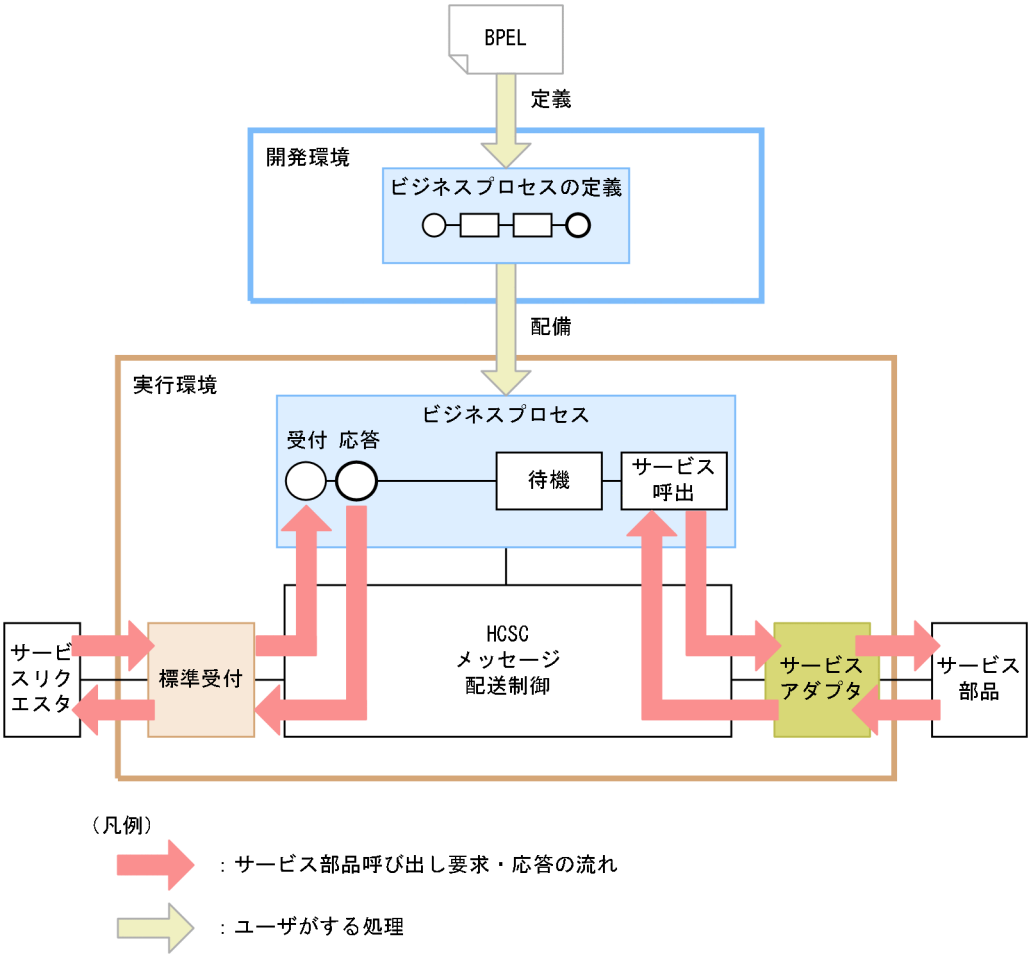
図 3-64　タイマ監視を行う場合のトランザクション



3.7　ビジネスプロセスを使用した待機処理

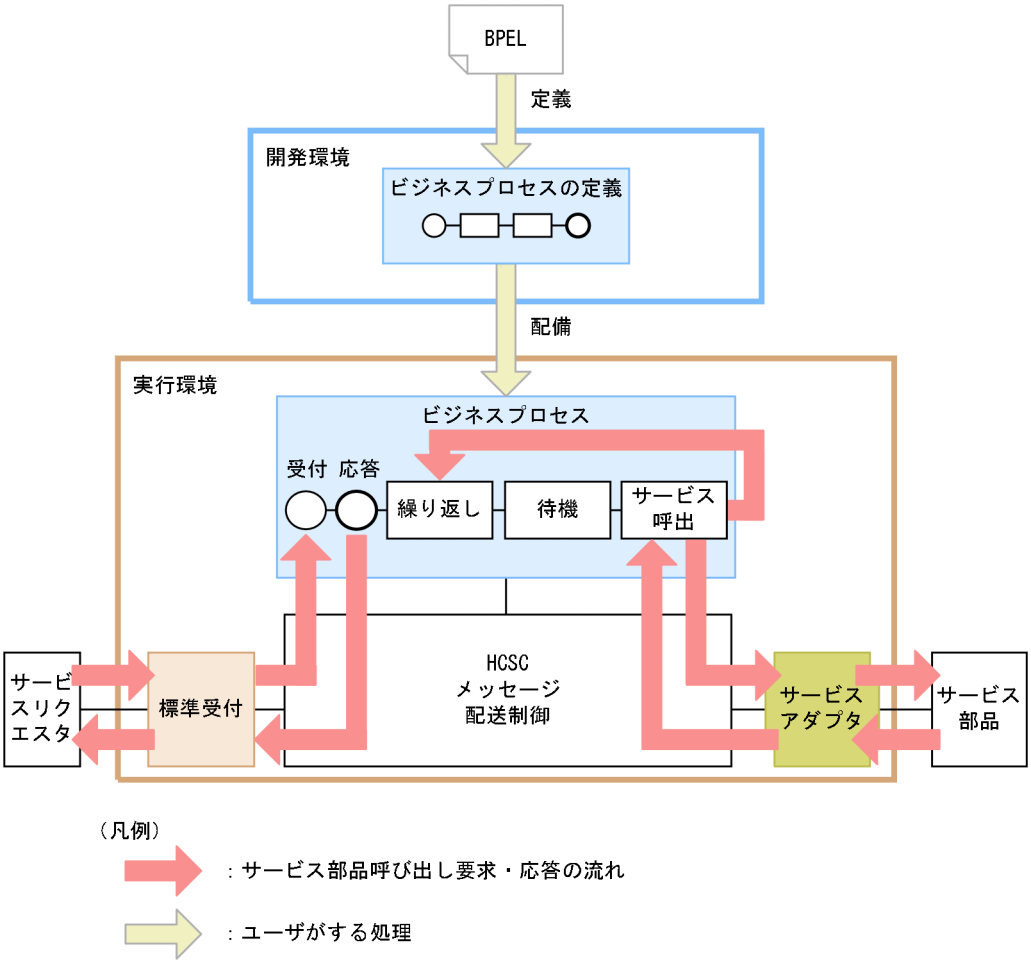
待機アクティビティを定義してビジネスプロセスを実行すると，特定の期限に達するまでビジネスプロセスの処理を遅延させることができます。待機アクティビティを定義したビジネスプロセスの待機処理を次の図に示します。

図 3-65　待機アクティビティを定義したビジネスプロセスの待機処理



繰り返しアクティビティの中に待機アクティビティを定義することで，日次処理などの一定間隔での処理を実現することもできます。繰り返しアクティビティの中に待機アクティビティを定義した場合の，ビジネスプロセスの待機処理を次の図に示します。

図 3-66　繰り返しアクティビティの中に待機アクティビティを定義したビジネスプロセスの待機処理



待機アクティビティの定義について次に説明します。

3.7.1　待機アクティビティでの待機時間の指定

待機アクティビティでは，待機時間として次のどちらかを指定します。

- 待機する間隔
- 待機を解除する期限

どちらの場合も XPath 式を使用して待機時間を指定します。XPath 式を使用すると，ビジネスプロセス内を流れる変数内の時刻に従って動作するため，待機時間を動的に指定できます。また，XPath 式の代わりに，待機する間隔または待機を解除する期限を直接指定することもできます。待機する間隔は XML Schema 型 duration，待機を解除する期限は XML Schema 型 dateTime で指定します。詳細については，マニュアル「サービスプラットフォーム 開発ガイド 基本開発編」の「5.6.11　待機アクティビティ」を参照してください。

3.7.2　待機アクティビティを実行するための条件

待機アクティビティは，永続化しないビジネスプロセスでは使用できない，応答アクティビティのあとに設定しなければならないなど，幾つかの条件があります。待機アクティビティを実行するための条件については，マニュアル「サービスプラットフォーム 開発ガイド 基本開発編」の「5.6.11　待機アクティビティ」を参照してください。

3.7.3　待機アクティビティに関する多重度

待機アクティビティを使用する上で性能を考えた場合，待機アクティビティに関する多重度を見直す必要があります。

待機アクティビティで指定した時間に待機処理を解除するプロセスインスタンスが複数存在している場合で，そのプロセスインスタンスの数より J2EE サーバの TimerService でタイムアウトメソッドをコールバックする最大スレッド数が小さいときは，性能上のボトルネックになることがあります。このような場合，J2EE サーバの環境設定で，タイムアウトメソッドのコールバックの最大スレッド数を設定します。

TimerService でタイムアウトメソッドをコールバックする最大スレッド数は，運用管理ポータルの［論理サーバの環境設定］で設定します。EJB コンテナの設定画面を開き，「タイムアウトメソッドのコールバックの最大スレッド数」※の値を，ビジネスプロセスのインスタンス数に合わせて変更してください。設定方法の詳細については，マニュアル「アプリケーションサーバ 運用管理ポータル操作ガイド」の「10.8.3 EJB コンテナの設定」を参照してください。

注※

J2EE サーバ用ユーザプロパティファイル（usrconf.properties）の ejbserver.ejb.timerservice.maxCallbackThreads キーに対応します。

3.8　応答アクティビティのあとに定義されたアクティビティの非同期実行

応答アクティビティのあとに任意のアクティビティを定義すると，そのアクティビティは Reliable Messaging または非同期 EJB を使用して非同期に実行できます。

応答後のアクティビティ実行で使用する機能は，データベースや Reliable Messaging の有無によって次のように異なります。

表 3-15　応答後のアクティビティ実行で使用する機能

Reliable Messaging	データベース	
	あり	なし
あり	• 永続化するビジネスプロセスの場合 Reliable Messaging • 永続化しないビジネスプロセスの場合 非同期 EJB※	－
なし	非同期 EJB※	非同期 EJB※

（凡例）

－：該当なし。

注※

応答後のアクティビティ実行に Reliable Messaging でなく非同期 EJB を使用する場合，次に示す機能は使用できません。

- 障害発生時に，Reliable Messaging のリトライ機能を利用したリカバリ処理が使用できません。リカバリ処理の詳細については，「3.4.7　応答アクティビティのトランザクション」を参照してください。
- Reliable Messaging を前提とする次の受付およびサービスアダプタは使用できません。
  標準受付（MDB（WS-R））
  標準受付（MDB（DB キュー））
  標準アダプタ（MDB（WS-R））
  標準アダプタ（MDB（DB キュー））

3.8.1　非同期 EJB を使用して非同期実行する場合の設定項目

非同期 EJB を使用する場合，HCSC サーバランタイム定義ファイルに次のプロパティで設定します。

- bp-reply-after-max-thread-pool-size
  プールに生成できるスレッドの最大数を設定します。デフォルト値は 32 です。
- bp-reply-after-min-thread-pool-size

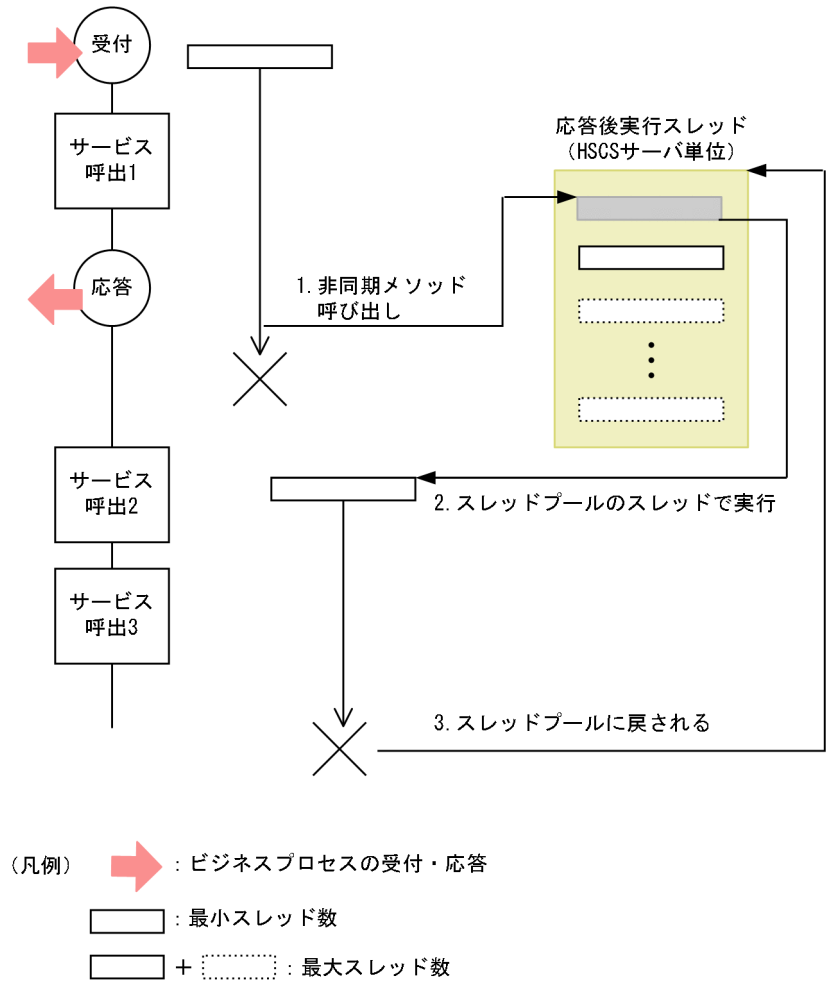
プール内で使用していないスレッドを幾つまで保存できるかを設定します。デフォルト値は 16 です。

- bp-reply-after-thread-pool-keep-alive
  使用していないスレッド数が min-thread-pool-size で設定した数を超えている場合に，スレッドを終了するまでに保存しておく時間（単位：秒）を設定します。デフォルト値は 60 です。

HCSC サーバランタイム定義ファイルの設定内容については，マニュアル「サービスプラットフォーム リファレンス」の「6.5.6　HCSC サーバランタイム定義ファイル」を参照してください。

応答後のアクティビティの実行の概要を次に示します。

図 3-67　応答後のアクティビティ実行の概要



3.8.2　非同期 EJB の実行で例外が発生した場合の再実行方法

応答アクティビティのあとに定義されたアクティビティで例外が発生した場合や，非同期 EJB の実行時に例外が発生した場合，プロセスは次の状態となり，一括再実行の対象になります。

- プロセスインスタンス：実行中状態
- 応答アクティビティ：完了状態（待ち受け）

- 応答アクティビティ以降に定義されたアクティビティ：未実行状態

プロセスインスタンスの再実行によって，応答アクティビティ以降に定義されたアクティビティを実行できます。

3.9　ログ出力アクティビティでのログ出力

ログ出力アクティビティを使用して，任意のログファイルを出力できます。

次のメッセージをログ出力アクティビティのログファイルとして出力できます。

- メッセージ定義ファイルに定義したメッセージ
- ［ログ出力アクティビティ］ダイアログで設定した任意の文字列

ログ出力アクティビティは，［ログ出力アクティビティ］ダイアログで設定します。ログ出力アクティビティの設定方法については，マニュアル「サービスプラットフォーム 開発ガイド 基本開発編」の「5.6.13 ログ出力アクティビティ」を参照してください。

ここでは，ログ出力アクティビティのログファイルに出力できる内容，取得方法，および出力方法について説明します。

3.9.1　設定に応じたログファイルの出力内容

［ログ出力アクティビティ］ダイアログでの設定によって，ログ出力アクティビティのログファイルに出力できる内容が異なります。

それぞれに必要な設定と出力されるログ出力アクティビティのログファイルの内容について説明します。

出力内容の詳細については，「3.9.3(3)　ログ出力アクティビティのログファイルの出力内容」を参照してください。

［ログ出力アクティビティ］ダイアログで設定した任意の文字列を出力する場合

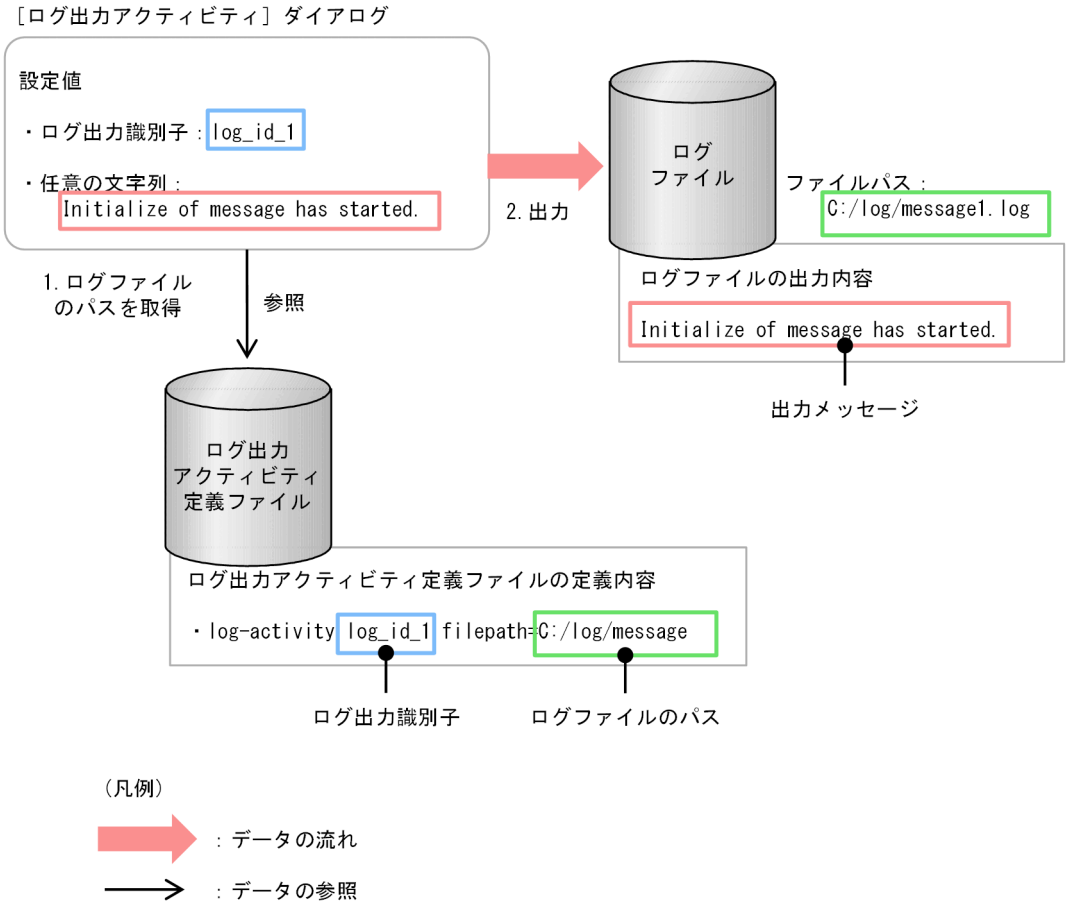
［ログ出力アクティビティ］ダイアログで設定した任意の文字列を出力する場合，ログ出力アクティビティ定義ファイルの設定が必要です。

ログ出力アクティビティ定義ファイルの詳細については，マニュアル「サービスプラットフォーム リファレンス」の「3.20.1　ログ出力アクティビティ定義ファイル」を参照してください。

なお，任意の文字列を出力する場合，ログレベルによるログ出力の制御はできません。自動的にログレベル「ERROR」が設定されます。

ログファイルが出力されるまでのログ出力アクティビティの動作について，次の図に示します。

図 3-68　ログ出力アクティビティの動作（[ログ出力アクティビティ] ダイアログで設定した任意の文字列を出力する場合）



1. メッセージとして［ログ出力アクティビティ］ダイアログで設定した任意の文字列を取得します。また，ログ出力識別子に該当するログファイルのファイルパスをログ出力アクティビティ定義ファイルから取得します。
2. 1.で作成したメッセージを，ログ出力アクティビティ定義ファイルに定義されているログファイルに出力します。

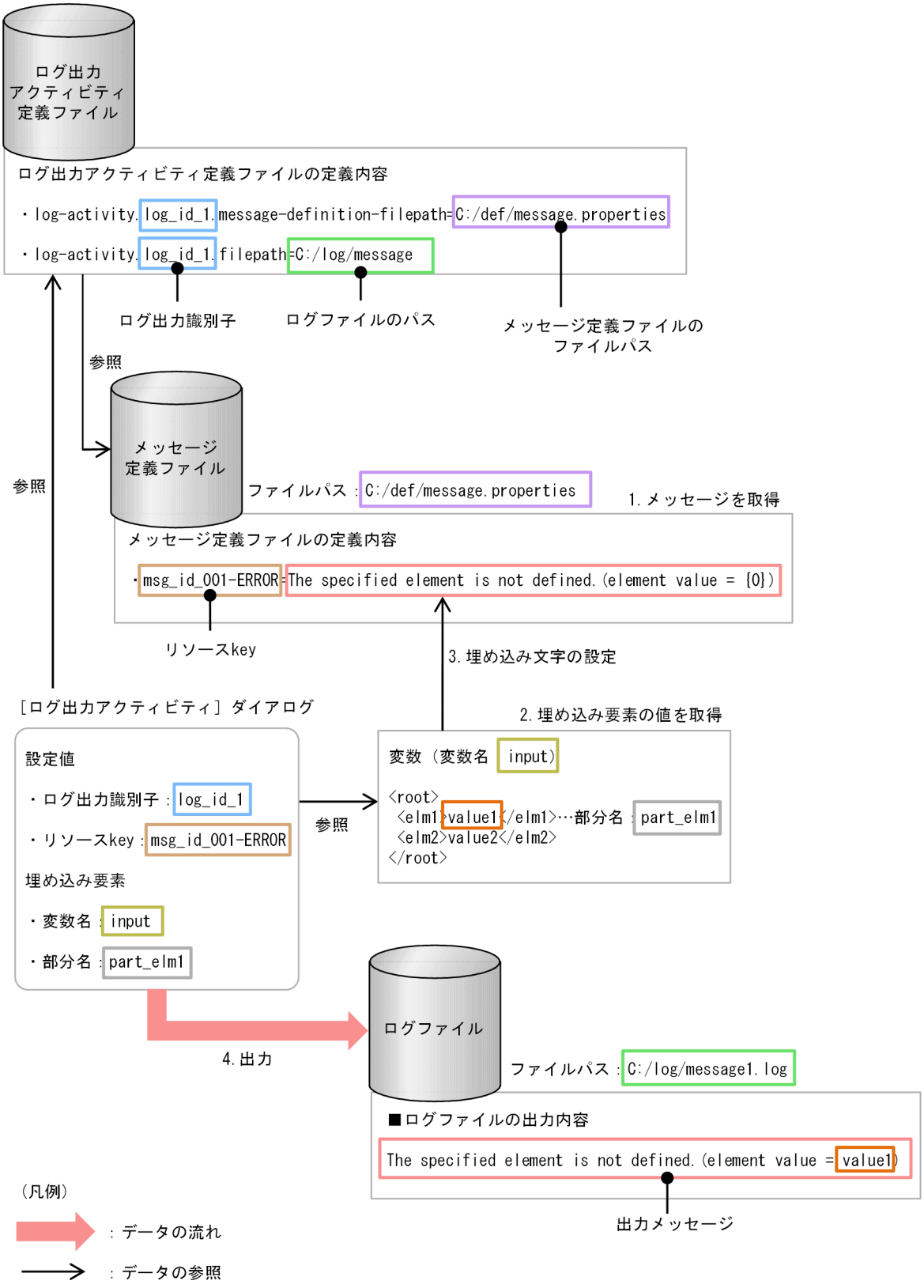
メッセージ定義ファイルに定義したメッセージを出力する場合

メッセージ定義ファイルに定義したメッセージを出力する場合，ログ出力アクティビティ定義ファイル，およびメッセージ定義ファイルの設定が必要です。

ログ出力アクティビティ定義ファイル，およびメッセージ定義ファイルの詳細については，マニュアル「サービスプラットフォーム リファレンス」の「3.20.1　ログ出力アクティビティ定義ファイル」および「3.20.2　メッセージ定義ファイル」を参照してください。

ログファイルが出力されるまでのログ出力アクティビティの動作について，次の図に示します。なお，この例ではメッセージに埋め込み要素を使用しています。

図 3-69　ログ出力アクティビティの動作（メッセージ定義ファイルに定義したメッセージを出力する場合）



1. ログ出力アクティビティ定義ファイルに設定されているメッセージ定義ファイルの場所を探し，ログ出力アクティビティで設定したリソース Key（msg_id_001-ERROR）に該当するメッセージを取得します。
2. [ログ出力アクティビティ] ダイアログで設定した埋め込み要素（変数名：input，部分名：part_elm1）の値（value1）を取得します。
3. 1.で取得したメッセージに 2.で取得した埋め込み文字（値：value1）を設定します。
4. 3.で作成したメッセージを，ログ出力アクティビティ定義ファイルに定義されているログファイルに出力します。

3.9.2　ログ出力アクティビティのログファイルの取得方法

ログ出力アクティビティを設定すると，ユーザが指定した任意のメッセージをログ出力アクティビティのログファイルに出力できます。

ログ出力アクティビティのログファイルを出力するには，ログ出力アクティビティ定義ファイルなどの定義ファイルを準備します。その後，ビジネスプロセス定義画面でログ出力アクティビティを配置し，[ログ出力アクティビティ] ダイアログで必要な情報を指定します。

[ログ出力アクティビティ] ダイアログの指定方法については，マニュアル「サービスプラットフォーム リファレンス」の「1.4.21　ログ出力アクティビティダイアログ」を参照してください。

ログ出力アクティビティの定義手順については，マニュアル「サービスプラットフォーム 開発ガイド 基本開発編」の「5.6.13　ログ出力アクティビティ」を参照してください。

出力されたログ出力アクティビティのログファイルの確認方法については，「3.9.3　ログ出力アクティビティのログファイルの出力」を参照してください。

ログ出力アクティビティで使用する定義ファイルを次に示します。

- ログ出力アクティビティ定義ファイル
- メッセージ定義ファイル
- HCSC サーバランタイム定義ファイル

これらの定義ファイルの定義内容を次に示します。

(1)　ログ出力アクティビティ定義ファイルの定義内容

ログ出力アクティビティ定義ファイルでは，ログ出力アクティビティのログファイルに関する情報を次に示すプロパティで指定します。ログ出力識別子は，ログ出力アクティビティのログファイルの出力先を一意に識別するための文字列です。

- log-activity.<ログ出力識別子>.filepath=出力先パス

- log-activity.<ログ出力識別子>.filesize=1 面当たりのファイルサイズ
- log-activity.<ログ出力識別子>.filenum=ファイルの面数
- log-activity.<ログ出力識別子>.level=ファイルの出力レベル

また，出力するメッセージの内容をメッセージ定義ファイルに定義する場合は，メッセージ定義ファイルのパスを次に示すプロパティで指定します。

- log-activity.<ログ出力識別子>.message-definition-filepath=メッセージ定義ファイルのファイルパス

ここで定義したログ出力識別子は，ログ出力アクティビティの設定時に［ログ出力アクティビティ］ダイアログで指定します。

ログ出力アクティビティ定義ファイルの定義方法の詳細については，マニュアル「サービスプラットフォーム リファレンス」の「3.20.1　ログ出力アクティビティ定義ファイル」を参照してください。

ログ出力アクティビティのログファイルは，log-activity.<ログ出力識別子>.filesize で指定したファイルサイズに達したときローテーションが実行されます。時刻によるログの出力先ファイルの切り替え，および出力先ファイル切り替え時のファイル名付与規則については，Application Server の設定に従います。

詳細については，マニュアル「アプリケーションサーバ リファレンス 定義編(サーバ定義)」の「2.2.3 usrconf.properties（J2EE サーバ用ユーザプロパティファイル)」を参照してください。

(2)　メッセージ定義ファイルの定義内容

ログ出力アクティビティのログファイルに出力する任意のメッセージは，［ログ出力アクティビティ］ダイアログで直接設定するほかに，メッセージ定義ファイルで定義することもできます。

メッセージ定義ファイルには，リソース Key とメッセージを定義します。ここで定義したリソース Key は，ログ出力アクティビティの設定時に［ログ出力アクティビティ］ダイアログで指定します。

メッセージ定義ファイルのファイルは，ログ出力識別子ごと（ログの出力先ごと）に作成します。作成したメッセージ定義ファイルのパスは，ログ出力アクティビティ定義ファイルに指定します。

メッセージ定義ファイルの定義方法の詳細については，マニュアル「サービスプラットフォーム リファレンス」の「3.20.2　メッセージ定義ファイル」を参照してください。

(3)　HCSC サーバランタイム定義ファイルの定義内容

HCSC サーバランタイム定義ファイルの次に示すプロパティで，ログ出力アクティビティ定義ファイルのパスを指定します。

- bp-log-activity-definition-filepath=ログ出力アクティビティ定義ファイルのパス

HCSC サーバランタイム定義ファイルの詳細については，マニュアル「サービスプラットフォーム リファレンス」の「6.5.6　HCSC サーバランタイム定義ファイル」を参照してください。

3.9.3　ログ出力アクティビティのログファイルの出力

ログ出力アクティビティを定義した場合，任意のメッセージをログファイルとして出力できます。

ログ出力アクティビティの定義手順については，マニュアル「サービスプラットフォーム 開発ガイド 基本開発編」の「5.6.13　ログ出力アクティビティ」を参照してください。

ログ出力アクティビティのログファイルの出力開始・停止のタイミングを次に示します。

- HCSC サーバが起動すると，ログファイルの出力が開始されます。
  出力開始に失敗した場合，エラーメッセージを出力して開始処理を終了します。障害対策後に HCSC サーバの起動を実行することで，ログファイルの出力が開始されます。
- HCSC サーバが停止すると，ログファイルの出力が停止されます。
  出力停止に失敗した場合，警告メッセージを出力して停止処理を続行します。
- ログファイルの出力開始後にメッセージの出力に失敗すると，警告メッセージを出力してログ出力アクティビティの処理を続行します。同じログ出力識別子を設定したログ出力アクティビティでは，メッセージは出力されなくなります。
  障害対策後に HCSC サーバの起動を実行することで，ログファイルの出力が開始されます。

この項では，ログ出力アクティビティのログファイルの出力先や出力される内容，および利用方法の例について説明します。ログ出力アクティビティのログファイルの取得方法については，「3.9.2　ログ出力アクティビティのログファイルの取得方法」を参照してください。

(1)　ログ出力アクティビティのログファイルの出力先

ログ出力アクティビティのログファイルの出力先は，ログ出力アクティビティ定義ファイルの log-activity.<ログ出力識別子>.filepath プロパティで設定した個所です。出力先はログ出力識別子によって異なります。

ログ出力アクティビティ定義ファイルのプロパティについては，マニュアル「サービスプラットフォーム リファレンス」の「3.20.1　ログ出力アクティビティ定義ファイル」を参照してください。

(2)　ログ出力アクティビティのログファイルの出力形式

ログ出力アクティビティのログファイルの出力形式を次に示します。

```
<番号> <日付> <時刻> <製品ID> <pid> <tid> <ID> <ログ出力文字列>
```

(3)　ログ出力アクティビティのログファイルの出力内容

ログ出力アクティビティのログファイルに出力される内容を次の表に示します。

表 3-16　ログ出力アクティビティのログファイルに出力される内容

項目	内容
番号	ログ出力アクティビティのログファイルの出力通番が表示されます。
日付	ログ出力アクティビティのログファイルの取得日付が yyyy/mm/dd の形式で出力されます。 • yyyy：西暦年 • mm：月 • dd：日
時刻	ログ出力アクティビティのログファイルの取得時刻が hh:mm:ss.SSS の形式で出力されます。 • hh：時 • mm：分 • ss：秒 • SSS：ミリ秒 ローカル時刻でミリ秒単位の時刻を示します。
製品 ID	製品を特定するための識別子として，「CSCBP」が表示されます。
pid	プロセスを識別するための ID が出力されます。
tid	スレッドを識別するための ID が出力されます。
ID	空白
ログ出力文字列	ログ出力アクティビティで定義した任意のメッセージが出力されます。

4 システム動作を円滑にするための機能

この章では，システム動作を円滑にするための機能について説明します。

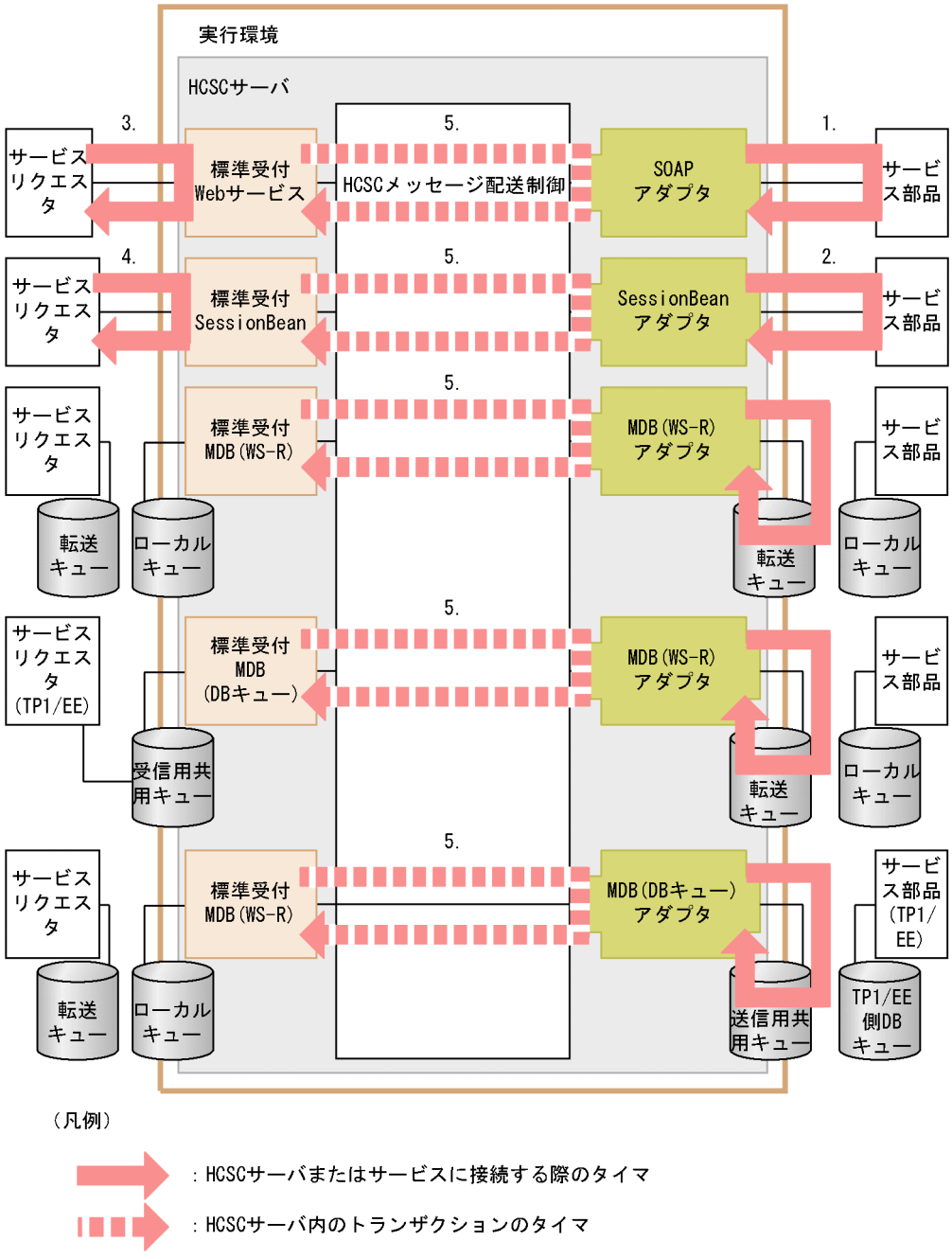
4.1　HCSC サーバに関連するタイマの設定

同期のサービス部品呼び出しの場合，応答が返って来ない場合のタイムアウト値を設定することで，無応答でハングアップすることを回避できます。また，サービス部品での実際の処理に掛かる最大時間を考慮して，タイマを設定する必要があります。

4.1.1　HCSC サーバで設定できるタイマ

HCSC サーバで設定できるタイマについて次の図に示します。

図 4-1　設定できるタイマ



「図 4-1　設定できるタイマ」の各項番は次のタイマを示しています。

1. サービス部品に接続するときのタイマ（Web サービス）
2. サービス部品に接続するときのタイマ（SessionBean）
3. HCSC サーバに接続するときのタイマ（Web サービス）
4. HCSC サーバに接続するときのタイマ（SessionBean）
5. HCSC サーバ内のトランザクションのタイマ

タイマは接続先やプロトコルによって設定方法が異なります。

(1)　サービス部品に接続するときのタイマ（Web サービス）

サービス部品（Web サービス）に接続する場合，書き込み，読み込み，および接続時のタイマを設定できます。

- **書き込み時のタイマ**
  HTTP および HTTPS（TCP/IP 通信）で使用するソケットに対する write（データ送信）処理でのタイムアウト値を設定します。接続確立後のデータ送信が完了しない場合などで有効となります。
- **読み込み時のタイマ**
  HTTP および HTTPS（TCP/IP 通信）で使用するソケットに対する read（データ受信）処理でのタイムアウト値を設定します。接続確立後のデータ受信が完了しない場合などで有効となります。
- **接続時のタイマ**
  HTTP および HTTPS（TCP/IP 通信）で使用するソケットに対する connect（ソケット接続）処理でのタイムアウト値を設定します。接続確立時の要求に対する応答がない場合などで有効となります。

タイムアウト値の設定は，SOAP 通信基盤を利用する場合と，JAX-WS エンジンを利用する場合で異なります。それぞれの場合の設定方法を説明します。

(a)　タイムアウト値の設定（SOAP 通信基盤）

SOAP 通信基盤（SOAP1.1 モード）を利用する場合のタイムアウト値の設定方法について説明します。タイムアウト値は，HCSC サーバ全体またはサービスアダプタ個別に設定できます。

- **HCSC サーバ全体の設定**
  タイムアウト値は，HCSC サーバ稼働マシンの SOAP 通信基盤のサーバ定義ファイルで定義します。サーバ定義ファイルの設定については，マニュアル「アプリケーションサーバ SOAP アプリケーション開発の手引」の「10.2　サーバ定義ファイルの設定」を参照してください。

表 4-1　HCSC サーバ全体のタイマ（Web サービス）の設定（SOAP 通信基盤）

値	キー名称	デフォルト値（秒）
サーバ兼クライアントのソケットの書き込みタイムアウト値	c4web.application.<識別子>.socket_write_timeout	60

値	キー名称	デフォルト値（秒）
サーバ兼クライアントのソケットの読み込みタイムアウト値	c4web.application.<識別子>.socket_read_timeout	300
サーバ兼クライアントのソケットの接続タイムアウト値	c4web.application.<識別子>.socket_connect_timeout	60

- サービスアダプタ個別の設定

  開発環境で設定する場合，サービスアダプタ定義画面のクライアント定義ファイルに指定します。クライアント定義ファイルの設定については，マニュアル「アプリケーションサーバ SOAP アプリケーション開発の手引」の「10.3　クライアント定義ファイルの設定」を参照してください。

表 4-2　サービスアダプタ個別のタイマ（Web サービス）の設定（SOAP 通信基盤）

値	キー名称	デフォルト値（秒）
クライアントのソケットの書き込みタイムアウト値	c4web.application.socket_write_timeout	60
クライアントのソケットの読み込みタイムアウト値	c4web.application.socket_read_timeout	300
クライアントのソケットの接続タイムアウト値	c4web.application.socket_connect_timeout	60

(b)　タイムアウト値の変更（SOAP 通信基盤）

すでに HCSC サーバに配備しているサービスアダプタに対して運用環境で定義する場合は，cscsvcctl コマンドを使用して，サービス部品呼び出しの通信タイムアウト値を変更します。変更方法の詳細については，マニュアル「サービスプラットフォーム システム構築・運用ガイド」の「5.3.28　サービス部品呼び出しの通信タイムアウト値を変更する」を参照してください。cscsvcctl コマンドの使い方については，マニュアル「サービスプラットフォーム リファレンス」の「cscsvcctl（サービス情報の管理）」を参照してください。

表 4-3　タイマ（Web サービス）の設定値の変更（SOAP 通信基盤）

値	キー名称
書き込みタイムアウト値	<クラスタ名>.<サービス ID>.WebService.c4web.application.socket_write_timeout
読み込みタイムアウト値	<クラスタ名>.<サービス ID>.WebService.c4web.application.socket_read_timeout
接続タイムアウト値	<クラスタ名>.<サービス ID>.WebService.c4web.application.socket_connect_timeout

(c)　タイムアウト値の設定（JAX-WS エンジン）

JAX-WS エンジン（SOAP1.1/1.2 併用モード）を利用する場合のタイムアウト値の設定方法について説明します。タイムアウト値は，HCSC サーバ全体またはサービスアダプタ個別に設定できます。

- HCSC サーバ全体の設定

  HCSC サーバ稼働マシンの JAX-WS エンジンの共通定義ファイルで定義します。共通定義ファイルの設定については，マニュアル「アプリケーションサーバ Web サービス開発ガイド」の「10.1.2　共通定義ファイルの設定項目」を参照してください。

表 4–4　HCSC サーバ全体のタイマ（Web サービス）の設定（JAX-WS エンジン）

値	キー名称	デフォルト値（ミリ秒）
サーバ兼クライアントのソケットの読み込みタイムアウト値	com.cosminexus.jaxws.request.timeout	300000
サーバ兼クライアントのソケットの接続タイムアウト値	com.cosminexus.jaxws.connect.timeout	60000

- サービスアダプタ個別の設定

  開発環境で設定する場合，サービスアダプタ定義画面のクライアント定義ファイルに指定します。クライアント定義ファイルの設定については，マニュアル「サービスプラットフォーム リファレンス」の「3.6.2　クライアント定義ファイル」を参照してください。

表 4–5　サービスアダプタ個別のタイマ（Web サービス）の設定（JAX-WS エンジン）

値	キー名称	デフォルト値（秒）
クライアントのソケットの読み込みタイムアウト値	com.cosminexus.csc.request.timeout	300
クライアントのソケットの接続タイムアウト値	com.cosminexus.csc.connect.timeout	60

(d)　タイムアウト値の変更（JAX-WS エンジン）

すでに HCSC サーバに配備しているサービスアダプタに対して運用環境で定義する場合は，cscsvcctl コマンドを使用して，サービス部品呼び出しの通信タイムアウト値を変更します。変更方法の詳細については，マニュアル「サービスプラットフォーム システム構築・運用ガイド」の「5.3.28　サービス部品呼び出しの通信タイムアウト値を変更する」を参照してください。cscsvcctl コマンドの使い方については，マニュアル「サービスプラットフォーム リファレンス」の「cscsvcctl（サービス情報の管理）」を参照してください。

表 4–6　タイマ（Web サービス）の設定値の変更（JAX-WS エンジン）

値	キー名称
読み込みタイムアウト値	<クラスタ名>.<サービス ID>.WebService.com.cosminexus.csc.request.timeout
接続タイムアウト値	<クラスタ名>.<サービス ID>.WebService.com.cosminexus.csc.connect.timeout

(2) サービス部品に接続するときのタイマ（SessionBean）

(a) タイムアウト値の設定

サービス部品に接続するときのタイマ（SessionBean）は，HCSC サーバ全体に設定したり，サービスアダプタ個別に設定したりできます。設定方法を次に示します。

- HCSC サーバ全体の設定

  HCSC サーバ稼働マシンのアプリケーションサーバの EJB コンテナでタイムアウトを設定します。RMI-IIOP 通信のタイムアウトの詳細については，マニュアル「アプリケーションサーバ 機能解説 基本・開発編(EJB コンテナ)」の「2.11.5　RMI-IIOP 通信のタイムアウト」を参照してください。

  また，設定するパラメタの詳細については，次に示すマニュアルを参照してください。

  - J2EE サーバの互換モードが推奨モードの場合

    マニュアル「アプリケーションサーバ リファレンス 定義編(サーバ定義)」の「2.2.3 usrconf.properties（J2EE サーバ用ユーザプロパティファイル)」

  - J2EE サーバの互換モードが V9 互換モードの場合

    マニュアル「アプリケーションサーバ 機能解説 互換編」の「10.1.1　usrconf.properties（J2EE サーバ用ユーザプロパティファイル)」

表 4-7　HCSC サーバ全体のタイマ（SessionBean）の設定

値	設定するファイル	キー名称	デフォルト値(秒)
RMI-IIOP 通信すべてに有効となるタイムアウト値	J2EE サーバ用ユーザプロパティファイル（usrconf.properties）	ejbserver.rmi.request.timeout	0（タイムアウトしません）
CORBA ネーミングサービス部品との通信でのタイムアウト値		ejbserver.jndi.request.timeout	0（タイムアウトしません）

- サービスアダプタ個別の設定

  開発環境で定義する場合，サービスアダプタ定義画面のクライアント定義ファイルに指定するファイルで設定します。クライアント定義ファイルの設定については，マニュアル「アプリケーションサーバ SOAP アプリケーション開発の手引」の「10.3　クライアント定義ファイルの設定」を参照してください。

表 4-8　サービスアダプタ個別のタイマ（SessionBean）の設定

値	キー名称	デフォルト値（秒）
呼び出しタイムアウト値	c4web.application.ejb_timeout	0（タイムアウトしません）

(b)　タイムアウト値の変更

すでに HCSC サーバに配備しているサービスアダプタに対して運用環境で定義する場合は，cscsvcctl コマンドを使用して，サービス部品呼び出しの通信タイムアウト値を変更します。変更方法の詳細については，マニュアル「サービスプラットフォーム システム構築・運用ガイド」の「5.3.28　サービス部品呼び出しの通信タイムアウト値を変更する」を参照してください。cscsvcctl コマンドの詳細は，マニュアル「サービスプラットフォーム リファレンス」の「cscsvcctl（サービス情報の管理）」を参照してください。

表 4-9　タイマ（SessionBean）の設定値の変更

値	キー名称
呼び出しタイムアウト値	<クラスタ名>.<サービス ID>.SessionBean.c4web.application.ejb_timeout

(3)　HCSC サーバに接続するときのタイマ（Web サービス）

HCSC サーバに接続するときのタイマ（Web サービス）は，サービスリクエスタ稼働マシン側で設定します。サービスリクエスタの稼働しているマシン全体に設定したり，サービスリクエスタ個別に設定したりできます。

SOAP 通信基盤および JAX-WS エンジンを使用する場合に分けて，設定方法を示します。

(a)　タイムアウト値の設定（SOAP 通信基盤）

SOAP 通信基盤を利用する場合のタイムアウト値の設定方法について説明します。タイムアウト値は，サービスリクエスタ稼働マシン全体またはサービスリクエスタ個別に設定できます。

- **サービスリクエスタ稼働マシン全体の設定**

  サービスリクエスタ稼働マシンでの SOAP 通信基盤のサーバ定義ファイルで定義します。設定内容は「4.1.1(1)(a)　タイムアウト値の設定（SOAP 通信基盤）」の場合と同様です。サーバ定義ファイルの設定については，マニュアル「アプリケーションサーバ SOAP アプリケーション開発の手引」の「10.2　サーバ定義ファイルの設定」を参照してください。

- **サービスリクエスタ個別の設定**

  作成するサービスリクエスタごとに個別に設定する場合，サービスリクエスタを実行するときに指定するクライアント定義ファイルで設定します。設定内容は「4.1.1(1)(a)　タイムアウト値の設定（SOAP 通信基盤）」の場合と同様です。クライアント定義ファイルの設定については，マニュアル「アプリケーションサーバ SOAP アプリケーション開発の手引」の「10.3　クライアント定義ファイルの設定」を参照してください。

(b)　タイムアウト値の設定（JAX-WS エンジン）

JAX-WS エンジンを利用する場合のタイムアウト値の設定方法について説明します。タイムアウト値は，サービスリクエスタ稼働マシン全体またはサービスリクエスタ個別に設定できます。

- **サービスリクエスタ稼働マシン全体の設定**

サービスリクエスタ稼働マシンでの JAX-WS エンジンの共通定義ファイルで定義します。設定内容は「4.1.1(1)(c)　タイムアウト値の設定（JAX-WS エンジン）」の場合と同様です。共通定義ファイルの設定については，マニュアル「アプリケーションサーバ Web サービス開発ガイド」の「10.1.2　共通定義ファイルの設定項目」を参照してください。

- **サービスリクエスタ個別の設定**

  作成するサービスリクエスタごとに個別に設定する場合，プロセス別の定義ファイルで設定します。プロセス別の定義ファイルの設定については，マニュアル「アプリケーションサーバ Web サービス開発ガイド」の「10.1.3　プロセス別の定義ファイルの設定」を参照してください。

(4)　HCSC サーバに接続するときのタイマ（SessionBean）

HCSC サーバに接続するときのタイマ（SessionBean）は，サービスリクエスタの稼働しているマシン全体に設定したり，サービスリクエスタ個別に設定したりできます。設定方法を次に示します。

- **サービスリクエスタ稼働マシン全体の設定**

  サービスリクエスタ稼働マシンのアプリケーションサーバで，EJB コンテナでタイムアウトを設定します。設定内容は「(2)　サービス部品に接続するときのタイマ（SessionBean）」の場合と同様です。RMI-IIOP 通信のタイムアウトの詳細については，マニュアル「アプリケーションサーバ 機能解説 基本・開発編(EJB コンテナ)」の「2.11.5　RMI-IIOP 通信のタイムアウト」を参照してください。設定するパラメタの詳細については，マニュアル「アプリケーションサーバ リファレンス 定義編(サーバ定義)」の「12.2.2　usrconf.properties（Java アプリケーション用ユーザプロパティファイル）」を参照してください。

- **サービスリクエスタ個別の設定**

  作成するサービスリクエスタごとに個別に，create から remove までの API 通信のタイムアウト値を設定する場合，サービスリクエスタのアプリケーション開発時に API（RequestTimeoutConfigFactory クラスおよび RequestTimeoutConfig クラスのメソッド）で設定します。

(5)　HCSC サーバ内のトランザクションのタイマ

HCSC サーバ内のトランザクションのタイマは，HCSC サーバ全体で設定します。HCSC サーバ稼働マシンのアプリケーションサーバの EJB コンテナでタイムアウトを設定します。設定するパラメタの詳細については，次に示すマニュアルを参照してください。

- J2EE サーバの互換モードが推奨モードの場合

  マニュアル「アプリケーションサーバ リファレンス 定義編(サーバ定義)」の「2.2.3　usrconf.properties（J2EE サーバ用ユーザプロパティファイル）」

- J2EE サーバの互換モードが V9 互換モードの場合

  マニュアル「アプリケーションサーバ 機能解説 互換編」の「10.1.1　usrconf.properties（J2EE サーバ用ユーザプロパティファイル）」

表 4-10　HCSC サーバ内のトランザクションのタイマの設定

値	設定するファイル	キー名称	デフォルト値（秒）
J2EE サーバ上で開始されるトランザクションのトランザクションタイムアウト値	J2EE サーバ用ユーザプロパティファイル（usrconf.properties）	ejbserver.jta.TransactionManager.defaultTimeOut	180

4.1.2　サービス部品のタイムアウト値の変更

通信タイムアウト値は開発環境で定義しますが，SOAP アダプタ，および SessionBean アダプタについては，サービス部品に接続するときの通信タイムアウト値を運用環境で変更できます。運用環境でサービス部品のタイムアウト値を変更する場合は，cscsvcctl コマンドを使用します。ただし，変更対象のサービスアダプタが未配備状態である必要があります。

変更方法の詳細については，マニュアル「サービスプラットフォーム システム構築・運用ガイド」の「5.3.28　サービス部品呼び出しの通信タイムアウト値を変更する」を参照してください。cscsvcctl コマンドの使い方については，マニュアル「サービスプラットフォーム リファレンス」の「cscsvcctl（サービス情報の管理)」を参照してください。

4.1.3　タイマ設定時の注意事項

(1)　サービス部品に接続するときのタイマと HCSC サーバに接続するときのタイマの関係

HCSC サーバに接続するときのタイマは，サービス部品に接続するときのタイマよりも大きい値を指定してください。

サービス部品呼び出しで時間が掛かり正常応答した場合でも，HCSC サーバに接続するときのタイマが先にタイムアウトし，処理が正常に完了しません。

なお，ビジネスプロセスを呼び出して，サービス部品を呼び出す場合は，複数のサービス部品を呼び出すことを考慮して，HCSC サーバに接続するときのタイマを設定する必要があります。

(2)　サービス部品に接続するときのタイマとトランザクションのタイマの関係

トランザクションのタイマは，サービス部品に接続するときのタイマよりも大きい値を指定してください。

サービス部品呼び出しで時間が掛かり正常応答した場合でも，トランザクションがタイムアウトすると，トランザクションがロールバックします。

4.2　サービス部品の接続先変更

サービス部品の接続先は開発環境で定義しますが，次に示すサービスアダプタについては，サービス部品に接続するための接続先を運用環境で変更できます。

- SOAP アダプタ
- SessionBean アダプタ
- MDB（WS-R）アダプタ
- MDB（DB キュー）アダプタ

運用環境でサービス部品の接続先を変更する場合は，cscsvcctl コマンドを使用します。ただし，変更対象のサービスアダプタが未配備状態である必要があります。

変更方法の詳細については，マニュアル「サービスプラットフォーム システム構築・運用ガイド」の「5.3.26　サービス部品の接続先を変更する」を参照してください。cscsvcctl コマンドの使い方については，マニュアル「サービスプラットフォーム リファレンス」の「cscsvcctl（サービス情報の管理)」を参照してください。

4.3　SOAP アダプタを利用したフォルトのキャッチ

Web サービスで業務エラー（商品の在庫がないなどの障害）が発生したときにビジネスプロセスで何らかの処理を実行するには，SOAP アダプタからフォルト電文を応答する必要があります。ビジネスプロセスでフォルト処理を定義する方法の詳細についてはマニュアル「サービスプラットフォーム 開発ガイド 基本開発編」の「5.4.4　フォルト処理を定義する」を参照してください。

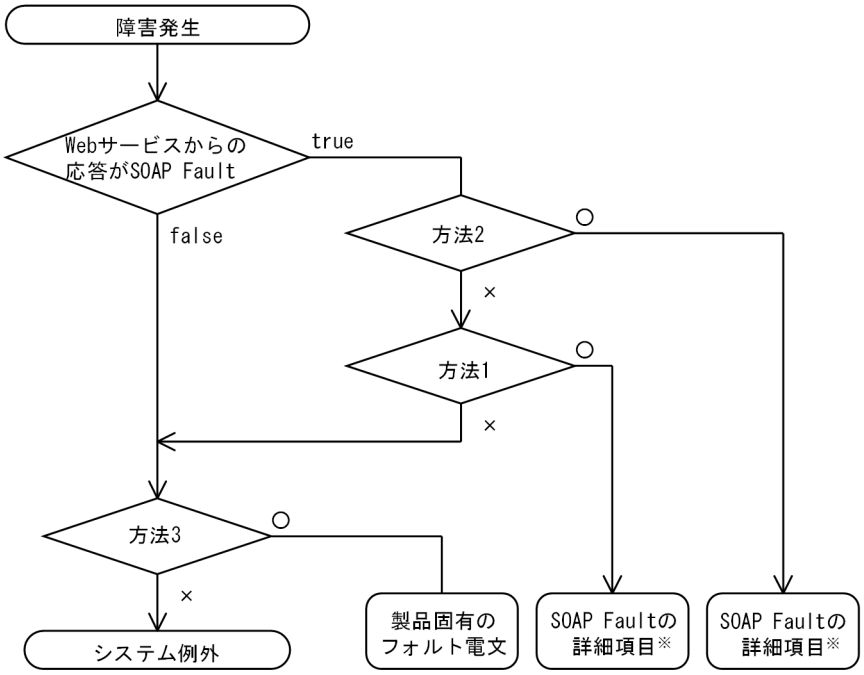
4.3.1　SOAP アダプタのフォルト判別の概要

SOAP アダプタからフォルト電文を応答する方法は次の 3 つです。

- SOAP Fault 電文判別
- SOAP Fault 動作定義ファイル
- システム例外をフォルトに変換

フォルト判別の流れを次の図に示します。

図 4-2　フォルト判別の流れ



（凡例）

○：フォルト電文で応答する条件を満たしている

×：フォルト電文で応答する条件を満たしていない

方法1：SOAP Fault電文判別

方法2：SOAP Fault動作定義ファイル

方法3：システム例外をフォルトに変換

注※

SOAP Fault の詳細項目とは detail 要素の子要素または Detail 要素の子要素，およびその子孫要素を指します。詳細項目が複数ある場合，その中の 1 つだけがフォルト電文として扱われます。詳細項目が存在しない場合，フォルト電文は null になります。

(1) SOAP Fault 電文判別

SOAP Fault によって伝搬されたエラー情報が，WSDL で定義されたユーザ定義例外の場合に，SOAP アダプタからフォルトを応答します。

SOAP メッセージの SOAP Fault は，SOAP メッセージ内のエラー情報を伝搬するために使用されます。サービスプラットフォームでは SOAP Fault のうち，ユーザ定義例外を伝搬する SOAP Fault がフォルトとして扱われます。フォルトとして扱われなかった SOAP Fault はシステム例外となり，KDEC03006-E または KDEC03024-E でフォルト情報が出力されます。

ユーザ定義例外を伝搬する SOAP Fault であるかどうかの判別方法は，HCSC サーバランタイム定義ファイルの soapfault-handling-mode プロパティの設定に従います。

soapfault-handling-mode プロパティの詳細は，マニュアル「サービスプラットフォーム リファレンス」の「6.5.6　HCSC サーバランタイム定義ファイル」を参照してください。

(2) SOAP Fault 動作定義ファイル

SOAP Fault 電文判別の方法でユーザ定義例外として判別できない SOAP Fault についても，SOAP Fault 動作定義ファイルを使用することで SOAP アダプタからフォルトとして応答できます。SOAP Fault 動作定義ファイルは HCSC サーバランタイム定義ファイルの soapfault-targetnamespace-filepath プロパティを設定することで使用できます。SOAP Fault 動作定義ファイルの詳細は，マニュアル「サービスプラットフォーム リファレンス」の「6.6.1　SOAP Fault 動作定義ファイル」を参照してください。

(3) システム例外をフォルトに変換

前述の方法によりフォルトとして扱われなかった SOAP Fault や，システム例外（通信エラーなど）を，製品固有のフォルトとして応答するための方法です。

システム例外をフォルト電文へ変換するかしないかは，サービスアダプタ定義画面でアダプタを定義するときに設定します。サービスアダプタの定義方法については，マニュアル「サービスプラットフォーム 開発ガイド 受付・アダプタ定義編」の「3.　サービスアダプタの定義」を，サービスアダプタ定義画面の詳細は，マニュアル「サービスプラットフォーム リファレンス」の「1.2.2　サービスアダプタ定義画面」を参照してください。

4.3.2　SOAP Fault の詳細項目によるフォルト判別

ここでは，HCSC サーバランタイム定義ファイルの soapfault-handling-mode プロパティに DETAIL を指定した場合のフォルト判別方法について説明します。

(1)　SOAP1.1 の SOAP Fault

次の条件を満たす SOAP Fault がフォルトとして扱われます。

- SOAP Fault の detail 要素の直下の子要素が，名前空間を含めて WSDL フォルト名と一致している

次の SOAP エンジンによってユーザ定義例外（フォルト bean）からマーシャルされた SOAP Fault の場合，この判別方法でフォルトとして扱われます。

- Cosminexus Web Services
- Cosminexus JAX-WS

フォルトとして扱われる SOAP1.1 の SOAP Fault の例を次に示します。

図 4-3　フォルトとして扱われる SOAP1.1 の SOAP Fault の例（詳細項目によるフォルト判別）

```
<S:Envelope xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">
  <S:Body>
    <S:Fault>
      <faultcode xmlns:ns1="http://server.sampletp">ns1:UserException</faultcode>
      <faultstring>KDCCP0015-E The user definition exception occurred. exceptionName =
sampletp.server.UserException</faultstring>
      <detail>
        <ns2:UserException xmlns:ns2="http://server.sampletp">
          <errorcode>0000</errorcode>
          <errormessage>Error Message</errormessage>
        </ns2:UserException>
      </detail>
    </S:Fault>
  </S:Body>
</S:Envelope>
```

フォルト判別に使用する

応答するフォルト電文

(2)　SOAP1.2 の SOAP Fault

次の条件を満たす SOAP Fault がフォルトとして扱われます。

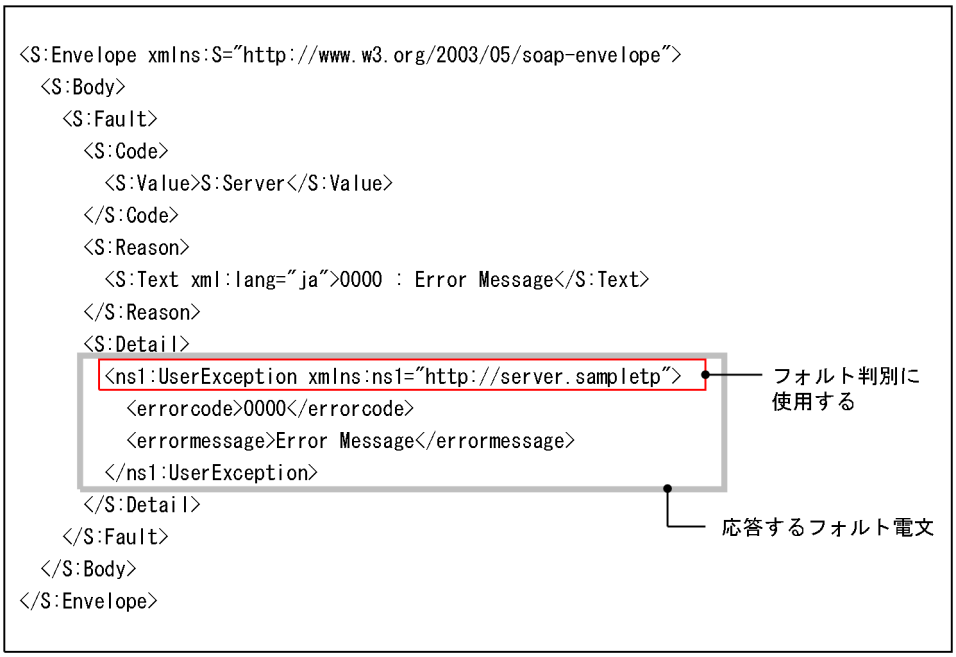
- SOAP Fault の Detail 要素の直下の子要素が，名前空間を含めて WSDL フォルト名と一致している

次の SOAP エンジンによってユーザ定義例外（フォルト bean）からマーシャルされた SOAP Fault の場合，この判別方法でフォルトとして扱われます。

- Cosminexus JAX-WS

フォルトとして扱われる SOAP1.2 の SOAP Fault の例を次に示します。

図 4-4　フォルトとして扱われる SOAP1.2 の SOAP Fault の例（詳細項目によるフォルト判別）



4.3.3　SOAP Fault の faultcode によるフォルト判別

ここでは，HCSC サーバランタイム定義ファイルの soapfault-handling-mode プロパティに FAULTCODE を指定した場合のフォルト判別方法について説明します。

(1)　SOAP1.1 の SOAP Fault

次の条件を満たす SOAP Fault がフォルトとして扱われます。

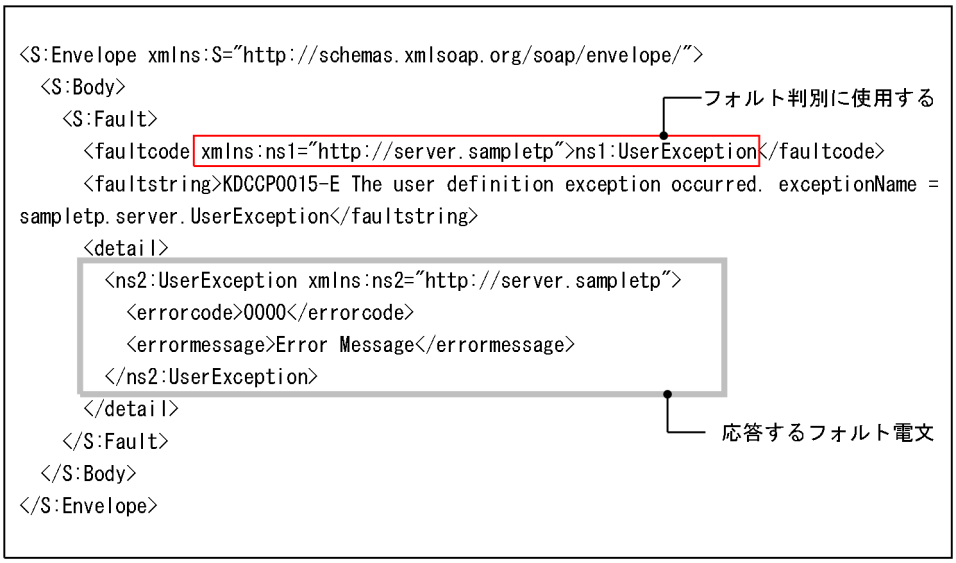
- SOAP Fault の faultcode 要素の値が，名前空間を含めて WSDL フォルト名と一致している

次の SOAP エンジンによってユーザ定義例外（フォルト bean）からマーシャルされた SOAP Fault の場合，この判別方法でフォルトとして扱われます。

- Cosminexus Web Services

フォルトとして扱われる SOAP1.1 の SOAP Fault の例を次に示します。

図 4-5　フォルトとして扱われる SOAP1.1 の SOAP Fault の例（faultcode によるフォルト判別）



(2)　SOAP1.2 の SOAP Fault

フォルトとして扱われません。

4.3.4　システム例外のフォルト変換

サービスアダプタからサービス部品（Web サービス）を呼び出したときに発生したエラーの例外を，そのままスローするか，サービスアダプタでフォルト電文に変換するか選択できます。例外をそのままスローする場合とフォルト電文に変換する場合の処理の違いについては，「4.8　例外発生時にサービスアダプタが取得するエラーの選択」を参照してください。

4.3.5　SOAP Fault 動作定義ファイルによるフォルトのキャッチ

SOAP アダプタでの SOAP Fault の扱いと，フォルトとして扱う場合の注意事項について説明します。

(1)　SOAP Fault の扱い

SOAP アダプタでは，サービス部品から返された SOAP Fault が次に示す条件をすべて満たす場合，フォルトとして扱われます。

- サービス部品の WSDL に定義されている
- フォルト名が faultCode に設定されている

このようなフォルトをユーザ定義例外と呼びます。ユーザ定義例外は，ビジネスプロセスでも認識されるため，フォルト発生後のプロセスの場合分け処理を定義できます。

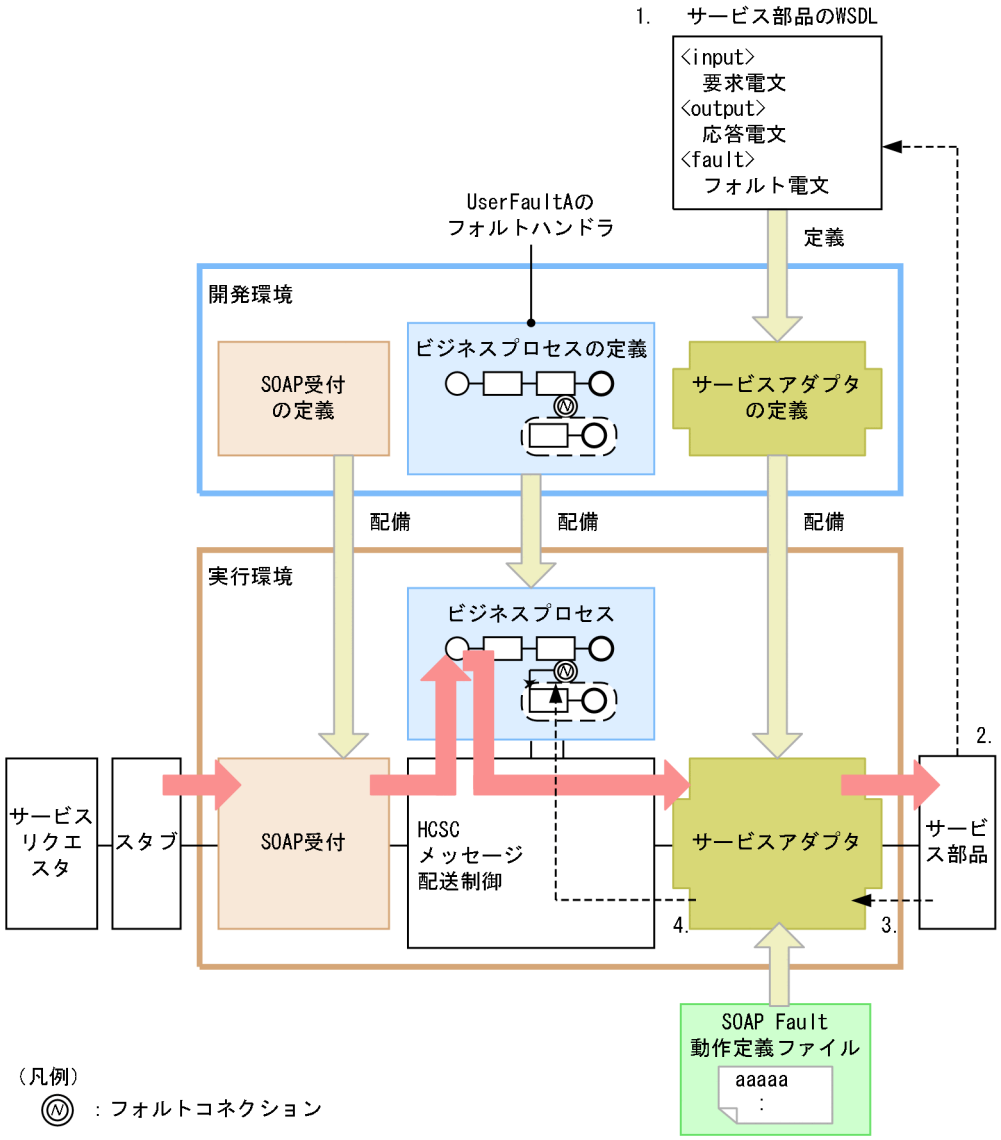
サービス部品の種類によっては，WSDL に定義されていない SOAP Fault が返されたり，faultCode に任意の値が設定されて SOAP Fault が返されたりすることがあります。このような SOAP Fault をフォルトとして扱いたいときは，SOAP Fault 動作定義ファイルを使用することで，サービス部品から返されるすべての SOAP Fault をフォルトとして扱うことができるようになります。

SOAP Fault 動作定義ファイルには，サービス部品の WSDL に宣言されている名前空間（targetNamespace の URI）を定義します。SOAP Fault 動作定義ファイルの詳細は，マニュアル「サービスプラットフォーム リファレンス」の「6.6.1　SOAP Fault 動作定義ファイル」を参照してください。

個々のサービスアダプタには，WSDL でサービス部品のインターフェースが定義されており，WSDL に定義されている名前空間が，SOAP Fault 動作定義ファイルに定義されている内容と一致するかどうかで，フォルトとして扱うかどうかを判断します。

SOAP Fault 動作定義ファイルを使用したときの SOAP Fault の扱いを次の図に示します。

図 4-6　SOAP Fault 動作定義ファイルを使用したときの SOAP Fault の扱い



1. サービス部品の WSDL を定義します。

2. サービス部品から返された SOAP Fault が，サービス部品の WSDL に定義されていればフォルトとして扱われます。
3. サービス部品から返された SOAP Fault が，サービス部品の WSDL に定義されていない，または WSDL の定義と異なる名前空間で定義されている場合は，SOAP Fault 動作定義ファイルに定義されている内容と一致するかどうかを照合します。
4. SOAP Fault 動作定義ファイルに定義されている内容と一致している場合は，フォルトとしてビジネスプロセスに返します。

(2) SOAP Fault をフォルトとして扱う場合の注意事項

サービス部品から返された SOAP Fault では，次の条件をすべて満たす場合にフォルト名が faultCode に設定され，フォルトとして扱われます。

- サービス部品側に SOAP 通信基盤を使用していること。
- fault メッセージを定義する wsdl:part 要素が，必ず element 属性で xsd:schema 要素内の element 要素を参照していること。
- WSDL fault 名（wsdl:fault 要素の name 属性値）と，schema（wsdl:types 内）に定義された例外型 complexType を指す element 名※が，名前空間を含めて一致していること。

これらの条件を満たしていない WSDL を利用する場合で，サービス部品から返された SOAP Fault をフォルトとして扱いたいときは，SOAP Fault 動作定義ファイルを定義してユーザ定義例外となるように設定します。

注※

element 名とは，「wsdl:fault が参照する wsdl:part 要素が，element 属性を用いて指定する要素」のことです。WSDL を次の順にたどって特定できる要素を指します。

1. wsdl:fault
2. wsdl:message（この要素をたどらない場合もあります）
3. wsdl:part
4. xsd:element

なお，サービスプラットフォームの開発支援コマンドである Java2WSDL コマンドを使用して，オペレーションの style 属性が「rpc」の WSDL ファイルを生成した場合，element 属性ではなく，type 属性で参照する WSDL ファイルが生成されます。そのため，生成された WSDL ファイルをそのまま使用しないで，修正および再生成してから使用します。WSDL ファイルの修正および再生成の方法については，マニュアル「サービスプラットフォーム 開発ガイド 基本開発編」の「4.3.2　サービス部品電文を作成する(Web サービスの場合)」を参照してください。

サービスプラットフォームでフォルトを使用する場合，document スタイルの WSDL を使用することを推奨します。

4.4　アプリケーションからの稼働状態の確認

4.4.1　稼働状態の確認の概要

サービスリクエスタなどの外部から Web サービス（SOAP 通信）を使用し，サービス名をキーにして稼働状態の確認要求をすると，HCSC サーバに配備されているサービスアダプタの情報を取得できます。

なお，稼働状態の確認は，サービス部品そのものの状態を確認するのではなく，HCSC サーバに配備されているサービスアダプタの状態を確認します。

4.4.2　稼働状態の確認の流れ

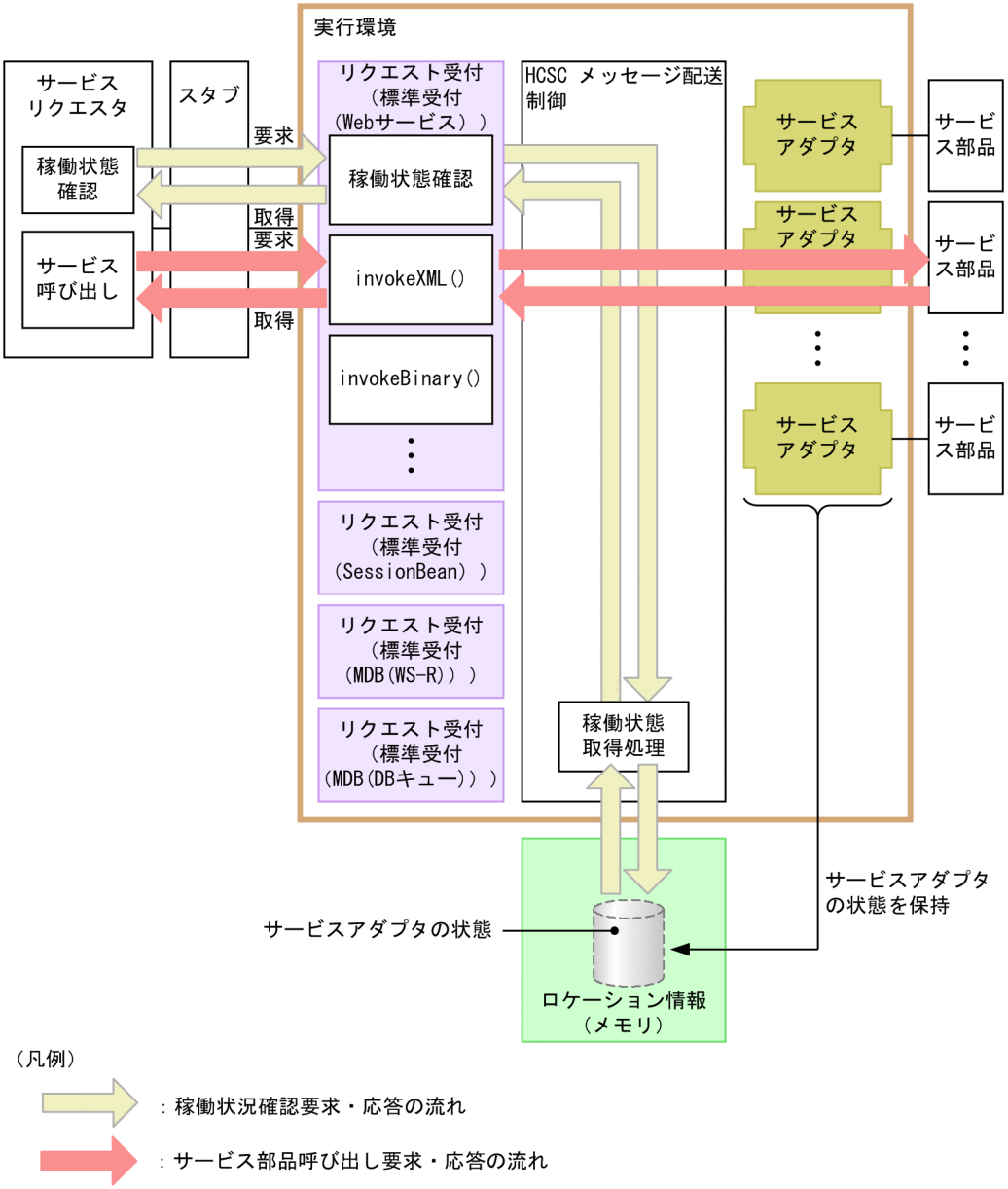
リクエスト受付（標準受付（Web サービス））にはサービスリクエスタからのメッセージを配送するためのメソッド（リクエスト受付のオペレーション）として invokeXML，invokeBinary などがあり，その中のサービスアダプタ稼働状態確認用のオペレーション（getServiceInfo）を使用し，稼働状態を確認します。

サービスリクエスタからサービスアダプタの稼働状態の確認要求をすると，HCSC サーバは指定されたサービス名に該当するサービスアダプタのロケーション情報（メモリ上の情報）を検索し，その状態を応答します。

なお，稼働状態確認の要求ができるのは，HCSC サーバのリクエスト受付が起動しているときだけです。

稼働状態の確認の流れを次の図に示します。

図 4-7　稼働状態の確認の流れ



4.4.3　稼働状態の確認時の注意事項

(1)　サービス部品の状態とサービスアダプタの状態の同期について

稼働状態確認機能は，サービス部品の状態ではなく，HCSC サーバに配備されているサービスアダプタの状態を確認する機能です。そのため，稼働状態を確認するときは，サービス部品とサービスアダプタが共に同期を取って起動および停止する運用が前提です。サービスアダプタが起動していてもサービス部品が停止している場合は，サービス部品呼び出しの要求はエラーになります。

(2) 稼働状態の確認結果と実状態の差異について

稼働状態の確認でサービスアダプタの起動を確認した場合でも，確認直後にサービスアダプタが停止すると，サービス部品呼び出しの要求をしたときにエラーになります。そのため，サービスリクエスタでは，サービスアダプタの稼働状態確認で起動を確認しても，サービス部品呼び出しを要求したときにサービスアダプタが停止状態となっていてエラーリターンする場合があることを考慮する必要があります。

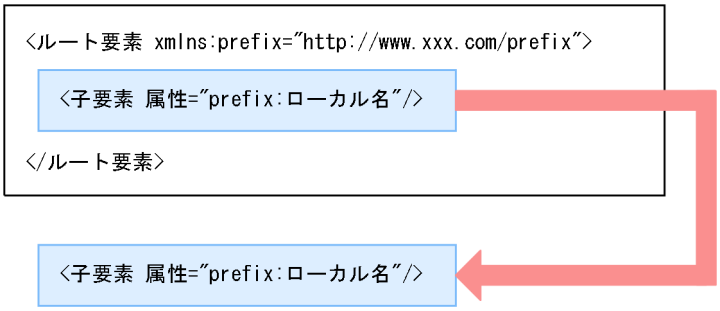
(3) リクエスト受付の状態について

HCSC サーバのリクエスト受付が起動していないときに，サービスリクエスタでサービスアダプタの稼働状態を確認するとエラーになります。そのため，サービスリクエスタでは，リクエスト受付が停止状態で，サービスアダプタの稼働状態確認要求がエラーリターンする場合を考慮する必要があります。

4.5　XML 電文内の属性値に指定した名前空間接頭辞の補完

HCSC サーバ内を流れるユーザ電文で，XML スキーマの属性値の型に QName 型を使用する場合，次の図のように XML 文書から「子要素」を部分木として切り出して出力すると，xmlns:prefix の名前空間宣言が欠落します。

図 4-8　部分木取得時の名前空間宣言の欠落



名前空間宣言が欠落した場合，次の問題が発生することがあります。

- ビジネスプロセスでフォルトコネクションを使用してフォルトをキャッチしようとしたが，サービス部品からのフォルトをキャッチできない。
- データ検証機能を使用している場合（HCSC サーバランタイム定義ファイルの telegram-validation プロパティに ON を指定），検証でエラーが発生する。
- 名前空間宣言が欠落した XML 電文をサービスリクエスタやサービス部品で受け取った場合，サービスリクエスタやサービス部品側でエラーが発生する。

なお，子要素のプレフィックスについては，部分木取得時に名前空間が解決されます。

このような問題を解決するには，HCSC サーバランタイム定義ファイルの xmltelegram-namespace-complement プロパティに ON を指定します。ただし，ON を指定すると，OFF を指定した場合よりも電文サイズが大きくなるため注意してください。

HCSC サーバランタイム定義ファイルについては，マニュアル「サービスプラットフォーム リファレンス」の「6.5.6　HCSC サーバランタイム定義ファイル」を参照してください。

4.6　任意形式（any 形式）の電文フォーマットの定義

開発環境で電文のフォーマット定義ファイルに任意形式（any 形式）を設定することで，電文の形式を意識しないサービス部品呼び出し要求・応答ができます。

なお，任意形式のデータを扱うことができる受付およびアダプタは，次のとおりです。

- ユーザ定義受付（カスタム受付）
- 汎用カスタムアダプタ

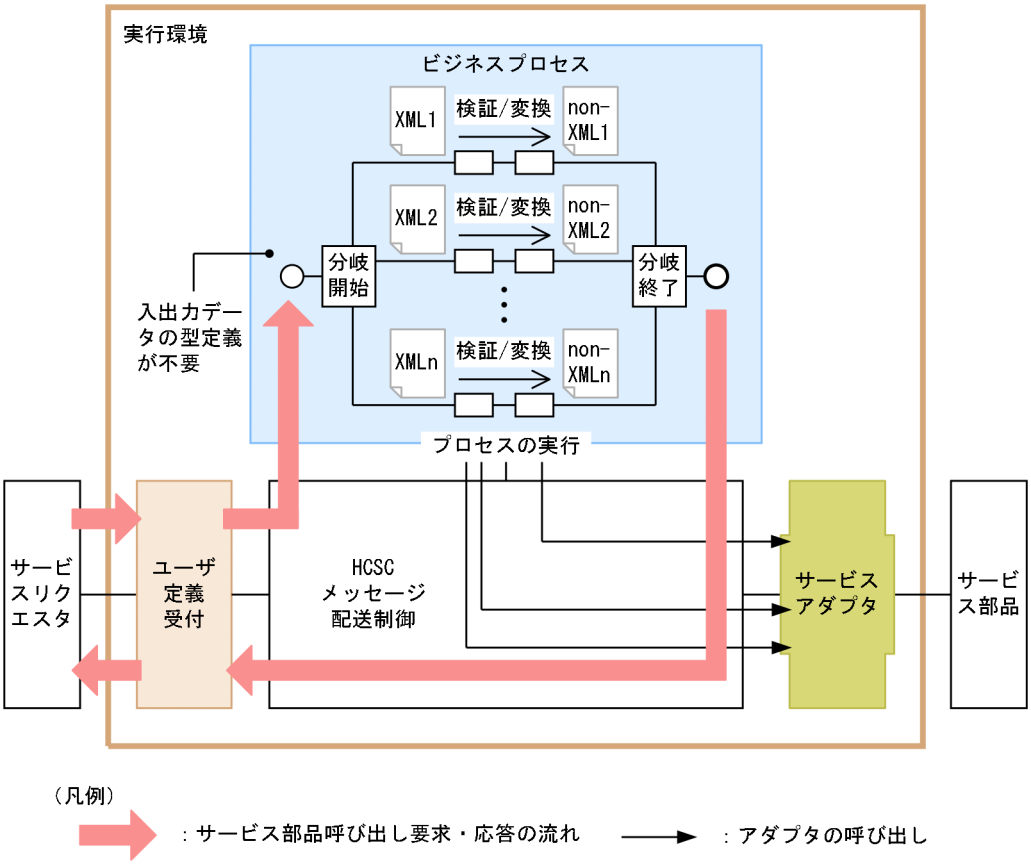
データ変換をしないメッセージ型の組み合わせについては「4.6.2　データ変換をしない場合の変数（メッセージ型）の組み合わせ」を参照してください。

4.6.1　任意形式（any 形式）を設定した場合の処理の流れ

電文のフォーマット定義ファイルに任意形式（any 形式）を設定した場合は，要求電文および応答電文のデータの型を意識する必要がありません。

電文のフォーマット定義ファイルに任意形式（any 形式）を設定した場合の処理の流れを次の図に示します。

図 4-9　電文のフォーマット定義ファイルに任意形式（any 形式）を設定した場合の処理の流れ

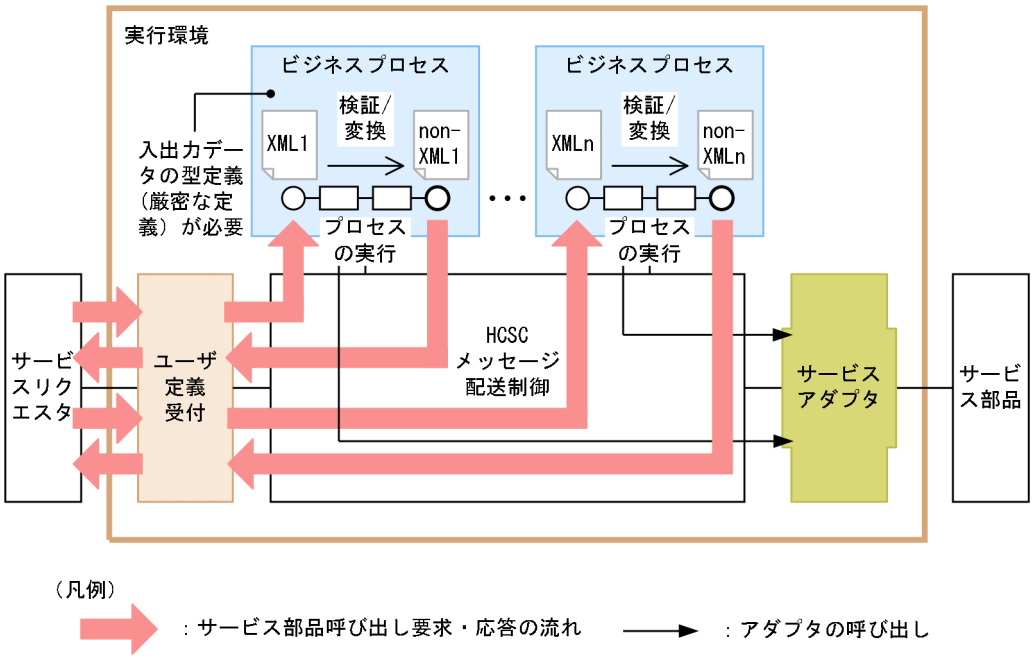


参考

電文のフォーマット定義ファイルに任意形式（any 形式）を設定しない場合は，要求電文および応答電文のデータ型を厳密に決める必要があります。また，開発環境で電文の形式ごとにコンポーネント（受付，ビジネスプロセス，およびサービスアダプタ）を作成する必要があります。

電文のフォーマット定義ファイルに任意形式（any 形式）を設定しない場合の処理の流れを次の図に示します。

図 4-10　電文のフォーマット定義ファイルに任意形式（any 形式）を設定しない場合の処理の流れ



4.6.2　データ変換をしない場合の変数（メッセージ型）の組み合わせ

任意形式（any 形式）を設定することで，データ変換をしない運用ができます。データ変換をしない場合の前提条件は次のとおりです。

- サービス呼出アクティビティのメッセージ型（XML，non-XML，any）とサービス呼出アクティビティから呼び出すコンポーネント（開発環境で作成する受付，ビジネスプロセス，およびサービスアダプタの総称）で設定したデータの型が一致している。
- ビジネスプロセスのサービス呼出アクティビティで任意形式（any 形式）を設定する場合は，開発環境の定義画面で呼び出し対象のコンポーネントを任意形式（any 形式）で定義している。

受付およびサービスアダプタでデータ変換をしない場合のメッセージ型の組み合わせを次に示します。

表 4-11　受付でデータ変換をしない場合のメッセージ型の組み合わせ

コンポーネント種別	データの型 または メソッド		変数（メッセージ型）					
			XML		non-XML		any	
			要求	応答	要求	応答	要求	応答
標準受付（Web サービス）	invokeXML		○	○	×	×	×	×
	invokeBinary		×	×	○	○	○	○
標準受付（SessionBean）	invokeXML		○	○	×	×	×	×
	invokeBinary		×	×	○	○	○	○
標準受付（MDB（WS-R））	onMessage		○	○	○	○	○	○
標準受付（MDB（DB キュー））	onMessage		○	○	○	○	○	○
ユーザ定義受付（SOAP 受付）			○	○	×	×	×	×
ユーザ定義受付（カスタム受付）	invokeXML	XML	○	○	×	×	×	×
		non-XML	×	×	×	×	×	×
		any	○	○	○	○	○	○
	invokeBinary	XML	×	×	×	×	×	×
		non-XML	×	×	○	○	×	×
		any	○	○	○	○	○	○
ユーザ定義受付（TP1/RPC 受付）	XML		×	×	×	×	×	×
	non-XML		×	×	○	○	○	○
	any		○	○	○	○	○	○
ユーザ定義受付（FTP 受付）	XML		○	○	×	×	×	×
	non-XML		×	×	×	×	×	×
	any		×	×	×	×	×	×
ユーザ定義受付（HTTP 受付）	XML		○	○	×	×	×	×
	non-XML		×	×	○	○	○	○
	any		×	×	○	○	○	○
ユーザ定義受付（Message Queue 受付）	XML		○	○	×	×	○	○
	non-XML		×	×	○	○	○	○
	any		○	○	○	○	○	○
ユーザ定義受付（ファイルイベント受付）	XML		○	○	×	×	×	×

コンポーネント種別	データの型 または メソッド	変数（メッセージ型）					
		XML		non-XML		any	
		要求	応答	要求	応答	要求	応答
ユーザ定義受付（ファイルイベント受付）	non-XML	×	×	×	×	×	×
	any	×	×	×	×	×	×
ユーザ定義受付（Kafka 受付）	XML	○	○	×	×	○	○
	non-XML	×	×	○	○	○	○
	any	○	○	○	○	○	○
ユーザ定義受付（gRPC 受付）	XML	○	○	×	×	×	×
	non-XML	×	×	×	×	×	×
	any	×	×	×	×	×	×
ユーザ定義受付（スケジュール駆動受付）	XML	○	○	×	×	×	×
	non-XML	×	×	×	×	×	×
	any	×	×	×	×	×	×

（凡例）

○：対応しています。

×：対応していません。

表 4-12　サービスアダプタでデータ変換をしない場合のメッセージ型の組み合わせ

コンポーネント種別	データの型	変数（メッセージ型）					
		XML		non-XML		any	
		要求	応答	要求	応答	要求	応答
SOAP アダプタ	XML	○	○	×	×	×	×
	non-XML	−	−	−	−	−	−
	any	−	−	−	−	−	−
SessionBean アダプタ	XML	○	○	×	×	×	×
	non-XML	−	−	−	−	−	−
	any	−	−	−	−	−	−
MDB（WS-R）アダプタ	XML	○	−	×	−	×	−
	non-XML	×	−	○	−	×	−
	any	−	−	−	−	−	−
MDB（DB キュー）アダプタ	XML	○	−	×	−	×	−
	non-XML	×	−	○	−	×	−

コンポーネント種別	データの型	変数（メッセージ型）					
		XML		non-XML		any	
		要求	応答	要求	応答	要求	応答
MDB（DB キュー）アダプタ	any	−	−	−	−	−	−
DB アダプタ	XML	○	○	×	×	×	×
	non-XML	×	×	×	×	×	×
	any	×	×	×	×	×	×
TP1 アダプタ	XML	×	×	×	×	×	×
	non-XML	×	×	○	○	×	×
	any	×	×	×	×	○	○
FTP アダプタ	XML	○	○	×	×	×	×
	non-XML	×	×	×	×	×	×
	any	×	×	×	×	×	×
ファイルアダプタ	XML	○	○	×	×	×	×
	non-XML	×	×	○	○	×	×
	any	×	×	×	×	×	×
Object Access アダプタ	XML	○	○	×	×	×	×
	non-XML	×	×	×	×	×	×
	any	×	×	×	×	×	×
Message Queue アダプタ	XML	○	○	×	×	×	×
	non-XML	×	×	○	○	×	×
	any	×	×	×	×	×	×
ファイル操作アダプタ	XML	○	○	×	×	×	×
	non-XML	×	×	×	×	×	×
	any	×	×	×	×	×	×
メールアダプタ	XML	○	○	×	×	×	×
	non-XML	×	×	×	×	×	×
	any	×	×	×	×	×	×
HTTP アダプタ	XML	○	○	×	×	×	×
	non-XML	×	×	○	○	×	×
	any	×	×	×	×	○	○
コマンドアダプタ	XML	○	×	×	×	×	×

コンポーネント種別	データの型	変数（メッセージ型）					
		XML		non-XML		any	
		要求	応答	要求	応答	要求	応答
コマンドアダプタ	non-XML	×	×	×	○	×	×
	any	×	×	×	×	×	○
SFTP アダプタ	XML	○	○	×	×	×	×
	non-XML	×	×	×	×	×	×
	any	×	×	×	×	×	×
Kafka アダプタ	XML	○	○	×	×	×	×
	non-XML	×	×	○	×	×	×
	any	×	×	×	×	○	×
gRPC アダプタ	XML	○	○	×	×	×	×
	non-XML	×	×	×	×	×	×
	any	×	×	×	×	×	×
汎用カスタムアダプタ	XML	○	○	×	×	×	×
	non-XML	×	×	○	○	×	×
	any	×	×	×	×	○	○

（凡例）

○：対応しています。

×：対応していません。

－：作成できません。

任意形式（any 形式）を設定している HCSC コンポーネントを単体サービス実行した場合の対応可否を次の表に示します。

表 4-13　HCSC コンポーネントを単体サービス実行した場合の対応可否

コンポーネント種別	メソッド	フォーマット定義ファイルの設定	
		any	
		要求	応答
標準受付（Web サービス）	invokeXML	×	×
	invokeBinary	○	○
標準受付（SessionBean）	invokeXML	×	×
	invokeBinary	○	○
標準受付（MDB（WS-R））	onMessage	○	○

コンポーネント種別	メソッド	フォーマット定義ファイルの設定	
		any	
		要求	応答
標準受付（MDB（DB キュー））	onMessage	○	○

（凡例）

○：対応しています。

×：対応していません。

4.7　システム例外をフォルトに変換する汎用フォルト

アクティビティ内で発生したシステム例外を汎用フォルトに変換して送出することで，フォルトを発生させることができないアクティビティでもフォルトを送出できます。

4.7.1　システム例外をフォルトに変換する汎用フォルトの概要

汎用フォルト電文送出とは，アクティビティ内で発生したシステム例外を，共通のフォーマットで作成したフォルト電文に変換して送出することで，フォルトを発生させることができないアクティビティでもフォルトを送出できるようにする機能です。

この機能を使用した場合，プロセスインスタンスを中断しないで続行するため，プロセスインスタンスの再実行はできません。

(1)　アクティビティ内で発生したシステム例外をフォルト電文に変換して送出するアクティビティ

汎用フォルト送出に対応するアクティビティと汎用フォルトに変換する例外を次に示します。

表 4–14　汎用フォルト送出に対応するアクティビティと汎用フォルトに変換する例外

項番	アクティビティ種別	汎用フォルトに変換する例外
1	Java 呼出アクティビティ	• 引数用割当変数が未初期化の場合に発生する例外 • 戻り値用割当変数に対応する型以外を設定した場合に発生する例外
2	データ変換アクティビティ	• 変換元変数が未初期化の場合に発生する例外 • データ変換処理で発生する例外
3	代入アクティビティ	• コピー元変数が不正または未初期化の場合に発生する例外 • 代入処理で発生する例外
4	フォルト送出アクティビティ	割当変数が未初期化の場合に発生する例外
5	検証アクティビティ	• 検証する変数が未初期化の場合に発生する例外 • 検証する変数が空の場合に発生する例外
6	繰り返しアクティビティ	• 繰り返し条件の判定で発生する例外 • 繰り返しリストに指定した繰り返し対象が不正な場合に発生する例外 • 最大繰り返し回数に指定した回数を超えて繰り返しを実行した場合に発生する例外
7	分岐アクティビティ	分岐条件の判定で発生する例外
8	ログ出力アクティビティ	変数が不正または未初期化の場合に発生する例外
9	リンクコネクション	リンク活性化条件の判定で発生する例外

設定条件の違いによるアクティビティ内で例外が発生したときの動作の違いを次の表に示します。

表 4–15　設定条件の違いによるアクティビティ内で例外が発生したときの動作の違い

HCSC サーバランタイム定義ファイルの設定	アクティビティ内で例外が発生したときの動作	対象アクティビティ
syserr-to-fault-convert=ON	汎用フォルト電文を送出	• データ変換アクティビティ※ • 代入アクティビティ※ • 分岐アクティビティ※
syserr-to-fault-convert=OFF	システム例外が発生	
syserr-to-fault-convert-all-activity=ON	汎用フォルト電文を送出	• リンクコネクション • Java 呼出アクティビティ • データ変換アクティビティ※ • 代入アクティビティ※ • フォルト送出アクティビティ • 検証アクティビティ • 繰り返しアクティビティ • 分岐アクティビティ※ • ログ出力アクティビティ
syserr-to-fault-convert-all-activity=OFF	システム例外が発生	

注※

syserr-to-fault-convert プロパティと syserr-to-fault-convert-all-activity プロパティのどちらかだけ ON を指定した場合は，ON が適用され汎用フォルトが発生します。

syserr-to-fault-convert プロパティと syserr-to-fault-convert-all-activity プロパティは対象となるアクティビティが異なるだけです。出力する汎用フォルト電文の内容は同じです。syserr-to-fault-convert プロパティと syserr-to-fault-convert-all-activity プロパティを同時に指定してもエラーにはなりません。

(2)　検証アクティビティ

検証アクティビティは，validate-fault-compatible プロパティを設定することによって，検証処理の結果が不正だったときに送出するフォルト電文の種類を変更できます。

設定条件の違いによる検証処理の結果が不正だったときの動作の違いを次の表に示します。

表 4–16　設定条件の違いによる検証処理の結果が不正だったときの動作の違い

HCSC サーバランタイム定義ファイルの設定	検証処理の結果が不正だったときの動作
validate-fault-compatible=ON	検証アクティビティ固有のフォルト電文を送出※
validate-fault-compatible=OFF	汎用フォルト電文を送出※

注※

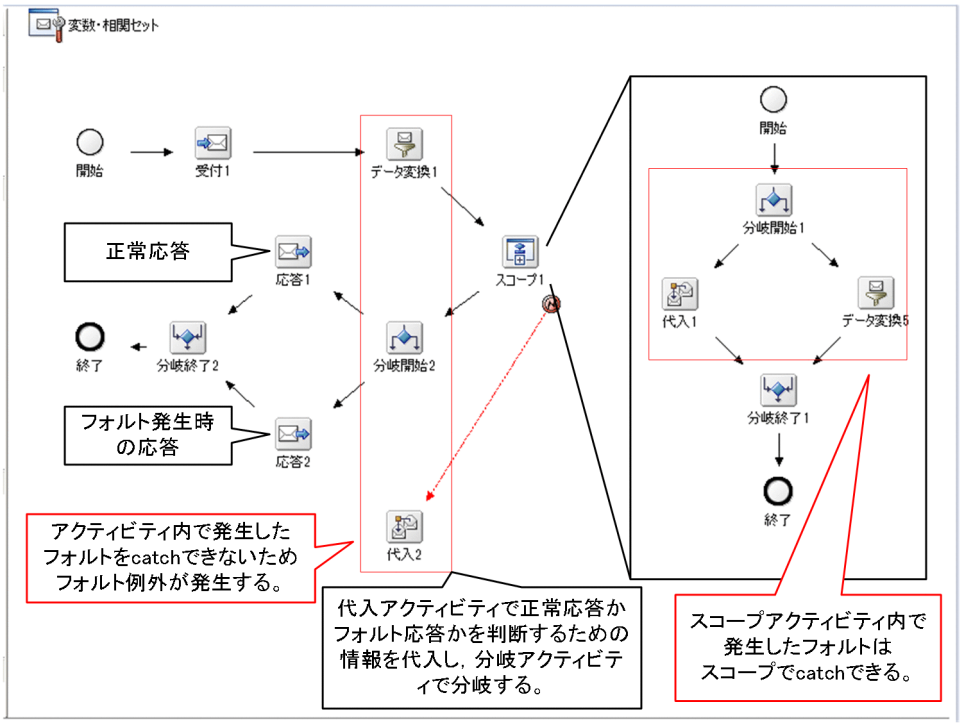
HCSC サーバランタイム定義ファイルの「validation-activity」の設定が ON になっている場合だけフォルト電文を送出します。「validation-activity」の設定が OFF の場合は検証処理そのものを実行しないため，フォルト電文も送出されません。

4.7.2　汎用フォルトを使用してシステム例外発生個所を特定する場合の流れ

アクティビティ内でシステム例外が発生する，または検証アクティビティの検証処理の結果が不正だった場合，発生した例外を汎用フォルト電文に変換して送出し，フォルト電文の情報から，メッセージログ内に出力されている障害発生個所特定用のメッセージ（KDEC20087-W）を引き当てることによって，ビジネスプロセス上の障害発生個所，障害原因の調査ができます。

汎用フォルトをスコープでキャッチするビジネスプロセス定義例を次に示します。

図 4-11　汎用フォルトをスコープでキャッチするビジネスプロセス定義例



汎用フォルトを使用してシステム例外の発生個所を特定する方法を次に示します。

1. スコープ 1 内に汎用フォルトを送出するアクティビティを定義します。
   アクティビティ内で例外が発生すると，汎用フォルトが送出されます。
2. スコープ 1 に定義されたフォルトハンドラで，送出された汎用フォルトをキャッチします（フォルト電文が変数に格納されます）。
3. 代入 2 で，汎用フォルト電文からルートアプリケーション情報を，応答電文に代入します。
   分岐アクティビティを使用することで正常時とフォルト発生時の応答を分けて定義することができます。
4. 応答 2 で，3.で汎用フォルト電文から取得したルートアプリケーション情報をリクエスタ（ユーザ）に通知します。
5. ユーザは，4.で取得したルートアプリケーション情報を基に，メッセージログを参照し，障害発生個所特定用のメッセージ（KDEC20087-W）の情報を参照します。

参考

ユーザは次の情報を参照することによって，障害発生個所や障害原因の調査が可能になります。

- 障害発生個所特定用のメッセージ（KDEC20087-W）に含まれるビジネスプロセス上の障害発生個所および障害原因の情報
- メッセージログ内の障害発生個所特定用のメッセージ（KDEC20087-W）付近に出力されるエラーメッセージ

4.7.3　汎用フォルト電文とメッセージログのマッピング

汎用フォルト電文に設定される情報（ルートアプリケーション情報）を基に，メッセージログに出力される障害発生個所特定用のメッセージ（KDEC20087-W）を参照することで，汎用フォルト電文とメッセージログのマッピングができます。

汎用フォルト電文とメッセージログのマッピングについては，「4.7.2　汎用フォルトを使用してシステム例外発生個所を特定する場合の流れ」を参照してください。

また，アクティビティでフォルトが発生した場合は，ユーザ電文トレースを出力することができます。ユーザ電文トレース機能を利用することによって，より詳細な情報を取得できます。

ユーザ電文トレースについては，マニュアル「サービスプラットフォーム システム構築・運用ガイド」の「7.4.4　ユーザ電文トレース」を参照してください。

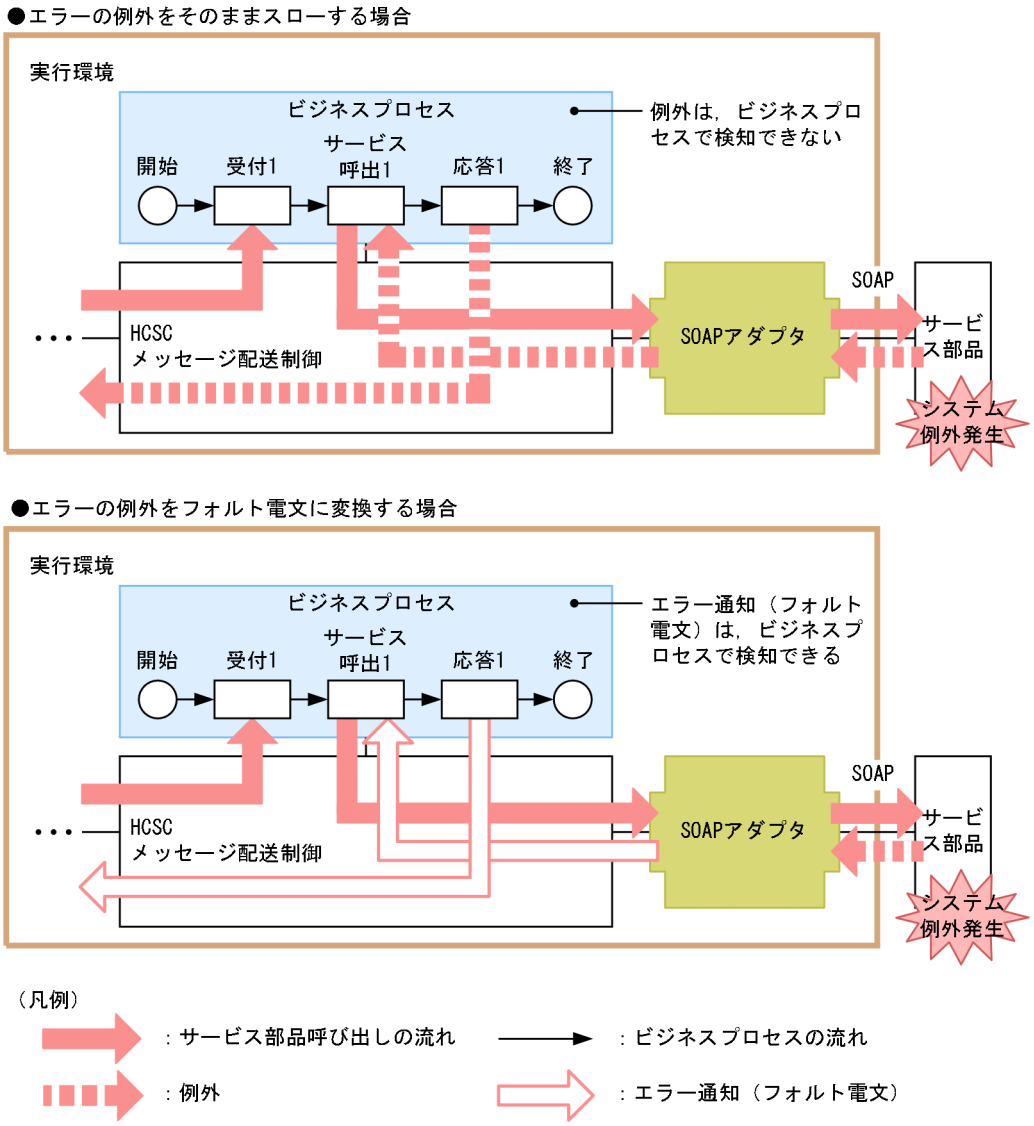
4.8　例外発生時にサービスアダプタが取得するエラーの選択

サービスアダプタからサービス部品（Web サービス）を呼び出したときに発生したエラーの例外を，そのままスローするか，サービスアダプタでフォルト電文に変換するか選択できます。

エラーの例外をサービスアダプタでフォルト電文に変換すると，ビジネスプロセスでエラーとしてキャッチできます。

例外をそのままスローする場合とフォルト電文に変換する場合の処理の違いを次に示します。

図 4-12　例外をそのままスローする場合とフォルト電文に変換する場合の処理の違い



なお，サービスアダプタで変換されたフォルト電文をビジネスプロセスでキャッチする場合，フォルト処理を行うアクティビティの配置が必要です。フォルト処理の定義については，マニュアル「サービスプラットフォーム 開発ガイド 基本開発編」の「5.4.4　フォルト処理を定義する」を参照してください。

例外をフォルト電文へ変換する場合の設定については，マニュアル「サービスプラットフォーム システム構築・運用ガイド」の「7.11　サービスアダプタでの例外発生時の運用」を参照してください。

4.9 UOC を利用したユーザ電文の値の変更

リクエスト電文をユーザ側で参照したり，編集したりするには，コンポーネント共通 UOC を利用します。また，コンポーネント共通 UOC を使用することで，受付およびサービスアダプタの入出力データの参照や更新ができます。

4.9.1 コンポーネント共通 UOC の概要

コンポーネント共通 UOC では，次の 2 つのことができます。

- UOC クラスのインスタンス生成
  ユーザが作成した java クラスをシステムクラスローダで読み込み，UOC クラスのインスタンスを作成します。
- UOC クラスの呼び出し
  ユーザが指定したコンポーネントで，ユーザが作成した java クラスを呼び出します。

(1) UOC クラスのインスタンス生成

UOC クラスのインスタンスは，HCSC サーバ起動時に作成され，メモリ上に保持されます。

UOC クラスのインスタンス生成の流れを次に示します。

1. コンポーネント共通 UOC 用プロパティファイルの読み込み
   コンポーネント共通 UOC 用プロパティファイルを読み込み，UOC クラスの呼び出し元 ID を key 情報，呼び出される UOC クラスのクラスパスを value とする Map 型のデータを作成し，jar ファイル作成の処理へ設定します。
   プロパティファイルの読み込みに失敗した場合，メッセージログに例外ログ（KDEC11003-E）を出力したあと，HCSC サーバ起動処理を中断します。
2. コンポーネント共通 UOC 用 jar ファイルの読み込みおよびインスタンス作成
   コンポーネント共通 UOC 用プロパティを読み込んで作成した Map 型データからクラスパスを取得し，UOC クラスのインスタンスを生成します。
   生成した UOC クラスのインスタンスは，クラスインスタンスを value，クラスパスを key とする Map 型データに格納します。
   クラスファイルの読み込みに失敗した場合，メッセージログに例外ログ（KDEC11002-E）を出力したあと，HCSC サーバ起動処理を中断します。

(2) UOC クラスの呼び出し

受付の共通処理部分，およびサービスアダプタの共通処理部分から呼び出すため，すべてのコンポーネントの処理で UOC クラスを呼び出すことができます。

コンポーネントの処理タイミングと，UOC クラスの呼び出しの有無を次に示します。

表 4-17　コンポーネントの処理タイミングと，UOC クラスの呼び出しの有無

種別		起動時	開始時	終了時			停止時
				正常	フォルト	システム例外	
標準受付		×	○	○	×	×	×
ユーザ定義受付	SOAP 受付の場合	×	○	○	×	×	×
	SOAP 受付以外の場合	○	○	○	×	×	○
ビジネスプロセス		×	×	×	×	×	×
サービスアダプタ		×	○	○	×	×	×

（凡例）

○：UOC クラスを呼び出します。

×：UOC クラスを呼び出しません。

(a)　UOC クラスへの入力情報

UOC クラスへの入力情報は，Map 型のオブジェクトにまとめて渡します。

UOC 用インターフェースで定義した定数（key）を基にデータを取得します。

定数（key）は，UOC クラスを作成する際に implements するインターフェースに定義します。

定義する定数を次に示します。

表 4-18　定義する定数

定数名（key 名）	データ型	設定内容
COMPO_TYPE	int	1：サービスアダプタ 2：受付
COMPO_KIND	String	受付の場合 • StandardWebService：標準受付（Web サービス） • StandardSessionBean：標準受付（SessionBean） • StandardWSR：標準受付（MDB（WS-R）） • StandardDBQ：標準受付（MDB（DB キュー）） • WebService：SOAP 受付 • TP1/RPC：TP1/RPC 受付 • FTP：FTP 受付 • HTTP：HTTP 受付

定数名（key 名）	データ型	設定内容
COMPO_KIND	String	• MessageQueue：Message Queue 受付 • FileEvent：ファイルイベント受付 • Kafka：Kafka 受付 • ScheduleTriggered：スケジュール駆動受付 • gRPC：gRPC 受付 • Custom：カスタム受付 サービスアダプタの場合 • WebService：SOAP アダプタ • SessionBean：SessionBean アダプタ • MDB_WSR：MDB（WS-R）アダプタ • MDB_DBQ：MDB（DB キュー）アダプタ • DB：DB アダプタ • TP1：TP1 アダプタ • File：ファイルアダプタ • ObjectAccess：Object Access アダプタ • MessageQueue：Message Queue アダプタ • FTP：FTP アダプタ • FileOperation：ファイル操作アダプタ • Mail：メールアダプタ • HTTP：HTTP アダプタ • Command：コマンドアダプタ • SFTP：SFTP アダプタ • Kafka：Kafka アダプタ • gRPC：gRPC アダプタ • Custom：汎用カスタムアダプタ
COMPO_ID	String	受付の受付 ID またはサービスアダプタのサービス ID
TELEGRAM_DATA	CSCMsgTelegramManager	リクエスト電文または応答電文
MONITOR_SEND_DATA	Map<String, Object>	送信データ格納 Map
DESTINATION_DATA	CSCMsgDestinationData	呼び出すサービスの宛先情報※

注※

ユーザ定義受付（SOAP 受付を除く）の場合だけ取得できます。

パラメタ Map から取得する電文の内容を次に示します。

●電文

リクエスト電文や応答電文は，電文取得用 API（CSCMsgTelegramManager）で UOC クラスに渡し，「#getHeader」または「#getBody」を使用して電文を取得します。

なお，電文はバイト配列で渡すため，XML 電文の場合は Document 型などに変換する必要があります。

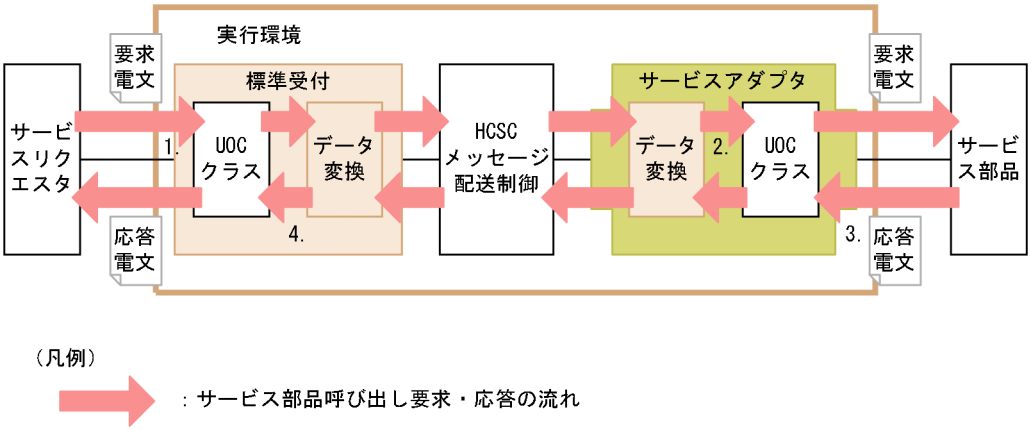
●宛先情報

SOAP ユーザ定義受付以外のユーザ定義受付の場合に呼び出すサービスの宛先情報が取得できます。また，ユーザ定義受付のリクエスト受付時にサービスの宛先情報取得用 API を使用し，任意の値を設定することで，呼び出すサービスを変更できます。

(b)　UOC クラスを呼び出すタイミング

UOC クラスを呼び出すタイミングと電文との関係を次に示します。

図 4-13　UOC クラスを呼び出すタイミングと電文との関係



1. 受付ではサービスリクエスタからの要求電文を UOC クラスに渡します。
2. サービスアダプタではサービス部品に渡す要求電文を UOC クラスに渡します。
3. サービスアダプタではサービス部品から返る応答電文を UOC クラスに渡します。
4. 受付ではサービスアダプタからの応答電文を UOC クラスに渡します。

注意

呼び出すサービスの宛先を変更した場合，リクエストトレースで出力される宛先は実際の宛先と異なることがあります。

(c)　性能解析トレースの出力

コンポーネント共通 UOC を使用すると，性能解析トレースは必ず出力されます。性能解析トレースが出力されるタイミングは，UOC クラスの呼び出し直前と直後になります。

性能解析トレースファイルに出力される形式は，J2EE サーバの性能解析トレースと同様です。性能解析トレースファイルの詳細については，マニュアル「アプリケーションサーバ 機能解説 保守／移行編」の「7.3 Management Server を利用した性能解析トレースファイルの収集」を参照してください。

4.9.2　コンポーネント共通 UOC の使用例

コンポーネント共通 UOC を使用して，受付で要求電文の値を変換および更新する例を次に示します。

なお，この例で使用するコンポーネントは次のとおりです。

- ユーザ定義受付名：サンプルユーザ定義受付 2
- 受付 ID：rcp2

コンポーネントとコンポーネント共通 UOC クラスの関連付けについては，マニュアル「サービスプラットフォーム 開発ガイド 基本開発編」の「付録 G　コンポーネント共通 UOC」を参照してください。

(1)　前提条件

前提となる動作環境を次に示します。

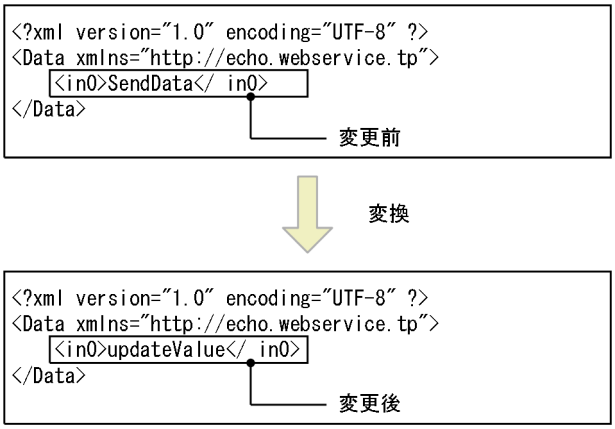
- J2EE サーバ名：J2EEServer
- HCSC サーバ名：HCSC
- SOAP モード：SOAP 1.1 モード
- サービスプラットフォームのインストールディレクトリ："C:¥Program Files¥Hitachi¥Cosminexus"
- jar ファイル格納ディレクトリ：C:¥uocjar
- プロパティ格納ディレクトリ：C:¥uocconf

(2)　処理内容

この例では，要求電文の一部を変換し，更新するという処理になります。

使用例の処理の内容を次に示します。

図 4-14　使用例の処理の内容



(3) UOC クラスの作成

要求電文のデータを変更する UOC クラスを作成する場合，次に示す処理を組み込む必要があります。

1. リクエストに対応する UOC クラス用インターフェースを implements する。
2. パラメタの CSCMsgTelegramManager オブジェクトから電文を取得する。
3. 電文を解析し，更新する。
   電文はバイト配列で取得されるため，Document 型などへの変換が必要です。
4. 電文をバイト配列に変換する。
5. 電文取得 API の電文更新用メソッドに，変換したバイト配列を設定する。

UOC クラスの作成例を次に示します。

図 4-15　UOC クラスの作成例

```
package jp.com.reception;

import java.util.Map;
          ：

        (中略)
          ：
public class Rcp2Uoc implements ComponentRequestUoc {                    ← 1.

    public void request(Map<String, Object> map) throws CSCUocSystemException {

        String nameSpase = "http://echo.webservice.tp";
        String tagName = "in0";
        byte[] bytes = null;                                                          ← 2.
        CSCMsgTelegramManager telegram = (CSCMsgTelegramManager)map.get(TELEGRAM_DATA);
        byte[] body = telegram.getBody();

        if( body != null ){
            DocumentBuilderFactory factory = DocumentBuilderFactory.newInstance();
            factory.setNamespaceAware(true);
            DocumentBuilder domBuilder = factory.newDocumentBuilder();
            Document dom = domBuilder.parse(new ByteArrayInputStream(body));

            NodeList list = dom.getElementsByTagNameNS(nameSpase,tagName);            ← 3.
            if((list != null) && (list.getLength() != 0)){
                if (list.item(0).getFirstChild() != null) {
                    list.item(0).getFirstChild().setNodeValue("updateValue");
                }
            }

            ByteArrayOutputStream os = new ByteArrayOutputStream();                   ← 4.
            TransformerFactory tfactory = TransformerFactory.newInstance();
            try {
                // Transformerオブジェクトを取得する
                Transformer transformer = tfactory.newTransformer();
                transformer.transform(new DOMSource(dom), new StreamResult(os));

                os.flush();
                // バイト配列を取得する
                bytes = os.toByteArray();
            } catch (Exception e) {
                e.printStackTrace();
                throw new CSCUocSystemException(e.getMessage());
            }

        }
        telegram.setBody(bytes);                                                  ← 5.
    }
}
```

注

　図中の番号は，組み込みが必要な処理の 1.～5.と対応しています。

(4) HCSC サーバの設定

(a) jar ファイルの準備

Ant などのビルドツールを使用して jar ファイルを作成し，jar ファイル格納ディレクトリ（この例では C:¥uocjar）に格納します。

(b) プロパティファイルの準備

コンポーネント共通 UOC 用プロパティファイルの設定内容を次に示します。

```
rcp2=jp.com.reception.Rcp2Uoc
```

HCSC サーバランタイム定義ファイルに，次に示す定義を追加します。

```
common-uoc-prop-path=C:¥¥uocconf
```

(c) J2EE サーバ用オプション定義ファイル（usrconf.cfg）の設定

usrconf.cfg ファイルに，次に示すプロパティ値を設定します。

```
add.class.path=C:¥uocjar¥uocClass2.jar
```

4.10　バイナリデータの変換エラー時のデバッグ情報の出力

バイナリデータの変換処理中にエラーが発生した場合に，原因を特定するための情報をデバッグ情報として出力できます。

4.10.1　デバッグ情報出力機能の概要

デバッグ情報出力機能は，バイナリデータの変換処理中にエラーが発生した場合に，原因を特定するための情報（デバッグ情報）をユーザ電文トレースに出力する機能です。

バイナリデータの変換処理では，バイナリデータと DOM（DOM パーサによって解析された XML 文書の内容を記述したもの）を，相互に変換します。

デバッグ情報は，バイナリデータの変換処理で，次の場合に出力されます。

- バイナリデータの読込処理（バイナリ→DOM の変換）でエラーが発生した場合
- バイナリデータの生成処理（DOM→バイナリの変換）でエラーが発生した場合

XML データ変換処理中にエラーが発生した場合は，デバッグ情報は出力されません。

また，バイナリデータの変換処理でエラーが発生した場合に，エラーの詳細情報として詳細情報メッセージが出力されます。

デバッグ情報出力機能が出力するデバッグ情報と，詳細情報メッセージの出力内容を併せて利用することで，バイナリデータの変換処理中のエラー原因を特定できます。

デバッグ情報の取得方法については，マニュアル「サービスプラットフォーム システム構築・運用ガイド」の「7.3.2　障害情報の取得方法（サービスプラットフォーム）」のデバッグ情報の取得方法に関する説明を参照してください。

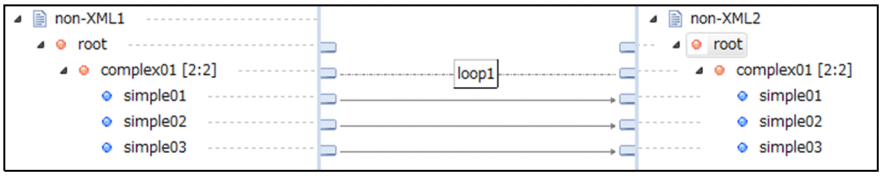
4.10.2　デバッグ情報の出力例

ここでは，バイナリデータの読込処理（バイナリ→DOM の変換）で，入力データの不正によってエラーになった場合のデバッグ情報の出力例について説明します。

(1)　データ変換の定義内容と入力データの内容

この例で使用するデータ変換のマッピング定義，変換元および変換先のバイナリフォーマット定義の設定値，変換するバイナリデータは次のとおりです。

●データ変換のマッピング定義



●変換元および変換先のバイナリフォーマット定義の設定値

```
【complex01の設定値】
 出現回数：2
【simple01～simple03の設定値】
 データ種別：文字列
 サイズ：3バイト
```

●変換するバイナリデータ

```
aaabbbcccdddeeeff
```

(2)　デバッグ情報の出力内容

(1)で示した内容でデータ変換を実行した場合，バイナリデータの読込処理（バイナリ→DOM の変換）で，入力データが短いため，エラー（KDEC40200-E）が発生します。

この場合のデバッグ情報の出力内容を次に示します。

(a)　デバッグ情報として出力する DOM の出力内容

デバッグ情報として出力する DOM の出力内容を次の図に示します。なお，次の図は，見やすさのためスペースと改行を挿入しています。

```
<?xml version="1.0" encoding="UTF-8"?>
<root>
  <complex01>
    <simple01>aaa</simple01>
    <simple02>bbb</simple02>
    <simple03>ccc</simple03>
  </complex01>
  <complex01>
    <simple01>ddd</simple01>
    <simple02>eee</simple02>
    <simple03/>
  </complex01>
</root>
```

説明

2 つ目の複合内容要素の子要素（simple03）のバイナリデータの読み込みでエラーが発生するため，ノードの値は出力されません。

次にトレース開始部の付加情報を示します。

```
DataType = DOM, Node = /root/complex01[2]/simple03, RootApInfo = XX.XXX.XXX.XXX/9496/0x00000
00000001cb0
```

注

「XX.XXX.XXX.XXX」の部分には IP アドレスが表示されます。

説明

トレース開始部の付加情報からは，次のことがわかります。

- 2 つ目の複合内容要素の子要素（simple03）の入力データ読み込み時にエラーが発生したこと
- ルートアプリケーション情報からメッセージログのエラー原因のメッセージ出力個所

(b) デバッグ情報として出力するバイナリデータの出力内容

読み込んだ位置までのバイナリデータが出力されます。

デバッグ情報として出力するバイナリデータの出力内容を次の図に示します。



次にトレース開始部の付加情報を示します。

```
DataType = Binary, InputDataSize = 17, Offset = 17, RootApInfo = XX.XXX.XXX.XXX/9496/0x0000000000001cb0
```

注

「XX.XXX.XXX.XXX」の部分には IP アドレスが表示されます。

説明

トレース開始部の付加情報からは，次のことがわかります。

- 入力データサイズの合計が 18 バイトではなく 17 バイトしかないこと
- ルートアプリケーション情報からメッセージログのエラー原因のメッセージ出力個所

(c) 詳細情報メッセージの出力内容

(1)で示した内容でデータ変換を実行した場合，データ変換でエラーになり，詳細情報メッセージが出力されます。詳細情報メッセージの出力例を次に示します。

```
…
KDEC40200-E バイナリデータの入力でエラーが発生しました。error=2
KDEC40310-E バイナリデータからDOMへの変換でエラーが発生しました。e=KDEC40200-E バイナリデータの入力でエラーが発生しました。error=2
KDEC40312-I バイナリデータからDOMへのデータ変換で発生したエラーの詳細情報を取得しました。node=/root/complex01[2]/simple03,offset=17,input data size=17,information=non-XML1
…
KDEC00001-E The service request was interrupted. (HCSCCommonID = CSC_HCSC_2017-07-26_10:21:45.166_7, ServiceRequestID = MSG_HCSC_SyncEJB_2017-07-26_10:21:45.166_7, RootApInfo = XX.XXX.XXX.XXX/9496/0x0000000000001cb0)
…
```

注

「XX.XXX.XXX.XXX」の部分には IP アドレスが表示されます。

説明

既存のエラーメッセージ（KDEC40200-E）では 3 バイト中，2 バイト読み込んだ際にエラーが発生したことがわかります。

詳細情報メッセージ（KDEC40312-I）からは，次のことがわかります。

- 入力データサイズの合計が 18 バイトではなく 17 バイトしかないこと
- 2 つ目の複合内容要素の子要素（simple03）の入力データ読み込み時にエラーが発生したこと

4.11　API を使用したデータ変換

API を使用したデータ変換について説明します。

4.11.1　データ変換 API を使用したデータ変換

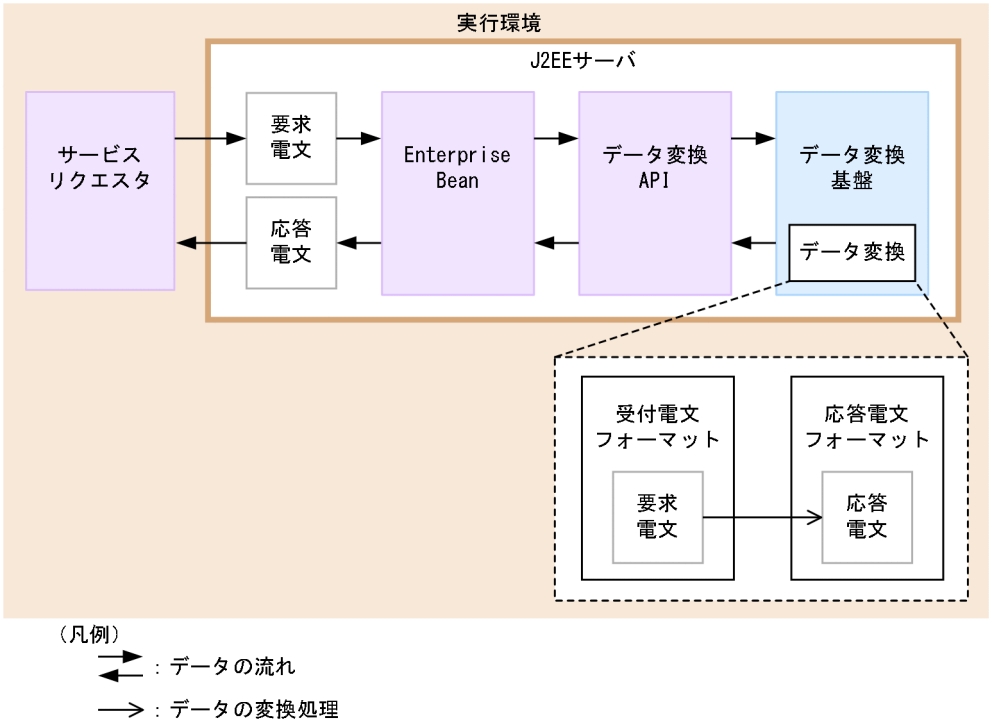
データ変換 API は，データ変換基盤を利用するためのインターフェースを定義した API です。

サービスプラットフォームでデータ変換を利用するには，HCSC サーバの構築やビジネスプロセスの定義が必要でした。データ変換 API を利用すれば，HCSC サーバの構築やビジネスプロセスを定義しなくてもデータ変換ができます。

データ変換 API を利用してデータ変換をする場合，データ変換 API を呼び出す Enterprise Bean を実装する必要があります。Enterprise Bean を J2EE サーバ上で稼働させ，サービスリクエスタから Enterprise Bean を経由してデータ変換 API を呼び出します。

データ変換 API を利用したデータ変換の実現方式を次の図に示します。

図 4-16　データ変換 API を利用したデータ変換の実現方式



データ変換 API では次の機能が利用できます。

- データ変換
  要求電文と定義情報を受け取り，データ変換後の応答電文を返す機能です。
- キャッシュ操作

定義情報のキャッシュを登録します。キャッシュ登録後の存在確認処理や削除処理も提供します。

- グループ管理

  キャッシュの保存するデータ変換 API のグループを管理する機能です。

- 障害対策

  データ変換やキャッシュ操作で障害が発生した際に，メッセージの出力とデータ変換 API 専用の例外クラスのオブジェクトをスローします。

- プロパティの変更

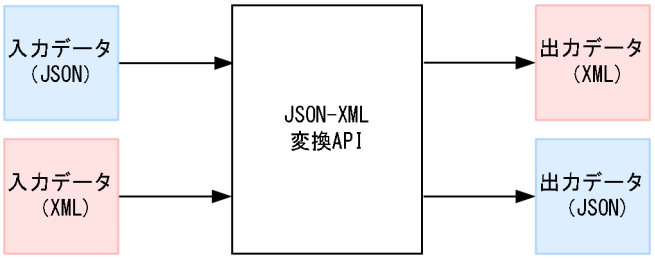
  データ変換に必要な設定（キャッシュやプロパティ）を保持するデータ変換 API を，業務ごとに管理する機能です。

4.11.2　JSON-XML 変換 API を使用した変換

JSON-XML 変換 API は，JSON 形式のデータと XML 形式のデータを相互に変換するための API です。JSON-XML 変換 API では，JSON 形式のデータと XML 形式のデータを一定の基準に従って変換します。

JSON-XML 変換 API を使用した変換例を次の図に示します。

図 4-17　JSON-XML 変換 API を使用した変換例



注意事項

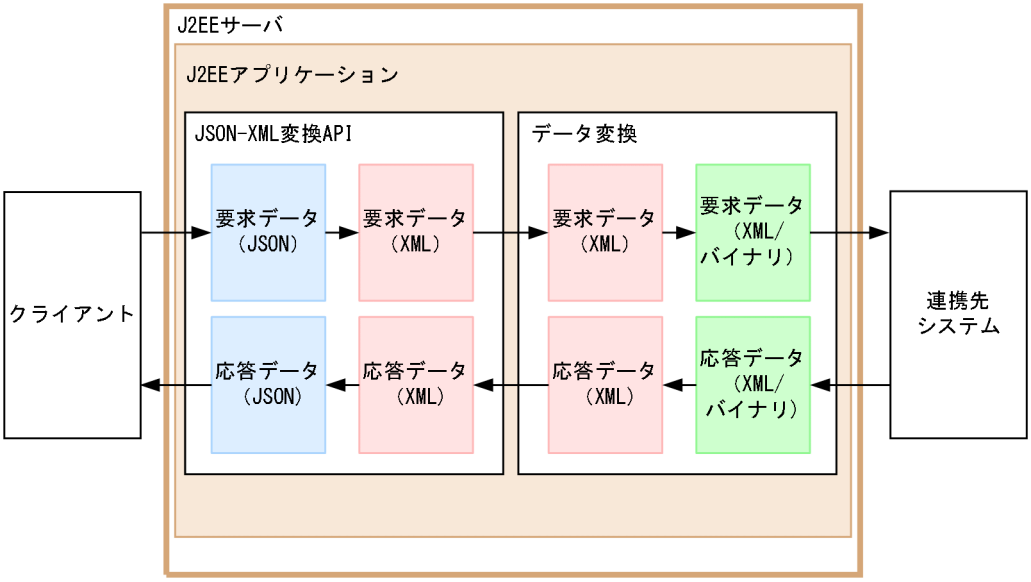
J2EE サーバ上で稼働する J2EE アプリケーション，またはビジネスプロセス中の Java 呼出アクティビティから，JSON-XML 変換 API を使用してください。

JSON-XML 変換 API を使用した変換の詳細については，マニュアル「サービスプラットフォーム 開発ガイド 基本開発編」の「13.2　JSON-XML 変換 API を使用したデータの形式変換」を参照してください。

JSON 形式のデータを XML 形式のデータに変換，または XML 形式のデータを JSON 形式のデータに変換することによって，サービスプラットフォームが提供するデータ変換機能と連携ができます。

データ変換機能と連携する変換例を次の図に示します。

図 4-18　データ変換機能と連携する変換例

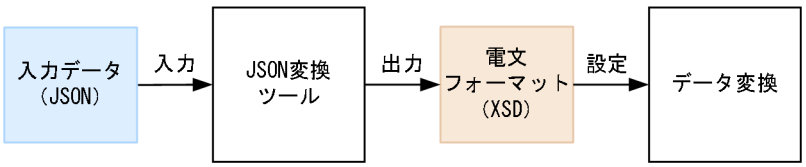


データ変換機能に設定するフォーマット（XML スキーマ）は，JSON 変換ツールで生成します。JSON 変換ツールは，JSON 形式のデータを受け取り，JSON-XML 変換後の XML 形式のデータに適合するフォーマットを生成します。

JSON 変換ツールの詳細については，マニュアル「サービスプラットフォーム 開発ガイド 基本開発編」の「14.　JSON 変換ツールを使用したデータ変換」を参照してください。

JSON 形式のデータからフォーマット（XML スキーマ）を作成し，設定するまでの流れを次の図に示します。

図 4-19　JSON 形式のデータからフォーマット（XML スキーマ）を作成，設定するまでの流れ



また，JSON 変換ツールを使用して，JSON-XML 変換の動作を確認できます。JSON-XML 変換をする前に JSON 変換ツールを実行して，JSON-XML 変換処理が正しくできるかどうか，および変換結果が正しいかどうかを確認してください。

JSON-XML 変換で扱う JSON と標準 JSON との差異，および JSON-XML 変換の規則の詳細については，マニュアル「サービスプラットフォーム 開発ガイド 基本開発編」の「11.　JSON データの変換」を参照してください。

JSON-XML 変換では，拡張モードを使用した JSON-XML 変換と標準モードを使用した JSON-XML 変換を行うことができます。

標準モードの JSON-XML 変換 API を使用したデータ変換の詳細については，マニュアル「サービスプラットフォーム 開発ガイド 基本開発編」の「13.2　JSON-XML 変換 API を使用したデータの形式変換」を参照してください。

拡張モードの JSON-XML 変換 API を使用したデータ変換の詳細については，マニュアル「サービスプラットフォーム 開発ガイド 基本開発編」の「12.　拡張モードでの JSON データの変換」を参照してください。

4.12　JSON 変換ツールを使用した変換

JSON 変換ツールは，サービスプラットフォームで JSON 形式のデータを利用するために必要な定義の開発およびデータ変換のテストを支援するためのツールです。

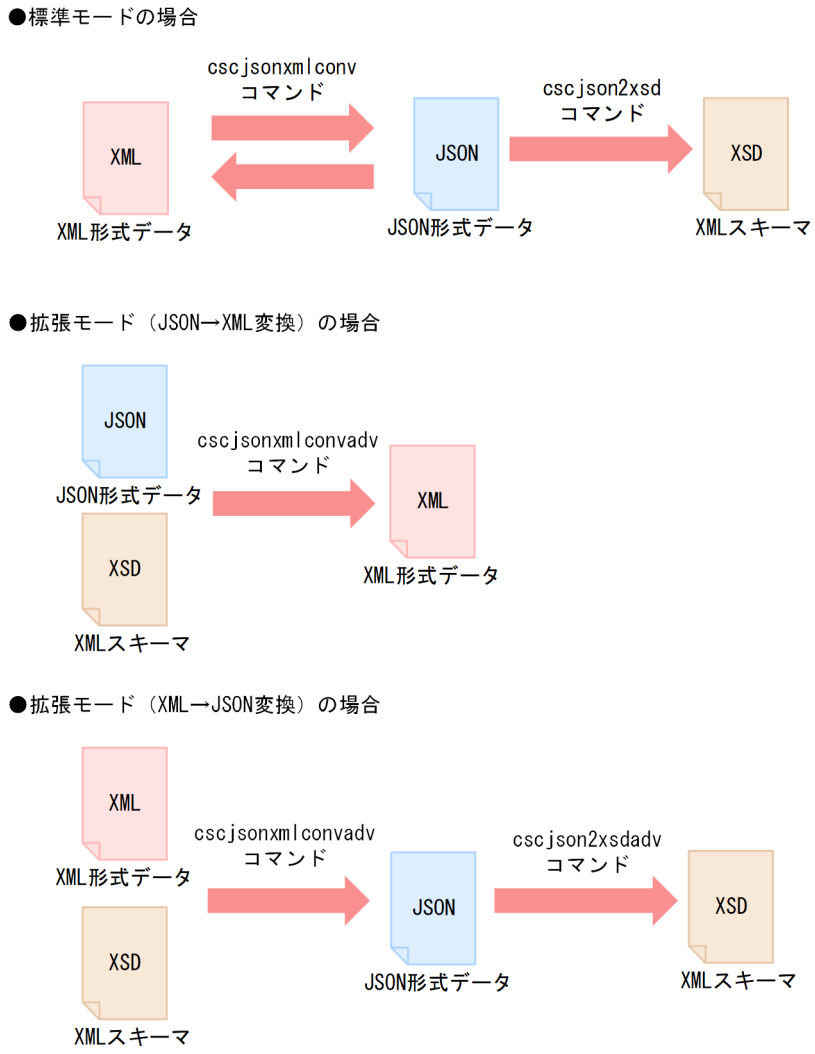
JSON 変換ツールは，次のコマンドで構成されています。

表 4-19　JSON 変換ツールの機能一覧

使用するモード	コマンド名	説明
標準モード	cscjsonxmlconv	JSON 形式データと XML 形式データとを相互に変換した結果をファイルとして生成します。サービスプラットフォームが自動的に変換する結果を，生成されたファイルと突き合わせて確認できます。
	cscjson2xsd	JSON 形式データに対応する XML スキーマを生成します。生成した XML スキーマをマッピング定義エディタで使用し，マッピング定義を作成します。
拡張モード	cscjsonxmlconvadv	JSON 形式データおよび XML スキーマを入力し，XML ファイルに変換します。また，その逆に XML ファイルと XML スキーマから JSON 形式データに変換することもできます。
	cscjson2xsdadv	JSON 形式データに対応する XML スキーマを生成します。

JSON 変換ツールとコマンドの関係を次に示します。

図 4-20　JSON 変換ツールとコマンドの関係



JSON 変換ツールを利用すると，JSON 形式データと XML 形式データの変換結果を取得し，どのように変換されるかを確認できます。

また，JSON 形式データから変換された XML 形式のデータに対応した，XML スキーマファイルを生成できます。

JSON-XML 変換の規則については，マニュアル「サービスプラットフォーム 開発ガイド 基本開発編」の「11.2　JSON-XML 変換の規則」を参照してください。

図中のそれぞれのコマンドについて説明します。

cscjsonxmlconv コマンドまたは cscjsonxmlconvadv コマンド

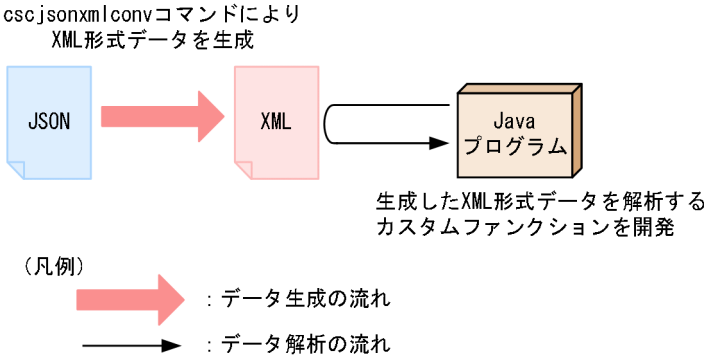
JSON 形式データと XML 形式データとを相互に変換した結果をファイルとして生成します。
cscjsonxmlconv コマンドは，JSON-XML 変換の標準モードで使用します。cscjsonxmlconvadv コマンドは，JSON-XML 変換の拡張モードで使用します。

JSON 形式データは，ビジネスプロセス上では XML 形式データとして扱うため，これらのコマンドによって生成した XML 形式データは，カスタムファンクションおよび Java アクティビティの設計およ

びテストに活用できます。また，データ変換コマンドまたはバイナリパースコマンドの入力データとしても利用できます。

JSON 形式データを利用した，カスタムファンクションおよび Java アクティビティの開発について次の図に示します。

図 4-21　JSON 形式データを利用したカスタムファンクションの開発



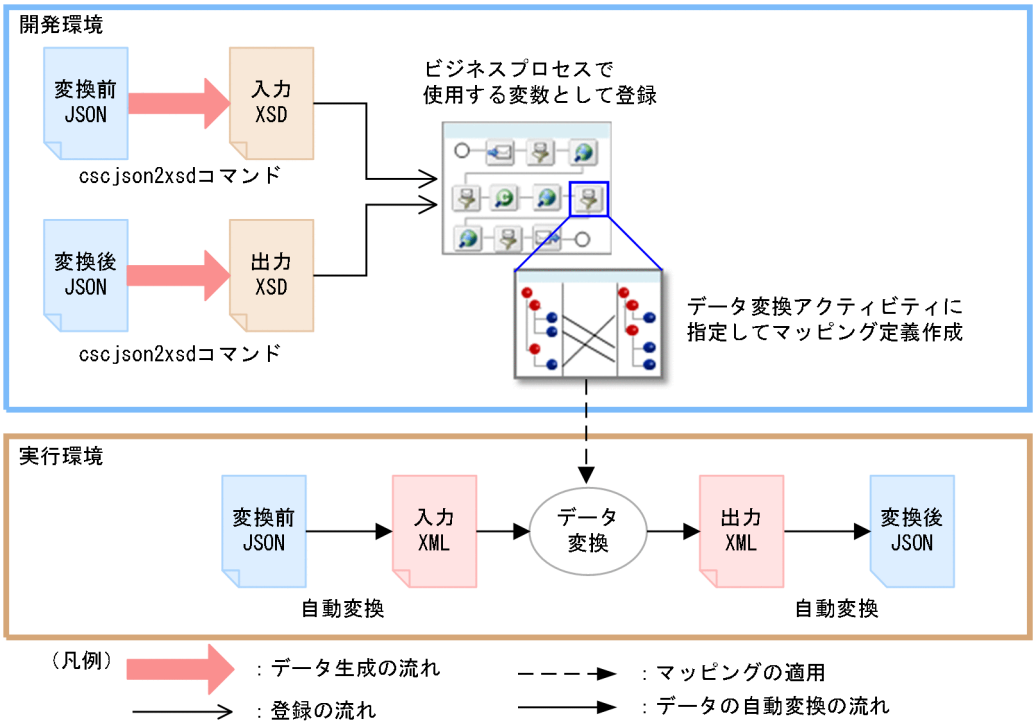
cscjson2xsd コマンドまたは cscjson2xsdadv コマンド

JSON 形式のデータをデータ変換するためには，サービスプラットフォームが自動的に変換する XML 形式のデータに対してマッピングの定義を作成する必要があります。cscjson2xsd コマンドは，JSON-XML 変換の標準モードで使用します。cscjson2xsdadv コマンドは，JSON-XML 変換の拡張モードで使用します。

これらのコマンドを利用することで，JSON 形式のデータに対応した XML スキーマファイルを生成できます。生成した XML スキーマを基にマッピング定義を作成します。cscjson2xsd コマンドまたは cscjson2xsdadv コマンドの入力となる JSON 形式のデータは事前に用意しておきます。

JSON 形式データでデータ変換する場合の，データ変換アクティビティを定義する流れを次の図に示します。

図 4-22　JSON 形式データを変換するデータ変換アクティビティの定義（cscjson2xsd コマンドの例）



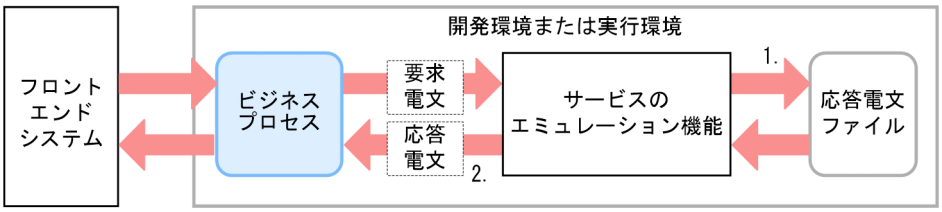
JSON 変換ツールの詳細は，マニュアル「サービスプラットフォーム 開発ガイド 基本開発編」の「14. JSON 変換ツールを使用したデータ変換」を参照してください。

4.13　サービスのエミュレーション機能の概要

サービスのエミュレーション機能は，サービス呼出アクティビティに設定したサービスアダプタの接続先（サービス部品）がなくても，ビジネスプロセスのテストができる機能です。

サービスのエミュレーション機能を使用したリクエストの流れを次の図に示します。

図 4-23　サービスのエミュレーション機能を使用したリクエストの流れ



1. エミュレーションする応答電文のデータを記載したファイル（応答電文ファイル）を読み込みます。
2. 読み込んだ内容を応答電文として返却します。

サービスのエミュレーション機能から返却された応答電文は，サービス呼出アクティビティに定義した応答電文のボディ割当変数に設定されます。サービスアダプタを呼び出すことなく，ビジネスプロセスを実行できます。

サービスのエミュレーション機能は，開発環境でも実行環境でも使用できます。使用する環境によって設定手順が異なります。ここでは，実行環境でのサービスのエミュレーション機能について説明します。開発環境でのサービスのエミュレーション機能を使用する場合は，マニュアル「サービスプラットフォーム 開発ガイド 基本開発編」の「10.2.2　サービスのエミュレーションの設定」を参照してください。

4.13.1　サービスのエミュレーション機能の設定

サービスのエミュレーション機能を使用するには，HCSC サーバランタイム定義ファイルの次のプロパティに，サービスのエミュレーション機能を使用するかどうかと，エミュレート設定情報定義ファイルのパスを設定する必要があります。

- service-emulate.enable（サービスエミュレーション機能の実行有無）
- service-emulate-def-filepath（エミュレート設定情報定義ファイルの絶対パス）

HCSC サーバランタイム定義ファイルについては，マニュアル「サービスプラットフォーム リファレンス」の「6.5.6　HCSC サーバランタイム定義ファイル」を参照してください。

また，エミュレート設定情報定義ファイルでエミュレーションに必要な情報を設定します。サービスのエミュレーションに必要な情報は，エミュレーションするサービスアダプタのオペレーション単位で設定します。エミュレーションに必要な情報を次に示します。

- エミュレーション対象のサービスアダプタ

- エミュレーション対象のサービスアダプタのオペレーション
- 応答電文の種別
- エミュレーションの実行有無
- 応答電文ファイル
- エミュレーションする条件
- 優先順位

エミュレート設定情報定義ファイルについては，マニュアル「サービスプラットフォーム リファレンス」の「6.5.7　エミュレート設定情報定義ファイル」を参照してください。

(1)　応答電文のエミュレーション

サービスのエミュレーション機能を使用すると，ユーザが作成した応答電文ファイルの内容が，サービス呼出アクティビティの応答電文のボディ割当変数に設定されます。ヘッダ割当変数はエミュレーションの対象外となります。サービス呼出アクティビティの応答電文のボディ割当変数には，種別が「XML」「non-XML」「any」の変数が設定できます。

(2)　要求電文の変換処理

応答電文ファイルに XSLT スタイルシート（ファイル拡張子「.xsl」）を指定すると，要求電文を XSLT スタイルシートで変換した結果を応答電文としてエミュレーションできます。このエミュレーションでの注意点を次に示します。

- 要求電文と応答電文の両方の変数種別が XML の場合だけ実行できます。それ以外の場合，動作は保証しません。
- 変換元（要求電文）を複数指定することはできません。
- XSLT スタイルシートには，マッピング定義エディタから生成したデータ変換定義ファイルで，かつファイル拡張子が「.xsl」のファイルを指定する必要があります。それ以外のファイルを指定した場合の動作は保証しません。

(3)　サービスのエミュレーション機能の実行条件

サービスのエミュレーション機能は，エミュレート設定情報定義ファイルの設定が，次のすべての条件に当てはまる場合だけ実行されます。

- エミュレーション機能の実行有無に有効（true）を設定している。
- エミュレーション対象のサービスアダプタがサービス呼出アクティビティに定義されているサービス名と一致する。
- エミュレーション対象のサービスアダプタのオペレーション名がサービス呼出アクティビティに定義されているオペレーション名と一致する。
- エミュレーションする条件を定義している場合，XPath 式の評価結果が真である。

実行条件に当てはまらない場合は，通常のサービス呼び出しが実行されます。

(4) 1 つのビジネスプロセス内で複数のサービスアダプタを呼び出している場合の定義例

1 つのビジネスプロセス内で複数のサービスアダプタを呼び出している場合，エミュレート設定情報定義ファイルで，emulate 要素を複数設定することでエミュレーションの定義を複数設定できます。

```
                              :
<viewState>
<emulate condition="" operation="operation1" service="SOAPAdapter" type="RESPONSE_NORMAL" va
lid="true" xmlfile="/data/telegram/test1.xml"/>
<emulate condition="" operation="exec" service="TP1Adapter" type="RESPONSE_NORMAL" valid="tr
ue" xmlfile="/data/telegram/test2.xml"/>
</viewState>
                              :
```

(5) サービスアダプタに複数のオペレーションが設定されている場合の定義例

サービスアダプタに複数のオペレーションが設定されている場合，エミュレート設定情報定義ファイルで，オペレーションごとに emulate 要素を定義することで同一のサービスアダプタの定義を複数設定できます。

```
                              :
<viewState>
<emulate condition="" operation="operation1" service="SOAPAdapter" type="RESPONSE_NORMAL" va
lid="true" xmlfile="/data/telegram/test1.xml"/>
<emulate condition="" operation="operation2" service="SOAPAdapter" type="RESPONSE_NORMAL" va
lid="true" xmlfile="/data/telegram/test2.xml"/>
</viewState>
                              :
```

4.13.2 サービスのエミュレーション機能を使用する場合の注意

サービスのエミュレーション機能を使用する場合の注意について説明します。

- サービスアダプタの通信モデルが非同期の場合，サービスのエミュレーション機能は実行できません。
- サービスのエミュレーションの実行時は，サービスアダプタを呼び出さないため，サービスアダプタで実行する機能や，ビジネスプロセスのサービス呼出アクティビティの機能の一部が使用できません。使用できない機能を次に示します。
  - サービスアダプタでデータ変換を定義している場合は，エミュレーション実行時にサービスアダプタのデータ変換は実行できません。
  - フォルト電文をエミュレーションする場合，サービスアダプタ定義画面で設定したフォルト名は設定されません。フォルト名は「invokeEmulateData」となります。
  - 応答電文のヘッダ電文は，エミュレーションされません。

- 実行環境のエミュレーションでは，サービスアダプタでコンポーネント共通 UOC は実行されません。
- HTTP アダプタで JSON-XML 変換は実行されません。
- Cookie 情報は引き継がれません。
- XML 形式の電文のエミュレーション実行時は，XML 電文内の属性値に指定した名前空間接頭辞は補完されません。
- 電文の妥当性検証は実行されません。
- リクエストトレースでは，サービスアダプタからサービス部品を呼び出したときの要求または応答の結果（取得位置：SVC）は出力されません。
- サービスアダプタの PRF トレースは出力されません。
- サービスアダプタからサービス部品を呼び出したときの要求または応答の電文トレース（取得位置：SVC）や，サービスアダプタでデータ変換を実行したときの変換前または変換後の電文トレース（CNVST）は出力されません。
- 各種サービスアダプタで出力している固有のトレースのうち，リクエスト実行時に出力しているトレースは出力されません。

- エミュレート設定情報定義ファイルや応答電文ファイルの内容を変更した場合，変更した内容は次のリクエストで反映されます。ただし，リクエスト実行中はファイルの内容は変更できません。
- 開発環境のビジネスプロセスのデバッグ機能で応答電文をエミュレーションした場合，エミュレート設定情報定義ファイルの設定は有効になりません。

5 性能向上のための機能

この章では，性能向上のための機能として，サービス部品呼び出し処理の多重度，XML 電文のキャッシュ機能，ビジネスプロセスの事前キャッシュ機能，データ変換処理の XML パーサプール機能，およびデータ変換のキャッシュ機能について説明します。

5.1　サービス部品呼び出し処理の多重度

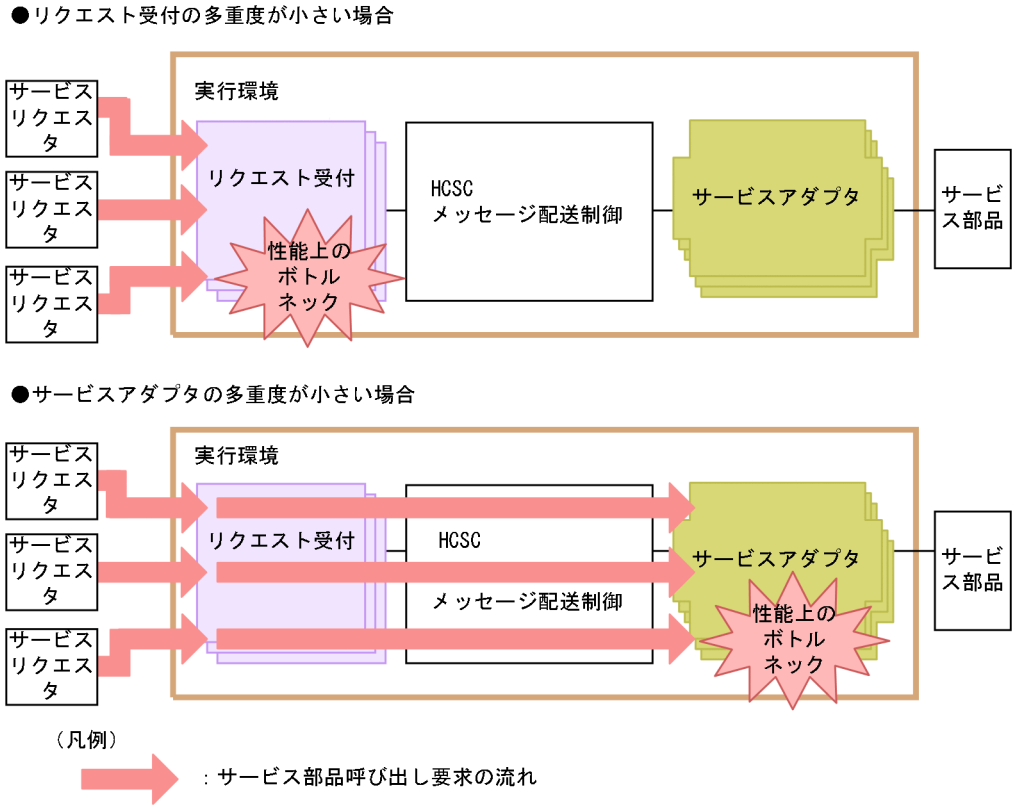
サービスリクエスタからのサービス部品の呼び出し要求は，同時に実行される場合があります。そのため，サービス部品呼び出し処理の多重度（同時に実行できる数）を設定する必要があります。

5.1.1　HCSC サーバ内の多重度

多重度の設定が良くない場合，性能上のボトルネックになるおそれがあります。

多重度の設定によってボトルネックになる例を次の図に示します。

図 5-1　多重度の設定によってボトルネックになる例



「図 5-1　多重度の設定によってボトルネックになる例」に示すように，リクエスト受付（標準受付またはユーザ定義受付）の多重度が小さいと，一度にたくさん要求が来ても同時に処理ができないため，一度に実行できる処理が少なくなります。また，リクエスト受付の多重度を大きくしても，サービスアダプタで定義している多重度が小さいと，サービスアダプタで一度に実行できる処理が少ないため，サービスアダプタがボトルネックになり，意図した性能を得られなくなります。

(1)　HCSC サーバでの多重度

多重度は，同期（Web サービス）の標準受付の場合は，スレッド数および最大同時実行数で設定します。同期（SessionBean）の標準受付の場合は，インスタンス数で設定します。HCSC サーバ内の多重度の設

定について，「表 5-1　標準受付の多重度を設定するプロパティ」，「表 5-2　ユーザ定義受付の多重度を設定するプロパティ」，および「表 5-3　サービスアダプタの多重度を設定する項目」に示します。

表 5-1　標準受付の多重度を設定するプロパティ

定義ファイル	プロパティ	内容
HCSC サーバランタイム定義ファイル	request-ejb.instance.minimum	同期（SessionBean）の標準受付のインスタンス最小数
	request-ejb.instance.maximum	同期（SessionBean）の標準受付のインスタンス最大数
	request-ejb.parallel.count	CTM がアプリケーションを呼び出すために用意するスレッド数
	request-soap.instance.minimum	同期（Web サービス・SOAP1.1）の標準受付の最小同時実行数
	request-soap.instance.maximum	同期（Web サービス・SOAP1.1）の標準受付の最大同時実行数
	request-soap1_2.instance.minimum	同期（Web サービス・SOAP1.2）の標準受付の最小同時実行数
	request-soap1_2.instance.maximum	同期（Web サービス・SOAP1.2）の標準受付の最大同時実行数
	request-jms.instance.maximum	非同期（MDB（WS-R））の標準受付のインスタンス最大数

表 5-2　ユーザ定義受付の多重度を設定するプロパティ

定義ファイル	プロパティ	内容
ユーザ定義受付ランタイム定義ファイル	user-defined-reception-soap.threads.maximum	最大同時実行数

表 5-3　サービスアダプタの多重度を設定する項目

画面	内容
開発環境でのサービスアダプタ定義画面	最大インスタンス数

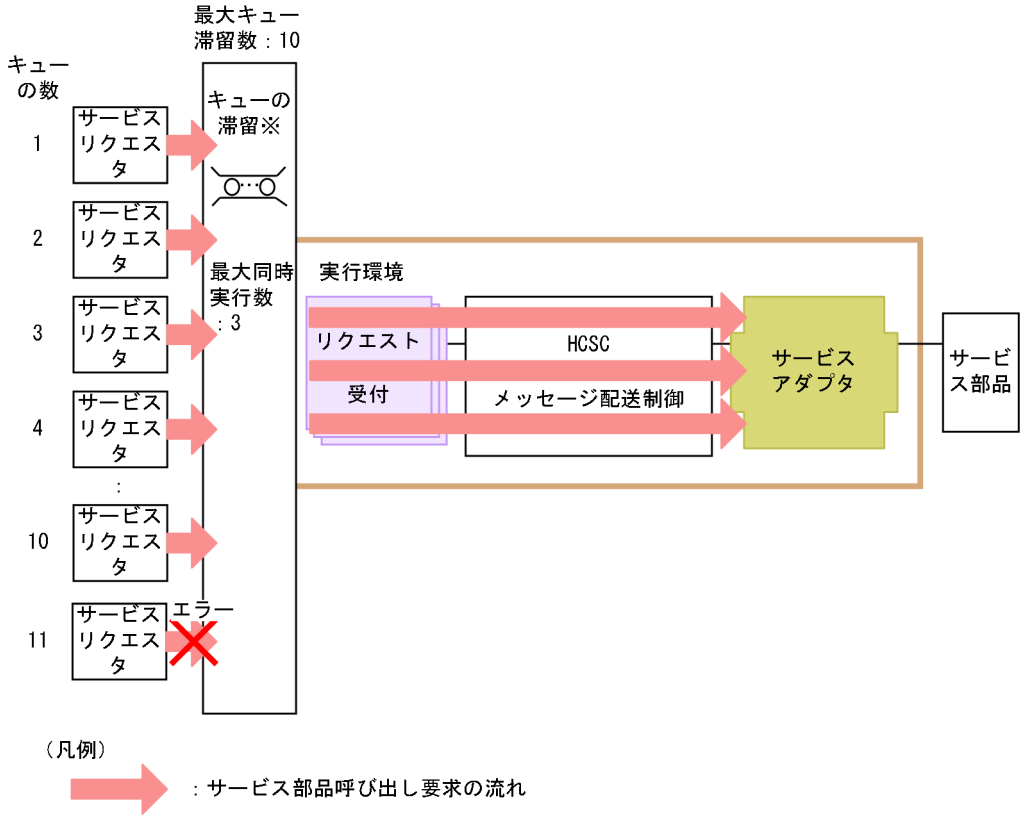
注

最大インスタンス数に 0 を指定した場合は，上限が設定されないで無制限になります。

(2) Web サービス（SOAP 通信）での多重度

Web サービス（SOAP 通信）の場合，同時実行数のほかに Web コンテナレベルでの最大キュー滞留数を設定できます。最大同時実行数と最大キュー滞留数を設定しておくと，サービスリクエスタからの要求を受けたときに，実際の処理をしないで処理を待たせておけるため，流量制御ができます。サービスリクエスタからの要求と最大キュー滞留数の関係を次の図に示します。

図 5-2　サービスリクエスタからの要求と最大キュー滞留数の関係



注※　Webサービスの実行待ちキューの滞留を示します。

例えば，「図 5-2　サービスリクエスタからの要求と最大キュー滞留数の関係」に示すように，最大同時実行数を 3 とした場合，最大キュー滞留数を 10 にしておくと，同時にきたサービス部品呼び出し要求のうち，3 件までは同時に処理されます。同時実行数より多くのサービス部品呼び出し要求（4 件以上）がきた場合は，最大キュー滞留数（10 件）までは，処理待ちの状態となり，先に実行されている処理が終わり次第，順次処理が実行されます。

最大キュー滞留数の数が上限（10 件）まで達している場合に，さらにサービスリクエスタからサービス部品呼び出し要求がきた場合は，要求は受け付けられないため，エラーとしてサービスリクエスタに応答されます。

HCSC サーバ内の最大キュー滞留数は，「表 5-4　標準受付の最大キュー滞留数を設定するプロパティ」および「表 5-5　ユーザ定義受付の最大キュー滞留数を設定するプロパティ」に示す定義ファイルのプロパティで設定します。

表 5-4　標準受付の最大キュー滞留数を設定するプロパティ

定義ファイル	プロパティ	内容
HCSC サーバランタイム定義ファイル	request-soap.queue-size	同期（Web サービス・SOAP1.1）の標準受付の実行待ちキューのサイズ
	request-soap1_2.queue-size	同期（Web サービス・SOAP1.2）の標準受付の実行待ちキューのサイズ

表 5-5　ユーザ定義受付の最大キュー滞留数を設定するプロパティ

定義ファイル	プロパティ	内容
ユーザ定義受付ランタイム定義ファイル	user-defined-reception-soap.queue-size	実行待ちキューサイズ

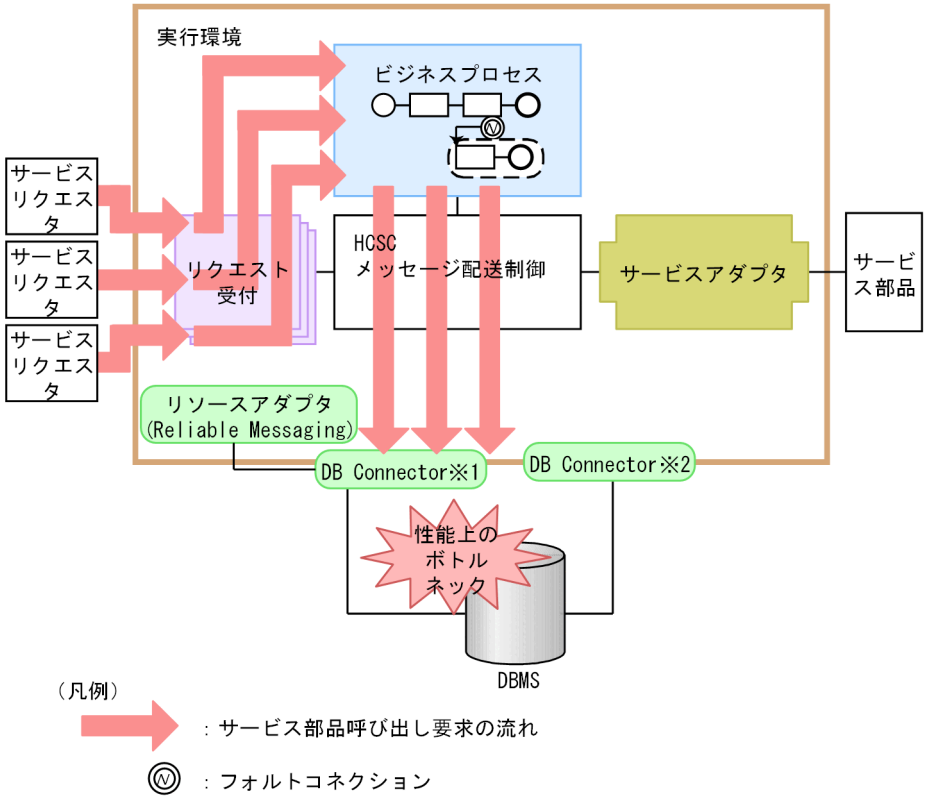
5.1.2　データベースへのアクセスに関する多重度

永続化するビジネスプロセスを使用する（プロセスインスタンスの実行履歴を採取する）場合，および非同期プロトコルを使用する場合は，HCSC サーバに設定する DB Connector の定義で，コネクションプール数を設定します。

(1)　永続化するビジネスプロセスを使用する場合

HCSC サーバに設定した 2 つの DB Connector のうち，HCSC サーバセットアップ定義ファイルの dbcon-xadisplayname プロパティに指定した DB Connector を使用します。リクエスト受付からビジネスプロセスを経由して，一度に多くの要求が来ても，DB Connector の部分でデータベースへの出力が一度にできなくなります。すると，性能上のボトルネックが発生して，その結果，ビジネスプロセスの同時実行処理性能が遅くなります。そのため，プロセスインスタンスの実行履歴を採取するときの多重度を設定する必要があります。永続化するビジネスプロセスを使用する場合のボトルネックを次の図に示します。

図 5-3　永続化するビジネスプロセスを使用する場合のボトルネック



注※1　dbcon-xadisplaynameプロパティに指定したLocalTransactionまたはXATransactionのDB Connectorです。

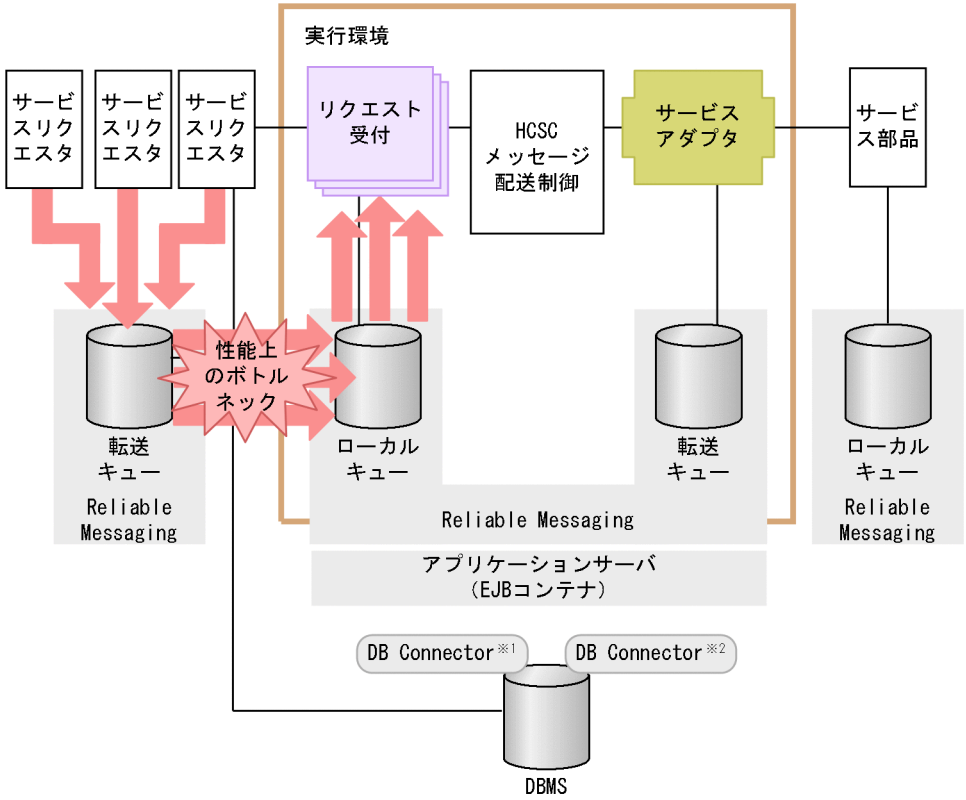
注※2　dbcon-noisplaynameプロパティに指定したNoTransactionまたはXATransactionのDB Connectorです。

(2)　非同期プロトコルを使用する場合

HCSC サーバに設定した 2 つの DB Connector のうち，HCSC サーバセットアップ定義ファイルの dbcon-xadisplayname プロパティに指定した DB Connector は，非同期（MDB（WS-R）または MDB（DB キュー））の標準受付でキューからメッセージを取り出したり，非同期（MDB（WS-R）または MDB（DB キュー））のサービスアダプタでメッセージを送信したりするときにも使用します。そのため，プロセスインスタンスの実行履歴を採取するときの多重度に加えて，HCSC サーバがリクエスト受付のキューからメッセージを取り出すときや，サービスアダプタが転送キューにメッセージを送信するときの多重度も含めて，コネクションプール数を検討する必要があります。

キューに対して一度に多くの要求が来ても，Reliable Messaging のリソースアダプタが同時にメッセージの取り出し処理ができないで，性能上のボトルネックが発生します。また，サービスアダプタの多重度を大きくしていても，サービスアダプタがキューにメッセージを送信するときの同時実行数が少ないと，やはり性能上のボトルネックが発生します。非同期プロトコルを使用する場合のボトルネックを次の図に示します。

図 5-4　非同期プロトコルを使用する場合のボトルネック



注※1　LocalTransactionまたはXATransaction用のDB Connectorです。
注※2　NoTransaction用のDB Connectorです。

(3)　データベースへのアクセスに関する多重度の設定

データベースへのアクセスに関する多重度は，次の表に示すように設定します。

表 5-6　DB Connector の多重度を設定するプロパティ

定義ファイル	プロパティ	内容
DB Connector（LocalTransaction または XATransaction）のセットアップ時に設定する属性ファイル	MinPoolSize	プールの最小値
	MaxPoolSize	プールの最大値

注 1

プールの最大値を増やした場合，データベース側の同時接続数も変更する必要があるため，注意が必要です（HiRDB の場合，pd_max_users（HiRDB の最大接続数）となります）。

注 2

実行環境が使用する DB コネクション数の最大値については，マニュアル「サービスプラットフォーム システム構築・運用ガイド」の「3.1.2　実行環境に必要なソフトウェアを設定する」を参照してください。

表 5-7　Reliable Messaging の多重度を設定するプロパティ

定義ファイル	プロパティ	内容
Reliable Messaging のセットアップ時に設定する属性ファイル	MinPoolSize	プールの最小値
	MaxPoolSize	プールの最大値

5.1.3　Web サービス（SOAP 通信）に関する多重度

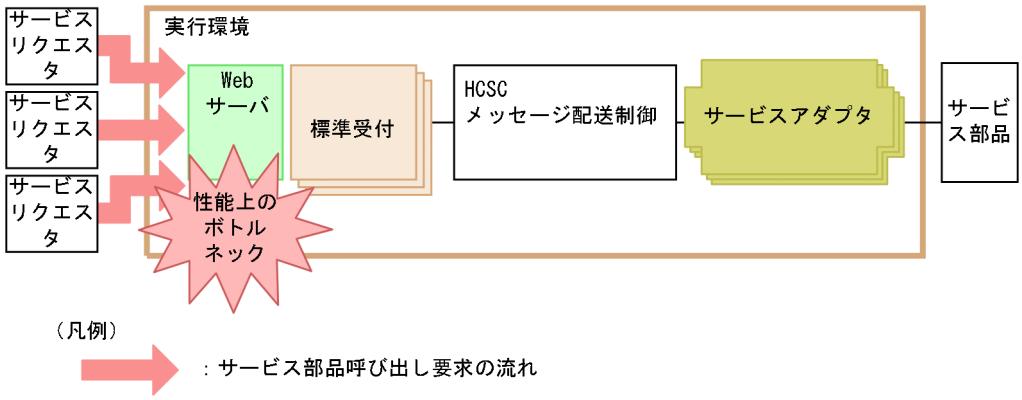
(1)　同時実行スレッド数

Web サービス（SOAP 通信）を使用する場合は，実行環境で設定している Web サーバ（HTTP サーバ）の同時実行スレッド数も関係します。

標準受付（Web サービス）および SOAP 受付が複数ある場合，各受付に設定した同時実行スレッド数の合計より，Web サーバに設定した全体の同時実行スレッド数が優先されます。そのため，Web サーバに設定した全体の同時実行スレッド数が，各受付に設定した同時実行スレッド数の合計より小さいと，一度に多くの要求が来ても同時に処理ができないため，性能上のボトルネックになることがあります。

Web サービス（SOAP 通信）を使用する場合のボトルネックを次の図に示します。

図 5-5　Web サービス（SOAP 通信）を使用する場合のボトルネック



Web サービス（SOAP 通信）に関する多重度は，次のように設定します。

- HTTP Server の場合

  コンフィグファイルに指定できるディレクティブのうち，Web サーバの性能についての定義に関するディレクティブを定義します。

  詳細については，マニュアル「HTTP Server」の「6.1.1　ディレクティブ一覧」を参照してください。

- インプロセス HTTP サーバ機能または NIO HTTP サーバ機能の場合

  運用管理ポータルの[論理サーバの環境設定]－[J2EE サーバ]－[J2EE サーバの論理サーバ名]－[HTTP サーバ]で次の内容を設定します。

  - 同時実行スレッド数

- 初期スレッド数

インプロセス HTTP サーバは，J2EE サーバの互換モードが V9 互換モードの場合，使用できます。推奨モードの場合は，NIO HTTP サーバを使用してください。

インプロセス HTTP サーバについては，マニュアル「アプリケーションサーバ 機能解説 互換編」の「6. インプロセス HTTP サーバ」を参照してください。

NIO HTTP サーバを使用する場合のシステム構築の詳細については，マニュアル「アプリケーションサーバ 機能解説 基本・開発編(Web コンテナ)」の「7. NIO HTTP サーバ」を参照してください。

(2) 占有スレッド数

Web サーバ（HTTP サーバ）の同時実行スレッド数が十分な数を確保できない状況であっても，個々のリクエスト受付で必ず最低限の実行をする場合は，占有スレッド数を定義します。

占有スレッド数を定義した場合は，Web サーバ（HTTP サーバ）の同時実行スレッド数以上の要求が個別のリクエスト受付に来た場合でも，必ず指定した数だけは同時に実行されます。

ほかのリクエスト受付に要求が来ていない場合は，最大同時実行数まで同時に実行できます。

HCSC サーバ内の占有スレッド数は，「表 5-8 標準受付の占有スレッド数を設定するプロパティ」および「表 5-9 ユーザ定義受付の占有スレッド数を設定するプロパティ」に示すように設定します。

表 5-8 標準受付の占有スレッド数を設定するプロパティ

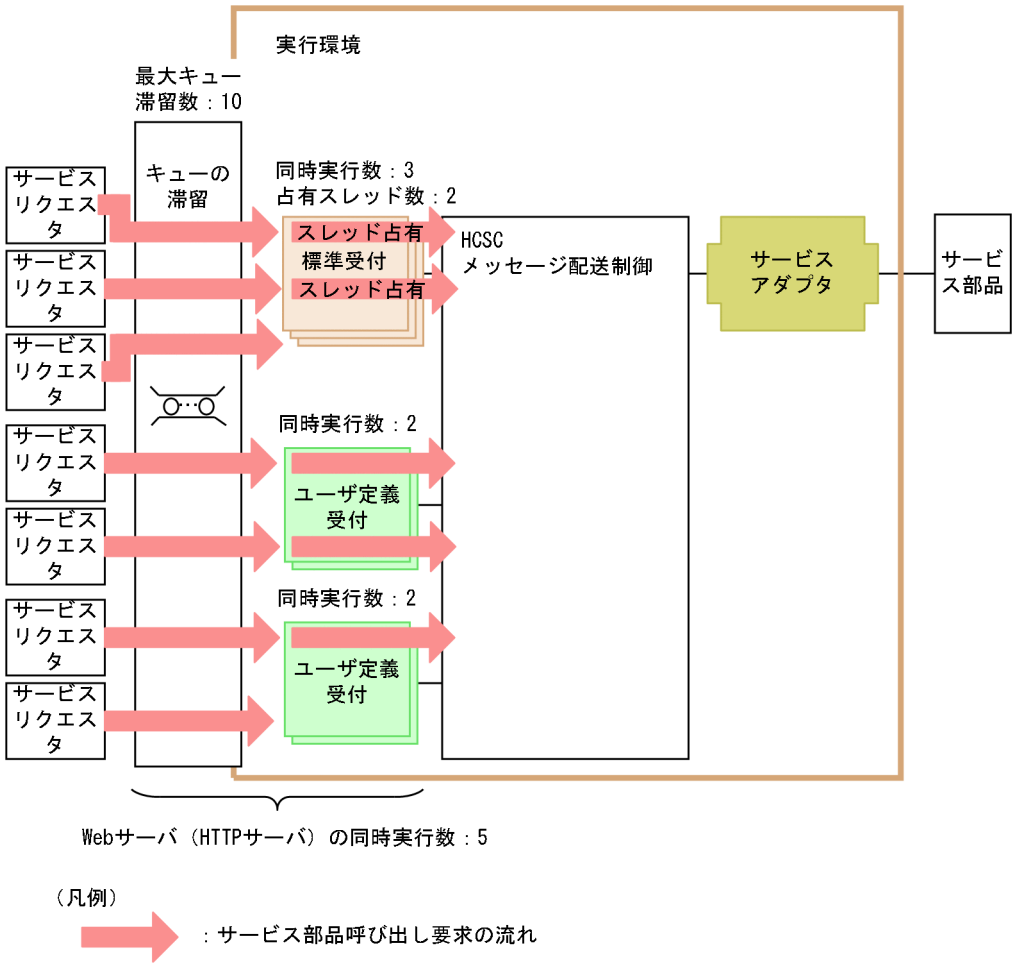
定義ファイル	プロパティ	内容
HCSC サーバランタイム定義ファイル	request-soap.exclusive.threads	同期（Web サービス・SOAP1.1）の標準受付の占有スレッド数
	request-soap1_2.exclusive.threads	同期（Web サービス・SOAP1.2）の標準受付の占有スレッド数

表 5-9 ユーザ定義受付の占有スレッド数を設定するプロパティ

定義ファイル	プロパティ	内容
ユーザ定義受付ランタイム定義ファイル	user-defined-reception-soap.exclusive.threads	占有スレッド数

最大同時実行数と占有スレッド数の関係を次の図に示します。

図 5-6　最大同時実行数と占有スレッド数の関係



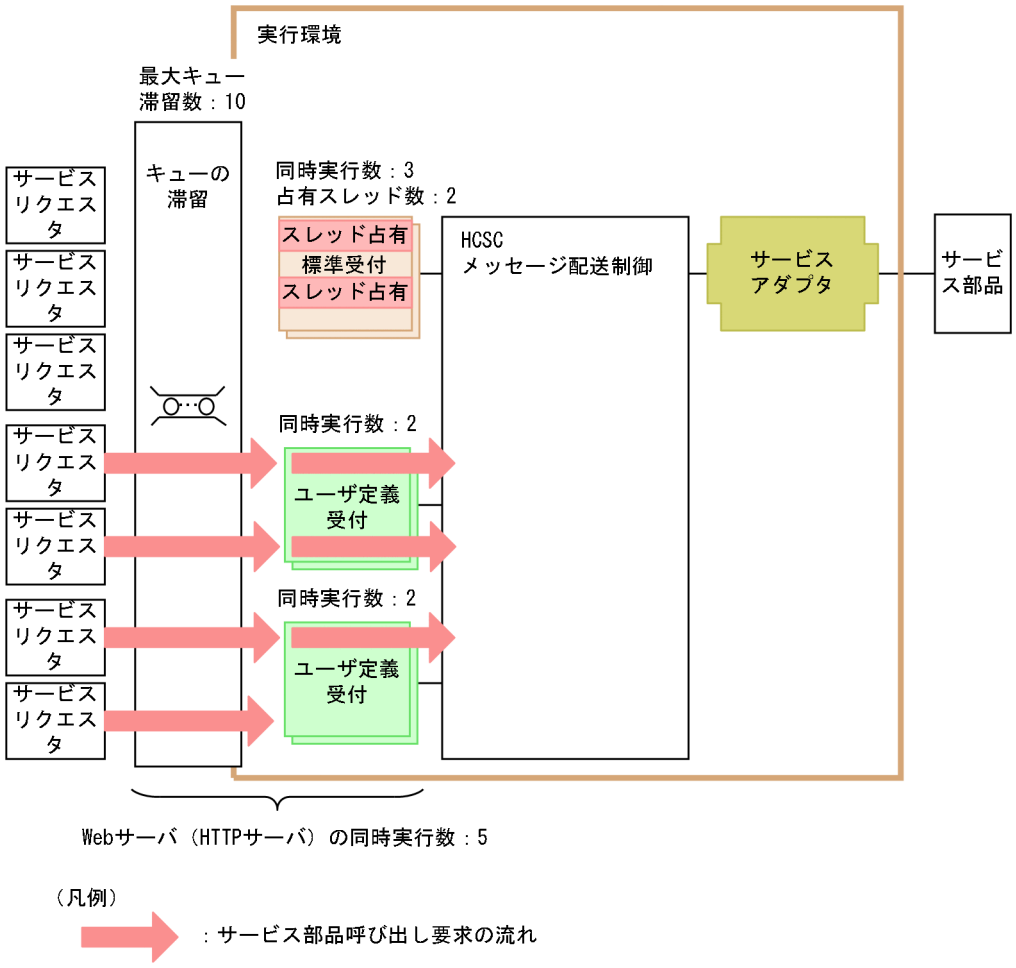
「図 5-6　最大同時実行数と占有スレッド数の関係」に示すように，標準受付，2 つのユーザ定義受付，および Web サーバ（HTTP サーバ）を次のように設定した場合，標準受付は必ず 2 つのスレッドを占有します。

- 標準受付：最大キュー滞留数 10，同時実行数 3，占有スレッド数 2
- ユーザ定義受付：同時実行数 2
- Web サーバ（HTTP サーバ）：同時実行数 5

そのため，Web サーバ（HTTP サーバ）の同時実行数に指定した 5 件以上の要求が来ると，2 つのスレッドは標準受付の実行のために使用され，残りの 3 つのスレッドを使ってユーザ定義受付が動作します。

ただし，最大キュー滞留数，標準受付の同時実行数，占有スレッド数，および Web サーバ（HTTP サーバ）の同時実行数が「図 5-6　最大同時実行数と占有スレッド数の関係」と同じ場合で，「図 5-7　占有スレッド数を指定した受付に要求が来ない場合」に示すように，占有スレッド数を指定した標準受付に要求が来ていないときでも，必ず 2 つはスレッドを占有して実行する状態になります。また，ほかのユーザ定義受付では，それぞれ同時実行数が 2 件で合計の同時実行数は 4 件になるはずですが，Web サーバ（HTTP サーバ）の同時実行数 5 件であるため，残り 3 つしか処理できなくなります。ほかのリクエスト受付の同時実行数が増えることはありません。

図 5-7　占有スレッド数を指定した受付に要求が来ない場合



5.1.4　XML 解析処理に関する多重度

要求電文および応答電文の処理に使用するクラスは複数のスレッドで共有できないため，複数のスレッドで同時使用する場合は，スレッドごとにインスタンスを生成する必要があります。

インスタンスの生成処理は時間を要するため，解析ごとに生成すると性能が劣化します。

インスタンスプール機能を使用すると，メモリ上に作成されたインスタンスプールに生成したインスタンスを貯めておくことができます。インスタンスプールへ貯めたインスタンスは，システム全体で共有し，再利用できるため，処理の効率向上が図れます。

インスタンスプールに生成したインスタンスを貯めておくには，次の表に示すように設定します。

表 5-10　インスタンスプールにキャッシュするインスタンス数を設定するプロパティ

定義ファイル	プロパティ	内容
HCSC サーバランタイム定義ファイル	xml-pooled-instance-minimum	要求電文および応答電文の処理に使用するクラスのインスタンスをキャッシュする最小数

定義ファイル	プロパティ	内容
HCSC サーバランタイム定義ファイル	xml-pooled-instance-maximum	要求電文および応答電文の処理に使用するクラスのインスタンスをキャッシュする最大数

作成済みのインスタンスを再利用する場合は，キャッシュする最小数を定義します。キャッシュする最小数を定義しておくと，HCSC サーバの起動時にプールが初期化され，定義した値の分のインスタンスを生成してプールに設定します。

キャッシュする最大数を設定しておくと，定義した値に達するまでインスタンスを貯めることができます。再利用によってインスタンスが減少しても，インスタンスを生成してプールに設定するため，枯渇することはありません。

xml-pooled-instance-minimum プロパティおよび xml-pooled-instance-maximum プロパティの詳細は，マニュアル「サービスプラットフォーム リファレンス」の「6.5.6　HCSC サーバランタイム定義ファイル」を参照してください。

5.2　XML 電文のキャッシュ機能

XML 電文のキャッシュ機能の概要と設定について，説明します。

5.2.1　XML 電文のキャッシュ機能の概要

XML 電文のキャッシュ機能を使用すると，次に示す条件が同一の XML 電文が流れる場合に，処理速度が向上することがあります。

- 要素の並び順が同じ
- 属性の並び順が同じ
- 要素の繰り返しが同じ

これは，一度解析した XML 文書の特徴を保持しているため，類似文書の場合にパース処理速度の向上が図れます。

5.2.2　XML 電文のキャッシュ機能の設定

XML 電文のキャッシュ機能を使用するには，HCSC サーバランタイム定義ファイルの xmltelegram-maxcache-num プロパティで値を設定します。

xmltelegram-maxcache-num プロパティの詳細は，マニュアル「サービスプラットフォーム リファレンス」の「6.5.6　HCSC サーバランタイム定義ファイル」を参照してください。

5.3　ビジネスプロセスの事前キャッシュ機能

事前キャッシュ機能を使用すると，ビジネスプロセスでリクエストを処理する前に，フォーマット定義および変換定義の解析と HCSC サーバ上への解析結果のキャッシュができます。キャッシュした定義は，HCSC サーバまたはコンポーネントを停止するまで保持されるため，リクエスト処理速度の向上が図れます。

事前キャッシュ機能を利用するには，cscprecache コマンドを使用します。cscprecache コマンドの詳細については，マニュアル「サービスプラットフォーム リファレンス」の「cscprecache（フォーマット定義およびデータ変換定義の事前キャッシュ)」を参照してください。

5.4　データ変換処理の XML パーサプール機能

データ変換の処理では，次に示す Java の XML パーサによって XML を解析しています。

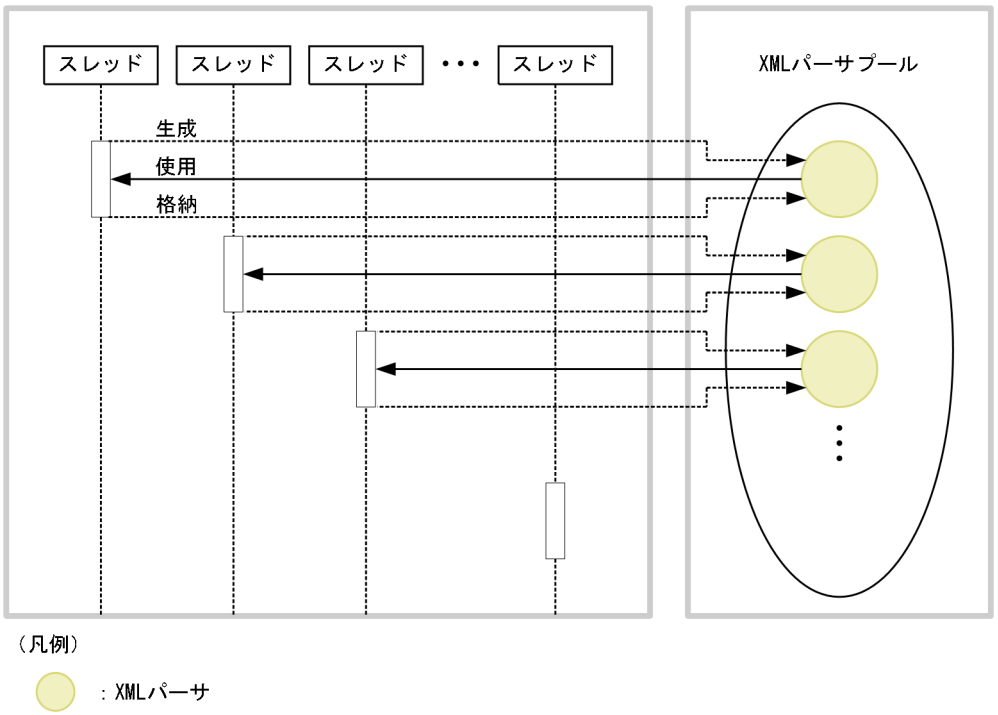
- DOM パーサ（javax.xml.DocumentBuilder クラス）
- SAX パーサ（org.xml.sax.XMLReader クラス）

これらの XML パーサのインスタンスは複数のスレッドで共有できないため，複数のスレッドで同時使用する場合は，スレッドごとに XML パーサを生成しています。XML パーサの生成処理は時間を要するため，解析ごとに生成すると性能が劣化します。

XML パーサプール機能を使用すると，メモリ上に作成された XML パーサプールに生成した XML パーサを貯めておくことができます。XML パーサプールへ貯めた XML パーサは，システム全体で共有し，再利用できるため，処理の効率向上が図れます。

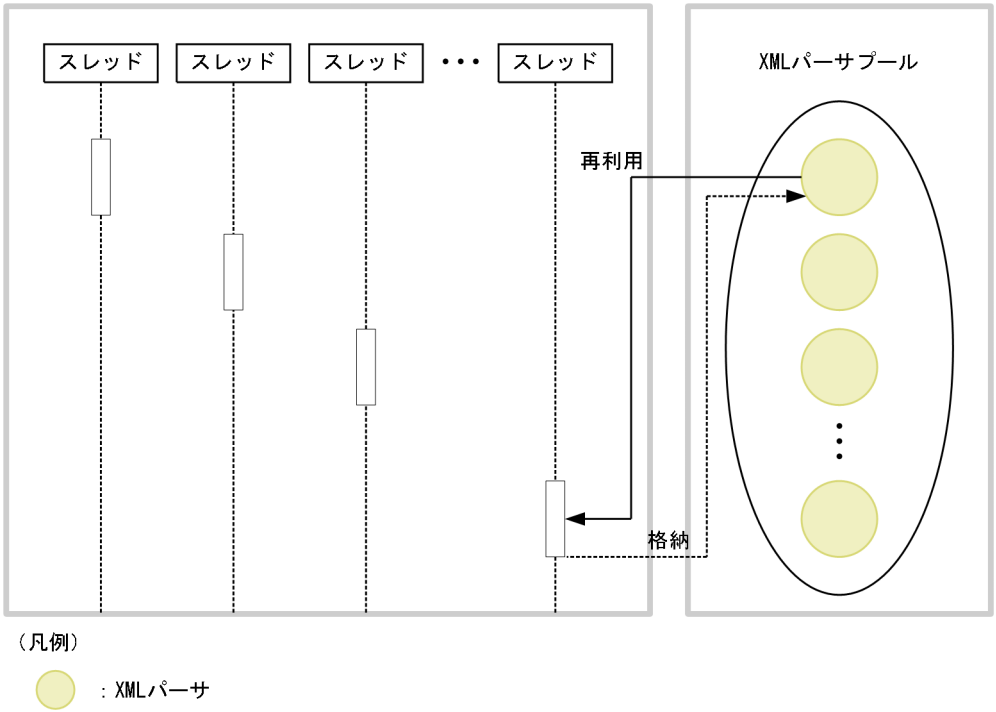
生成した XML パーサをメモリ上に作成された XML パーサプールに貯めておく処理の流れを次の図に示します。

図 5-8　生成した XML パーサをメモリ上に作成された XML パーサプールに貯めておく処理の流れ



XML パーサプールへ貯めた XML パーサを再利用する処理を次の図に示します。

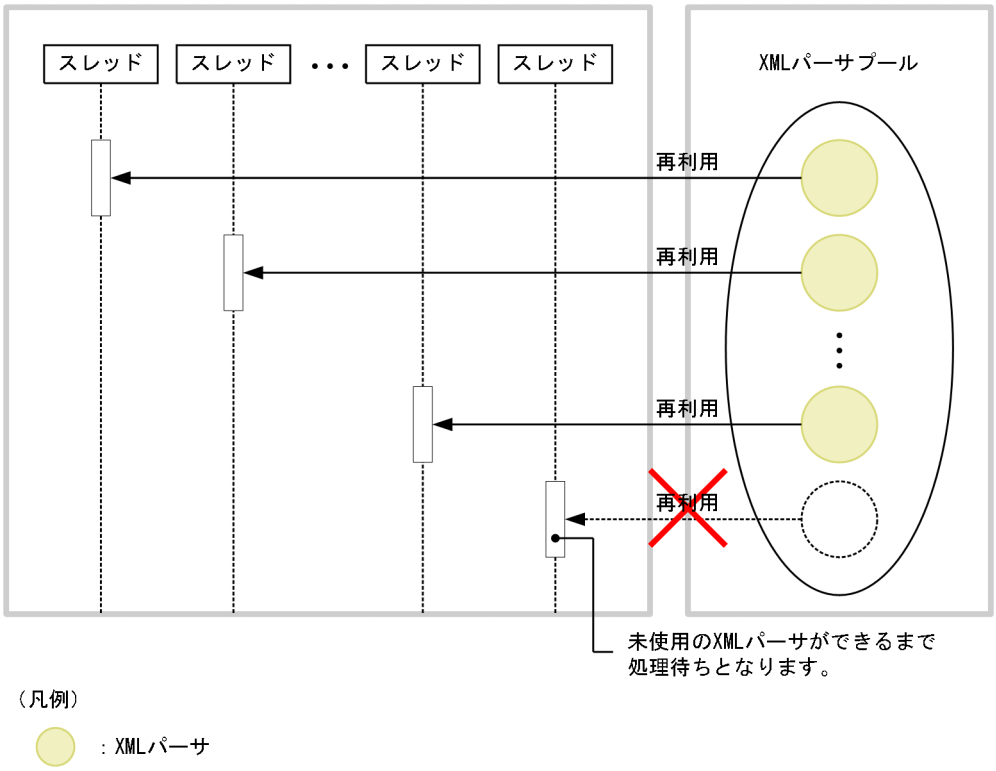
図 5-9　XML パーサプールへ貯めた XML パーサを再利用する処理



XML パーサプールに貯めることができる XML パーサの最大数は，オプションで指定した最大プール数の 2,147,483,647 です。XML パーサを同時使用するスレッド数が最大プール数を超える場合は，他のスレッドでの使用が終わるまで処理を停止し，未使用となり次第処理を続行します。

XML パーサを同時使用するスレッド数が最大プール数を超えた場合の処理を次の図に示します。

図 5-10　XML パーサを同時使用するスレッド数が最大プール数を超えた場合の処理



XML パーサプール機能を使用するための設定方法については，マニュアル「サービスプラットフォーム システム構築・運用ガイド」の「2.3.9　J2EE サーバ用ユーザプロパティファイル（usrconf.properties）を設定する」の XML パーサプール機能オプションに関する記述を参照してください。

5.5　データ変換のキャッシュ機能

コマンドやリクエストの処理によって，データ変換基盤にフォーマット定義とデータ変換定義がキャッシュとして登録されます。登録されたキャッシュは，コマンドの実行によって削除されます。また，キャッシュ機能を利用し，オブジェクトを再利用することもできます。

登録されているキャッシュ数を確認したい場合は，cscresinfo コマンドで取得します。cscresinfo コマンドの詳細については，マニュアル「サービスプラットフォーム リファレンス」の「cscresinfo（リソース稼働情報の表示)」を参照してください。

5.5.1　キャッシュの登録および削除のタイミング

HCSC コンポーネントごとのキャッシュの登録および削除のタイミングを次に示します。

表 5-11　キャッシュの登録および削除のタイミング

キャッシュの動作	HCSC コンポーネント		
	ビジネスプロセス	ユーザ定義受付	サービスアダプタ
キャッシュの登録	次の場合に，キャッシュが登録されます。 • ビジネスプロセスへの初回リクエスト※ • cscprecache コマンドの事前キャッシュ処理	ユーザ定義受付への初回リクエスト※	サービスアダプタへの初回リクエスト※
キャッシュの削除	csccompostop コマンドでの，ビジネスプロセスの停止処理中	csccompostop コマンドでの，ユーザ定義受付の停止処理中	cscsvstop コマンドでの，HCSC サーバ停止処理中

注※

初回リクエスト時に，表 5-12 のキャッシュ契機に記載されている動作が行われた場合にキャッシュが登録されます。ただし，cscprecache コマンド実行時は除きます。

5.5.2　キャッシュ機能を利用したオブジェクトの再利用

サービスプラットフォームではデータ変換などの生成に時間が掛かるオブジェクトを扱う場合，キャッシュ機能を利用し，オブジェクトを再利用しています。各オブジェクトの再利用とキャッシュ数の調整方法を次に示します。なお，アダプタ，ビジネスプロセスを再起動した場合は，新しいオブジェクトが生成されるため，以前のオブジェクトは再利用されません。

表 5-12　オブジェクトの再利用とキャッシュ数の調整方法

機能	キャッシュ契機	キャッシュ再利用契機	キャッシュ数調整方法
データ変換定義オブジェクトのキャッシュ	• cscprecache コマンド実行時 • データ変換時（XML・バイナリ）	データ変換時（XML・バイナリ）	HCSC サーバランタイム定義ファイルの transformdef-maxcache-num プロパティの値を調整します。
フォーマット定義オブジェクトのキャッシュ	• cscprecache コマンド実行時 • XML→バイナリ変換後※1 • バイナリ→XML 変換前※1 • バイナリ→バイナリ変換前後 • ビジネスプロセスのフォルト処理時 • バリデーション時※2	• XML→バイナリ変換後 • バイナリ→XML 変換前 • バイナリ→バイナリ変換前後 • ビジネスプロセスのフォルト処理時 • バリデーション時	HCSC サーバランタイム定義ファイルの formatdef-maxcache-num プロパティの値を調整します。
XML 文書解析オブジェクトのキャッシュ	• XML 変換前後 • ビジネスプロセスの XPath 操作時	• XML 変換前後 • ビジネスプロセスの XPath 操作時	HCSC サーバランタイム定義ファイルの xmltelegram-maxcache-num プロパティの値を調整します。

注※1

XML スキーマはキャッシュされません。

注※2

次の機能の実行時を示します。

・ビジネスプロセスの検証アクティビティ実行時

・データ検証機能実行時

5.5.3　サービスアダプタでのキャッシュの注意事項

サービスアダプタで電文のデータ変換を行う場合，フォーマット定義とデータ変換定義のオブジェクトが生成されます。しかし，オブジェクト名の一部にデプロイの時刻が設定されているため，サービスアダプタを再デプロイした場合に前回のオブジェクトが参照されずに，新しいオブジェクトが生成されます。

また，サービスアダプタは csccompostop コマンドでキャッシュが削除されないため，サービスアダプタのデプロイ・アンデプロイ，およびキャッシュの登録を繰り返すと，利用しないキャッシュが増え続けます。

不要なキャッシュを削除するため，cscsvstop コマンドを実行し，HCSC サーバを停止後，再デプロイを実施してください。

6 障害調査のための機能

この章では，サービスプラットフォームで発生した障害を調査する機能について説明します。

6.1　障害情報の取得・障害回復・障害解析機能

6.1.1　障害情報の取得機能

サービスプラットフォームの運用環境には，運用・実行環境での障害の発生に備えて，トレース情報を保存する機能があります。トレース情報は，次に示すファイルに格納されます。

- メッセージログファイル
- リクエストトレースファイル
- 性能解析トレースファイル
- 保守情報（例外ログ，メンテナンスログ，メソッドトレースなど）

これらのトレース情報を確認することで，障害の発生時刻，発生場所，原因などを把握できます。

また，サービスプラットフォームの開発環境および運用環境の画面では，画面操作時に自動的に実行されるチェックによって出力されたメッセージは，コンソールビューに表示されます。コンソールビューに表示されるメッセージの内容を確認し，各種定義や設定を変更してから再度操作できます。

6.1.2　障害回復機能

次に示す運用時に発生した障害についても，コマンドで情報の確認，ファイルのバックアップ，環境の復旧などができます。

- システムのセットアップまたはアンセットアップ時
- オペレーション実行時
- サービス部品呼び出し実行時

6.1.3　障害解析機能

サービスプラットフォームでは，性能解析トレースファイルを利用して障害解析を補助する**メッセージトラッキング機能**を提供しています。メッセージトラッキングとは，障害発生個所を特定し，障害の切り分けや障害調査を補助する障害解析ツールです。メッセージトラッキングを起動すると，専用の対話型プロンプトが表示されます。表示された対話型プロンプトでコマンドを実行し，性能解析トレースファイルを解析します。

メッセージトラッキング機能は，障害発生時の性能解析トレースファイルがあれば，Service Architect，または Service Platform がインストールされているマシンならどこでも利用できます。そのため，障害が発生した HCSC サーバ稼働マシンだけでなく，障害が発生した HCSC サーバとは別のリモート環境でも障害を解析できます。

6.2　データ検証機能

6.2.1　データ検証機能の概要

サービスアダプタでは，要求電文や応答電文が，定義したフォーマット定義に対して妥当かどうかを検証できます。また，データ変換機能を利用している場合は，変換前後の電文が，定義したフォーマット定義に対して妥当かどうかを検証できます。

サービスアダプタに指定したフォーマット定義の形式と，要求時および応答時に送られてくる電文の形式が妥当であるかどうか検証し，電文がフォーマットの形式に対して妥当でない場合は，サービスを実行する前にエラーを返します。

テスト環境などで，サービスを実行する前に妥当な電文かどうかを検証できるため，サービスの実行の失敗を事前に回避できます。

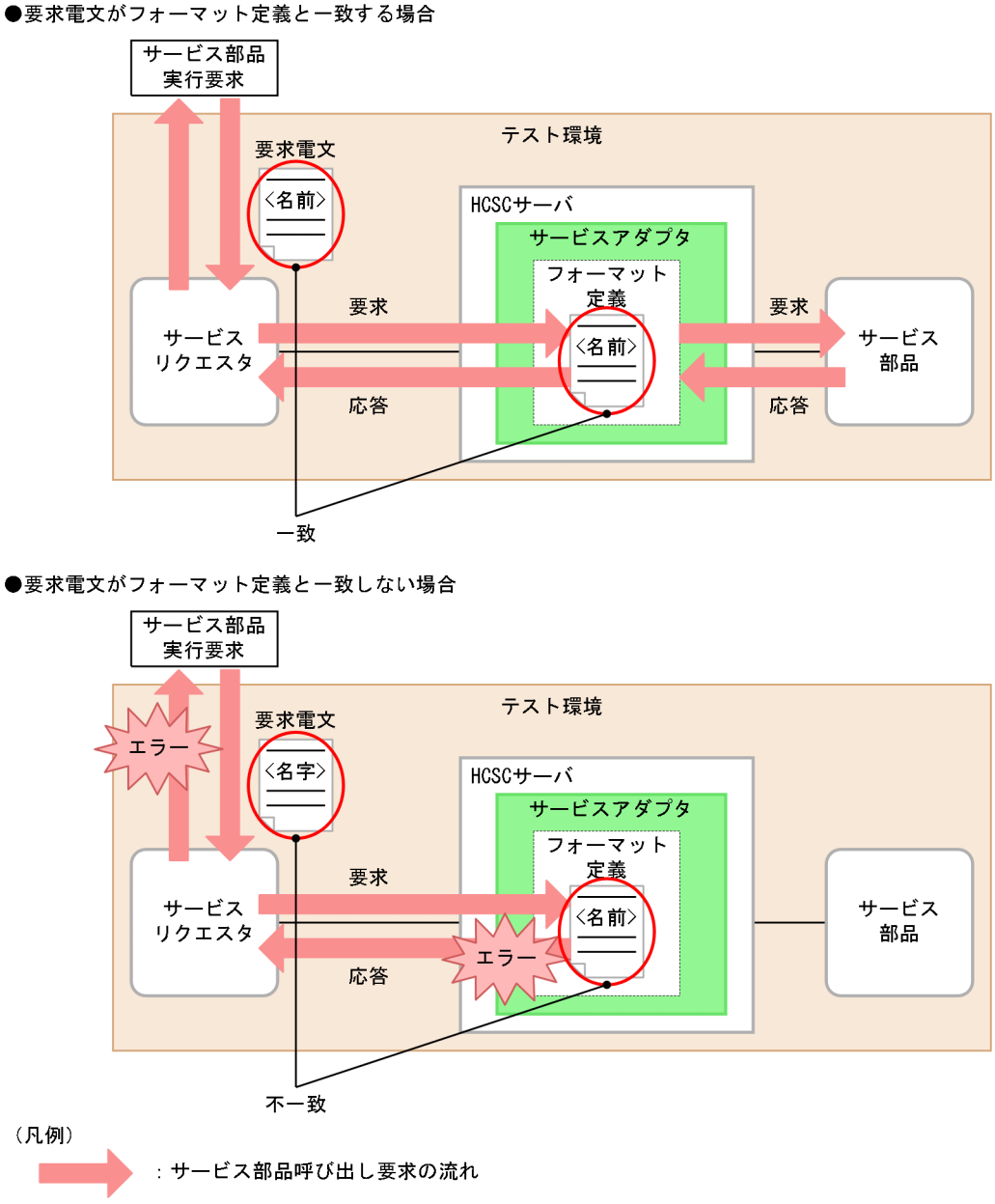
注意事項

データ検証機能が有効になっている場合，サービスリクエスタやサービスから受信した電文が，サービスアダプタ定義時に指定した電文フォーマットに適合しているかどうかのチェックが実施されます。データ検証機能が無効になっている場合，このチェックは実施されません。

データ検証機能が無効の状態で，HCSC サーバが電文フォーマットに適合しない電文を受信した場合，動作は保証しません。

要求電文とフォーマット定義が一致する場合と一致しない場合の電文の流れを次の図に示します。

図 6-1　要求電文とフォーマット定義が一致する場合と一致しない場合の電文の流れ



6.2.2　データ検証機能の設定

データ検証機能を使用するには，HCSC サーバランタイム定義ファイルの telegram-validation プロパティで値を設定します。

telegram-validation プロパティの詳細は，マニュアル「サービスプラットフォーム リファレンス」の「6.5.6　HCSC サーバランタイム定義ファイル」を参照してください。

6.2.3　データ検証機能の有効範囲

データ検証機能の有効範囲を次の表に示します。なお，データ検証機能が有効な場合，検証の対象は HCSC サーバ全体になります。

表 6-1　データ検証機能の有効範囲

サービスアダプタの種別	電文種別	要求/応答	サービスアダプタでのデータ変換有無	検証機能有効/無効	検証される電文
SOAP アダプタ，SessionBean アダプタ，MDB（WS-R）アダプタ，MDB（DB キュー）アダプタ	XML	要求	データ変換なし	有効	サービス部品電文
			標準電文→サービス部品電文へのデータ変換あり	有効	標準電文，サービス部品電文
		応答	データ変換なし	有効	サービス部品電文
			サービス部品電文→標準電文へのデータ変換あり	有効	標準電文，サービス部品電文
	バイナリ	要求	データ変換なし	無効	－
			リクエスタ→標準電文→サービス部品電文へのデータ変換以外	無効	－
		応答	データ変換なし	無効	－
			サービス部品電文→標準電文→リクエスタへのデータ変換以外	無効	－
DB アダプタ	XML	すべて	すべて	無効	－

（凡例）

－：該当しません。

電文の検証機能が有効になるサービスアダプタの詳細については，マニュアル「サービスプラットフォーム システム構築・運用ガイド」の「7.7.24　ユーザ電文に誤りがある場合の障害対策」を参照してください。

6.3　ユーザ電文トレース機能

6.3.1　ユーザ電文トレース機能の概要

ユーザ電文トレース機能は，HCSC サーバ上を流れる電文の内容をユーザ電文トレースファイルに出力する機能です。

次の内容をユーザ電文トレースとして取得できます。

- リクエスト受付で受け付けたサービス部品呼び出し要求または応答の電文
- ビジネスプロセスを呼び出した時の要求または応答の電文
- サービスアダプタからサービス部品を呼び出したときの要求または応答の電文
- データ変換を行ったときの変換前または変換後の電文
- バイナリデータの変換エラー時のデバッグ情報

ユーザ電文トレース機能を使用することによって，設計したとおりの電文が流れているかを確認できます。また，各ポイント（リクエスタ側とサービス側，ビジネスプロセスのアクティビティ，データ変換など）も確認できるため，開発の容易性が増します。

6.3.2　ユーザ電文トレース機能の設定

ユーザ電文トレース機能を使用するには，次に示す HCSC サーバランタイム定義ファイルのプロパティで値を設定します。

- telegramtrace プロパティ
- telegramtrace-filepath プロパティ
- telegramtrace-filesize プロパティ
- telegramtrace-filenum プロパティ
- telegramtrace-trigger プロパティ
- telegramtrace-component-id-filepath プロパティ
- telegramtrace-component-type-filepath プロパティ
- telegramtrace-format プロパティ

各プロパティの詳細は，マニュアル「サービスプラットフォーム リファレンス」の「6.5.6　HCSC サーバランタイム定義ファイル」を参照してください。

ユーザ電文トレースを取得する設定にすると，HCSC サーバ全体に対して有効となります。

6.3.3　ユーザ電文トレース機能使用時の注意事項

ユーザ電文トレース機能使用時の注意事項を次に示します。

- ユーザ電文トレース機能は，電文の内容をファイルに出力します。そのため，電文の内容が情報漏洩する懸念があり，セキュリティ上の問題が発生します。したがって，出力したユーザ電文トレースファイルの取り扱いに注意してください。
- ユーザ電文トレースファイルが電文の内容を出力する際は，出力先の環境に設定された文字コード（J2SE の file.encoding プロパティで設定された文字コード）を使用します。

6.4　アクティビティトレース機能

アクティビティトレースを使うと，アクティビティの処理経路を出力できるようになります。アクティビティトレースに出力されるのは，リクエスト時にビジネスプロセス内で実行されたアクティビティのログです。

アクティビティトレースを使用することで，設計したとおりにアクティビティが遷移しているかを確認できます。また，そのほかのトレース情報や設計情報と突き合わせることで，アクティビティが意図した分岐先に遷移しなかった場合などの原因の調査にも役立ちます。

アクティビティトレースの出力方法については，マニュアル「サービスプラットフォーム システム構築・運用ガイド」の「7.4.5　アクティビティトレース」を参照してください。

7 他製品と連携して利用する機能

この章では，JP1 と連携した運用管理，接続できるデータベース，および HiRDB と連携した実行履歴の管理について説明します。

7.1　JP1 と連携した運用管理

JP1 と連携してサービスプラットフォームを運用すると，業務システム全体の監視，問題の検知などの運用を効率良くできるようになります。また，サーバやアプリケーションの起動・停止を自動化することで，システムの日常運用の効率化を図れます。

JP1 と連携したシステムの運用については，マニュアル「アプリケーションサーバ 機能解説 運用／監視／連携編」の「12.2　JP1 との連携」も参照してください。

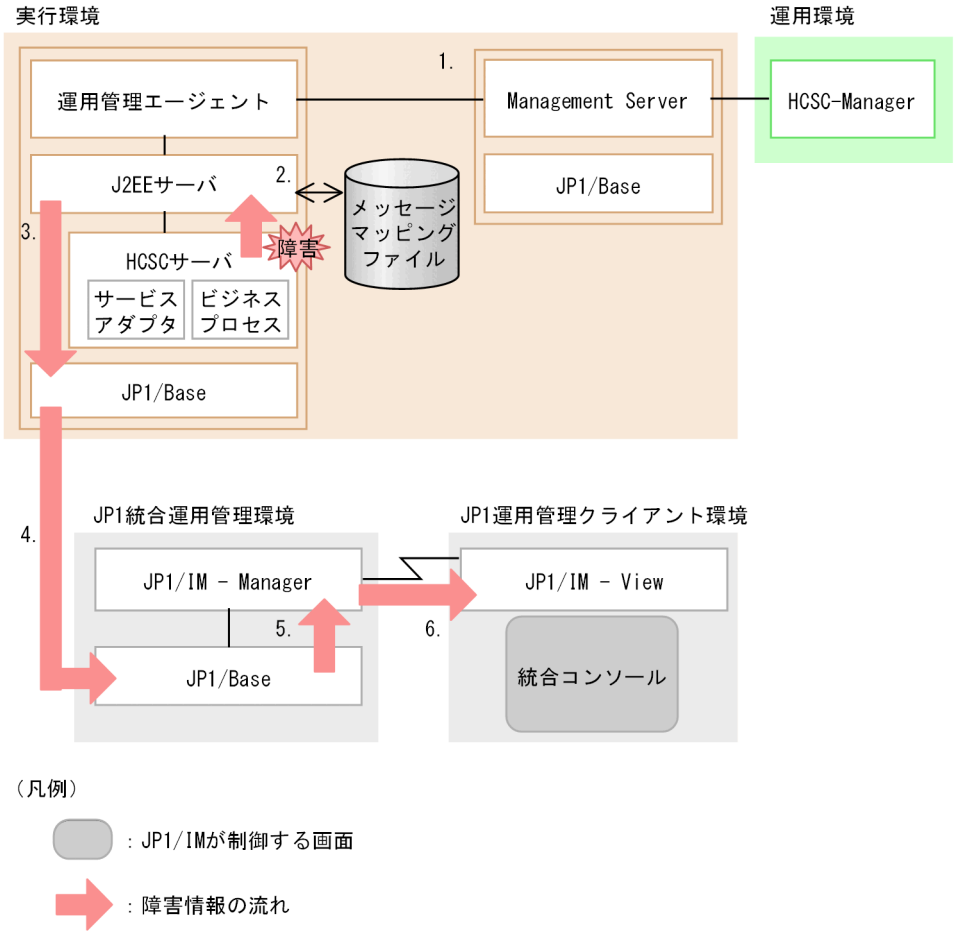
7.1.1　JP1/IM との連携による障害検知

システムで発生した障害を JP1 イベントによって集中管理することができます。サービスプラットフォームでは JP1/IM - Manager と連携し，システムで発生した障害を検知して，障害メッセージを JP1 イベントとして JP1 へ通知できます。また，JP1/IM - View では，JP1/IM - Manager のホストに接続して，実行環境から発行された JP1 イベントを表示できます。JP1 イベントとして発行されるメッセージは，ユーザの環境に合わせてカスタマイズできます。

どの障害が発生した場合に，どのメッセージが通知されるかをあらかじめ設定しておくことで，JP1/IM - View で表示されるメッセージログを監視して，障害発生時の障害の種類や原因などの調査を簡略化できます。

JP1/IM と連携した障害監視の概要を次の図に示します。

図 7-1　JP1/IM と連携した障害監視の概要



1. Management Server で，JP1/IM で監視する JP1 イベント発行を設定します。
2. 実行環境で発生した障害のメッセージが JP1 イベントに変換されます。
3. JP1/Base を使用して，システムで発生した障害のメッセージが JP1 イベントとして発行されます。
4. JP1 イベントが JP1 統合運用管理環境に転送されます。
5. JP1 統合運用管理環境に転送されてきた JP1 イベントが，JP1/IM - Manager によって収集されます。
6. 統合コンソールから JP1/IM - View を使用し，JP1/IM - Manager に接続して，JP1 イベントの詳細を確認できます。

7.1.2　JP1 によるプロセス監視

サービスプラットフォームのシステムを JP1 で監視する場合の推奨プロセスについて説明します。

(1) 監視推奨プロセス

HCSC サーバ起動時の各論理サーバのプロセスは，Manager で監視されていて，プロセスダウンやハングアップなどの障害が発生すると，論理サーバを自動で再起動します。そのため，サービスプラットフォームのシステムを JP1 で監視する場合は，次のプロセスを監視の対象とすることを推奨します。

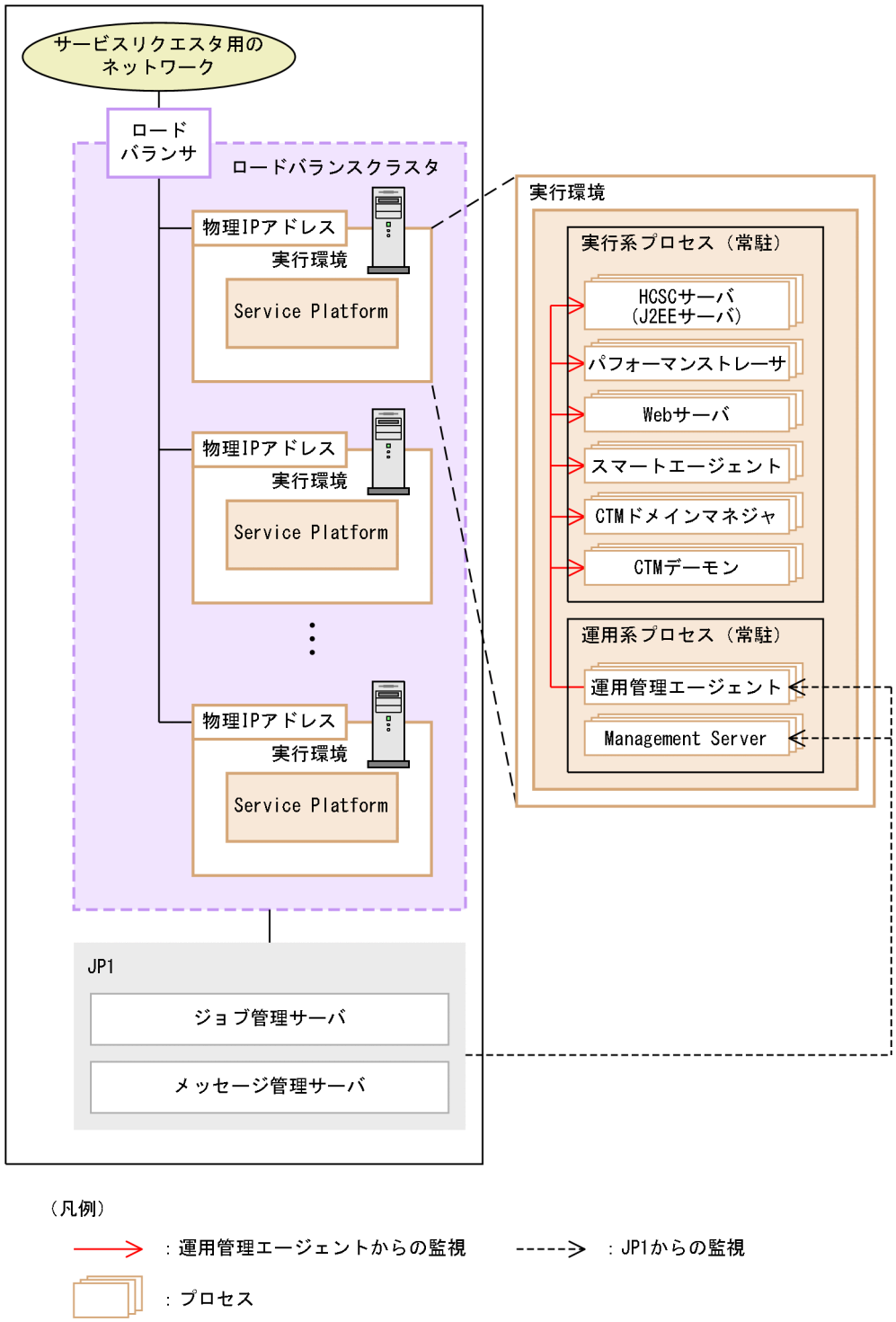
- Management Server
- 運用管理エージェント

ポイント

各論理サーバのプロセスのうち，上記推奨プロセス以外のプロセスを JP1 で監視すると，プロセスの二重監視になります。

HCSC サーバ起動時のプロセス構成を次の図に示します。

図 7-2　HCSC サーバ起動時のプロセス構成



JP1 の監視推奨となるプロセス名を次の表に示します。

表 7-1　JP1 からの監視推奨となるプロセス名

監視推奨プロセス	プロセス名	
	Windows の場合	UNIX の場合
Management Server	mngsvr.exe	cjstartsv※

監視推奨プロセス	プロセス名	
	Windows の場合	UNIX の場合
運用管理エージェント	adminagentsv.exe	adminagent

注※

コマンドライン名（ps -ef 相当）で監視します。実行例を次に示します。

/opt/Cosminexus/CC/server/bin/cjstartsv cosmi_m

「cosmi_m」はデフォルトのサーバ名です。「cosmi_m」以外のサーバ名に変更している場合は，監視するコマンドライン名も変更する必要があります。

また，データベースを使用している場合は，使用しているデータベースのプロセスも JP1 の監視推奨プロセスとなります。それらのプロセス名については，各データベースのマニュアルを参照してください。

(2) プロセスを監視するときの注意事項

プロセス数の上限と下限で監視する場合，上限を 1，下限を 1 で監視すると，一時的に cjstartsv や adminagent のプロセスが 1 より大きくなり，障害を検知してしまうことがあります。Management Server，運用管理エージェントについては複数起動を抑止する機能があるため，Manager の推奨としては上限なしで監視してください。

7.2　接続できるデータベース

ここでは，サービスプラットフォームから接続できるデータベースおよび接続に使用する JDBC ドライバについて説明します。データベースごとに接続に使用する JDBC ドライバが異なります。なお，使用する機能によって接続できるデータベースおよび使用する JDBC ドライバが異なります。

サービスプラットフォームから接続できるデータベース，JDBC ドライバおよび接続に使用する機能の対応を，次の表に示します。

表 7-2　サービスプラットフォームから接続できるデータベース

データベース	JDBC ドライバ	メッセージング機能およびビジネスプロセス機能		DB アダプタ機能	
		ローカルトランザクション	グローバルトランザクション	ローカルトランザクション	グローバルトランザクション
HiRDB	HiRDB Type4 JDBC Driver	◎	◎	◎	◎
Oracle	Oracle JDBC Thin Driver	○	○	○	○

（凡例）

◎：使用できます。かつ，使用を推奨します。

○：使用できます。

7.3 HiRDB と連携した実行履歴の管理

HiRDB と連携してプロセスインスタンスの実行履歴を管理すると，検索したり削除したりするときの効率化を図れます。

7.3.1 実行履歴の分割格納

データベースに HiRDB を使用する場合で，かつプロセスインスタンスの実行履歴が膨大になる場合は，実行履歴の分割格納をすると，実行履歴を検索したり削除したりするときの効率が向上します。

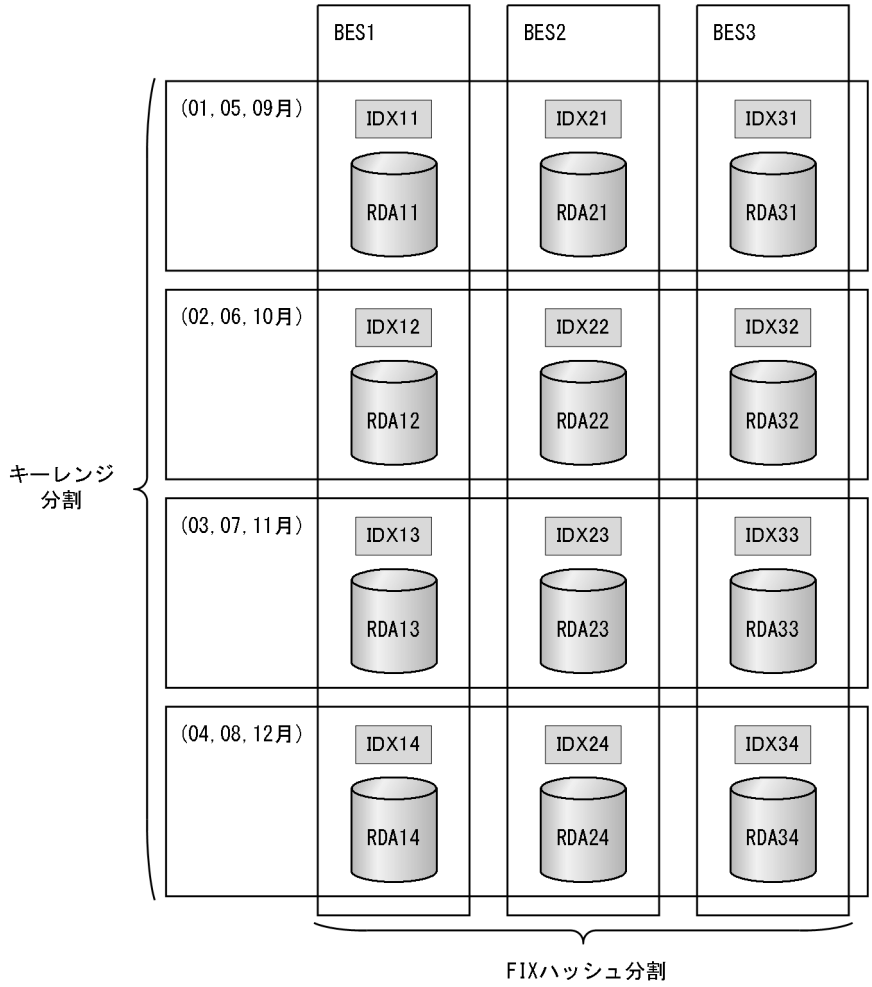
サービスプラットフォームでは，日付データによる境界値指定のキーレンジ分割が利用できます。また，分割されたデータをさらにプロセスインスタンス識別子による FIX ハッシュ分割で分割するマトリクス分割も利用できます。

なお，マトリクス分割を利用するためには，HiRDB Advanced High Availability が必要です。

キーレンジ分割および FIX ハッシュ分割の詳細は，マニュアル「HiRDB システム導入・設計ガイド」の表の横分割の種類に関する説明を参照してください。

マトリクス分割の例として，日付データによる境界値指定のキーレンジ分割とプロセスインスタンス識別子の組み合わせを次の図に示します。

図 7-3　日付データによる境界値指定のキーレンジ分割とプロセスインスタンス識別子を組み合わせたマトリクス分割の例



(1)　日付データによる境界値指定のキーレンジ分割

サービスプラットフォームでは，プロセスインスタンスの実行履歴の管理に関するテーブルの列名（SplitKey）をキーレンジ分割のキーとして利用できます。

DB サーバ内の横分割された RD エリアに実行履歴を分割して格納することで，RD エリア単位の操作ができます。そのため，実行履歴が膨大になるほど削除効率が向上します。

実行履歴を RD エリア単位で削除する例については，マニュアル「サービスプラットフォーム システム構築・運用ガイド」の「6.1.4　プロセスインスタンスの実行履歴を削除する」を参照してください。

(2) プロセスインスタンス識別子による FIX ハッシュ分割

サービスプラットフォームでは，プロセスインスタンスの実行履歴の管理に関するテーブルの列名（ProcessID）を FIX ハッシュ分割のキーとして利用できます。

検索条件に分割キーを指定されていると，該当データがあると予測される RD エリアだけを検索するため，検索効率が向上します。

注意事項

検索条件に分割キーが指定されていない場合，分割したすべての RD エリアが探索対象となるため，サーバ間の通信が発生する分，分割しなかった場合よりも検索効率が劣化するおそれがあります。

リクエスト処理時に発行される SQL は，ビジネスプロセスの内容によって異なります。発行される SQL の詳細は，SQL トレース機能を利用してご確認ください。SQL トレース機能については，マニュアル「HiRDB UAP 開発ガイド」を参照してください。

(3) 実行履歴を分割するときの注意事項

プロセスインスタンスの実行履歴を分割する場合は，次の点に注意してください。

- HCSC サーバがセットアップ済みの環境からは移行できません。
- RD エリア単位に実行履歴を削除する場合は，すべての HCSC サーバの受付を停止させてください。
- 相関セットの重複チェックを強化してください。相関セットの重複チェックを強化するには，HCSC サーバセットアップ定義ファイルの db-tbl-split-corcheck-use プロパティの設定を ON にしてください。設定を OFF にすると，リクエストが正しく処理されないおそれがあります。db-tbl-split-corcheck-use プロパティの詳細は，マニュアル「サービスプラットフォーム リファレンス」の「6.4.3　HCSC サーバセットアップ定義ファイル」を参照してください。
- 相関セットが同じリクエストを要求した場合，要求したリクエストがすべてエラーになるおそれがあります。相関セットが同じリクエストを要求しないでください。

8 FTP と連携してファイルを転送する機能（FTP 連携）

この章では，FTP と連携してファイルを転送する機能（FTP 連携）について説明します。

8.1　FTP 連携の概要

FTP（ファイル転送プロトコル）を利用して，さまざまなシステム間のファイル転送を中継する機能を **FTP 連携**といいます。

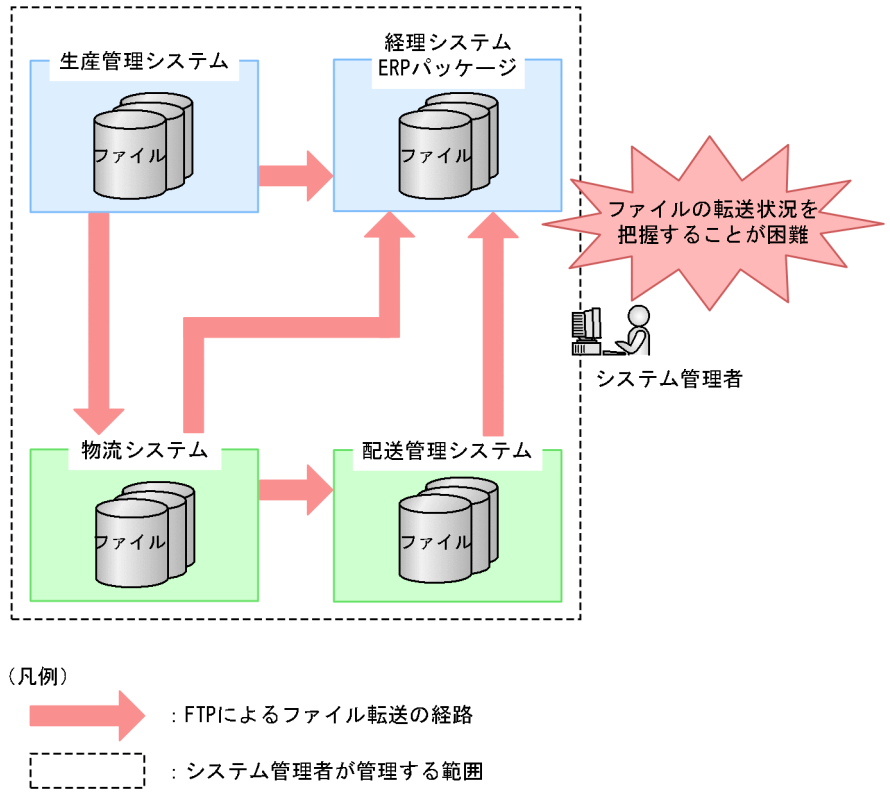
ここでは，FTP 連携の目的および特長について説明します。

8.1.1　FTP 連携の目的

FTP を利用した既存のシステムでは，ファイルを転送するシステム同士が個別に連携しています。このため，システムの管理者にとって，システム全体の転送状況（転送経路やデータの種類など）を管理することが困難です。

既存のシステムで FTP による転送状況を管理する例を次の図に示します。

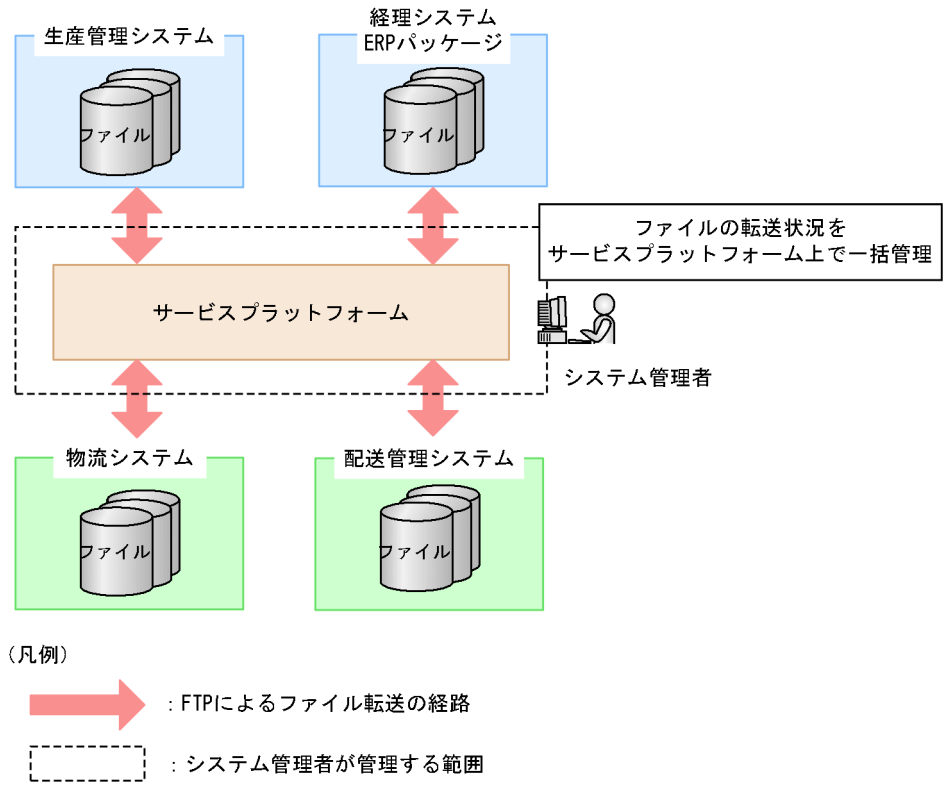
図 8-1　既存のシステムの例



FTP 連携を利用することで，これまで個別に管理していたシステム間の転送状況をサービスプラットフォーム上で一括して管理できるようになります。

サービスプラットフォームを利用したシステムで FTP による転送状況を管理する例を次の図に示します。

図 8-2　サービスプラットフォームを利用したシステムの例



システム間の転送状況を一括して管理することで，システムの改修時にシステム全体の影響範囲を把握しやすくなるため，改修時のコストを削減できるメリットもあります。

8.1.2　FTP 連携の特長

FTP 連携には次の特長があります。

- FTP によるシステム間の転送状況を管理できます。

  ビジネスプロセスのユーザ定義 Java クラスを使用することで，転送状況を把握するために必要な次の情報をログとして取得できます。

  - システム間の転送経路
  - 転送の開始・終了時刻，および転送に要した時間
  - 転送の頻度
  - 転送のデータ量

- FTP を利用したさまざまなシステムと連携できます。

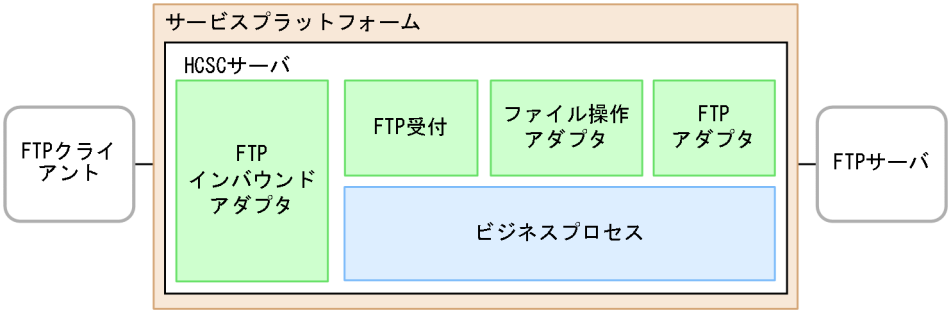
  既存のシステムと接続してファイルの送受信を中継できます。システム間でファイル形式や扱う文字コードが異なる場合，レイアウト・文字コード変換の機能を利用して対応することができます。接続する FTP クライアントおよび FTP サーバに対してソフトウェアの追加は不要です。

8.2　FTP 連携を利用するためのシステムの構成

FTP によるファイル転送では，FTP クライアントが FTP サーバに転送を要求し，ファイルをアップロードしたりダウンロードしたりします。このマニュアルでは，転送を要求する側のシステムを **FTP クライアント**，転送要求を受け付ける側のシステムを **FTP サーバ**と呼びます。

FTP 連携を利用するための，FTP サーバ，FTP クライアント，およびサービスプラットフォーム内のシステムの構成を次に示します。

図 8-3　FTP 連携を利用するためのシステムの構成



システムを構成する要素について説明します。

FTP インバウンドアダプタ

FTP クライアントからの要求の処理や，FTP クライアントの認証，ファイルのデータ転送処理などを行うためのリソースアダプタです。FTP クライアントからの要求は FTP インバウンドアダプタを経由して FTP 受付に送られます。

FTP 受付

FTP クライアントからの要求を受け付けるためのインターフェースです。FTP クライアントとビジネスプロセスとの間でファイルの送受信を行います。

ビジネスプロセス

ファイル操作アダプタや FTP アダプタを呼び出すためのアクティビティを定義します。

ファイル操作アダプタ

ファイルのレイアウト変換や文字コード変換，複製，削除などのファイル操作を行うためのアダプタです。

FTP アダプタ

FTP サーバと接続してファイルの送受信を行うためのアダプタです。

8.3 FTP 連携を利用したファイル転送パターンの例

FTP 連携を利用して実現できるファイル転送パターンには，次の種類があります。

- FTP クライアントから FTP サーバへのファイル転送
- FTP サーバから FTP クライアントへのファイル転送
- FTP クライアントから別の FTP クライアントへのファイル転送
- FTP サーバのファイルリストの取得

各ファイル転送パターンは，ビジネスプロセス，FTP 受付，FTP アダプタ，およびファイル操作アダプタを組み合わせることで実現できます。

ここでは，各ファイル転送パターンの特長，およびファイル転送の流れについて説明します。

8.3.1 FTP クライアントから FTP サーバへのファイル転送

この転送パターンでは，転送コマンドの STOR コマンドまたは APPE コマンドを使用して，FTP クライアントから FTP サーバにファイルを送信します。

この転送パターンには次の 2 種類があります。

同期型ビジネスプロセスを利用する場合

FTP サーバへのファイル転送が完了してから FTP クライアントに応答を返す転送パターンです。

この転送パターンでは，FTP 連携を利用しない場合と同様に，FTP クライアントと FTP サーバの処理の状態を同期させることができます。

非同期型ビジネスプロセスを利用する場合

FTP クライアントに応答を返してから FTP サーバにファイルを送信する転送パターンです。

この転送パターンでは，FTP クライアントから見た応答時間が短縮されるため，次の場合に活用できます。

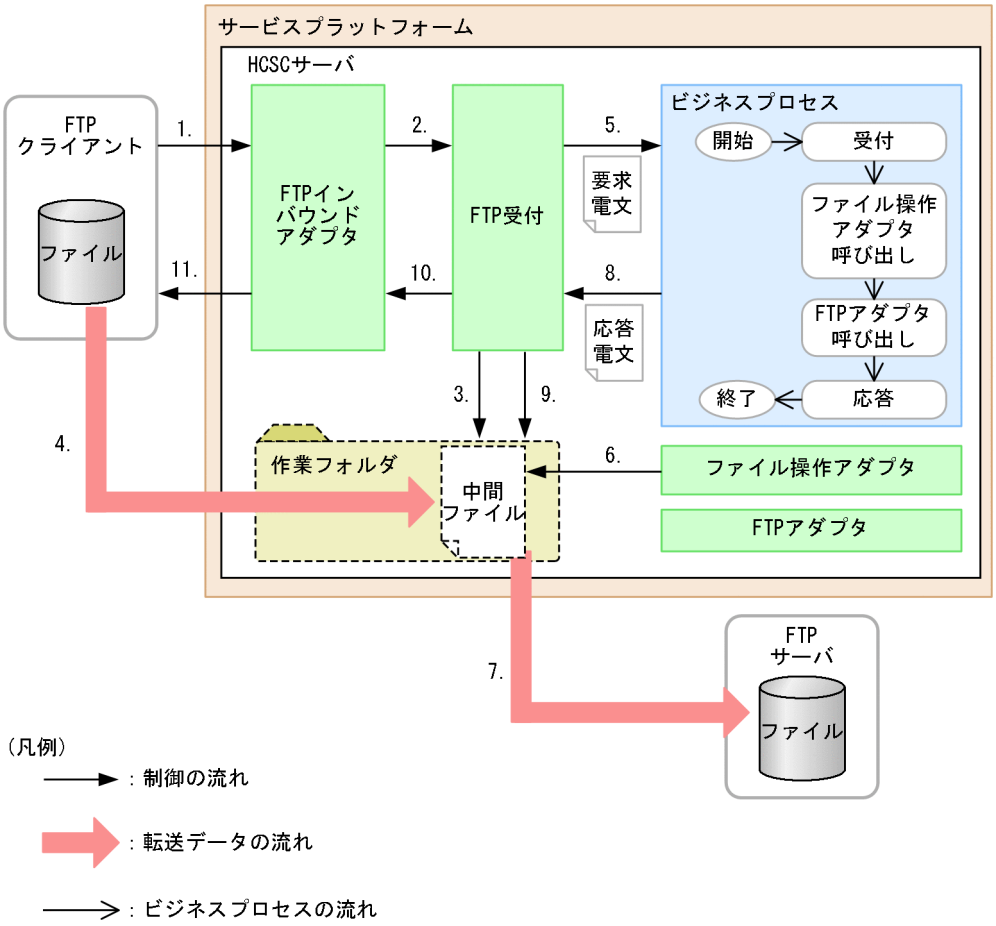
- ある拠点から複数の拠点に対して同じ情報を同報送信する場合
- FTP クライアントの無通信監視時間を超えるような大規模ファイルを送信する場合

それぞれのファイル転送の流れを次に示します。

(1) 同期型ビジネスプロセスを利用したファイル転送の流れ

ファイル転送の流れを次の図に示します。

図 8-4　FTP クライアントから FTP サーバへのファイル転送の流れ（同期型ビジネスプロセスを利用する場合）



図中の制御，転送データ，およびビジネスプロセスの流れについて説明します。次の番号は，図中の番号と対応しています。

1. FTP クライアントが転送コマンド（STOR コマンドまたは APPE コマンド）を実行します。
2. 転送コマンドを受け取った FTP インバウンドアダプタが FTP 受付を呼び出します。
3. FTP 受付が作業フォルダ（HCSC サーバ上でファイルを扱うための一時的なフォルダ）を作成します。
4. FTP クライアントから転送されたファイルのデータが，作業フォルダ内に中間ファイルとして出力されます。
5. FTP 受付がビジネスプロセスを呼び出します。このとき，FTP クライアントから受け取った転送コマンドや中間ファイルのファイル名などの情報は，要求電文に格納されてビジネスプロセスに渡されます。
6. ビジネスプロセスからファイル操作アダプタが呼び出されたあと，ファイル操作アダプタは作業フォルダ内の中間ファイルを読み込んで，レイアウト変換やコード変換などの処理をします。
7. ビジネスプロセスから FTP アダプタが呼び出されたあと，FTP アダプタは中間ファイルのデータを FTP サーバに転送します。
8. ビジネスプロセスで応答アクティビティが実行されたあと，FTP 受付に応答電文が渡されます。
9. FTP 受付が作業フォルダおよび作業フォルダ中の中間ファイルを削除します。

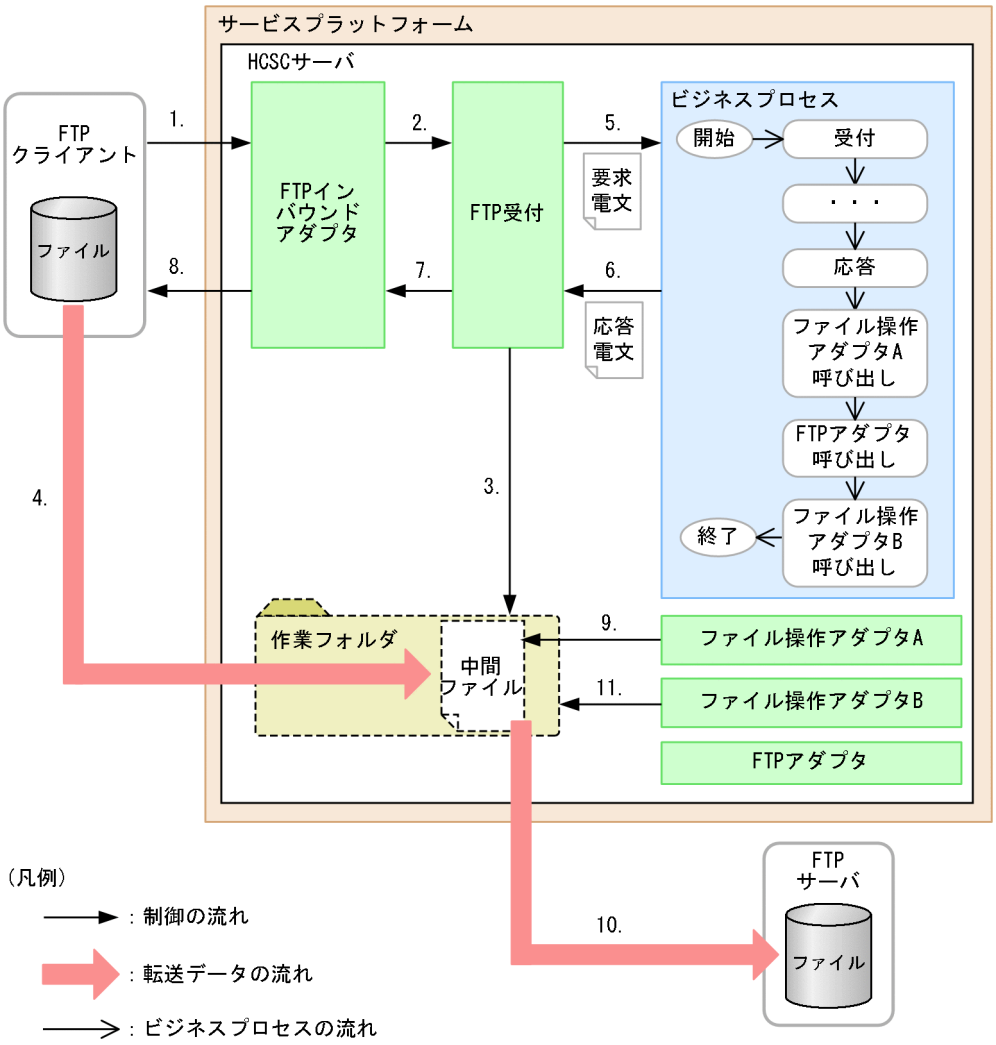
10. FTP 受付から FTP インバウンドアダプタに応答を返します。

11. 応答を受け取った FTP インバウンドアダプタは，転送コマンドの実行結果を FTP クライアントに返します。

(2) 非同期型ビジネスプロセスを利用したファイル転送の流れ

ファイル転送の流れを次の図に示します。

図 8-5 FTP クライアントから FTP サーバへのファイル転送の流れ（非同期型ビジネスプロセスを利用する場合）



図中の制御，転送データ，およびビジネスプロセスの流れについて説明します。次の番号は，図中の番号と対応しています。

1. FTP クライアントが転送コマンド（STOR コマンドまたは APPE コマンド）を実行します。
2. 転送コマンドを受け取った FTP インバウンドアダプタが FTP 受付を呼び出します。
3. FTP 受付が作業フォルダを作成します。

4. FTP クライアントから転送されたファイルのデータが，作業フォルダ内に中間ファイルとして出力されます。
5. FTP 受付がビジネスプロセスを呼び出します。このとき，FTP クライアントから受け取った転送コマンドや中間ファイルのファイル名などの情報は，要求電文に格納されてビジネスプロセスに渡されます。
6. ビジネスプロセスで応答アクティビティが実行されたあと，FTP 受付に応答電文が渡されます。
7. FTP 受付から FTP インバウンドアダプタに応答を返します。
8. 応答を受け取った FTP インバウンドアダプタは，転送コマンドの実行結果を FTP クライアントに返します。
9. ビジネスプロセスからファイル操作アダプタ A が呼び出されたあと，ファイル操作アダプタ A は作業フォルダ内の中間ファイルを読み込んで，レイアウト変換やコード変換などの処理をします。
10. ビジネスプロセスから FTP アダプタが呼び出されたあと，FTP アダプタは中間ファイルのデータを FTP サーバに転送します。
11. ビジネスプロセスからファイル操作アダプタ B が呼び出されたあと，ファイル操作アダプタ B はビジネスプロセスが終了する前に作業フォルダおよび作業フォルダ中の中間ファイルを削除します。

8.3.2 FTP サーバから FTP クライアントへのファイル転送

この転送パターンでは，FTP クライアントから転送コマンドの RETR コマンドを使用して，FTP サーバから FTP クライアントにファイルを送信します。

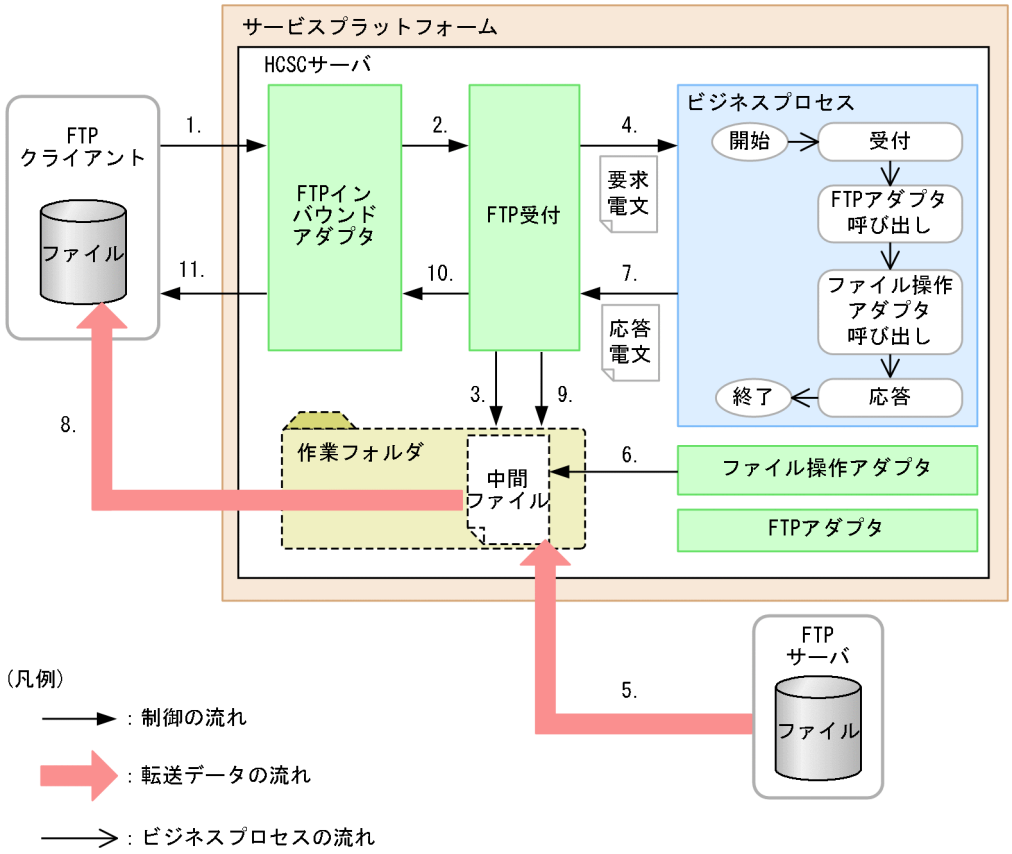
この転送パターンでは中間ファイルの格納先に作業フォルダと共通フォルダが利用できます。

それぞれのファイル転送の流れを次に示します。

(1) 作業フォルダを利用したファイル転送の流れ

ファイル転送の流れを次の図に示します。

図 8-6　FTP サーバから FTP クライアントへのファイル転送の流れ（作業フォルダを利用する場合）



図中の制御，転送データ，およびビジネスプロセスの流れについて説明します。次の番号は，図中の番号と対応しています。

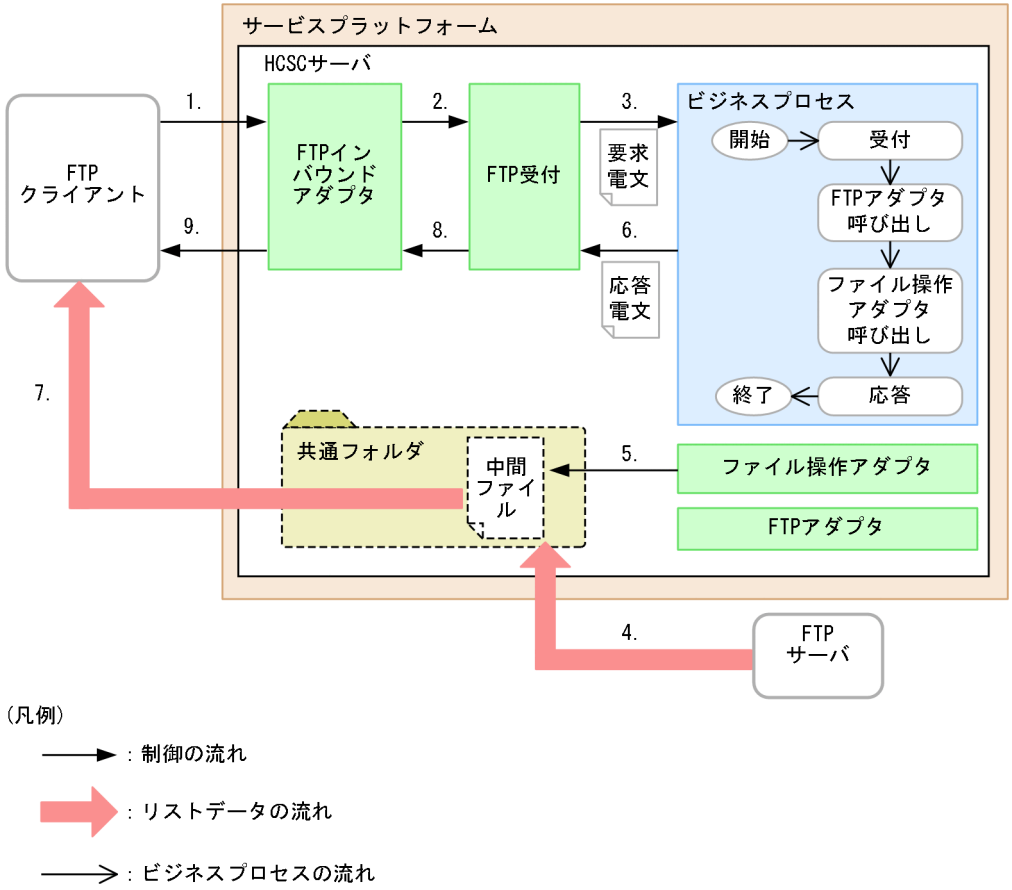
1. FTP クライアントが転送コマンド（RETR コマンド）を実行します。
2. 転送コマンドを受け取った FTP インバウンドアダプタが FTP 受付を呼び出します。
3. FTP 受付が作業フォルダを作成します。
4. FTP 受付がビジネスプロセスを呼び出します。このとき，FTP クライアントから受け取った情報（転送コマンドなど）は，要求電文に格納されてビジネスプロセスに渡されます。
5. ビジネスプロセスから FTP アダプタが呼び出されたあと，FTP アダプタは FTP サーバからファイルを取得し，作業フォルダ内に中間ファイルとして出力します。
6. ビジネスプロセスからファイル操作アダプタが呼び出されたあと，ファイル操作アダプタは作業フォルダ内の中間ファイルを読み込んで，レイアウト変換やコード変換などの処理をします。
7. ビジネスプロセスで応答アクティビティが実行されたあと，FTP 受付に中間ファイルの情報を含んだ応答電文が渡されます。
8. 応答電文で示された中間ファイルのデータが，作業フォルダから FTP クライアントに転送されます。
9. FTP 受付が作業フォルダおよび作業フォルダ中の中間ファイルを削除します。
10. FTP 受付から FTP インバウンドアダプタに応答を返します。

11. 応答を受け取った FTP インバウンドアダプタは，FTP クライアントに転送コマンドの実行結果を返します。

(2) 共通フォルダを利用したファイル転送の流れ

ファイル転送の流れを次の図に示します。

図 8-7 FTP サーバから FTP クライアントへのファイル転送の流れ（共通フォルダを利用する場合）



図中の制御，転送データ，およびビジネスプロセスの流れについて説明します。次の番号は，図中の番号と対応しています。

1. FTP クライアントが転送コマンド（RETR コマンド）を実行します。
2. 転送コマンドを受け取った FTP インバウンドアダプタが FTP 受付を呼び出します。
3. FTP 受付がビジネスプロセスを呼び出します。このとき，FTP クライアントから受け取った情報（転送コマンドなど）は，要求電文に格納されてビジネスプロセスに渡されます。
4. ビジネスプロセスから FTP アダプタが呼び出されたあと，FTP アダプタは FTP サーバからファイルを取得し，共通フォルダ内に中間ファイルとして出力します。
5. ビジネスプロセスからファイル操作アダプタが呼び出されたあと，ファイル操作アダプタは共通フォルダ内の中間ファイルを読み込んで，変換処理（レイアウト変換やコード変換など）をします。

6. ビジネスプロセスで応答アクティビティが実行されたあと，FTP 受付に中間ファイルの情報を含んだ応答電文が渡されます。
7. 応答電文で示された中間ファイルのデータが，共通フォルダから FTP クライアントに転送されます。
8. FTP 受付から FTP インバウンドアダプタに応答を返します。
9. 応答を受け取った FTP インバウンドアダプタは，FTP クライアントに転送コマンドの実行結果を返します。

8.3.3 FTP クライアントから別の FTP クライアントへのファイル転送

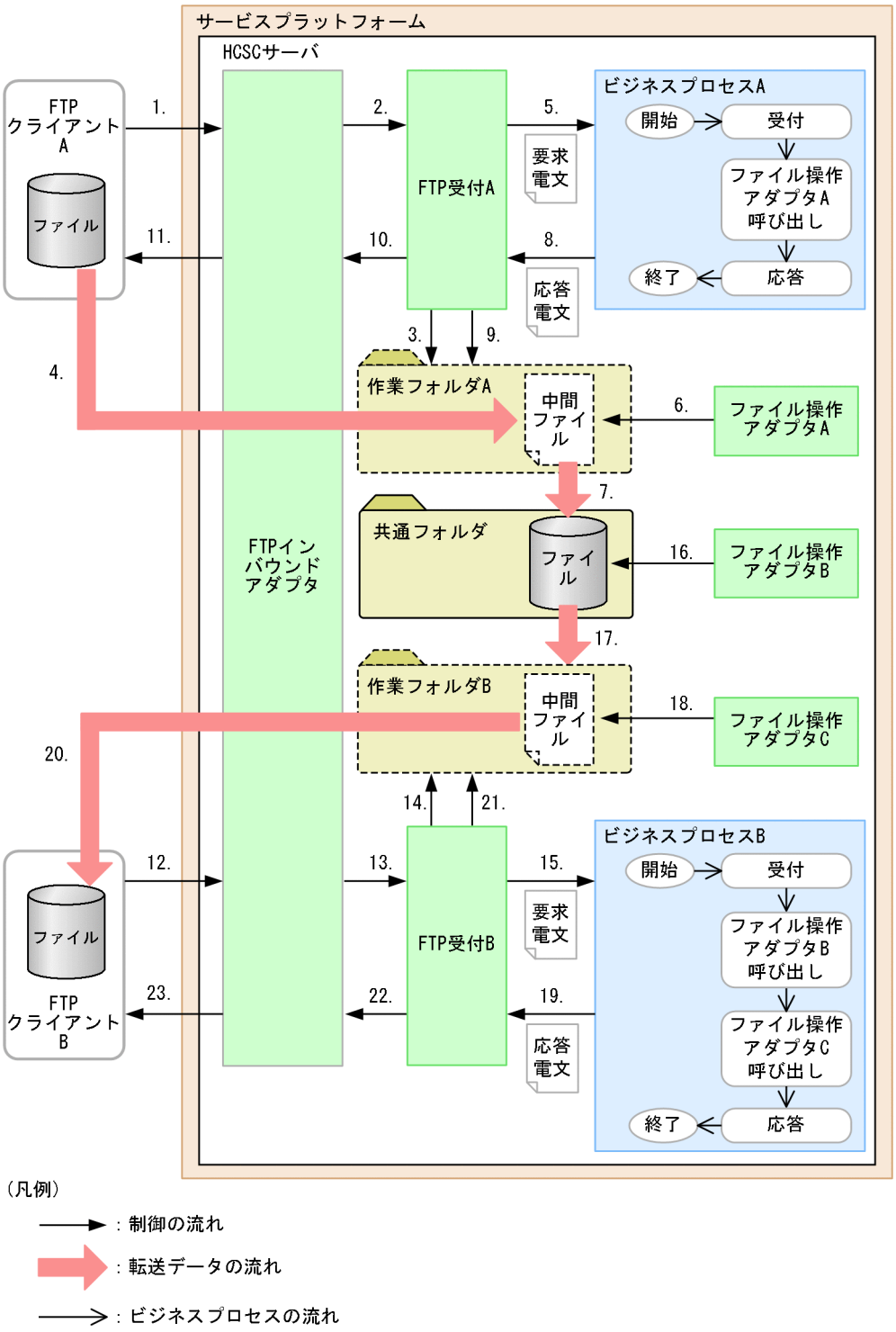
1 対 1 または 1 対複数の FTP クライアント間でファイルを送受信する転送パターンについて説明します。

この転送パターンでは，ファイルの送信側と受信側のシステムが任意のタイミングで HCSC サーバにアクセスできるため，海外拠点のように運用時間帯が異なるシステム間でファイル転送を行う場合に活用できます。

ファイル転送の流れを次の図に示します。

この図では，FTP クライアント A から STOR コマンドまたは APPE コマンドを実行して HCSC サーバにファイルを送信（図中の番号 1.～11.）したあと，FTP クライアント B から RETR コマンドを実行して HCSC サーバ内のファイルを取得（図中の番号 12.～23.）しています。

図 8-8　FTP クライアントから別の FTP クライアントへのファイル転送の流れ



図中の制御，転送データ，およびビジネスプロセスの流れについて説明します。次の番号は，図中の番号と対応しています。

1. FTP クライアント A が転送コマンド（STOR コマンドまたは APPE コマンド）を実行します。
2. 転送コマンドを受け取った FTP インバウンドアダプタが FTP 受付 A を呼び出します。
3. FTP 受付 A が作業フォルダ A を作成します。

4. FTP クライアント A から転送されたファイルのデータが，作業フォルダ A 内に中間ファイルとして出力されます。
5. FTP 受付 A がビジネスプロセス A を呼び出します。このとき，FTP クライアント A から受け取った転送コマンドや中間ファイルのファイル名などの情報は，要求電文に格納されてビジネスプロセス A に渡されます。
6. ビジネスプロセス A からファイル操作アダプタ A が呼び出され，作業フォルダ A の中間ファイルにアクセスします。
7. ファイル操作アダプタ A は，作業フォルダ A 内の中間ファイルを共通フォルダ（ユーザが作成する任意のフォルダ）内に複製します。
8. ビジネスプロセス A で応答アクティビティが実行されたあと，FTP 受付 A に応答電文が渡されます。
9. FTP 受付 A が作業フォルダ A および作業フォルダ A 内の中間ファイルを削除します。
10. FTP 受付 A から FTP インバウンドアダプタに応答を返します。
11. 応答を受け取った FTP インバウンドアダプタは，転送コマンドの実行結果を FTP クライアント A に返します。
12. FTP クライアント B が転送コマンド（RETR コマンド）を実行します。
13. 転送コマンドを受け取った FTP インバウンドアダプタが FTP 受付 B を呼び出します。
14. FTP 受付 B が作業フォルダ B を作成します。
15. FTP 受付 B がビジネスプロセス B を呼び出します。このとき，FTP クライアント B から受け取った転送コマンドなどの情報は，要求電文に格納されてビジネスプロセス B に渡されます。
16. ビジネスプロセス B からファイル操作アダプタ B が呼び出され，共通フォルダ内のファイルにアクセスします。
17. ファイル操作アダプタ B は，共通フォルダ内のファイルを作業フォルダ B に複製します。
18. ビジネスプロセス B からファイル操作アダプタ C が呼び出されたあと，ファイル操作アダプタ C は，作業フォルダ B 内の中間ファイルを読み込んで，レイアウト変換やコード変換などの処理をします。
19. ビジネスプロセス B で応答アクティビティが実行されたあと，FTP 受付 B に中間ファイルの情報を含んだ応答電文が渡されます。
20. 応答電文で示された中間ファイルのデータが FTP インバウンドアダプタから FTP クライアント B に転送されます。
21. FTP 受付 B は，作業フォルダ B および作業フォルダ B 内の中間ファイルを削除します。
22. FTP 受付 B から FTP インバウンドアダプタに応答を返します。
23. 応答を受け取った FTP インバウンドアダプタは，FTP クライアント B に転送コマンドの実行結果を返します。

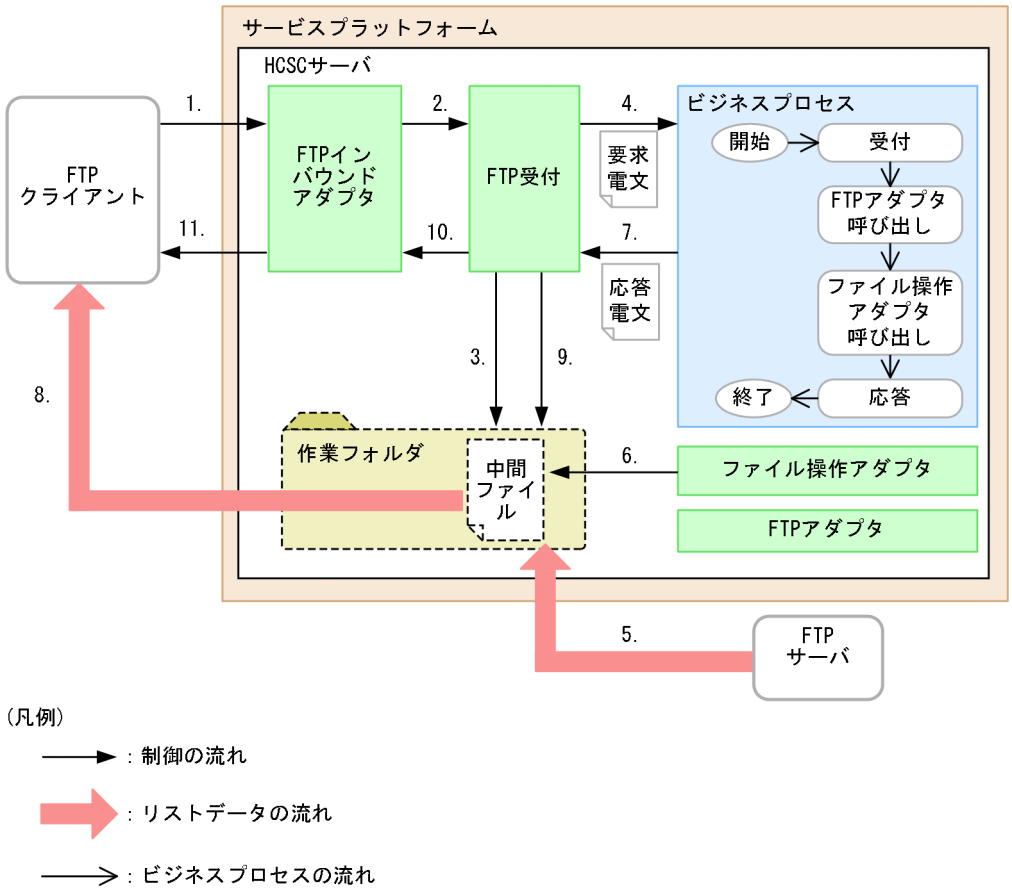
8.3.4　FTP サーバのファイルリストの取得

この転送パターンでは，FTP クライアントまたはサービスリクエスタからリストコマンドを使用して，FTP サーバのファイルリストを取得しています。

(1)　ファイルリストを作業フォルダに出力する場合

FTP サーバから取得したファイルリストやファイル操作アダプタで取得した共通フォルダ内のファイルリストを作業フォルダに出力して，FTP クライアントに送信する例について説明します。

図 8-9　FTP サーバのファイルリストの取得の流れ（作業フォルダに出力する場合）



図中の制御，リストデータ，およびビジネスプロセスの処理の流れについて説明します。次の番号は，図中の番号と対応しています。

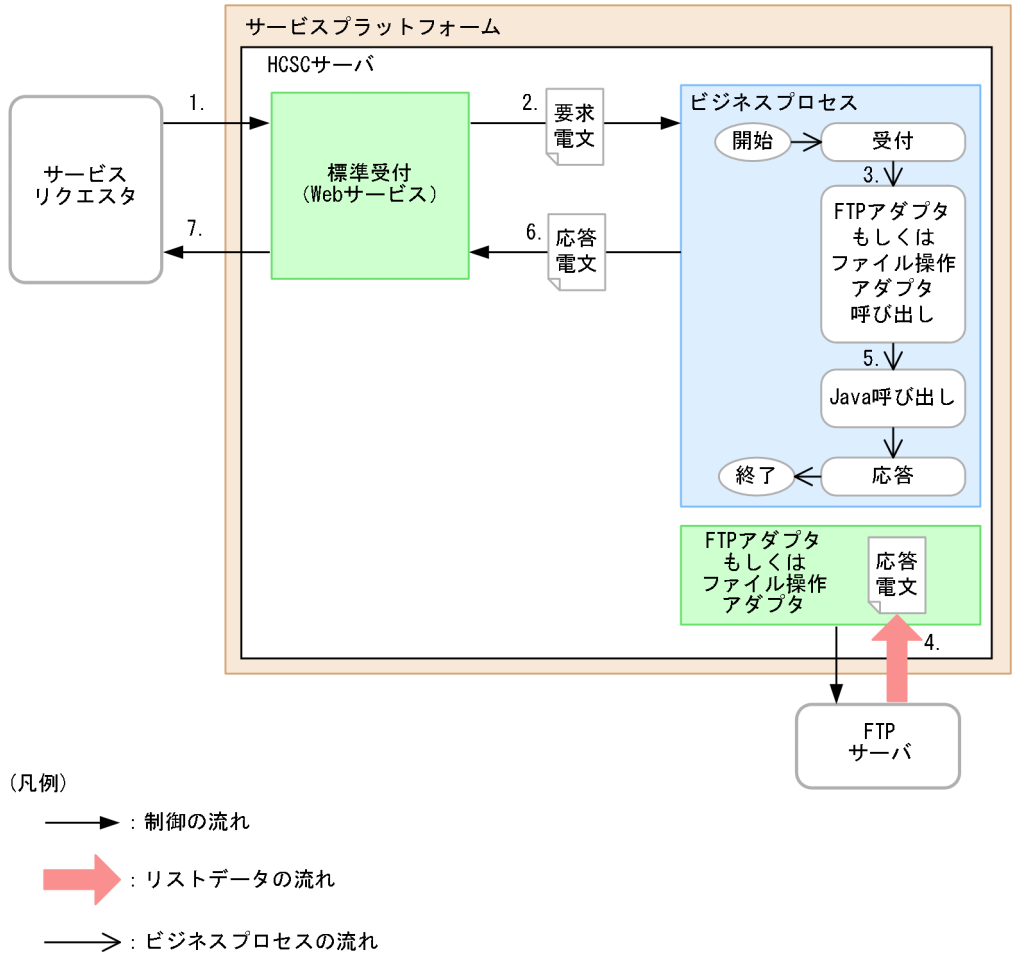
1. FTP クライアントがリストコマンドを実行します。
2. リストコマンドを受け取った FTP インバウンドアダプタが FTP 受付を呼び出します。
3. FTP 受付が作業フォルダを作成します。
4. FTP 受付がビジネスプロセスを呼び出します。このとき，FTP クライアントから受け取った FTP コマンドの情報などは，要求電文に格納されてビジネスプロセスに渡されます。
5. ビジネスプロセスから FTP アダプタが呼び出されたあと，FTP アダプタは FTP サーバからファイルリストを取得し，作業フォルダ内に中間ファイルとして出力します。

6. ビジネスプロセスからファイル操作アダプタが呼び出されたあと，ファイル操作アダプタは作業フォルダ内の中間ファイルを読み込んで，レイアウト変換やコード変換などの処理をします。共通フォルダ内のファイルリストを取得する場合は，ファイル操作アダプタのファイル一覧書き込みオペレーションでファイルリストを作業フォルダ内に中間ファイルとして出力します（ファイル一覧書き込みオペレーションの詳細は，「8.7.10　ファイル一覧書き込みオペレーション」を参照してください）。
7. ビジネスプロセスで応答アクティビティが実行されたあと，FTP 受付に中間ファイルの情報を含んだ応答電文が渡されます。
8. 応答電文で示された中間ファイルのデータが，作業フォルダから FTP クライアントに転送されます。
9. FTP 受付が作業フォルダおよび作業フォルダ中の中間ファイルを削除します。
10. FTP 受付から FTP インバウンドアダプタに応答を返します。
11. 応答を受け取った FTP インバウンドアダプタは，FTP クライアントにリストコマンドの実行結果を返します。

(2)　ファイルリストを応答電文に出力する場合

FTP サーバから取得したファイルリストを応答電文に出力して，ビジネスプロセスで処理する例について説明します。

図 8-10　FTP サーバのファイルリストの取得の流れ（応答電文に出力する場合）



図中の制御，リストデータ，およびビジネスプロセスの処理の流れについて説明します。次の番号は，図中の番号と対応しています。

1. サービスリクエスタがリストコマンドを要求するリクエストを実行します。
2. 標準受付（Web サービス）がビジネスプロセスを呼び出します。このとき，サービスリクエスタから受け取った FTP コマンドの情報などは，要求電文に格納されてビジネスプロセスに渡されます。
3. ビジネスプロセスから FTP アダプタもしくはファイル操作アダプタを呼び出します。ファイル操作アダプタの場合は，ファイル一覧取得オペレーションで共通フォルダ内のファイルリストを応答電文に格納します（ファイル一覧取得オペレーションの詳細は，「8.7.9　ファイル一覧取得オペレーション」を参照してください）。
4. 3.で呼び出したアダプタは FTP サーバからファイルリストを取得したあと，メモリに保持して応答電文に格納します。
5. 3.で呼び出したアダプタの応答電文のリストデータを Java 呼出アクティビティなどの要求電文で取得し，解析します。
6. ビジネスプロセスから標準受付（Web サービス）に応答電文を返します。
7. 標準受付（Web サービス）はサービスリクエスタに応答を返します。

8.4　FTP 受付の機能

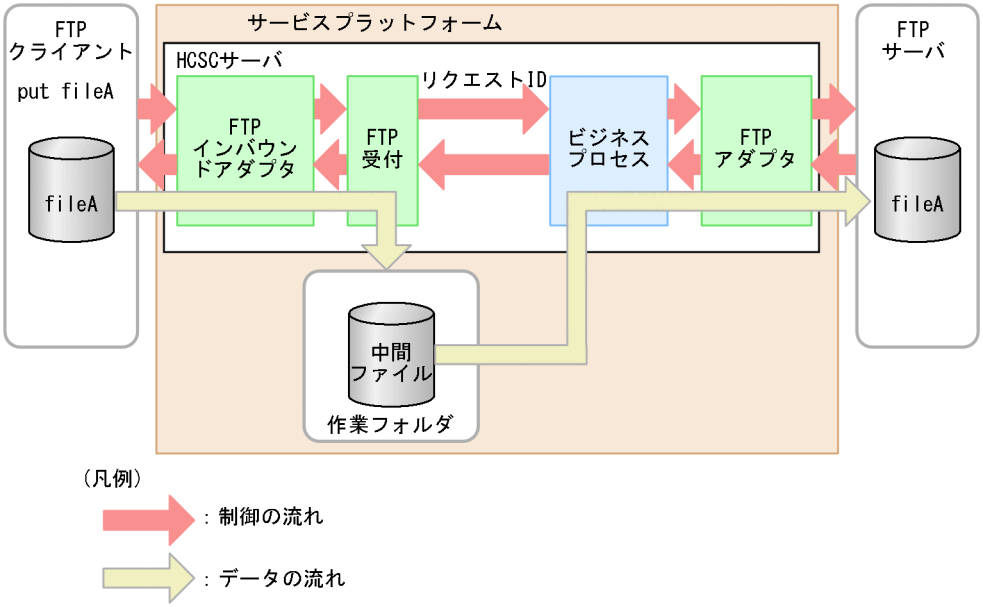
ここでは，FTP 受付の目的，および FTP 受付を利用したファイル送受信の概要について説明します。

FTP 受付とは，FTP クライアントから FTP インバウンドアダプタを経由して接続を受け付け，ビジネスプロセスを呼び出したり，FTP クライアントとビジネスプロセスの間で送受信されるファイルのデータやファイルの一覧情報を仲介したりするインターフェースです。

ビジネスプロセスから呼び出されるファイル操作アダプタや FTP アダプタと連携して動作することによって，ビジネスプロセスを介した FTP クライアントと FTP サーバ間のファイル転送，およびファイルの一覧情報取得を実現できます。

FTP サーバへファイルを転送する場合を例に FTP 受付を使用した FTP 連携の流れを次の図に示します。

図 8-11　FTP 受付を使用した FTP 連携の流れ



8.4.1　FTP 受付でサポートするオペレーションと FTP コマンド

FTP 受付では，ファイルに対する操作の単位をオペレーションと呼びます。

FTP 受付でサポートするオペレーションには，次の種類があります。

PUT オペレーション

FTP クライアント側から FTP サーバ側へファイルを転送するオペレーションです。

GET オペレーション

FTP サーバ側から FTP クライアント側へファイルを転送するオペレーションです。

GETINFO オペレーション

FTP サーバ側のファイル一覧情報を取得して FTP クライアント側へ送付するオペレーションです。

FTP クライアントから FTP インバウンドアダプタに対して FTP コマンドが呼び出された場合に，これらのオペレーションが FTP インバウンドアダプタから呼び出されます。

注意事項

1 つのビジネスプロセスに対して複数の FTP 受付を関連づける場合，これらの FTP 受付にはすべて同じオペレーションを選択してください。異なるオペレーションの FTP 受付を 1 つのビジネスプロセスに複数定義した場合は，動作を保証しません。

(1) FTP 受付でサポートするオペレーションと FTP コマンドの種類

FTP 受付でサポートする FTP コマンドを次の表に示します。

表 8-1 FTP 受付でサポートする FTP コマンド

FTP コマンド	用途	呼び出される FTP 受付のオペレーション
STOR	ファイルの内容をサーバ側のファイルに蓄積します。すでにサーバにファイルが存在する場合，既存のファイルの内容は上書きされます。	PUT オペレーション
APPE	ファイルの内容をサーバ側のファイルに蓄積します。すでにサーバにファイルが存在する場合，転送したデータが既存のファイルに追加されます。	
RETR	サーバ側に存在するファイルの複製を取得します。	GET オペレーション
LIST	サーバ側に存在するファイルの情報の一覧を取得します。	GETINFO オペレーション
NLST	サーバ側に存在するファイルの名前の一覧を取得します。	
SITE CSCTHR	転送コマンド実行前，転送コマンド実行後，リストコマンド実行前，またはリストコマンド実行後にビジネスプロセスで実行する FTP コマンドを設定します。 FTP クライアントが転送コマンドまたはリストコマンドを実行する前にこのコマンドを実行しておくと，そのあとに実行した転送コマンドまたはリストコマンドの情報とともに，このコマンドで指定した情報がビジネスプロセスに渡されます。	－

（凡例）

－：呼び出されるオペレーションはありません。

(2) FTP コマンドの実行形式と注意事項

FTP コマンドの実行形式と注意事項を次に示します。

実行形式

```
<FTPコマンド> /<FTP受付の受付ID>/<リモートファイルパス>
```

注意事項

- FTP 受付の受付 ID の先頭に指定する記号およびリモートファイルパスに指定する区切り文字は必ずスラッシュ（/）を指定してください。
- 転送コマンドの引数の指定についての制限は FTP プロトコルの仕様に従ってください。

8.4.2　FTP 受付で使用できる通信モデルおよびビジネスプロセス

ここでは，FTP 受付で使用できる通信モデルおよびビジネスプロセスの型について説明します。

(1)　FTP 受付で使用できる通信モデル

FTP 受付で使用できる通信モデルは，同期通信です。

(2)　FTP 受付で使用できるビジネスプロセス

FTP 受付と関連づけられるビジネスプロセスは，同期型，および非同期型の 2 種類です。それぞれのビジネスプロセスの特徴を次の表に示します。

表 8–2　ビジネスプロセスの特徴

ビジネスプロセスの型	特徴	備考
同期型	FTP 受付から呼び出されたビジネスプロセスが，目的の処理を完了したあとに応答を返します。応答を返したあとに実行するアクティビティはありません。	オペレーションの種別やファイル連携方式に関係なく使用できます。
非同期型	FTP 受付から呼び出されたビジネスプロセスが，目的の処理を完了する前に応答を返します。応答を返したあとに実行するアクティビティはあります。	ビジネスプロセスのうち，非同期に実行される処理（FTP 受付に応答を返したあとに実行される処理）で，GET オペレーションおよび GETINFO オペレーションに関する処理は実行できません。

8.4.3　FTP 受付使用時のトランザクション

FTP インバウンドアダプタにはトランザクションで FTP 受付以降の処理に引き継ぐ情報がありません。そのため，FTP 受付では，FTP インバウンドアダプタのトランザクションとは異なる新たなトランザクションを開始します。ただし，ビジネスプロセス内ではさらに FTP 受付と異なる新たなトランザクションが開始されます。

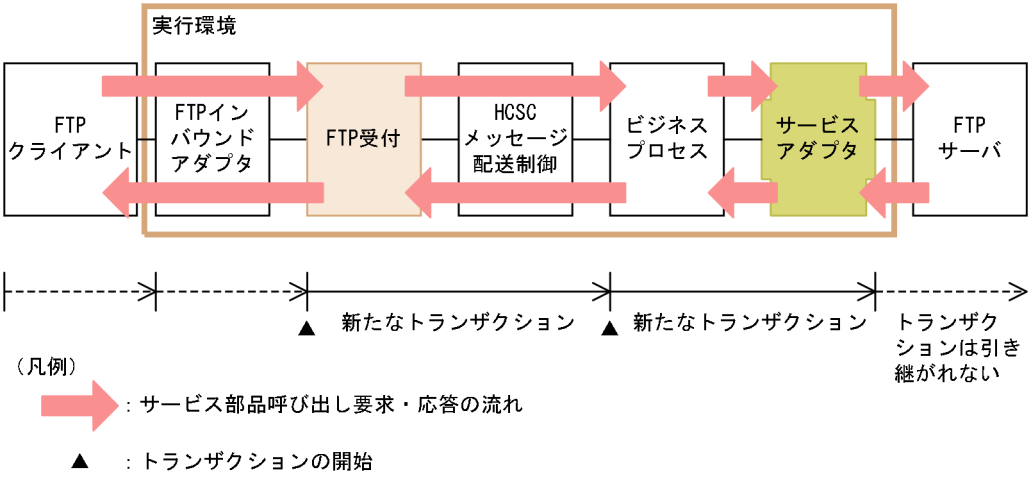
新たなトランザクションが開始されるタイミングは，FTP 受付のタイムアウトの監視をする場合としない場合とで異なります。

FTP 受付のタイムアウトの監視の詳細は，「8.4.9　タイムアウトの監視」を参照してください。

(1)　FTP 受付のトランザクション（タイムアウトの監視をしない場合）

FTP 受付のタイムアウトの監視をしない場合で，ビジネスプロセスを呼び出すときの FTP 受付のトランザクションを次に示します。

図 8-12　FTP 受付のトランザクション（タイムアウトの監視をしない場合）

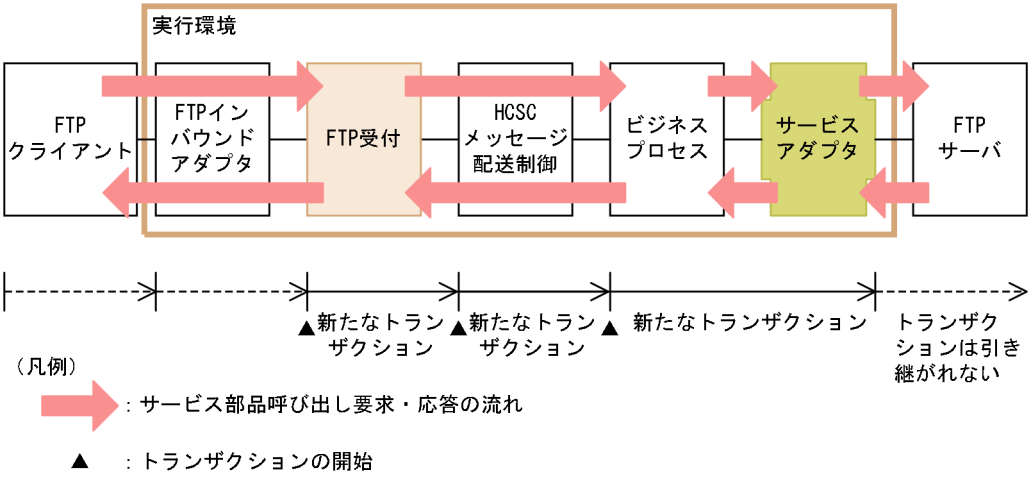


なお，ビジネスプロセス内で FTP アダプタを使用してサーバ側にファイルを転送する場合，サーバ側へのファイル転送が完了したあとにビジネスプロセスの処理でエラーが発生すると，トランザクションはロールバックされて，FTP クライアント側にエラーが返ります。ただし，サーバ側のファイルは転送後の状態のままとなり，転送前の状態に戻されません。

(2)　FTP 受付のトランザクション（タイムアウトの監視をする場合）

FTP 受付のタイムアウトの監視をする場合，異なるスレッドでビジネスプロセスが呼び出されるため，HCSC メッセージ配送制御で新たなトランザクションが開始します。FTP 受付のタイムアウトの監視をする場合で，ビジネスプロセスを呼び出すときの FTP 受付のトランザクションを次に示します。

図 8-13　FTP 受付のトランザクション（タイムアウトの監視をする場合）



HCSC メッセージ配送制御で開始するトランザクションは，J2EE サーバのトランザクションタイムアウトの値で監視されます。トランザクションタイムアウトの設定については，マニュアル「アプリケーションサーバ システム設計ガイド」の「8.6.4　トランザクションタイムアウトを設定する」を参照してください。

なお，FTP 受付のタイムアウト値を設定する場合，J2EE サーバのトランザクションタイムアウトの値よりも大きい値を設定することを推奨します。

8.4.4　リクエスト ID の生成

FTP 受付は，FTP クライアントからのリクエストを受け付けると，リクエストごとに**リクエスト ID** を生成します。ビジネスプロセスの処理は，生成したリクエスト ID を指定して実行されます。

リクエスト ID は次の形式で生成されます。

```
<受付ID><FTPクライアントのIPアドレス><日付><時刻><追番><分割キー><HCSCサーバ名>
```

リクエスト ID のそれぞれの項目について次の表に示します。

表 8-3　リクエスト ID の項目

項目	桁数	内容
<受付 ID>	8 桁	FTP 受付の設定で指定した受付 ID です。受付 ID が 8 桁に満たない場合は，受付 ID の右側に足りない分の「-」が設定されます。
<FTP クライアントの IP アドレス>	IPv4 の場合：12 桁 IPv6 の場合：32 桁	要求元の FTP クライアントの IP アドレスです。IPv4 アドレスのそれぞれのバイトが 3 桁に満たない場合は，それぞれのバイトの左側に足りない分の「0」が設定されます。IPv6 アドレスはそれぞれのバイトが 4 桁に満たない場合は，それぞれのバイトの左側に足りない分の「0」が設定されます。英字は小文字で表記されます。
<日付>	8 桁	FTP 受付がリクエストを受け付けた日付が yyyymmdd の形式で設定されます。 • yyyy：西暦年 • mm：月 • dd：日
<時刻>	9 桁	FTP 受付がリクエストを受け付けた時刻が hhmmssSSS の形式で設定されます。 • hh：時 • mm：分 • ss：秒 • SSS：ミリ秒

項目	桁数	内容
<追番>	3 桁	フォルダ名を一意にするための整数値が「000」からの連番で設定されます。<日付><時刻><追番>がすでにあるリクエスト ID と重複する場合は追番を 1 つ繰り上げます。 「001」から「999」までのどの整数値を設定しても重複する場合，<日付>および<時刻>を変更して追番には「000」を設定します。
<分割キー>	1 桁	HCSC サーバ名が付加されていることを示す分割キーです。半角スラッシュ（/），半角アットマーク（@）または半角ピリオドが使用されます。 HCSC サーバランタイム定義ファイルで変更可能です。 詳細は，マニュアル「サービスプラットフォーム リファレンス」の「6.5.6　HCSC サーバランタイム定義ファイル」を参照してください。
<HCSC サーバ名>	最大 8 桁	FTP 受付が配備されている HCSC サーバの名称です。 HCSC サーバ名が 8 桁に満たない場合，不足分は補われません。

8.4.5　作業フォルダと中間ファイルの作成

FTP 受付は，ビジネスプロセスとの間のファイルデータを受け渡すために，FTP クライアントから送られてきたファイルのデータやファイルの一覧情報を一時的に格納する中間ファイルを作成します。また，中間ファイルは，リクエストごとに FTP 受付が生成する作業フォルダに格納されます。

作業フォルダを作成するディレクトリは HCSC サーバランタイム定義ファイルの work-folder プロパティで設定してください。詳細は，マニュアル「サービスプラットフォーム リファレンス」の「6.5.6　HCSC サーバランタイム定義ファイル」を参照してください。

なお，作業フォルダと作業フォルダに格納された中間ファイルは，次の条件を満たした場合に削除されます。

- FTP 受付定義ファイルの urecp-ftp.work-dir.auto-delete プロパティの値が「true」である。
- FTP 受付がビジネスプロセスを呼び出して，呼び出したビジネスプロセスから，<success>タグの値が true の応答電文を受け取った。
- FTP 受付と FTP インバウンドアダプタの間のデータ（ファイルの内容やリストデータ）の受け渡しが正常に終了した。または，GETINFO オペレーションの応答電文の<reply-code>の値が 400～599 の値である。

上記の条件を満たしていない場合や，プロセスが異常終了した場合および cscpireexec コマンド（プロセスインスタンスの再実行）などでビジネスプロセスを再実行した場合は，作業フォルダおよび作業フォルダ内の中間ファイルが自動で削除されません。FTP 受付が作業フォルダを削除できなかった場合，次の方法で作業フォルダと中間ファイルを削除してください。

FTP 受付定義ファイルの urecp-ftp.work-dir.auto-delete プロパティの値が「false」の場合

ビジネスプロセスが実行する最後の処理でファイル操作アダプタを呼び出して作業フォルダと中間ファイルを削除してください。または，cscfswls コマンドで作業フォルダを参照し，削除対象かどうか確認したあと，cscfswrm コマンドで作業フォルダを削除してください。

FTP 受付定義ファイルの urecp-ftp.work-dir.auto-delete プロパティの値が「true」の場合

cscfswls コマンドで作業フォルダを参照し，削除対象かどうか確認したあと，cscfswrm コマンドで作業フォルダを削除してください。

詳細は，マニュアル「サービスプラットフォーム リファレンス」の「cscfswls（作業フォルダの参照)」または「cscfswrm（作業フォルダの削除)」を参照してください。

共通フォルダ内のファイルを中間ファイルとして使用する場合も，リクエストを受け付けた時点で作業フォルダが作成されます。自動で削除されなかった作業フォルダについては，cscfswrm コマンドで作業フォルダを削除してください。

8.4.6 ビジネスプロセスのアクセス制御

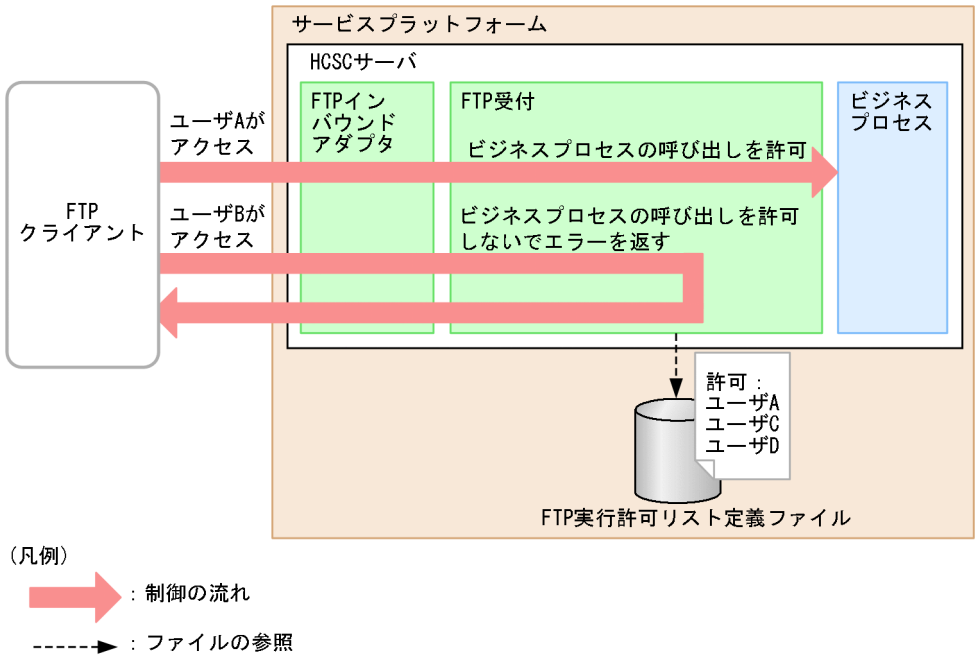
ここでは，FTP 受付の機能として，ビジネスプロセスのアクセス制御と，転送前または転送後に実行する FTP コマンドの管理について説明します。

FTP 受付は，ビジネスプロセスへのアクセスを許可および禁止するユーザを設定することで，ビジネスプロセスへのアクセスをユーザごとに制御できます。

ビジネスプロセスのアクセス制御の設定は，FTP 実行許可リスト定義ファイルで設定できます。FTP 受付は，FTP 実行許可リスト定義ファイルの設定内容を基に，FTP クライアントからアクセスしてきたユーザに対してビジネスプロセスのアクセスを許可するかどうかを判断して，許可する場合にだけビジネスプロセスを呼び出し，許可しない場合はビジネスプロセスを呼び出さずにエラーを返します。

ビジネスプロセスのアクセス制御の流れを次の図に示します。

図 8-14　ビジネスプロセスのアクセス制御の流れ



この図では，FTP 実行許可リスト定義ファイルで許可されているユーザ A からのリクエストはビジネスプロセスを呼び出していますが，FTP 実行許可リスト定義ファイルで許可されていないユーザ B のリクエストに対しては，ビジネスプロセスを呼び出さずにエラーを返しています。

FTP 実行許可リスト定義ファイルで設定する内容を次の表に示します。

表 8-4　FTP 実行許可リスト定義ファイルで設定する内容

内容	指定方法	デフォルト値
FTP 受付が関連づいたビジネスプロセスへのアクセスを許可するユーザ	urecp-ftp.permission.allow プロパティでユーザ名を指定します。 複数のユーザを指定する場合は，ユーザ名とユーザ名の間をコンマ（,）で区切ってください。 アスタリスク（*）を指定した場合は，すべてのユーザのアクセスを許可します。	*
FTP 受付が関連づいたビジネスプロセスのアクセスを許可しないユーザ	urecp-ftp.permission.deny プロパティでユーザ名を指定します。 複数のユーザを指定する場合は，ユーザ名とユーザ名の間をコンマ（,）で区切ってください。	なし

注

両方に同じユーザが指定されている場合，アクセスを許可しない設定が優先されます。

なお，FTP 実行許可リスト定義ファイルは，任意のファイル名で任意のフォルダに格納できますが，FTP 実行許可リスト定義ファイルの絶対パスを FTP 受付コンフィグファイルに指定する必要があります。FTP 受付コンフィグファイルに FTP 実行許可リスト定義ファイルの絶対パスが指定されていない場合は，すべてのユーザに対してビジネスプロセスへのアクセスが許可されます。

また，FTP 実行許可リスト定義ファイルの内容を変更した場合，変更した FTP 許可リスト定義ファイルを使用している FTP 受付を再起動する必要があります。

詳細は，マニュアル「サービスプラットフォーム リファレンス」の「6.8.7　FTP 実行許可リスト定義ファイル」または「6.8.5　FTP 受付コンフィグファイル」を参照してください。

8.4.7　転送前／転送後またはリスト前／リスト後に実行する FTP コマンドの管理

SITE CSCTHR コマンドを実行することによって，転送コマンド実行前，転送コマンド実行後，リストコマンド実行前，またはリストコマンド実行後にビジネスプロセスで実行する FTP コマンドを設定できます。FTP クライアントが転送コマンドまたはリストコマンドを実行する前に SITE CSCTHR コマンドを実行しておくと，そのあとに実行した FTP コマンドの情報とともに，SITE CSCTHR コマンドで指定した情報がビジネスプロセスに渡されます。

SITE CSCTHR コマンドの実行形式を次に示します。

実行形式

```
SITE CSCTHR {-B | -A} <FTPコマンド>[;<FTPコマンド> …]
```

FTP コマンド以外の，FTP サーバが独自にサポートしているコマンドを指定する場合は，次のように<FTP コマンド>の部分に SITE コマンドを指定して，SITE コマンドの引数にコマンドを指定してください。

```
SITE CSCTHR {-B | -A} SITE <FTPサーバが独自にサポートしているコマンド>
```

また，オプションに-B，または-A を指定することで，指定した FTP コマンドを転送コマンド実行前／リストコマンド実行前に実行するか，または転送コマンド実行後／リストコマンド実行後に実行するかを設定できます。

-B オプションを使用した場合の動作

「<FTP コマンド>[;<FTP コマンド> …]」の部分に指定した FTP コマンドが，FTP 受付の要求電文の<ftp-commands-before>タグに設定されます。この形式で指定した FTP コマンドは，転送コマンド実行前またはリストコマンド実行前に実行されます。SITE CSCTHR コマンドをこの形式で複数回実行した場合，「<FTP コマンド>[;<FTP コマンド> …]」の部分が SITE CSCTHR コマンドを実行した順番にセミコロン（;）で接続されて<ftp-commands-before>タグに追加されます。

-A オプションを使用した場合の動作

「<FTP コマンド>[;<FTP コマンド> …]」の部分に指定した FTP コマンドが，FTP 受付の要求電文の<ftp-commands-after>タグに設定されます。この形式で指定した FTP コマンドは，転送コマンド実行後またはリストコマンド実行後に実行されます。SITE CSCTHR コマンドをこの形式で複数回実行した場合，「<FTP コマンド>[;<FTP コマンド> …]」の部分が SITE CSCTHR コマンドを実行した順番にセミコロン（;）で接続されて<ftp-commands-after>タグに追加されます。

また，転送コマンド実行前，転送コマンド実行後，リストコマンド実行前，またはリストコマンド実行後にビジネスプロセスで実行を許可するコマンドは，FTP コマンド許可リスト定義ファイル（FTP 受付用）に指定して管理できます。FTP コマンド許可リスト定義ファイル（FTP 受付用）に指定されていない FTP

コマンドが SITE CSCTHR コマンドに含まれていても，FTP 受付の要求電文の<ftp-commands-before>タグ，または<ftp-commands-after>タグに追加されません。その場合，ビジネスプロセスが呼び出されないため，エラーが発生します。

なお，FTP コマンド許可リスト定義ファイル（FTP 受付用）は，任意のファイル名で任意のフォルダに格納できますが，FTP コマンド許可リスト定義ファイル（FTP 受付用）の絶対パスを FTP 受付コンフィグファイルに指定する必要があります。FTP 受付コンフィグファイルに FTP コマンド許可リスト定義ファイル（FTP 受付用）の絶対パスが指定されていない場合は，すべての FTP コマンドに対してビジネスプロセスでの実行が禁止されます。

また FTP コマンド許可リスト定義ファイル（FTP 受付用）の内容を変更した場合，変更した FTP コマンド許可リスト定義ファイル（FTP 受付用）を使用している FTP 受付を再起動する必要があります。

FTP コマンド許可リスト定義ファイル（FTP 受付用）で編集できる定義内容については，マニュアル「サービスプラットフォーム リファレンス」の「6.8.6　FTP コマンド許可リスト定義ファイル」を参照してください。

8.4.8　リストコマンドオプションの設定による LIST／NLST コマンドの実行

LIST コマンドまたは NLST コマンドの引数オプションをリストコマンドオプション定義ファイルで定義しておくことで，引数オプションを指定してリストコマンドを実行できます。

引数オプションを指定した場合のコマンドの実行形式を次に示します。

- LIST コマンドの場合

```
LIST [ <オプション>][ <パス>]
```

- NLST コマンドの場合

```
NLST [ <オプション>][ <パス>]
```

リストコマンドオプション定義ファイルの詳細は，マニュアル「サービスプラットフォーム リファレンス」の「6.8.9　リストコマンドオプション定義ファイル」を参照してください。

リストコマンドオプション定義ファイルは，任意のパスに任意のファイル名で保持できます。ただし，リストコマンドオプション定義ファイルの絶対パスが FTP 受付コンフィグファイルで指定されている必要があります。

FTP 受付コンフィグファイルについては，マニュアル「サービスプラットフォーム リファレンス」の「6.8.5　FTP 受付コンフィグファイル」を参照してください。

リストコマンドオプション定義ファイルに引数オプションを指定した場合，FTP クライアントからリストコマンドが実行されたとき，リストコマンドの引数に指定された情報がリストコマンドオプション定義ファイルの定義に従って<オプション>と<パス>に分割され，GETINFO オペレーションの要求電文の<getinfo-option>と<getinfo-path>に設定されます。

リストコマンドオプション定義ファイルに引数オプションの指定がない場合，またはFTP受付コンフィグファイルでリストコマンドオプション定義ファイルのパスが指定されていない場合，リストコマンドの引数で指定された情報は，すべて<パス>の情報として要求電文の<getinfo-path>に設定されます。

リストコマンドオプション定義ファイルの設定を変更した場合，設定を有効にするには，リストコマンドオプション定義ファイルに関連づけられたFTP受付を再起動する必要があります。

8.4.9　タイムアウトの監視

FTP受付コンフィグファイルのurecp-ftp.request-timeoutにタイムアウト値を設定することで，ビジネスプロセスに要求電文を渡してから応答電文が返ってくるまでの処理時間を監視できます。設定した時間を経過しても応答が返ってこない場合，障害としてタイムアウト例外が発生します。

なお，デフォルトではFTP受付のタイムアウトの監視をしません。

FTP受付でタイムアウトを監視するとき，HCSCメッセージ配送制御内部にFTP受付単位でスレッドが生成されます。タイムアウトの監視に使われるスレッドのうち，受付起動時に生成されて常駐するスレッドの数（常駐スレッド数）は，FTP受付コンフィグファイルのurecp-ftp.resident-thread.countプロパティで設定できます。

常駐スレッド数を超える数のタスクを受け付けた場合，タスクを処理する際にスレッドが一時的に追加生成されます。追加生成されたスレッドは，タスク処理後に300秒間待機状態となり，次のタスクが来なければ破棄されます。

FTP受付のタイムアウトを監視するために生成されるスレッドの最大数（常駐スレッド数＋追加生成されたスレッド数）は，FTP受付コンフィグファイルのurecp-ftp.timer-thread.maximumプロパティで設定できます。なお，常駐スレッド数と最大スレッド数を同じ値に設定した場合，スレッドは追加生成されません。

FTP受付コンフィグファイルで設定できるプロパティの詳細は，マニュアル「サービスプラットフォームリファレンス」の「6.8.5　FTP受付コンフィグファイル」を参照してください。

8.4.10　共通フォルダと中間ファイルの作成

FTP受付は，共通フォルダ内のファイルを中間ファイルとして使用することができます。

(1)　共通フォルダにアクセスできるオペレーション

共通フォルダ内のファイルにアクセスできるのは，GETオペレーションだけです。PUTオペレーションおよびGETINFOオペレーションは共通フォルダにアクセスできません。

(2) 共通フォルダにアクセスする方法

FTP 受付は，次の条件を満たした場合に，共通フォルダ内のファイルを中間ファイルとして使用できます。

- GET オペレーションの応答電文の<folder-name>の値に共通フォルダ定義名が設定されている。
- GET オペレーションの応答電文の<file-name>の値に共通フォルダ内のファイルが設定されている。
- GET オペレーションの応答電文の<success>の値に true が設定されている。

(3) 中間ファイルとして指定可能なファイル

中間ファイルとして指定できるファイルを次に示します。

- 共通フォルダ内のファイルであること。フォルダを指定することはできません。
- 共通フォルダからの相対パスが指定できるが，共通フォルダの外部のファイルを指定することはできません。なお，パスの先頭のスラッシュ（/）は省略することができます。
- シンボリックリンクのファイルでないこと。

(4) 共通フォルダ内のファイルの削除

共通フォルダ内のファイルを中間ファイルとして使用した場合，FTP 受付でファイルを削除することができます。ファイルが削除される条件を次に示します。

- 共通フォルダ内のファイルを中間ファイルとして使用した。
- FTP 受付定義ファイルの urecp-ftp.common-dir-file.auto-delete プロパティの値が「true」である。
- FTP 受付がビジネスプロセスを呼び出して，呼び出したビジネスプロセスから，<success>タグの値が true の応答電文を受け取った。
- FTP 受付と FTP インバウンドアダプタの間のデータ（ファイルの内容やリストデータ）の受け渡しが正常に終了した。

上記の条件を満たしていない場合や，プロセスが異常終了した場合および cscpireexec コマンド（プロセスインスタンスの再実行）などでビジネスプロセスを再実行した場合は，共通フォルダ内の中間ファイルが自動で削除されません。

上記の条件を満たしていない場合や，プロセスが異常終了した場合およびビジネスプロセスの再実行を要求した場合（cscpireexec コマンドなど）は，共通フォルダ内の中間ファイルが自動で削除されません。FTP 受付が共通フォルダ内の中間ファイルを削除できなかった場合，次の方法で共通フォルダ内の中間ファイルを削除してください。

FTP 受付定義ファイルの urecp-ftp.common-dir-file.auto-delete プロパティの値が「false」の場合

ビジネスプロセスが実行する最後の処理でファイル操作アダプタを呼び出して共通フォルダ内の中間ファイルを削除してください。または，cscfscls コマンドで共通フォルダを参照し，削除対象かどうか確認したあと，cscfscrm コマンドで共通フォルダ内の中間ファイルを削除してください。

FTP 受付定義ファイルの urecp-ftp.common-dir-file.auto-delete プロパティの値が「true」の場合

cscfscls コマンドで共通フォルダを参照し，削除対象かどうか確認したあと，cscfscrm コマンドで共通フォルダ内の中間ファイルを削除してください。

詳細は，マニュアル「サービスプラットフォーム リファレンス」の「cscfscls（共通フォルダの参照)」または「cscfscrm（共通フォルダの削除)」を参照してください。

(5) 共通フォルダ内のファイルの排他制御

GET オペレーションで共通フォルダ内のファイルへアクセスする場合，ファイルの読み込み時に共有ロックを取得します。また共通フォルダのファイルを削除する場合，削除するファイルに対して占有ロックを取得します。ファイルロック取得時のロックリトライ回数およびロックリトライ間隔は，FTP 受付コンフィグファイルのプロパティで変更できます。

FTP 受付コンフィグファイルの詳細は，マニュアル「サービスプラットフォーム リファレンス」の「6.8.5 FTP 受付コンフィグファイル」を参照してください。また排他制御については，「8.8.2 共通フォルダを利用したファイルアクセス」を参照してください。

8.4.11 リストコマンドパスのデータ抽出機能

リストコマンドの引数のパス情報は，GETINFO オペレーションの要求電文の<getinfo-path>に設定されます。しかし，特定の情報（FTP 受付の受付 ID など）をパス情報で渡したい場合，データ変換処理が複雑になります。その場合にリストコマンドパスのデータ抽出機能を使用すると，パス情報に設定した特定の情報を簡単に抜き出してビジネスプロセスに渡すことができます。

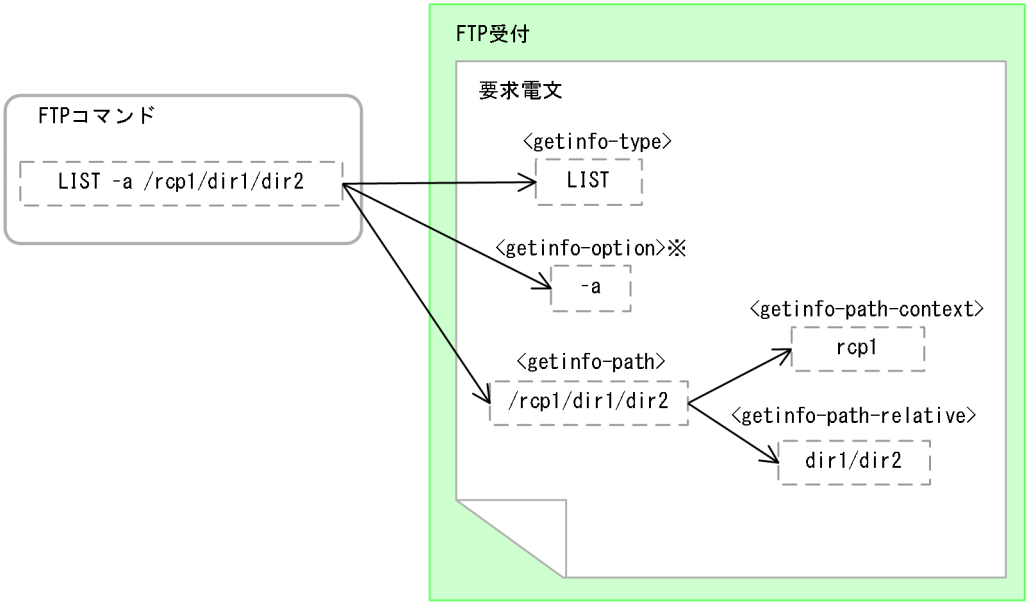
(1) データ抽出の前提条件

LIST コマンドまたは NLST コマンドの引数のパス情報が次の形式と一致する場合，パスの部分文字列を抜き出し，GETINFO オペレーションの要求電文の<getinfo-path-context>と<getinfo-path-relative>に設定することができます。なお，リストコマンドのパス情報は，GETINFO オペレーションの要求電文の<getinfo-path>の値を指します。

- パスの形式

  /<1 バイト以上の値>/<1 バイト以上の値>

図 8-15　GETINFO オペレーションの要求電文の生成イメージ



（凡例）

⟶：FTPコマンドのデータの流れ

注※

<getinfo-option>を生成する場合は，リストコマンドオプション定義ファイルの登録が必要です。

(2)　<getinfo-path-context>の生成

(a)　設定される値

パス情報の先頭のスラッシュ（/）から次のスラッシュ（/）より前の部分が設定されます。このタグにスラッシュ（/）は含まれません。

(b)　タグの生成される条件

次のすべての条件を満たしている場合に生成されます。条件に合致しない場合はエラーおよび警告メッセージが出力されません。

- FTP 受付定義ファイルの urecp-ftp.list-command-path-extraction プロパティの値が true である。
- <getinfo-path>が GETINFO オペレーションの要求電文に出力されている。
- <getinfo-path>の先頭がスラッシュ（/）である。
- <getinfo-path>に 2 つ目のスラッシュ（/）が存在する。
- <getinfo-path>の先頭のスラッシュ（/）と 2 つ目のスラッシュ（/）の間に 1 バイト以上の値が存在する。

FTP 受付定義ファイルの詳細は，マニュアル「サービスプラットフォーム リファレンス」の「3.5.1　FTP 受付定義ファイル」を参照してください。

(3) <getinfo-path-relative>の生成

(a) 設定される値

パス情報の 2 つ目のスラッシュ（/）より後ろの値が設定されます。

(b) タグの生成される条件

次のすべての条件を満たしている場合に生成されます。条件に合致しない場合はエラーおよび警告メッセージが出力されません。

- FTP 受付定義ファイルの urecp-ftp.list-command-path-extraction プロパティの値が true である。
- <getinfo-path-context>が GETINFO オペレーションの要求電文に出力されている。
- <getinfo-path>に 2 つ目のスラッシュ（/）より後ろの値が存在する。

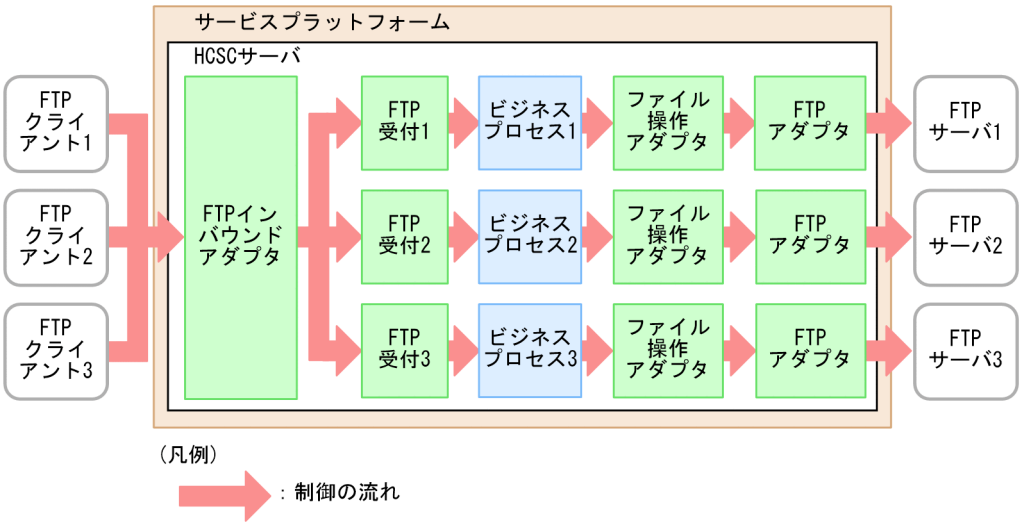
FTP 受付定義ファイルの詳細は，マニュアル「サービスプラットフォーム リファレンス」の「3.5.1　FTP 受付定義ファイル」を参照してください。

8.5　FTP インバウンドアダプタの機能

FTP インバウンドアダプタとは，FTP クライアントからの転送要求を受信して，ビジネスプロセスに関連づけられた FTP 受付に振り分けるアダプタです。

FTP 連携での FTP インバウンドアダプタの位置づけを次の図に示します。

図 8-16　FTP 連携での FTP インバウンドアダプタの位置づけ



ここでは，FTP インバウンドアダプタでサポートする次の機能について説明します。

- 送受信後のファイルサイズ確認
- タイムアウトの監視
- セキュリティに関する機能

FTP インバウンドアダプタのセキュリティに関する機能および設定方法については，マニュアル「サービスプラットフォーム 開発ガイド 受付・アダプタ定義編」の「付録 H.2　セキュアプロトコルによる暗号化・認証（FTP インバウンドアダプタ）」を参照してください。

8.5.1　送受信後のファイルサイズ確認

FTP インバウンドアダプタでは，JP1/File Transmission Server/FTP と連携することによって，転送前と転送後のファイルのサイズを比較します。FTP クライアントから SITE FSIZE コマンドが実行されると，ファイルが転送処理の間に破損していないかなどの確認がされるため，送受信されたファイルの信頼性を確保できます。

JP1/File Transmission Server/FTP については，マニュアル「JP1/File Transmission Server」を参照してください。

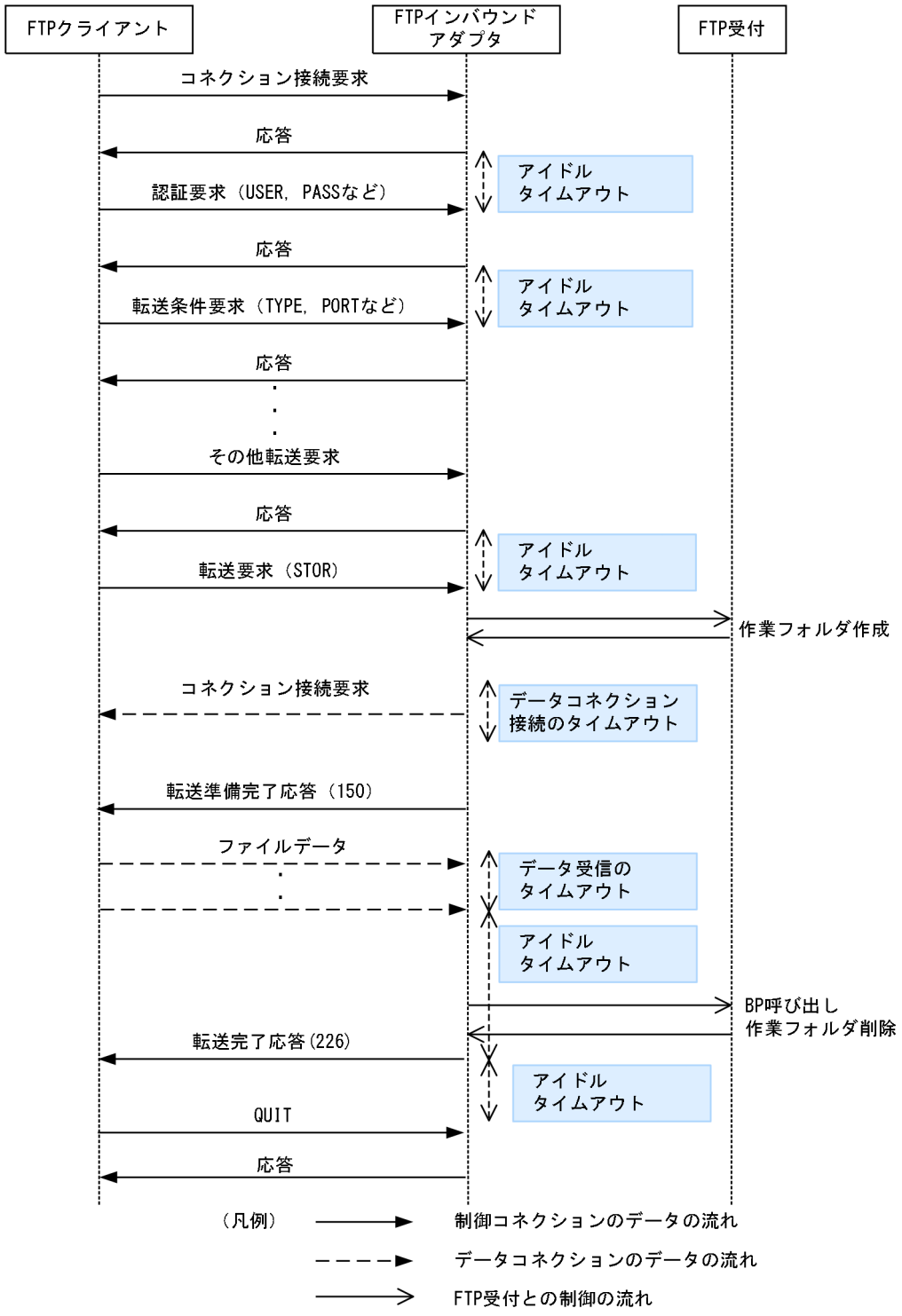
8.5.2　タイムアウトの監視

FTP インバウンドアダプタと FTP クライアントとの通信中や，FTP 受付への振り分け中に，FTP クライアントやネットワークの異常によって通信処理が無応答になったり，FTP 受付以降のファイル操作処理に時間が掛かってしまったりしている場合に，FTP インバウンドアダプタでタイムアウトを検知します。

タイムアウトが検知された場合，メッセージが出力されたりコネクションが切断されたりして，システムの処理能力の低下を防止します。

FTP インバウンドアダプタでタイムアウトを監視するポイントを，Active モードおよび STOR によるファイル転送の場合を例に，次の図に示します。

図 8-17　FTP インバウンドアダプタでタイムアウトを監視するポイント（Active モードおよび STOR によるファイル転送の場合の例）



(1)　タイムアウトしたときの動作

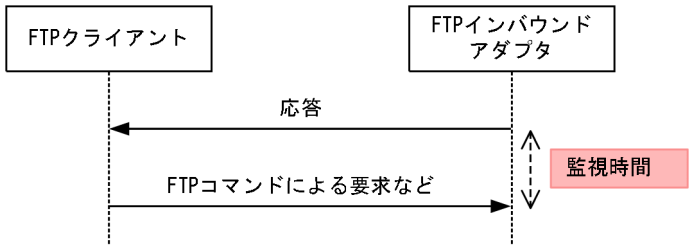
それぞれのポイントでの，タイムアウトしたときの動作，および注意事項について説明します。

(a) アイドルタイムアウト

制御コネクションに対して，FTP インバウンドアダプタから応答を返してから，FTP クライアントから次の FTP コマンドが送信されてくるまでの時間を監視します。

アイドルタイムアウトの監視の流れを次の図に示します。

図 8-18　アイドルタイムアウトの監視の流れ



次に，タイムアウトしたときの動作および注意事項を示します。

タイムアウトしたときの動作

- メッセージを出力します。
- タイムアウトが発生した制御コネクションを切断します。

注意事項

データ送受信中のアイドルタイムアウトの監視範囲は次のとおりです。

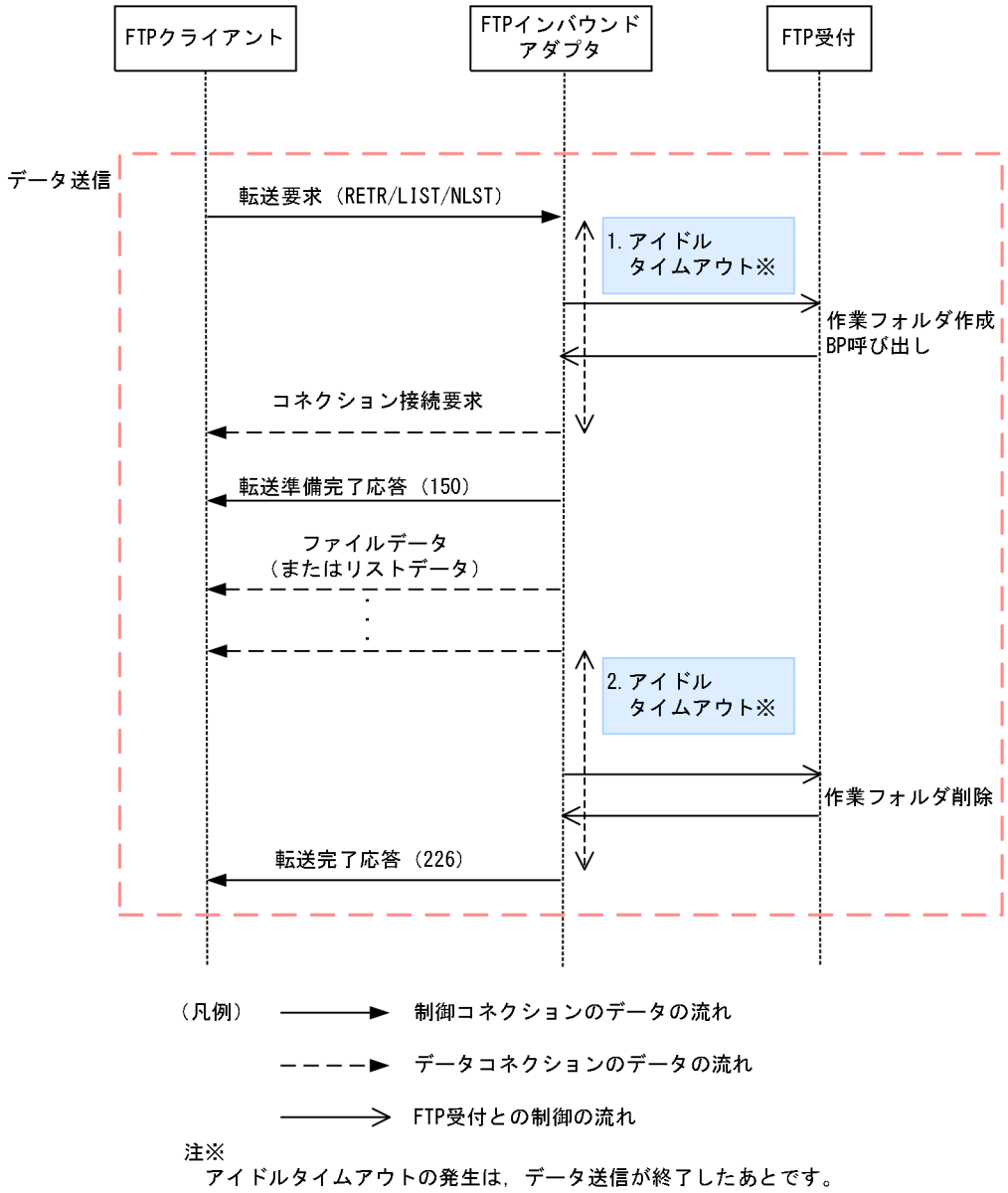
1. FTP クライアントから転送要求を受信してから，FTP 受付に制御が移り，FTP インバウンドアダプタがコネクション接続要求を完了するまでの間
2. FTP クライアントとのファイルデータ（またはリストデータ）の送受信が完了してから，FTP 受付に制御が移り，FTP インバウンドアダプタが転送完了応答を返すまでの間

上記 1.および 2.の合計時間がアイドルタイムアウトの監視時間を超えると，アイドルタイムアウトが発生します。ただし，アイドルタイムアウトの発生は，データ送受信が終了したあと（転送完了応答を送信したあと）に発生します。

なお，コネクション接続要求が完了してから，ファイルデータ（またはリストデータ）の送受信が完了するまでの間は，アイドルタイムアウトの監視をしません。

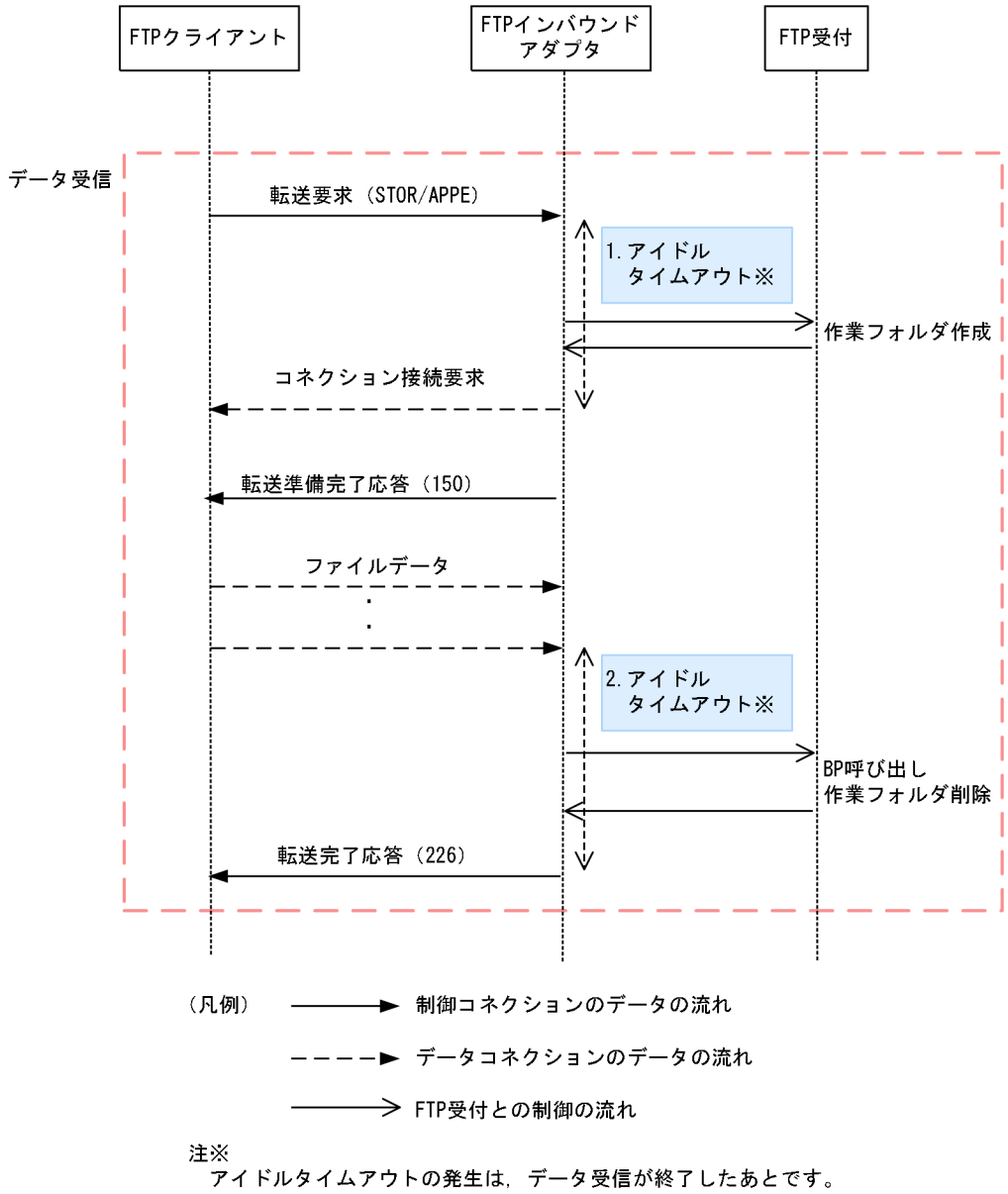
データ送信中のアイドルタイムアウト，およびデータ受信中のアイドルタイムアウトについて，次の図に示します。

図 8-19　データ送信中のアイドルタイムアウト



注※
アイドルタイムアウトの発生は，データ送信が終了したあとです。

図 8-20　データ受信中のアイドルタイムアウト

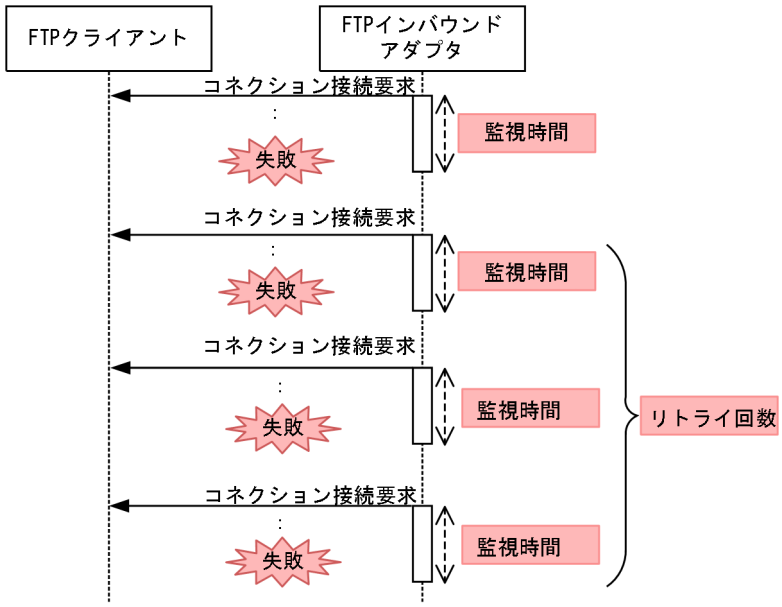


注※
アイドルタイムアウトの発生は，データ受信が終了したあとです。

(b)　データコネクション接続のタイムアウト

データコネクションの接続方式がアクティブモードの場合，FTP クライアントへのデータコネクションの接続を開始してから完了するまでの時間を監視します。時間の監視は，コネクション接続の監視時間，およびリトライ回数で指定します。データコネクション接続のタイムアウトの監視の流れを次の図に示します。

図 8-21　データコネクション接続のタイムアウトの監視の流れ



次に，タイムアウトまたはデータコネクション接続でエラーが発生したときの動作および注意事項を示します。

タイムアウトしたとき，またはデータコネクション接続でエラーが発生したときの動作

- メッセージを出力します。
- 指定したリトライ回数に達していない場合，データコネクション接続をリトライします。
- 指定したリトライ回数に達した場合，制御コネクションでレスポンスコード「425 Can't open data connection.」を返します。

注意事項

指定した監視時間より，TCP/IP で設定されたコネクション接続の監視時間が短い場合は，TCP/IP で設定された監視時間でタイムアウトが発生します。また，データコネクションの接続でエラーが発生した場合は，監視時間を待たないで，すぐにエラーとなります。

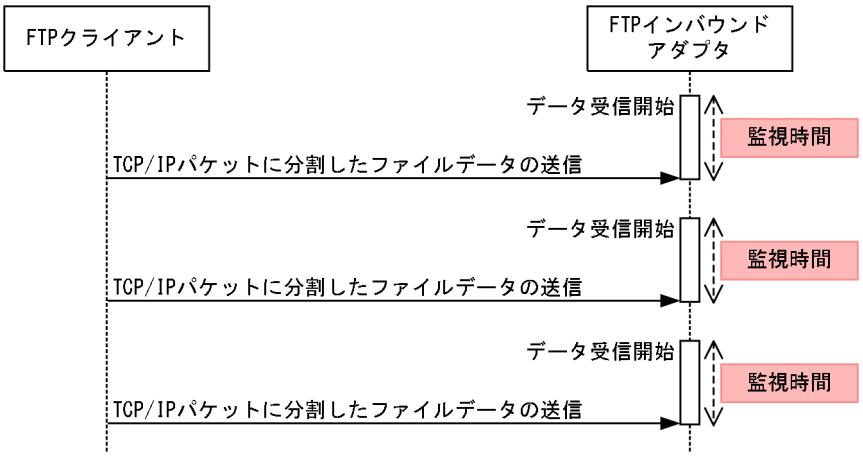
TCP/IP で設定された監視時間でタイムアウトが発生した場合，およびデータコネクションの接続でエラーが発生した場合は，指定したリトライ回数だけリトライします。リトライ回数を超えたときは，データコネクションの接続処理を終了します。

(c)　データ受信のタイムアウト

FTP クライアントから TCP/IP パケットに分割したファイルデータが受信できる状態となってから，TCP/IP パケットに分割したファイルデータが送信されてくるまでの時間を監視します。

データ受信のタイムアウトの監視の流れを次の図に示します。

図 8-22　データ受信のタイムアウトの監視の流れ



次に，タイムアウトしたときの動作を示します。

タイムアウトしたときの動作

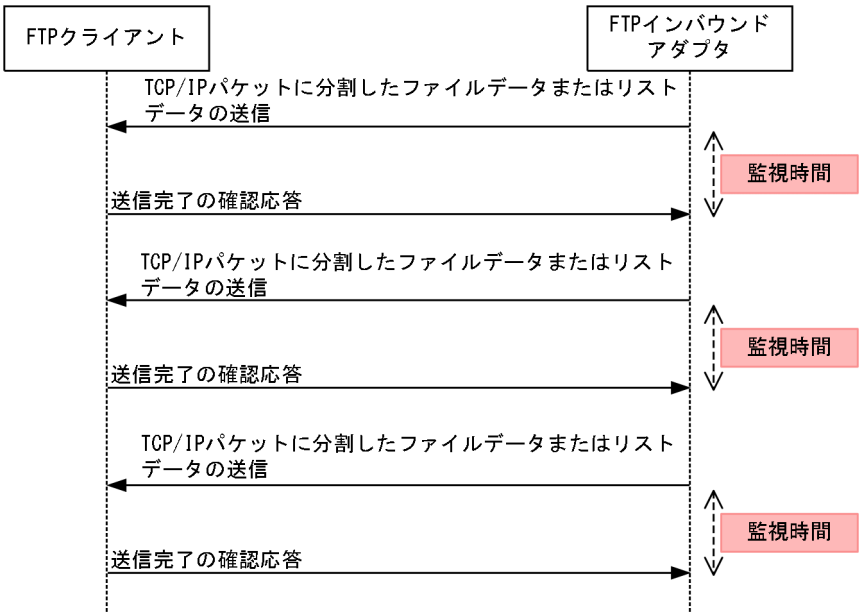
- メッセージを出力します。
- タイムアウトが発生したデータコネクションを切断します。
- 制御コネクションでレスポンスコード「551 <ファイルパス>: Error on output file.」を返します。
- FTP 受付に例外「FileTransferFailedException」を通知します。

(d)　データ送信のタイムアウト

FTP クライアントへ TCP/IP パケットに分割したファイルデータまたはリストデータの送信を開始してから，TCP/IP パケットに分割したファイルデータまたはリストデータの送信完了の確認応答までの時間を監視します。

データ送信のタイムアウトの監視の流れを次の図に示します。

図 8-23　データ送信のタイムアウトの監視の流れ



次に，タイムアウトしたときの動作を示します。

タイムアウトしたときの動作

- メッセージを出力します。
- タイムアウトが発生したデータコネクションを切断します。
- 制御コネクションで次のレスポンスコードを返します。
  ・ファイルデータ転送の場合
  「551 <ファイルパス>: Error on input file.」を返します。
  ・リストデータ転送の場合
  「551 Requested action aborted.」を返します。
- FTP 受付に例外「FileTransferFailedException」を通知します。

(2)　設定方法

cjgetrarprop コマンドで取得した connector 属性ファイルの次の項目を設定してください。

- nioListener_idleTimeout（アイドルタイムアウト）
- nioListener_dataConnection_active_connectRetryCount（データコネクション接続のリトライ回数）
- nioListener_dataConnection_active_connectRetryInterval（データコネクション接続の監視時間）
- nioListener_dataConnection_idleTimeout（データ送信およびデータ受信のタイムアウト）

設定方法の詳細は，マニュアル「サービスプラットフォーム システム構築・運用ガイド」の「3.2.3　FTP インバウンドアダプタをセットアップする」を参照してください。

8.6　FTP アダプタの機能

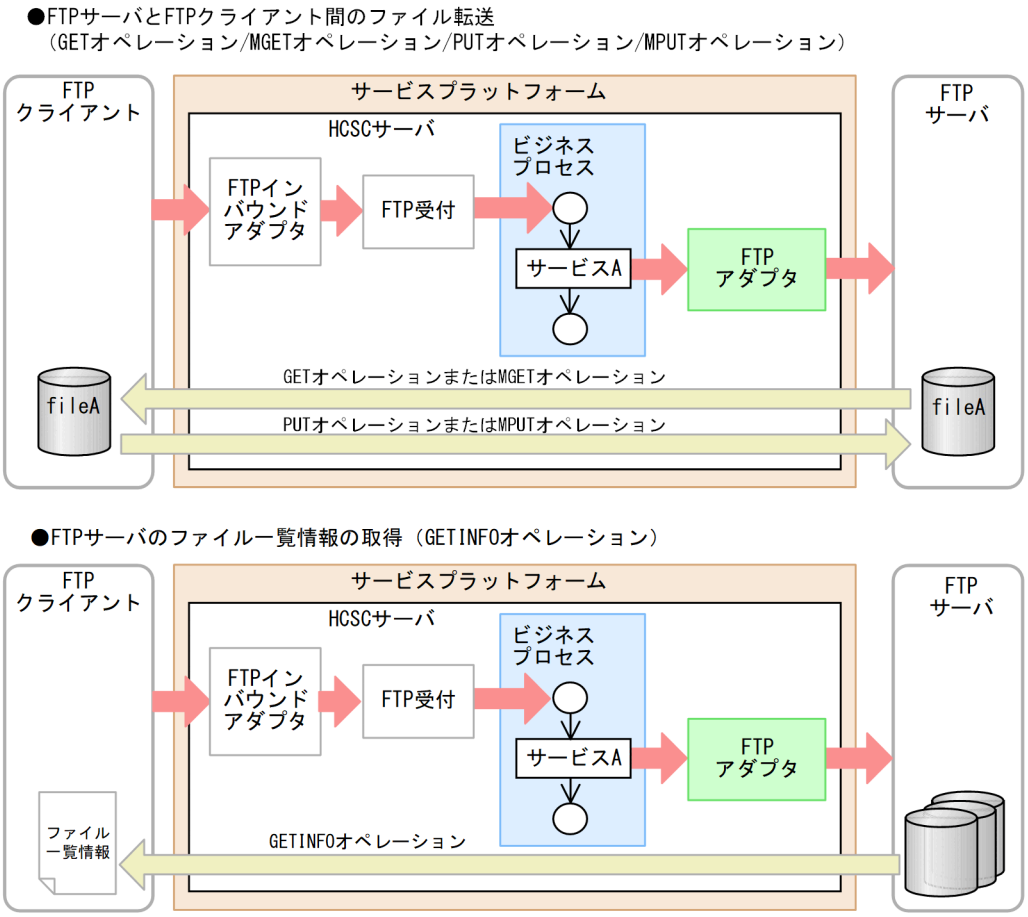
ここでは，FTP アダプタの役割，および FTP アダプタを利用したファイル送受信の流れについて説明します。

FTP アダプタとは，FTP サーバとサービスプラットフォームとの間でファイルを送受信するためのサービスアダプタです。また，FTP サーバのファイル一覧情報の取得，ファイル名の変更，ログインセッションの維持も FTP アダプタで実施できます。

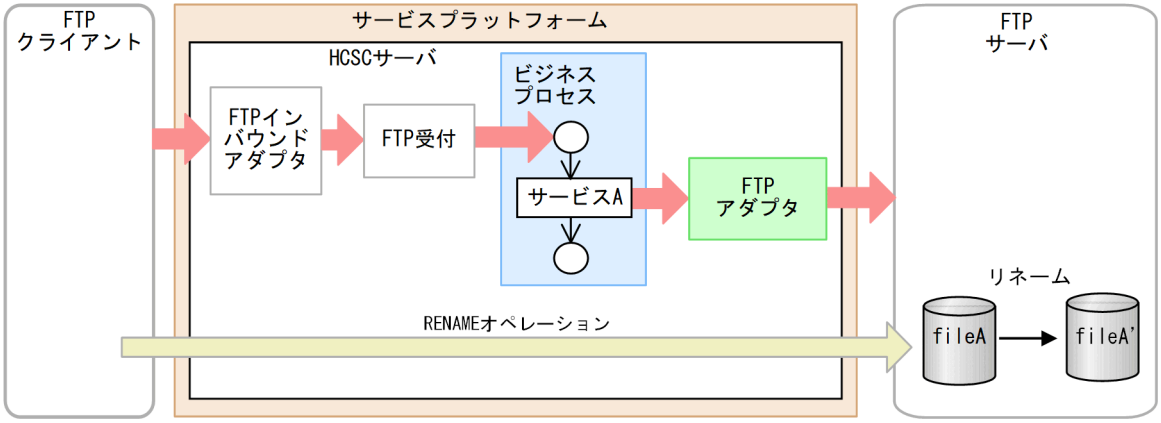
FTP 連携を利用したシステムでは，FTP クライアントと FTP サーバの間でファイルを送受信するために，接続する FTP サーバごとに FTP アダプタが必要になります。

FTP アダプタを利用したファイル送受信の流れを次の図に示します。

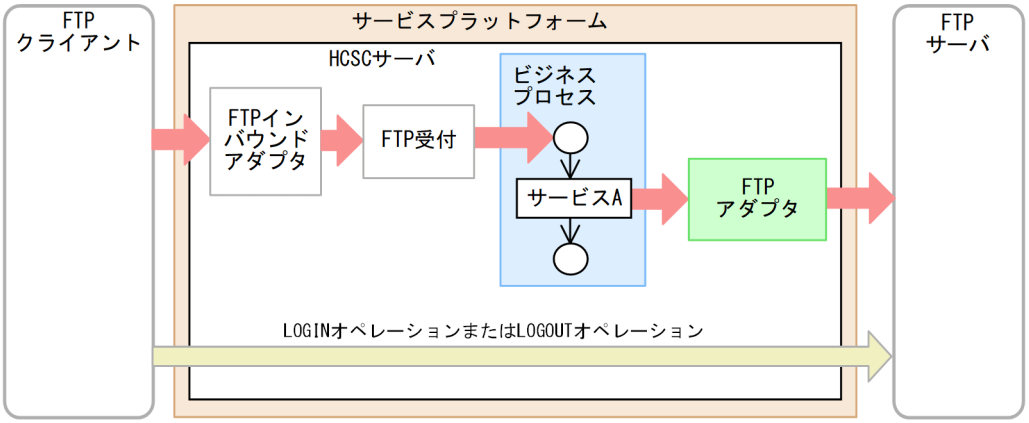
図 8-24　FTP アダプタを利用したファイルの送受信の流れ

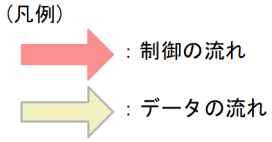


●リモートファイルのファイル名変更（RENAMEオペレーション）



●ログインセッションの確立と切断（LOGINオペレーション/ LOGOUTオペレーション）





図中の制御およびデータの流れについて説明します。

FTP サーバから FTP クライアントへのファイル転送の場合（GET オペレーションまたは MGET オペレーション）

FTP クライアントからファイル取得コマンドを実行すると，コマンドの実行要求が FTP インバウンドアダプタを経由して FTP 受付に届けられます。そのあとビジネスプロセスが開始され，ビジネスプロセスから FTP アダプタが呼び出されます。

ビジネスプロセスから実行要求を受け取った FTP アダプタは，FTP サーバと接続して FTP クライアントにファイルを送信します。

MGET オペレーションを利用した場合，複数のファイルを取得できます。

FTP クライアントから FTP サーバへのファイル転送の場合（PUT オペレーションまたは MPUT オペレーション）

FTP クライアントからファイル送信コマンドを実行すると，GET オペレーションの場合と同様に，コマンドの実行要求が FTP アダプタに届けられます。そのあと，FTP アダプタは，FTP サーバと接続して FTP クライアントから FTP サーバにファイルを送信します。

MPUT オペレーションを利用した場合，複数のファイルを転送できます。

FTP サーバのファイル一覧情報を取得して FTP クライアント側へ送付する場合（GETINFO オペレーション）

FTP クライアントから情報取得コマンド（LIST コマンドまたは NLST コマンド）を実行すると，実行したコマンドに応じて FTP サーバ側に存在するファイルの情報の一覧またはファイルの名前の一覧を取得します。

リモートファイルのファイル名を変更する場合（RENAME オペレーション）

FTP クライアントからファイルのリネームコマンド（RNFR コマンドおよび RNTO コマンド）を実行すると，コマンドで指定したファイル名を変更します。

FTP サーバへのログインセッションを確立する場合（LOGIN オペレーション）

FTP サーバに接続し，ログインセッションを確立します。ほかのオペレーション実行時に，FTP サーバへのログイン処理を省略できるため，処理速度が向上します。

FTP サーバのログインセッションを切断する場合（LOGOUT オペレーション）

維持しているログインセッションに QUIT コマンドを発行し，FTP サーバのログインセッションを切断します。

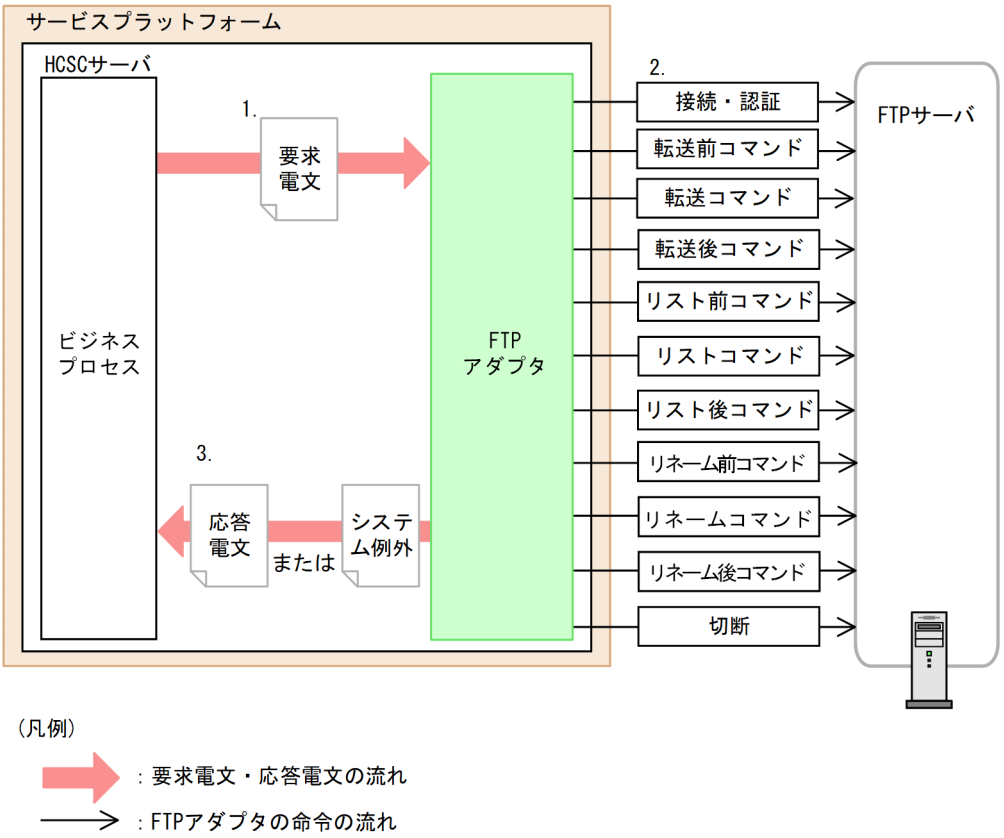
FTP 連携システム全体のファイル送受信の流れについては，「8.3　FTP 連携を利用したファイル転送パターンの例」を参照してください。

8.6.1　FTP アダプタの処理の流れ

FTP アダプタを中心としたファイル送受信時の処理の流れについて説明します。

ビジネスプロセスから実行要求を受けたときの FTP アダプタの処理の流れを次の図に示します。

図 8-25　FTP アダプタの処理の流れ



1. ビジネスプロセスから実行要求を受けて FTP アダプタが呼び出されます。このとき，ファイルの転送情報は要求電文に格納されて FTP アダプタに渡されます。
2. 要求電文を受け取った FTP アダプタは，FTP サーバに対して接続・認証，転送コマンド，リストコマンド，リネームコマンド，切断などの処理を実行します。
3. FTP サーバの呼び出しが完了したあと，FTP アダプタは結果を応答電文に設定し，ビジネスプロセスに返します。

   エラーが発生した場合は，応答電文の代わりにシステム例外を返します。

8.6.2　FTP コマンドの実行

ここでは，FTP アダプタの機能として，FTP アダプタが実行する FTP コマンドの種類，実行条件，およびタイマ管理などについて説明します。

FTP アダプタは，ビジネスプロセスから実行要求を受けたあと，条件に応じて FTP サーバに認証・接続や転送コマンドなどの FTP コマンドを実行します。

FTP アダプタが実行する FTP コマンドの種類，および実行条件を次の表に示します。

表 8-5　FTP コマンドの種類とコマンドの実行可否

分類	FTP コマンド	コマンドの実行可否
接続・認証	（connect 処理）	◎
	AUTH	△※1
	USER	◎※2
	PASS	◎※2
	ACCT	△※3
保護レベル	PBSZ	△※4
	PROT	△※4
転送パラメタ	TYPE	△※3
	MODE	△※3
	PASV，PORT，EPSV，EPRT	○※5
ファイルサイズ確認	SITE FSIZE	△※6
転送前コマンド・転送後コマンド， リスト前コマンド・リスト後コマンド，ファイルのリネーム前コマンド・ファイルのリネーム後コマンド※7	MDTM，SIZE，STAT，DELE，MKD，XMKD，RMD，XRMD，PWD，XPWD，CDUP，XCUP，CWD，XCWD，NOOP，SITE，RNFR，RNTO，ALLO，OPTS，STRU，SYST	△※8
転送または情報取得	STOR，APPE，RETR，LIST，NLST	○※3
	ABOR	△※9
ファイルのリネーム	RNFR，RNTO	△※3
切断	QUIT	◎※2

（凡例）

◎：必ず実行されます。

○：どれか 1 つが実行されます。

△：条件に依存します。

注※1

FTPS を使用する場合で，Explicit モードのときに実行されます。FTPS に関する設定については，マニュアル「サービスプラットフォーム 開発ガイド 受付・アダプタ定義編」の「付録 H.1　FTPS を使用したセキュアな接続（FTP アダプタ）」を参照してください。

注※2

ログインセッション維持が有効の場合，または RENAME オペレーションを実行する場合，一部のコマンドの実行が省略されることがあります。

注※3

要求電文フォーマットの指定によっては実行されます。実行条件については，マニュアル「サービスプラットフォーム 開発ガイド 受付・アダプタ定義編」の「3.3.10　FTP アダプタを定義する」を参照してください。

注※4

FTPS を使用する場合に実行されます。

注※5

FTP アダプタ実行環境プロパティファイルの ftpadp.pasvport プロパティの設定に依存します。FTP アダプタ実行環境プロパティファイルの詳細は，マニュアル「サービスプラットフォーム リファレンス」の「6.8.3　FTP アダプタ実行環境プロパティファイル」を参照してください。

注※6

SITE FSIZE コマンドの実行条件については，「8.6.4　送受信後のファイルサイズ確認」を参照してください。

注※7

ファイルの転送コマンド実行前，転送コマンド実行後，リストコマンド実行前，リストコマンド実行後，ファイルのリネームコマンド実行前，またはファイルのリネームコマンド実行後に実行されるコマンドです。

注※8

要求電文フォーマットの<ftp-commands-before>タグ，および<ftp-commands-after>タグに実行指示がある場合に実行されます。

注※9

データ転送時にタイムアウトが発生した場合に実行されます。

注意事項

FTP アダプタが実行する FTP コマンドの仕様は RFC959，RFC2228，RFC2428，および RFC3659 に準拠しています。ただし，次のコマンドは，RFC の仕様と FTP アダプタの仕様とで機能に差異があります。

- TYPE

  FTP アダプタの TYPE コマンドに EBCDIC は指定できません。
- MODE

  FTP アダプタの MODE コマンドにブロックモードは指定できません。
- LIST

  コマンドの引数でオプションを指定できます。ただし，接続先 FTP サーバがオプションの指定をサポートしている必要があります。
- NLST

  コマンドの引数でオプションを指定できます。ただし，接続先 FTP サーバがオプションの指定をサポートしている必要があります。
- EPSV

  FTP アダプタの EPSV コマンドにネットワークプロトコルおよび ALL は指定できません。

8.6.3　処理に使用するフォルダの指定

(1)　処理に使用するフォルダの指定

PUT オペレーション，MPUT オペレーション，GET オペレーション，および MGET オペレーションの場合，HCSC サーバ内で FTP 受付が管理する作業フォルダ，またはクラスタ構成内で共通に利用する共通フォルダを使用してファイルを送受信します。GETINFO オペレーションの場合，作業フォルダおよび共通フォルダのほかに応答電文にファイルのリストを出力することもできます。なお，RENAME オペレーションの場合，作業フォルダおよび共通フォルダは使用しません。

出力先は FTP アダプタの要求電文で指定できます。各オペレーションの出力先の種類と指定方法を次に示します。

- PUT オペレーション，MPUT オペレーション，GET オペレーション，および MGET オペレーションの場合

出力先	指定方法
共通フォルダ	要求電文の<local-folder>タグの属性に common="true"を指定します。
作業フォルダ	要求電文の<local-folder>タグの属性に common="false"を指定します。

- GETINFO オペレーションの場合

出力先	指定方法
共通フォルダ	要求電文の<local-folder>タグの属性に common="true"を指定します。
作業フォルダ	要求電文の<local-folder>タグの属性に common="false"を指定し，かつ<request-id>タグを指定します。
応答電文	要求電文の<local-folder>タグおよび<request-id>タグを省略します。

FTP アダプタの要求電文フォーマットの内容については，マニュアル「サービスプラットフォーム 開発ガイド 受付・アダプタ定義編」の「3.3.10　FTP アダプタを定義する」を参照してください。

(2)　サブフォルダの指定

FTP アダプタでは共通フォルダルートの配下にサブフォルダを作成できます。

共通フォルダ内のサブフォルダを相対パスで指定したファイルの転送ができます。

- **相対パスの指定方法**

  各オペレーションの要求電文または応答電文のローカル側ファイル名（local-file-name）に，共通フォルダルートからの相対パスを指定します。

  - ファイル名を指定する場合の指定例

```
<request>
  :
  <local-folder common=”true”>default</local-folder>
```

```
  <local-file-name>a.txt</local-file-name>
   :
</request>
```

- 相対パスを指定する場合の指定例

```
<request>
   :
  <local-folder common="true">default</local-folder>
  <local-file-name>/folder1/folder2/a.txt</local-file-name>
   :
</request>
```

- 注意事項
  - 作業フォルダについては，相対パスを指定できません。
  - 相対パスの先頭の「/」は省略できます。
  - 共通フォルダルートより上位のフォルダへのアクセスはできません。アクセスした場合はエラーになります。
  - シンボリックリンクのファイルは指定できません。

8.6.4　送受信後のファイルサイズ確認

送受信後のファイルサイズを確認する機能について説明します。

(1)　ファイルサイズ確認とは

FTP サーバに JP1/File Transmission Server/FTP を使用している場合，FTP アダプタと FTP サーバとの間で送受信されたファイルのサイズが，送信側と受信側で正しいかどうかを確認できます。

転送中にファイルが破損した場合，FTP アダプタからの応答コードが正常終了しているときはファイルが破損しているかどうかをユーザから確認できません。このような場合に，FTP アダプタは転送元のファイルサイズと転送先のファイルサイズを比較して，ファイルサイズが異なる場合はビジネスプロセスにシステム例外（転送エラー）を返します。

なお，改行コードのバイト数が異なる OS 間で ASCII モードのファイル転送をした場合，自動的に改行コードが変換されるためファイルサイズが増減しますが，この場合はエラーになりません。

JP1/File Transmission Server/FTP を使用したファイルサイズ確認機能については，マニュアル「JP1/File Transmission Server」を参照してください。

(2)　設定方法

ファイルサイズ確認を有効にするには，FTP アダプタ実行環境プロパティファイルの ftpadp.file-size-check プロパティを「TRUE」に設定している必要があります。

FTP アダプタ実行環境プロパティファイルの詳細は，マニュアル「サービスプラットフォーム リファレンス」の「6.8.3　FTP アダプタ実行環境プロパティファイル」を参照してください。

(3)　注意事項

FTP サーバに JP1/File Transmission Server/FTP を使用していない場合に FTP アダプタ実行環境プロパティファイルの ftpadp.file-size-check プロパティを「TRUE」に設定していると，FTP アダプタは例外を返して，リクエスト処理を終了します。

8.6.5　ファイルのリネーム

FTP アダプタのリモートファイルのファイル名をリネームする方法について説明します。

ファイル名のリネームには，RENAME オペレーションを使用します。RENAME オペレーションの要求電文に指定されたリモートファイルのパスを使用してリネームが行われます。RENAME オペレーションの要求電文については，マニュアル「サービスプラットフォーム 開発ガイド 受付・アダプタ定義編」の「3.3.10　FTP アダプタを定義する」を参照してください。

なお，RENAME オペレーション実行時は，データコネクションは確立されません。

(1)　RENAME オペレーションで実行するコマンド

RENAME オペレーションでは，次の FTP コマンドを実行します。

表 8-6　RENAME オペレーションで実行する FTP コマンド

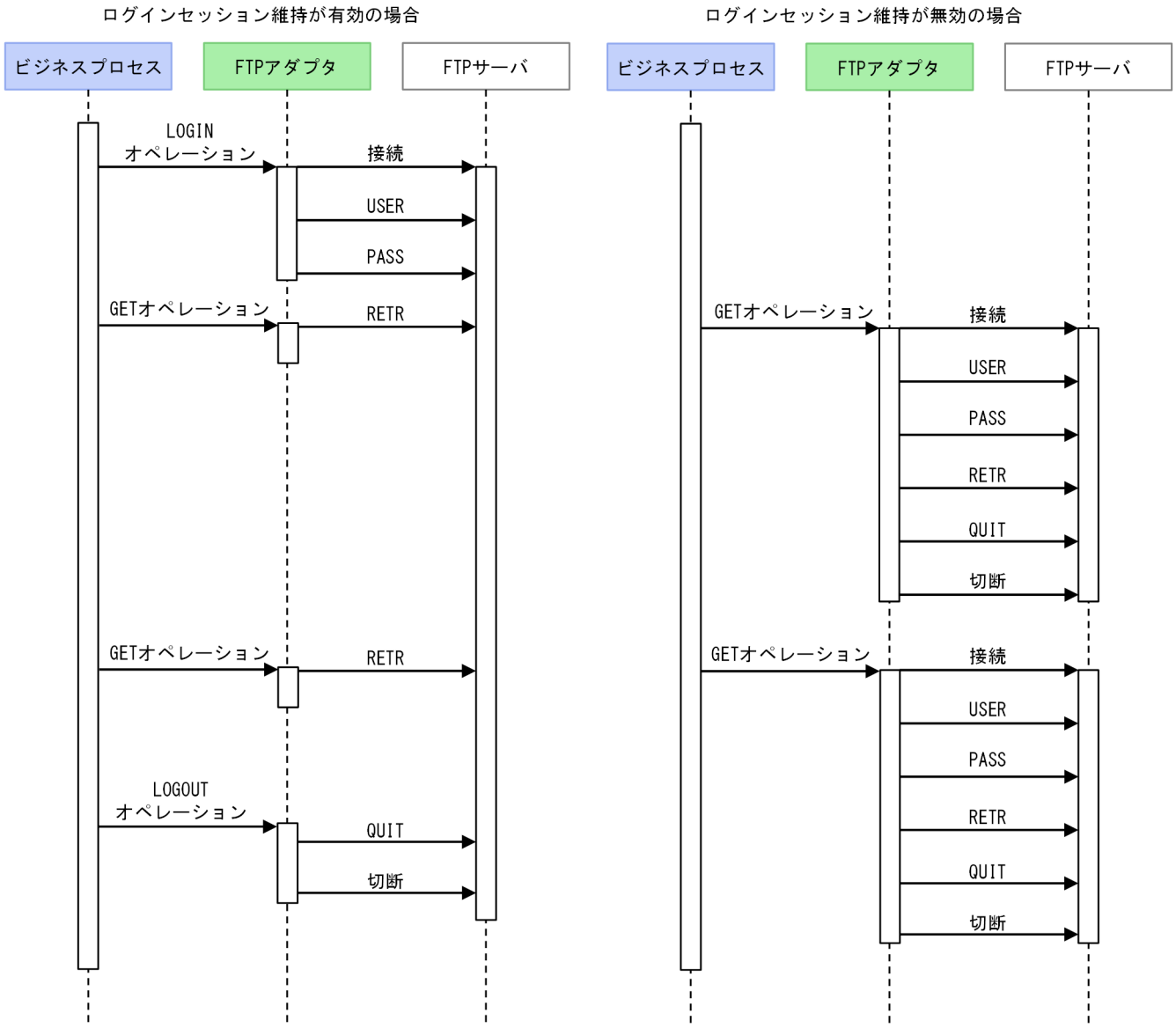
コマンド形式	説明	コマンド引数に指定する値
RNFR ファイルパス	名称変更元を設定するコマンドです。変更したいファイルの元のパス名を指定します。	要求電文の「rename-from-file-path」の要素値
RNTO ファイルパス	名称変更先を設定するコマンドです。RNFR コマンドで指定したファイルの新しいパス名を指定します。	要求電文の「rename-to-file-path」の要素値

RFC959 に従って，RNFR コマンドを実行した直後に，RNTO コマンドを続けて実行します。

8.6.6　ログインセッションの維持

FTP アダプタでは，FTP サーバにログインした接続（ログインセッション）を維持できます。ほかのオペレーションの実行時に，FTP サーバへのログイン処理を省略できるため，処理速度が向上します。

図 8-26　ログインセッション維持の処理の例（2 ファイル取得時）



ログインセッション維持は，FTP アダプタ実行環境プロパティファイルの ftpadp.session.keep プロパティで設定します。

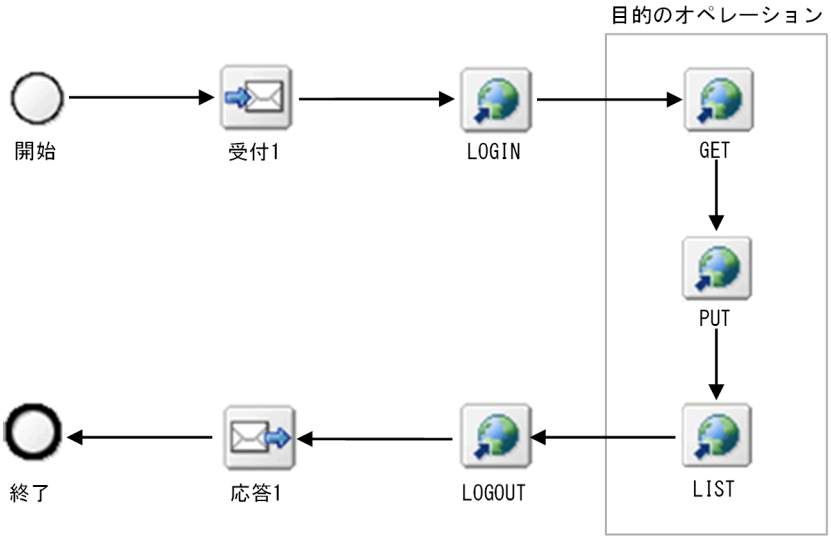
(1)　ログインセッション維持を利用したビジネスプロセスの作成条件

ログインセッション維持を利用したビジネスプロセスを作成する場合，FTP サーバへの接続ごとに，次の要件を満たす必要があります。

- 目的の FTP オペレーションの前に，LOGIN オペレーションを実行する。
- GET オペレーション，PUT オペレーションなどの，目的の FTP オペレーションを実行する。
- 目的の FTP オペレーションの後に，LOGOUT オペレーションを実行する。

この条件を満たした，ビジネスプロセスの設定例を次に示します。

図 8-27　ログインセッション維持を利用したビジネスプロセスの設定例



(2)　最大ログインセッション数の設定

ログインセッションの最大数は，FTP アダプタ実行環境プロパティファイルの ftpadp.session.max で設定できます。

ftpadp.session.max には，FTP アダプタを利用する各ビジネスプロセスのログインセッション必要数の合計より大きい値を指定してください。

ビジネスプロセスのログインセッションの必要数は次の計算式で求めます。

接続条件が異なるFTPサーバへの接続数×ビジネスプロセスの実行多重度

接続条件とは，LOGIN オペレーションの要求電文および FTP 受付実行環境プロパティで指定する IP アドレス，ポート，ログインユーザ，課金情報，FTPS などの条件を指します。

ログインセッション数が超過した場合の処理の例を次に示します。

図 8-28　ログインセッション数が超過した場合の処理（未使用セッションあり）

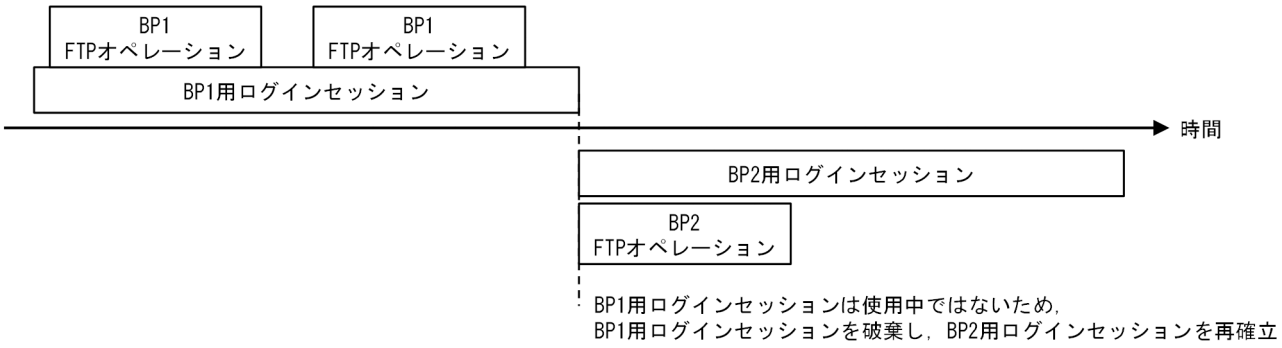


図 8-28 では，最大ログインセッション数が 1 で，並行して動作する 2 つのビジネスプロセスから FTP オペレーションが交互に実行されています。

必要なセッションが最大ログインセッションの設定値を超えた場合，FTP オペレーションで使用されていないログインセッションから，アイドル時間が最も長いログインセッションを破棄した後に，ログインセッションを確立または再確立します。

図 8–29　ログインセッション数が超過した場合の処理（未使用セッションなし）

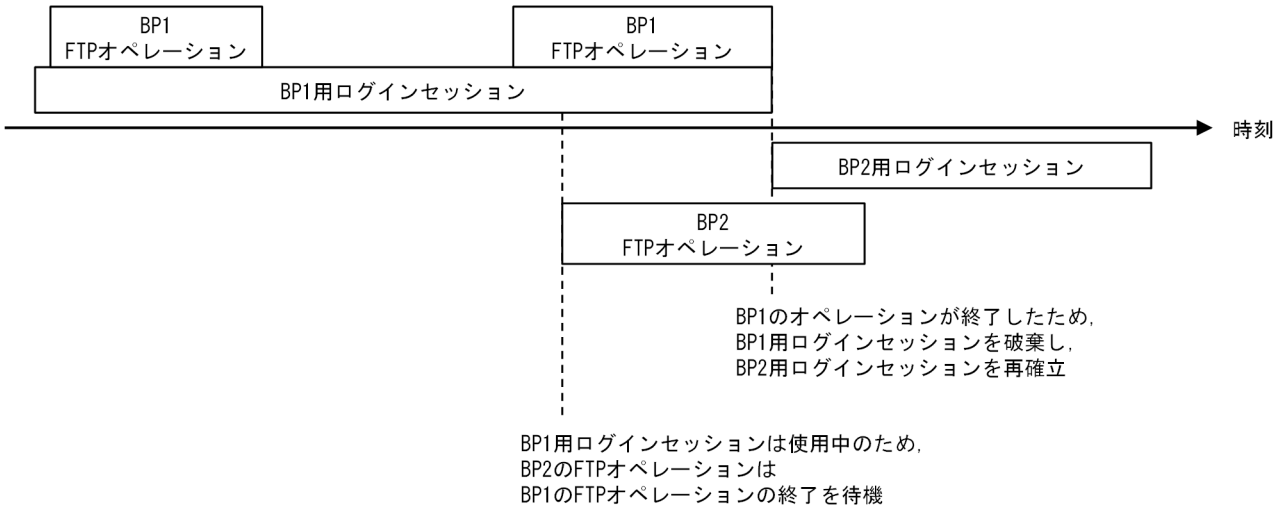


図 8-29 では，最大ログインセッション数が 1 で，並行して動作する 2 つのビジネスプロセスから FTP オペレーションが同時に実行されています。

未使用のログインセッションがなく，すべてのログインセッションが FTP アダプタの実行中オペレーションで使用中の場合，エラーメッセージを出力し，実行中のオペレーションの終了を待機します。実行中のオペレーションが終了した後，終了したオペレーションで使用していたログインセッションを破棄し，新たにログインセッションを確立または再確立します。

(3)　ログインセッションに記録される付加情報

FTP アダプタから FTP サーバに接続する場合，ログインセッションは，セッションに設定されている次の引数の値を付加情報として記録します。

表 8–7　ログインセッションに記録される付加情報

付加情報の種類	引数	FTP コマンド
FTP タイプ	ASCII または Image	TYPE
FTP モード	Stream または Compressed	MODE

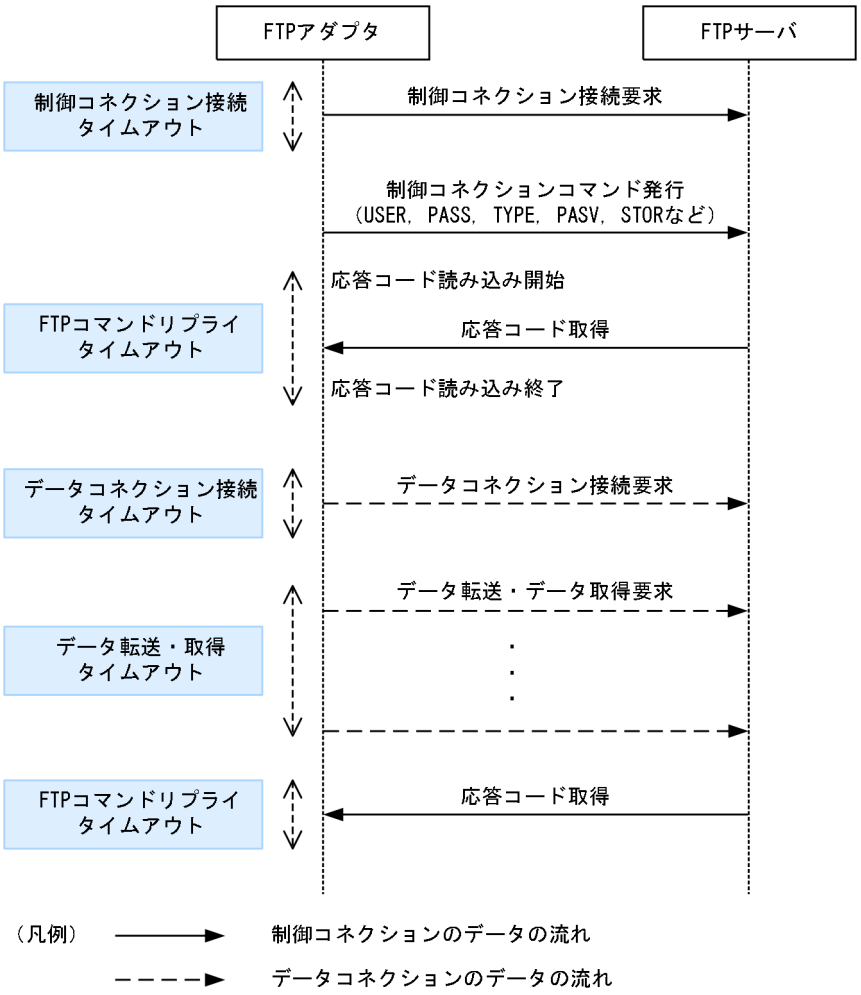
FTP アダプタは，オペレーション実行時に，要求電文および FTP アダプタ実行環境プロパティファイルの設定値と，ログインセッションに記録されている値を比較し，設定値が異なる場合だけ FTP サーバに該当する FTP コマンドを送信します。これによって，不要なコマンド送信を削減できます。

8.6.7　タイマ管理

FTP アダプタのタイマ管理の機能，およびタイムアウト値の設定方法について説明します。

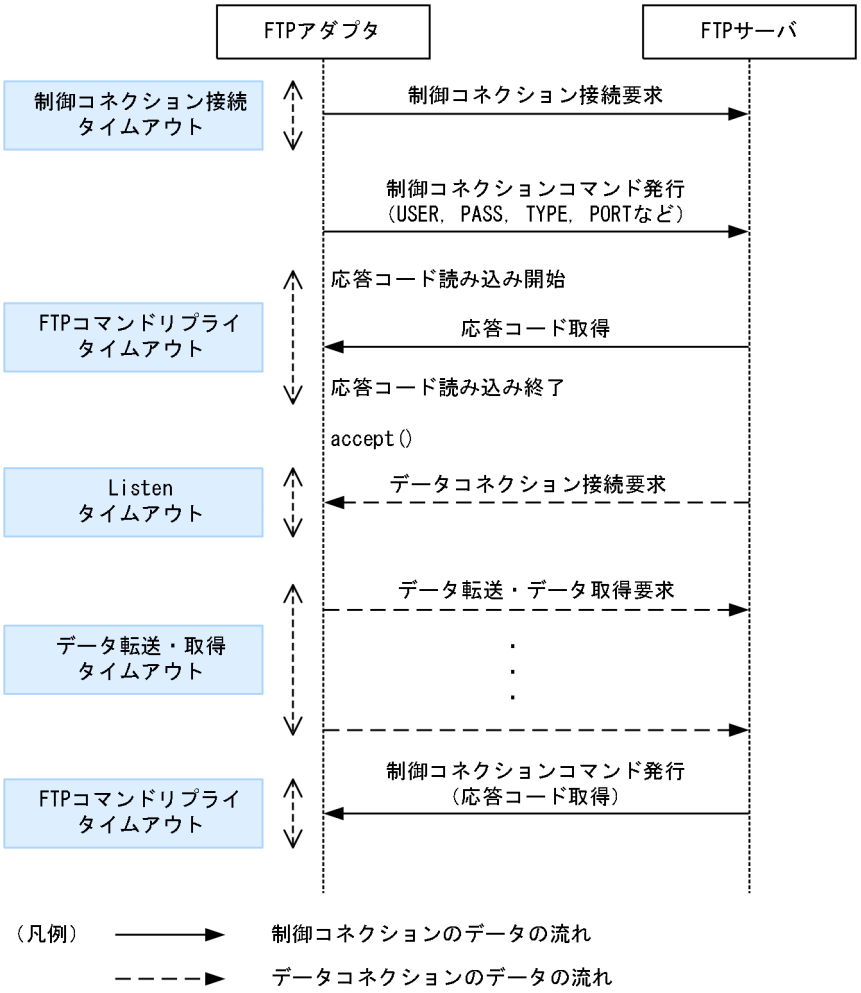
FTP アダプタでタイムアウトを監視するポイントを，Passive モードおよび STOR によるファイル転送の場合を例に，次の図に示します。

図 8-30　FTP アダプタでタイムアウトを監視するポイント（Passive モードおよび STOR によるファイル転送の場合の例）



FTP アダプタでタイムアウトを監視するポイントを，Active モードおよび STOR によるファイル転送の場合を例に，次の図に示します。

図 8-31　FTP アダプタでタイムアウトを監視するポイント（Active モードおよび STOR によるファイル転送の場合の例）



FTP アダプタでは，次のタイムアウトを設定できます。

1. 制御コネクション接続タイムアウト
2. FTP コマンドリプライタイムアウト
3. 排他取得タイムアウト
4. データコネクション接続タイムアウト
5. データ転送・取得タイムアウト
6. Listen タイムアウト
7. データ読み込みタイムアウト
8. ログインセッション維持のアイドルタイムアウト

各タイムアウト値は，FTP アダプタ実行環境プロパティファイルで設定できます。

FTP アダプタ実行環境プロパティファイルの詳細は，マニュアル「サービスプラットフォーム リファレンス」の「6.8.3　FTP アダプタ実行環境プロパティファイル」を参照してください。

(1) 制御コネクション接続タイムアウト

FTP アダプタと FTP サーバの間のコマンド実行やその応答などは，制御コネクション接続によって通信されます。

FTP アダプタからの制御コネクションの接続要求時に相手先ビジーなどでコネクションを確立できなかった場合，指定間隔おきに指定回数まで接続処理をリトライします。接続処理をリトライした場合，リトライの状況を保守用ログに出力します。

設定するパラメタを次の表に示します。

表 8-8 制御コネクション接続タイムアウトの設定

値	キー名称	デフォルト値
制御コネクション接続リトライ回数（回）	ftpadp.control-con.retry.count	5
制御コネクション接続リトライ間隔（秒）	ftpadp.control-con.retry.interval	2
1 回の制御コネクション接続タイムアウト時間（秒）	ftpadp.control-con.connect.timeout	10

タイムアウト時間は，これらのパラメタに指定した値を使った次の計算式で求められます。

（1 回の制御コネクション接続タイムアウト時間＋制御コネクション接続リトライ間隔）×制御コネクション接続リトライ回数

(2) FTP コマンドリプライタイムアウト

FTP アダプタから FTP コマンドを発行したあと，指定した時間を経過しても応答がない場合，例外が発生します。FTP コマンドリプライタイムアウトを設定することで，例外が発生するまでの時間を設定できます。

設定するパラメタを次の表に示します。

表 8-9 FTP コマンドリプライタイムアウトの設定

値	キー名称	デフォルト値（秒）
FTP コマンドリプライタイムアウト値	ftpadp.ftp-command-reply.timeout	60

(3) 排他取得タイムアウト

共通フォルダ内のファイルを PUT オペレーションまたは MPUT オペレーションで FTP サーバに転送する場合，共通フォルダ内のファイルに対して共有ロックが設定されます。

また，GET オペレーションまたは MGET オペレーションで共通フォルダのファイルを取得する場合，共通フォルダのファイルに占有ロックが設定されます。

共有ロックまたは占有ロックの設定時に他プロセスとの競合によって設定が失敗したとき，指定間隔おきに指定回数まで排他取得処理をリトライします。リトライ間隔とリトライ回数の積に，各排他取得時のファイルアクセス時間を加えた時間がタイムアウト時間となります。

設定するパラメタを次の表に示します。

表 8-10　排他取得タイムアウトの設定

値	キー名称	デフォルト値
共有ロックリトライ回数（回）	ftpadp.read-lock.retry.count	0
共有ロックリトライ間隔（秒）	ftpadp.read-lock.retry.interval	1
占有ロックリトライ回数（回）	ftpadp.write-lock.retry.count	0
占有ロックリトライ間隔（秒）	ftpadp.write-lock.retry.interval	1

GETINFO オペレーションで共通フォルダにファイルを出力する場合，すでに同名のファイルが存在するときは，ファイルが上書きされます。同名のファイルがほかのプロセスで使用されているときは，占有ロックリトライ回数および占有ロックリトライ間隔で指定したタイムアウト時間で処理がリトライされます。

(4)　データコネクション接続タイムアウト

FTP アダプタ，FTP サーバ間でのデータ転送は，データコネクション接続によって通信されます。

接続モードが PASV モードの場合，データコネクション接続要求時に相手先ビジーなどによってコネクションを確立できなかったときは，指定間隔おきに指定回数まで接続処理をリトライします。接続処理をリトライした場合，リトライの状況を保守用ログに出力します。

接続モードが PORT モードの場合，データコネクション接続タイムアウトは FTP サーバの仕様に依存するため，FTP アダプタでは設定できません。

設定するパラメタを次の表に示します。

表 8-11　データコネクション接続タイムアウトの設定

値	キー名称	デフォルト値
PASV モードのデータコネクション接続リトライ回数（回）	ftpadp.data-con.retry.count	5
PASV モードのデータコネクション接続リトライ間隔（秒）	ftpadp.data-con.retry. interval	2
1 回のデータコネクション接続タイムアウト時間（秒）	ftpadp.data-con.connect.timeout	10

タイムアウト時間は，これらのパラメタに指定した値を使った次の計算式で求められます。

（1 回のデータコネクション接続タイムアウト時間＋ PASV モードのデータコネクション接続リトライ間隔）×PASV モードのデータコネクション接続リトライ回数

(5) データ転送・取得タイムアウト

FTP アダプタからデータ転送コマンドまたはリストコマンドを実行したあと，指定した時間を経過してもデータ転送またはリストデータの取得が完了しない場合は例外が発生します。データ転送タイムアウトおよびデータ取得タイムアウトを設定することで，例外が発生するまでの時間を指定できます。

この例外が発生した場合のローカルファイルおよびリモートファイルの扱いを次に示します。

- RETR コマンド，LIST コマンド，または NLST コマンド実行中のローカルファイル
  FTP アダプタが削除します。
- STOR コマンドまたは APPE コマンド実行中のリモートファイル
  FTP サーバの仕様に依存します。

設定するパラメタを次の表に示します。

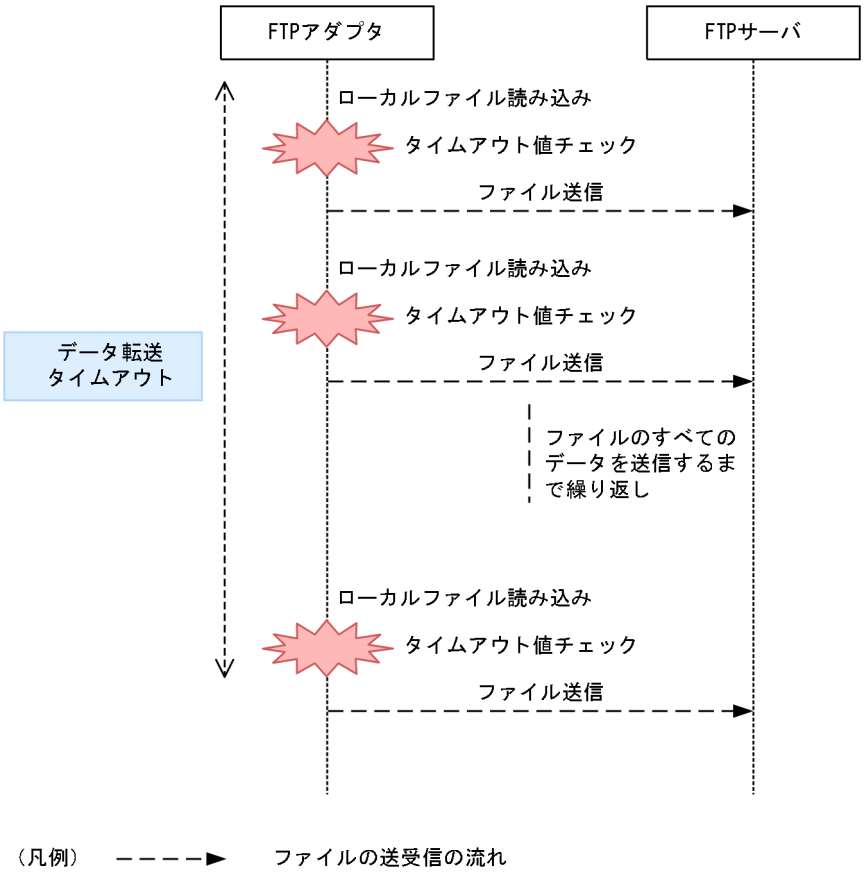
表 8-12　データ転送・取得タイムアウトの設定

値	キー名称	デフォルト値（秒）
データ転送タイムアウト値	ftpadp.data-con-put.timeout	60
データ取得タイムアウト値	ftpadp.data-con-get.timeout	60

ファイル送信は複数回実行されます。

ローカルファイルの読み込みが終わったあとで，時間をチェックして時間が経過していた場合に，タイムアウトが発生します。

図 8-32　データ転送タイムアウトの流れ



ファイル受信は複数回実行されます。

ファイル受信が終わったあとで，時間をチェックして時間が経過していた場合に，タイムアウトが発生します。

図 8-33　データ取得タイムアウトの流れ（Passive の例）

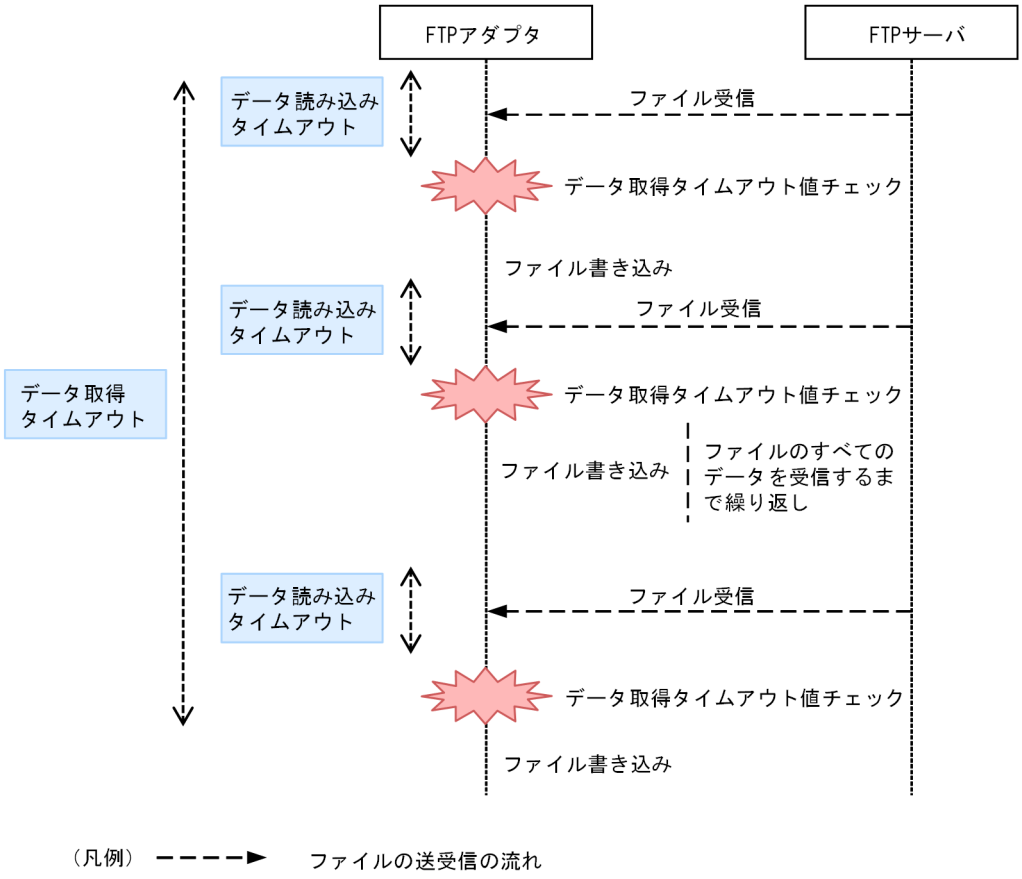
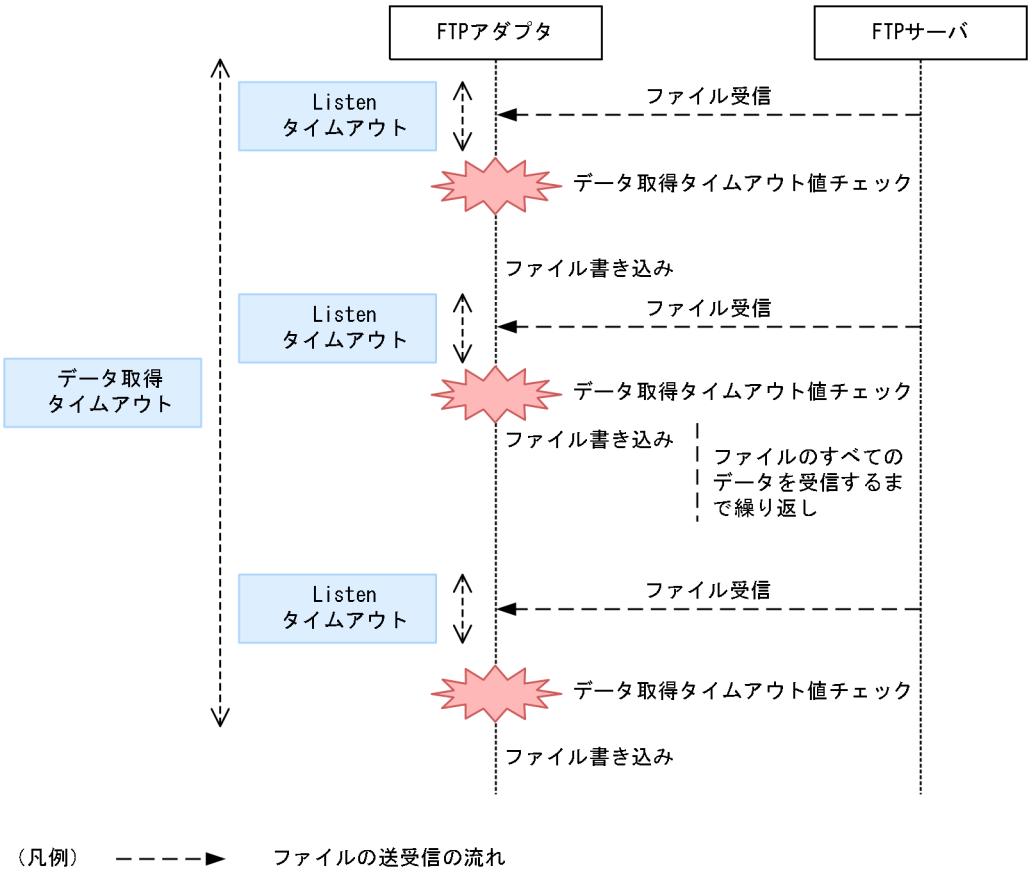


図 8-34　データ取得タイムアウトの流れ（Active の例）



(6)　Listen タイムアウト

接続モードが PORT モードで，次のどちらかの場合，例外が発生します。

- FTP アダプタが PORT コマンドの応答を受けたあと，Listen 状態で一定時間 FTP サーバからデータコネクションの接続要求がなかった場合
- データコネクションの接続完了後，FTP サーバから FTP アダプタにファイルが送信されてこない場合

Listen タイムアウトを設定することで，例外が発生するまでの時間を指定できます。

設定するパラメタを次の表に示します。

表 8-13　Listen タイムアウトの設定

値	キー名称	デフォルト値（秒）
Listen タイムアウト値	ftpadp.listen.timeout	60

(7)　データ読み込みタイムアウト

接続モードが PASV モードで，データコネクションの接続完了後，FTP サーバから FTP アダプタにファイルが送信されてこない場合に，例外が発生します。

データ読み込みタイムアウトを設定することで，例外が発生するまでの時間を指定できます。

設定するパラメタを次の表に示します。

表 8-14　データ読み込みタイムアウト設定

値	キー名称	デフォルト値（秒）
データ読み込みタイムアウト値	ftpadp.data-con.read.timeout	0

(8)　ログインセッション維持のアイドルタイムアウト

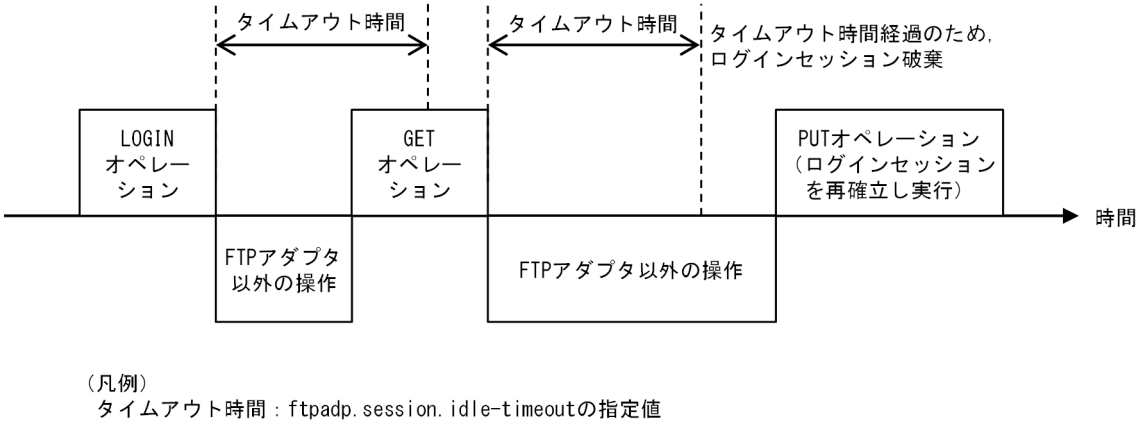
フォルトなどでビジネスプロセスの実行が途中で中断された場合，LOGOUT オペレーションが呼び出されないで，ビジネスプロセスが終了する場合があります。このようなログインセッションの切断漏れを防ぐため，ログインセッションアイドルタイマを使用します。

ログインセッションアイドルタイマは，FTP オペレーションの実行終了時に開始します。指定時間内に該当するログインセッションが使用されない場合，エラーメッセージを出力し，セッションを切断します。

FTP オペレーションの実行が終了し，次の FTP オペレーションを実行するまでの時間間隔は，ftpadp.session.idle-timeout で指定するアイドルタイムアウト以内に収めるように調整してください。オペレーションの実行間隔がアイドルタイムアウトを超えると，ログインセッションアイドルタイマによってログインセッションが破棄され，次の FTP オペレーション実行時に FTP アダプタがログインセッションを再確立します。

ログインセッション維持のアイドルタイムアウトの例を次に示します。

図 8-35　ログインセッション維持のアイドルタイムアウトの例



（凡例）
タイムアウト時間：ftpadp.session.idle-timeoutの指定値

8.7　ファイル操作アダプタの機能

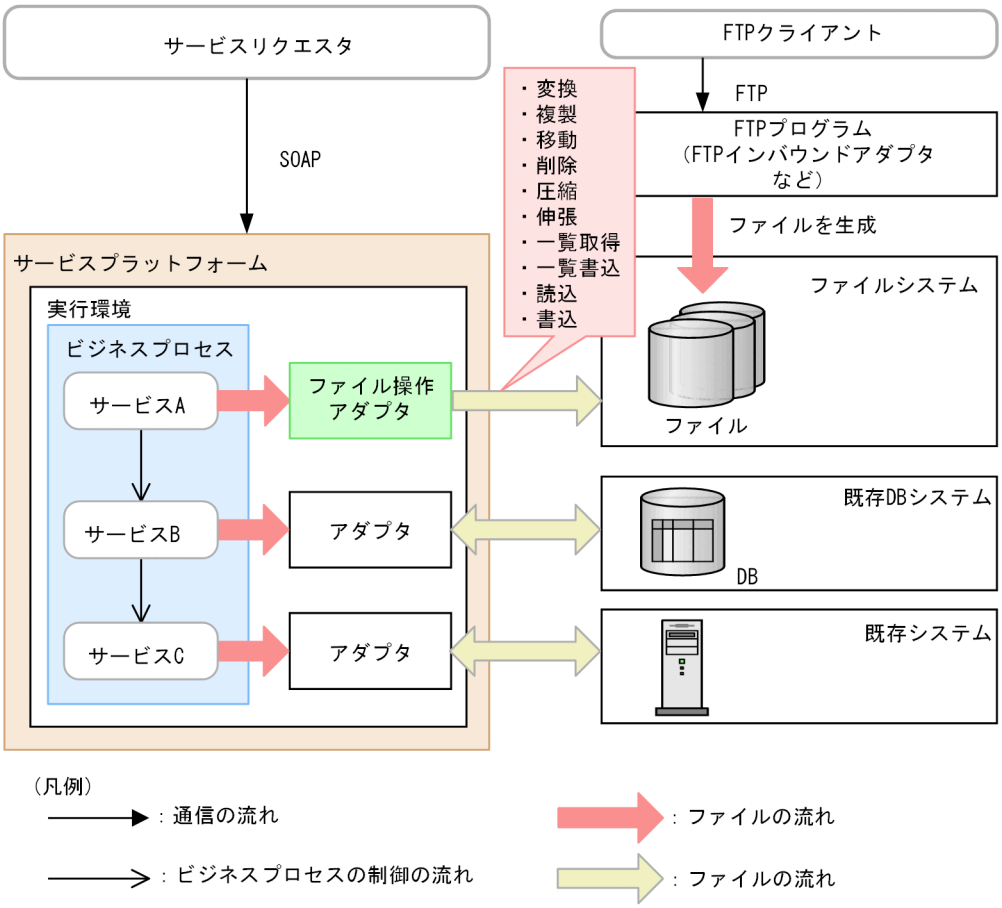
ここでは，ファイル操作アダプタの構成およびサポートするオペレーションについて説明します。

ファイル操作アダプタとは，FTP プログラムなどを利用して生成したファイルに対してレイアウト変換や文字コード変換，複製，移動，削除，圧縮，伸張，一覧取得などのファイル操作をするためのアダプタです。ファイル操作アダプタでは，ファイルに対する操作の単位をオペレーションと呼びます。

FTP 連携を利用する場合，FTP 受付および FTP アダプタを使用してオペレーションごとにファイル操作アダプタを定義する必要があります。

ファイル操作アダプタの構成を次の図に示します。

図 8-36　ファイル操作アダプタの構成



8.7.1　ファイル操作アダプタでサポートするオペレーション

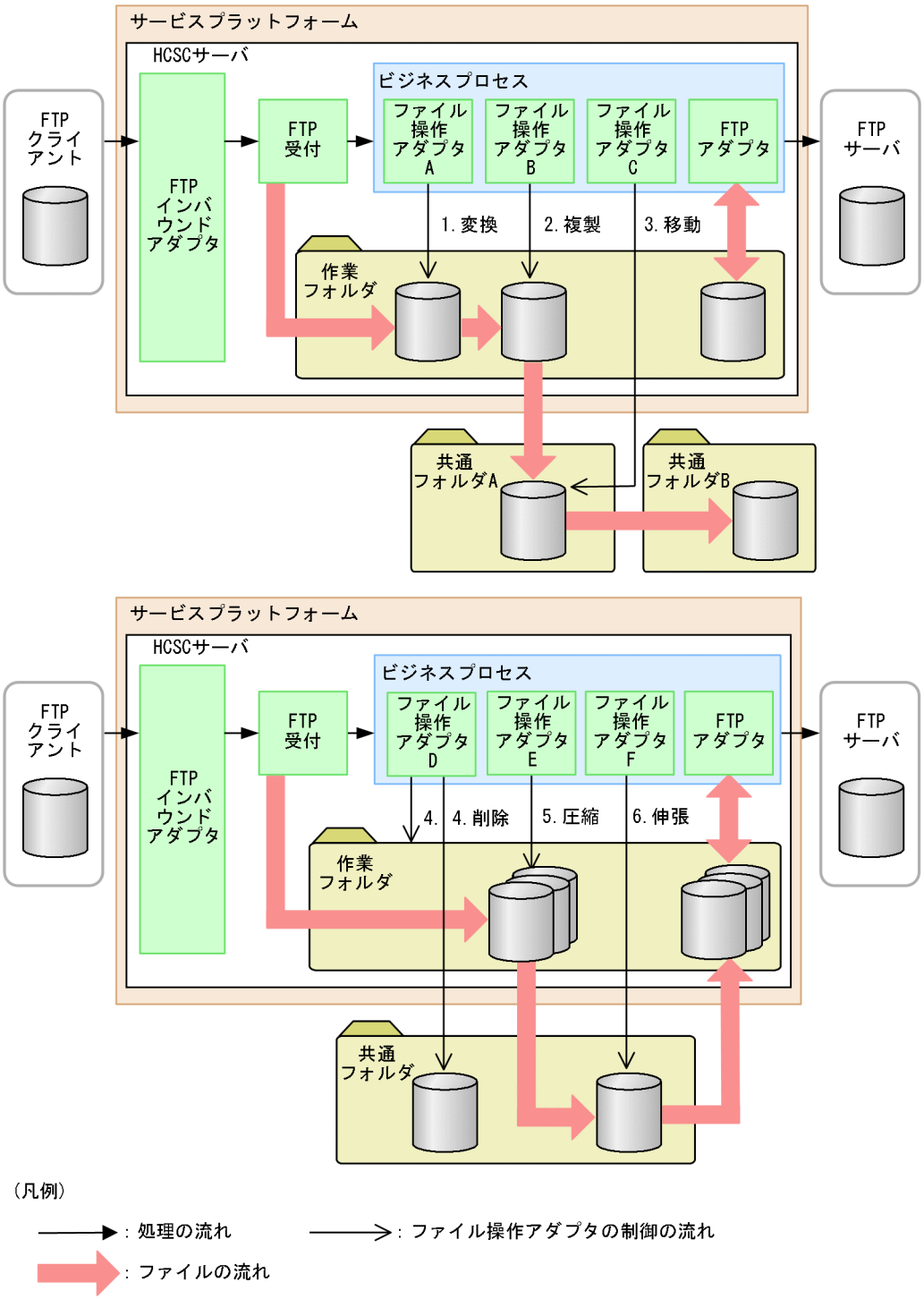
ファイル操作アダプタでサポートするオペレーションには，次の種類があります。

- ファイル変換オペレーション
- ファイル複製オペレーション

- ファイル移動オペレーション
- ファイル・フォルダ削除オペレーション
- ファイル圧縮オペレーション
- ファイル伸張オペレーション
- ファイル一覧取得オペレーション
- ファイル一覧書き込みオペレーション
- ファイル読み込みオペレーション
- ファイル書き込みオペレーション

それぞれのオペレーションの動作を次の図に示します。

図 8-37　ファイル操作アダプタのオペレーションの動作（1/3）



（凡例）

：処理の流れ　　：ファイル操作アダプタの制御の流れ

：ファイルの流れ

図 8-38　ファイル操作アダプタのオペレーションの動作（2/3）

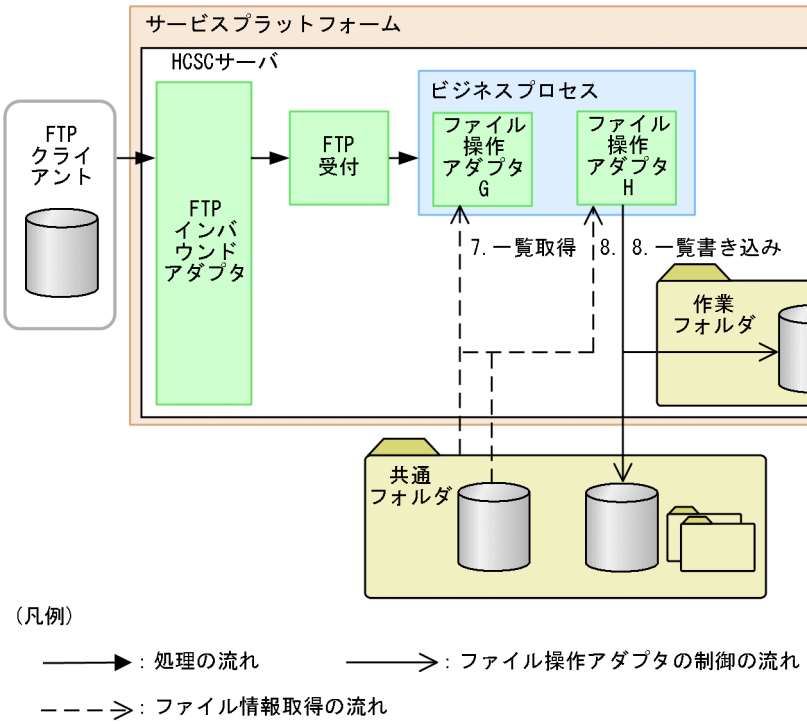
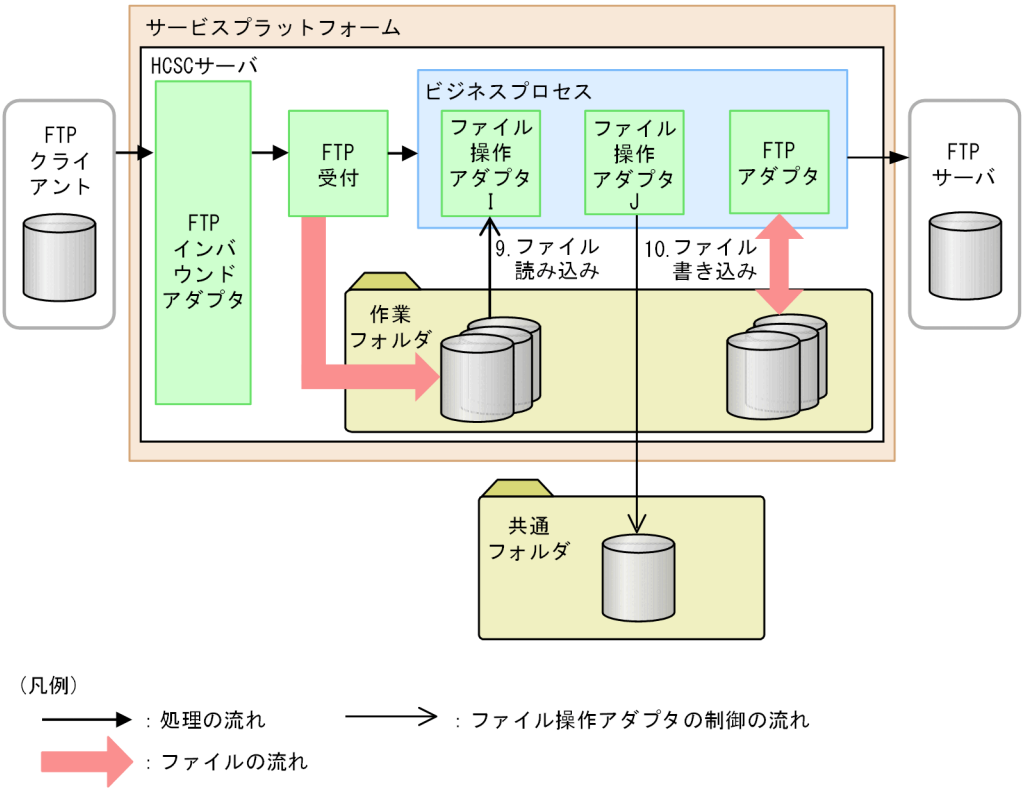


図 8-39　ファイル操作アダプタのオペレーションの動作（3/3）



図中の 1.～10.について説明します。次の番号は図中の番号と対応しています。

1. ファイル操作アダプタ A（ファイル変換オペレーション）は，作業フォルダ（または共通フォルダ）から変換元の入力ファイルを読み込み，変換後の出力ファイルを作業フォルダ（または共通フォルダ）に書き込みます。
2. ファイル操作アダプタ B（ファイル複製オペレーション）は，作業フォルダ（または共通フォルダ）から入力ファイルを読み込み，作業フォルダ（または共通フォルダ）に出力ファイルを複製します。
3. ファイル操作アダプタ C（ファイル移動オペレーション）は，作業フォルダ（または共通フォルダ）のファイルを作業フォルダ（または共通フォルダ）へ移動します。
4. ファイル操作アダプタ C（ファイル・フォルダ削除オペレーション）は，共通フォルダのファイル，サブフォルダ，または作業フォルダを削除します。
5. ファイル操作アダプタ D（ファイル圧縮オペレーション）は，作業フォルダ（または共通フォルダ）から圧縮元の入力ファイルを読み込み，作業フォルダ（または共通フォルダ）に圧縮後の出力ファイルを書き込みます。
6. ファイル操作アダプタ E（ファイル伸張オペレーション）は，作業フォルダ（または共通フォルダ）から伸張元の入力ファイルを読み込み，作業フォルダ（または共通フォルダ）に伸張後の出力ファイルを書き込みます。
7. ファイル操作アダプタ G（ファイル一覧取得オペレーション）は，共通フォルダ（または共通フォルダのサブフォルダ）直下のファイル情報もしくは共通フォルダのファイル単体の情報を取得し，応答電文で返します。
8. ファイル操作アダプタ H（ファイル一覧書き込みオペレーション）は，共通フォルダ（または共通フォルダのサブフォルダ）直下のファイル情報もしくは共通フォルダのファイル単体の情報を取得し，作業フォルダ（または共通フォルダ）のファイルに取得した情報を書き込みます。
9. ファイル操作アダプタ I（ファイル読み込みオペレーション）は，作業フォルダ（または共通フォルダ）からファイルを読み込み，ほかのサービスに渡します。
10. ファイル操作アダプタ J（ファイル書き込みオペレーション）は，ほかのサービスから取得したデータを，作業フォルダ（または共通フォルダ）に書き込みます。

ファイル操作アダプタを使用してファイルの変換や削除などをする場合，オペレーションごとにファイル操作アダプタを定義する必要があります。

それぞれのオペレーションのファイル操作アダプタ同士を組み合わせて使用することもできます。

8.7.2　各オペレーションの要求電文および応答電文

ファイル操作アダプタで使用する要求電文および応答電文の電文フォーマットは，オペレーションごとに異なります。

各オペレーションの要求電文および応答電文の電文フォーマットについては，マニュアル「サービスプラットフォーム 開発ガイド 受付・アダプタ定義編」の「3.3.11　ファイル操作アダプタを定義する」を参照してください。

8.7.3　ファイル変換オペレーション

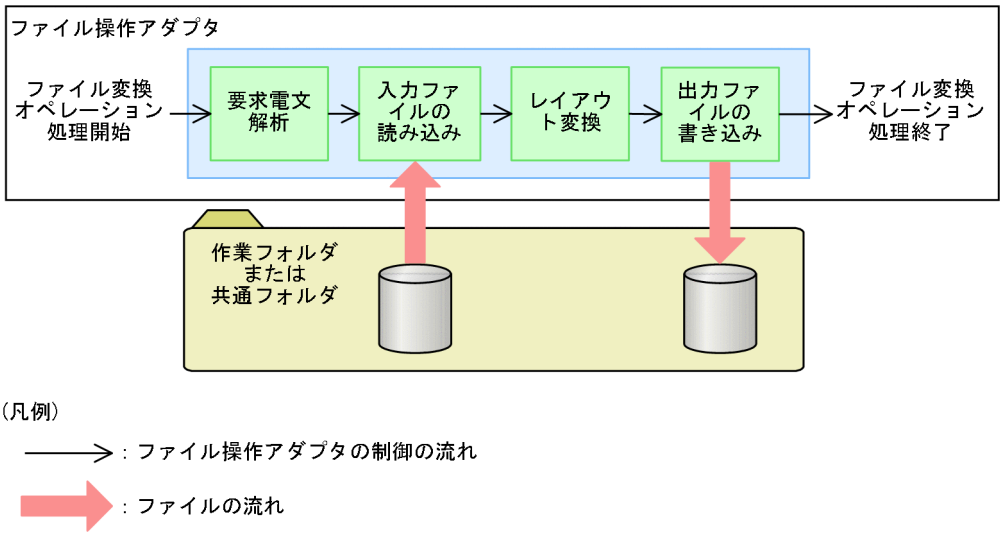
ここでは，ファイル変換オペレーションでサポートする機能やファイル形式について説明します。

(1)　ファイル変換オペレーションの機能概要

ファイル変換オペレーションは，作業フォルダまたは共通フォルダから変換元の入力ファイルを読み込み，変換後の出力ファイルを作業フォルダまたは共通フォルダに書き込みます。

ファイル変換オペレーションの処理の概要について，次の図に示します。

図 8-40　ファイル変換オペレーションの処理



(2)　サポートする入出力ファイルの形式

ファイル変換オペレーションでサポートする入力ファイル，および出力ファイルのファイル形式について次の表に示します。

表 8-15　ファイル変換オペレーションでサポートするファイル形式

項番	処理方式	入力ファイルの形式		出力ファイルの形式	サポート可否
1	一括処理方式	バイナリ形式		バイナリ形式	○
2				XML 形式	○
3		XML 形式		バイナリ形式	○
4				XML 形式	○
5	分割処理方式	バイナリ形式	固定長形式	バイナリ形式	○
6				XML 形式	○
7			レングスタグ形式	バイナリ形式	×
8				XML 形式	×

項番	処理方式	入力ファイルの形式		出力ファイルの形式	サポート可否
9	分割処理方式	バイナリ形式	改行セパレータ形式	バイナリ形式	○
10				XML 形式	○
11			EOR セパレータ形式	バイナリ形式	○
12				XML 形式	○
13			上記以外	バイナリ形式	×
14				XML 形式	×
15		XML 形式		バイナリ形式	○
16				XML 形式	○

（凡例）

○：サポートしています。

×：サポートしていません。

固定長形式，レングスタグ形式，改行セパレータ形式，および EOR セパレータ形式の違いを次に示します。

- 固定長形式

  レコードのバイト長をあらかじめ決めておく形式です。

- レングスタグ形式

  データ要素に対して，その長さまたは個数を表す数値情報を付けた形式です。

- 改行セパレータ形式

  改行コード（LF（0x0a）または CR+LF（0x0d 0x0a））の出現をレコードの区切りとする形式です。改行セパレータ形式の詳細については，「(4)(a)　レコード単位の判定形式の種類」を参照してください。

- EOR セパレータ形式

  ファイル操作アダプタ定義ファイルで設定した EOR コード（レコードの終端を示すコード）の出現をレコードの区切りとする形式です。EOR セパレータ形式を使用すると，レコードの区切りが改行コード以外のファイルも処理できます。

ファイル変換時にレイアウト変換ができます。また，入出力形式がバイナリ形式の場合は，改行コードの付与・削除，文字コード変換もできます。

参考

入力ファイルに 1 バイトでもデータが含まれている場合は，ファイル変換を実行します（データが空文字，改行文字，EOF 文字だけの場合であっても，ファイル変換を実行します）。入力ファイルが 0 バイトの場合は，0 バイトのファイルを出力します。エラーにはなりません。

(3)　一括処理方式

一括処理方式では，入力ファイルを一括して読み込み，一括して変換します。変換後，一括して出力ファイルへ書き込みます。

一括処理方式で対象となるデータフォーマット，および対象とする変換内容を次に示します。

- **対象とするデータフォーマット**

  1 つのバイナリフォーマット定義または XML スキーマで定義できる範囲です。

- **対象とする変換内容**

  1 つのデータ変換定義ファイル（拡張子：xsl）で定義できる範囲です。

注意事項

入力ファイルおよび出力ファイルのサイズが 500MB 以上のファイルは変換できません。500MB 以上のファイルを変換した場合，動作を保証しません。

(4)　分割処理方式

入力ファイルの読み込み，変換，出力ファイルへの書き込みを，入力ファイルのレコード単位で処理します。入力ファイルの終端まで繰り返し処理を実行します。

分割処理方式でバイナリ形式から XML 形式のファイルに変換する場合，および XML 形式からバイナリ形式のファイルに変換する場合の定義例については，マニュアル「サービスプラットフォーム 開発ガイド 受付・アダプタ定義編」の「付録 E　ファイル操作アダプタの定義例」を参照してください。

(a)　レコード単位の判定形式の種類

入力ファイルの形式によってレコード単位の判定形式が異なります。

バイナリ形式の場合

レコード単位の判定には，次の形式があります。

- **固定長形式**

  レコードのバイト長をあらかじめ指定する形式です。入力ファイルが固定長形式の場合に使用します。

- **改行セパレータ形式**

  改行コード（LF（0x0a）または CR+LF（0x0d 0x0a））の出現をレコードの区切りとして判定する形式です。入力ファイルが改行セパレータ形式の場合に使用します。改行セパレータ形式については，「(d) レコードの分割（入力ファイルがバイナリ形式の場合)」の説明を参照してください。

  分割処理方式で，改行セパレータ形式を選択した場合，バイナリフォーマット定義に関係なく，改行コード（LF（0x0a）または CR+LF（0x0d 0x0a））の出現でレコードの区切りを判断しているため，変換元ファイルにある文字コードの改行のバイトパターンが，改行コード以外のとき（文字コード例：UTF-16_BIG，UTF-16_LITTLE，KEIS+EBCDIC，KEIS+EBCDIK，KEIS），改行

セパレータ形式の分割処理方式では正常に処理できません。なお，CUSTOM を選択した場合，処理対象のデータに従います。

- EOR セパレータ形式

  ファイル操作アダプタ定義ファイルで設定した EOR コード（レコードの終端を示すコード）の出現をレコードの区切りとする形式です。EOR セパレータ形式を使用すると，レコードの区切りが改行コード以外のファイルも処理できます。

XML 形式の場合

ファイル操作アダプタ定義ファイルでレコードとして扱う要素を指定します。

注意事項

入力ファイルおよび出力ファイルのサイズに制限はありません。ただし，入力ファイルおよび出力ファイルの 1 レコードのサイズが 500MB 以上のファイルは入力ファイルの形式に関係なく変換できません。1 レコードのサイズが 500MB 以上のファイルを変換した場合，動作を保証しません。また，レコードを構成する 1 要素のサイズについても，入力ファイルの形式に関係なく，500MB 以上を超える場合は動作を保証しません。

(b) 対象とするデータフォーマット

分割処理方式で対象とする入力ファイルのデータフォーマットは，バイナリフォーマット定義および XML スキーマです。また，対象とするデータ構造は，ヘッダレコード部，データレコード部，トレーラレコード部がこの順番で出現する構造である必要があります。

データレコード部は，決められた要素の並びが連続して繰り返されている構造を持ったデータです。データベースのテーブル構造と同じで，決められた要素の並び（バイナリ形式の場合は行，XML 形式の場合は特定の要素以下のデータ）をレコードと呼び，これが n 回繰り返されているイメージです。レコード内の要素数およびレコード数に制限はありません。

ヘッダレコード部およびトレーラレコード部は，任意の要素が並ぶ複合内容要素で，レコードが繰り返されることはありません。また，これらの部分は省略できます。

データファイル構造とデータレコード部のイメージを次に示します。

図 8-41　データファイル構造

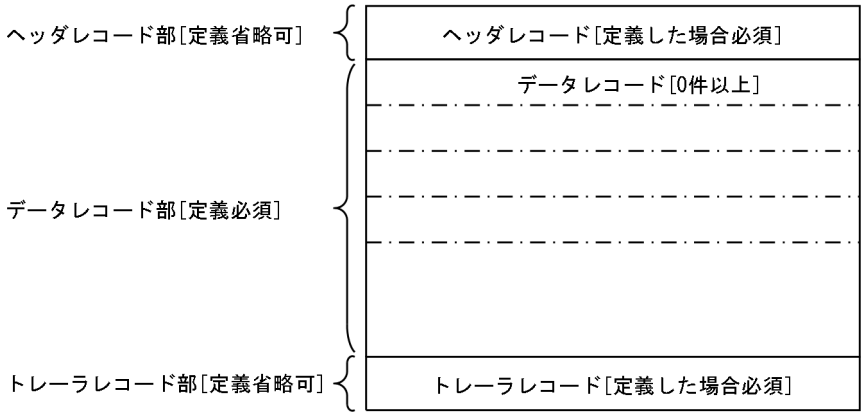
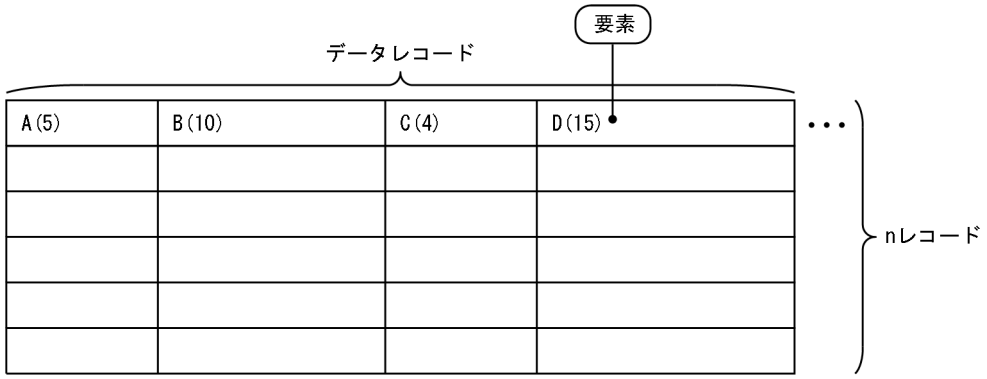


図 8-42　データレコード部のイメージ



（凡例）

・・・：繰り返しの省略

データフォーマットはバイナリ形式（固定長形式，改行セパレータ形式，および EOR セパレータ形式），および XML 形式に対応します。また，入力および出力フォーマットとも上記の形式の構造であることが必要です。

バイナリ形式のデータの例を次に示します。

図 8-43　固定長形式のデータ例



図 8-44　改行セパレータ形式のデータ例

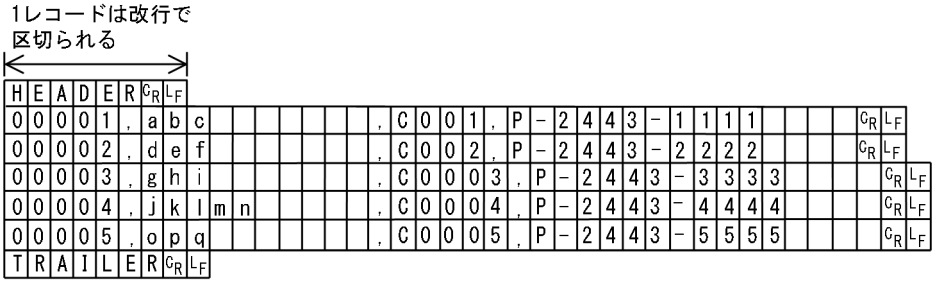
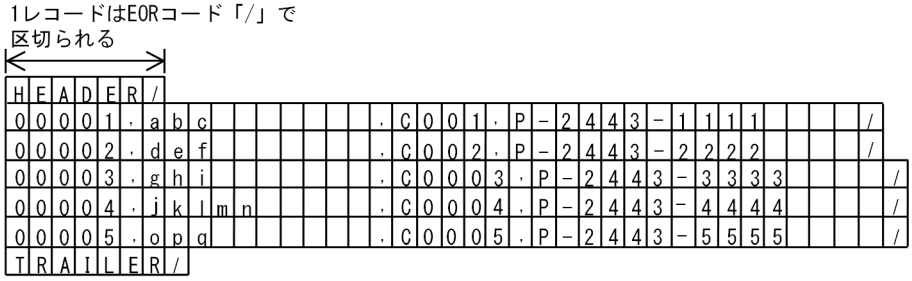


図 8-45　EOR セパレータ形式のデータ例（EOR コードが「/（0x2F)」の場合）



(c)　分割処理方式で必要なファイル

分割処理方式で定義が必要なバイナリフォーマット定義，XML スキーマ，およびデータ変換定義を次に示します。

バイナリフォーマット定義および XML スキーマ

入力ファイルおよび出力ファイルの形式によって必要な定義が異なります。

入力ファイルまたは出力ファイルがバイナリ形式の場合

入力側と出力側のファイルフォーマットを定義する必要があるため，バイナリフォーマット定義エディタを使用して次のうち必要なフォーマットを作成します。

- 入力側のヘッダレコード部のフォーマット
- 入力側のデータレコード部のフォーマット
- 入力側のトレーラレコード部のフォーマット
- 出力側のヘッダレコード部のフォーマット
- 出力側のデータレコード部のフォーマット
- 出力側のトレーラレコード部のフォーマット

入力ファイルまたは出力ファイルが XML 形式の場合

入力側と出力側の XML スキーマを作成します。バイナリ形式と異なり，1 つの XML スキーマで，ヘッダレコード，データレコード，およびトレーラレコードを定義することができます。XML スキーマはデータ変換定義を作成するために必要となりますが，変換オペレーションの実行時には使用しません。

データ変換定義

入力側と出力側のフォーマット間のデータ変換を定義する必要があるため，マッピング定義エディタを使用して次のデータ変換定義を作成してください。

- ヘッダレコード部のデータ変換定義
- データレコード部のデータ変換定義
- トレーラレコード部のデータ変換定義

なお，出力側が XML 形式の場合，マッピング対象を指定した複合内容要素の配下に制限をしてマッピングできます。1 つの XML スキーマにヘッダレコード，データレコード，およびトレーラレコードを定義した場合，各レコードのルート要素となる複合内容要素の配下に制限してマッピングすることで，各レコード部のデータ変換定義を生成できます。詳細はマニュアル「サービスプラットフォーム 開発ガイド 基本開発編」の「7.4.8　マッピング対象の範囲を限定する」を参照してください。

(d)　レコードの分割（入力ファイルがバイナリ形式の場合）

レコード単位の判定方法

レコードの区切りの判定は，固定長形式，改行セパレータ形式，または EOR セパレータ形式から選択して指定できます。

固定長形式，改行セパレータ形式，または EOR セパレータ形式の種類は，ファイル操作アダプタ定義ファイルの次のパラメタで指定してください。

- csc.adapter.fileOperation.transform.format

  内容については，マニュアル「サービスプラットフォーム リファレンス」の「3.5.2　ファイル操作アダプタ定義ファイル」を参照してください。

- 固定長形式の場合

  レコードの区切りは，ファイル操作アダプタ定義ファイルの次のパラメタに指定した個所です。

  ・csc.adapter.fileOperation.transform.header.fixedSize

  ・csc.adapter.fileOperation.transform.data.fixedSize

  ・csc.adapter.fileOperation.transform.trailer.fixedSize

- 改行セパレータ形式の場合

  レコードの区切りは，入力ファイルに改行コード（LF（0x0a）または CR+LF（0x0d 0x0a））が出現する個所です。ファイル終端もレコードの区切りとして扱われます。なお，改行直後にファイルの終端が出現する場合は，直前の改行と合わせて 1 つのレコードセパレータとして扱われます。

- EOR セパレータ形式の場合

  レコードの区切りは，ファイル操作アダプタ定義ファイルに設定した EOR コードが出現する個所です。EOR コードは csc.adapter.fileOperation.transform.EOR プロパティに 16 進数で設定します。16 進数の先頭には「0x」を付与しません。

  EOR コードが「/（0x2F)」の場合の設定例を次に示します。

  ・csc.adapter.fileOperation.transform.EOR=2F

EOR コードが出現した個所までのデータが 1 レコードとして扱われます。また，ファイルの終端もレコードの区切りとして扱われます。なお，EOR コードが出現した直後にファイルの終端があった場合，EOR コードが出現した個所までのデータが最終レコードとして扱われます。

レコード部の位置と長さ

分割処理方式の場合，現在読み込まれているレコードが，ヘッダレコード部，データレコード部，トレーラレコード部のうちどれなのかを判定する必要があります。固定長形式，改行セパレータ形式，または EOR セパレータ形式ごとに，レコード部の位置と長さを判定する方法を次に示します。

- 固定長形式の場合

  レコード部の位置と長さの判定について，次の表に示します。

表 8-16　レコード有無指定によるレコード種別（固定長形式）

レコード有無指定	ヘッダレコード部指定有無	有	有	無	無
	トレーラレコード部指定有無	有	無	有	無
レコード部の位置と長さ	ヘッダレコード部	ファイル先頭 ～ ヘッダレコード長	ファイル先頭 ～ ヘッダレコード長	－	－
	データレコード部	ヘッダレコード部終了位置 ～ データレコード長＊データレコード数	ヘッダレコード部終了位置 ～ データレコード長＊データレコード数	ファイル先頭 ～ データレコード長＊データレコード数	ファイル先頭 ～ データレコード長＊データレコード数
	トレーラレコード部	データレコード部終了位置 ～ トレーラレコード長	－	データレコード部終了位置 ～ トレーラレコード長	－

（凡例）

－：該当しません。

データレコード数は，次の計算式に従って算出します。

```
データレコード部サイズ＝データファイルサイズ －（ヘッダレコード長＋トレーラレコード長）
レコード数 ＝ データレコード部サイズ／データレコード長（小数点以下切り捨て）
```

レコード数の計算結果にあまりがある場合，メッセージ（KDEC80045-W）が出力されたあと，処理が継続されます。この場合，次に示すような可能性があるため，確認して原因を取り除いてください。

・入力ファイルの末尾に EOD がある。

・入力ファイルに誤りがある。
・ファイル操作アダプタ定義ファイルの次の項目の設定に誤りがある。
csc.adapter.fileOperation.transform.headerRecord
csc.adapter.fileOperation.transform.trailerRecord
csc.adapter.fileOperation.transform.header.fixedSize
csc.adapter.fileOperation.transform.data.fixedSize
csc.adapter.fileOperation.transform.trailer.fixedSize

- 改行セパレータ形式および EOR セパレータ形式の場合
  レコード部の位置と長さの判定について，次の表に示します。

表 8-17　レコード有無指定によるレコード種別（改行セパレータ形式）

レコード有無指定	ヘッダレコード部指定有無	有	有	無	無
	トレーラレコード部指定有無	有	無	有	無
レコード部の位置と長さ	ヘッダレコード部	先頭行	先頭行	－	－
	データレコード部	先頭行以降 ～ 最終行直前	先頭行以降 ～ 最終行	先頭行 ～ 最終行直前	先頭行 ～ 最終行
	トレーラレコード部	最終行	－	最終行	－

（凡例）
－：該当しません。

メッセージ（KDEC80047-E）が出力されて処理が終了された場合，次に示すような可能性があるため，確認して原因を取り除いてください。
・入力ファイルのレコード数が不足している。
・ファイル操作アダプタ定義ファイルの次の項目の設定に誤りがある。
csc.adapter.fileOperation.transform.headerRecord
csc.adapter.fileOperation.transform.trailerRecord

表 8-18　エラー判定パターン

ヘッダレコード部	トレーラレコード部	変換するファイルの行数
有	有	1 行以下
有	無	0 行以下
無	有	0 行以下

(e) レコードの分割（入力ファイルが XML 形式の場合）

入力 XML ファイルのヘッダレコード，データレコード，およびトレーラレコードのレコード区切りを指定するには，ファイル操作アダプタ定義ファイルに読み込む要素のパスを設定します。

ファイル操作アダプタ定義ファイルの設定項目と設定内容を次の表に示します。

表 8-19　ファイル操作アダプタ定義ファイルの設定（入力 XML ファイルの分割入力）

設定項目	設定内容
csc.adapter.fileOperation.transform.header.inElement	入力 XML ファイルのヘッダレコードに対応する要素のパスを指定します。 ヘッダレコードを使用しない場合，指定は不要です（指定しても無視されます）。
csc.adapter.fileOperation.transform.data.inElement	入力 XML ファイルのデータレコードに対応する要素のパスを指定します。
csc.adapter.fileOperation.transform.trailer.inElement	入力 XML ファイルのトレーラレコードに対応する要素のパスを指定します。 トレーラレコードを使用しない場合，指定は必要ありません（指定しても無視されます）。

入力 XML ファイルとファイル操作アダプタ定義ファイルの設定例を示します。

図 8-46　入力 XML ファイルとファイル操作アダプタ定義ファイルの設定例

＜入力XMLファイルの例＞

```
<?xml version="1.0" encoding="UTF-8"?>
<root>
 <hns:header xmlns:hns="http://header">
   <hns:id>12345 </hns:id>
   <hns:name>ADDRESS</hns:name>
 </hns:header>
 <dns:datas xmlns:dns="http://data">
   <dns:data>
     <dns:name>John</dns:name>
     <dns:country>USA</dns:country>
   </dns:data>
   <dns:data>
     <dns:name>Mary</dns:name>
     <dns:country>UK</dns:country>
   </dns:data>
 </dns:datas>
 <tns:trailer xmlns:tns="http://trailer">
   <tns:id>12345</tns:id>
   <tns:name>ADDRESS</tns:name>
 </tns:trailer>
</root>
```

ヘッダレコードに対応する要素

データレコードに対応する要素

トレーラレコードに対応する要素

＜cscFileOperation.properties（ファイル操作アダプタ定義ファイル）の設定例＞

```
csc.adapter.fileOperation.transform.header.inElement =
root/{http://header}header

csc.adapter.fileOperation.transform.data.inElement =
root/{http://data}datas/{http://data}data

csc.adapter.fileOperation.transform.trailer.inElement =
root/{http://trailer}trailer
```

(f)　レコードのマージ（出力ファイルが XML 形式の場合）

入力ファイルが分割処理方式の場合，各レコードの変換結果を 1 つの XML ファイルにマージするための出力用テンプレート XML ファイルを使用します。

出力用テンプレート XML ファイルの内容は，「(b) 対象とするデータフォーマット」で作成した出力 XML スキーマに沿った形式である必要があります。整形式の XML 文書でない場合，変換時にエラーとなります。

出力用テンプレート XML ファイルの仕様を次の表に示します。

表 8-20　出力用テンプレート XML ファイルの仕様

項目	仕様
ファイル名	ファイル操作アダプタ定義ファイルの csc.adapter.fileOperation.transform.outputTemplateXmlFile プロパティで指定したファイル名
文字エンコーディング	UTF-8

出力用テンプレート XML ファイル中の特定の要素をレコード（出力データ）に置き換えて，出力 XML ファイルが作成されます。

レコードに置き換える要素は，ファイル操作アダプタ定義ファイルで指定します。ファイル操作アダプタ定義ファイルの設定項目と設定内容を次の表に示します。

表 8-21　ファイル操作アダプタ定義ファイルの設定（出力 XML ファイルのマージ）

設定項目	設定内容
csc.adapter.fileOperation.transform.header.outElement	出力 XML ファイルのヘッダレコードに対応する要素のパスを指定します。 ヘッダレコードを使用しない場合，指定は不要です（指定しても無視されます）。
csc.adapter.fileOperation.transform.data.outElement	出力 XML ファイルのデータレコードに対応する要素のパスを指定します。
csc.adapter.fileOperation.transform.trailer.outElement	出力 XML ファイルのトレーラレコードに対応する要素のパスを指定します。 トレーラレコードを使用しない場合，指定は必要ありません（指定しても無視されます）。

出力用テンプレート XML ファイル，およびファイル操作アダプタ定義ファイルの設定例を次の図に示します。

図 8-47　出力用テンプレート XML ファイルとファイル操作アダプタ定義ファイルの設定例

＜出力用テンプレートXMLファイルの例＞

```
<?xml version="1.0" encoding="UTF-8"?>
<root>
  <hns:header xmlns:hns="http://header">
    <hns:id/>
    <hns:name/>
  </hns:header>
  <dns:datas xmlns:dns="http://data">
    <dns:data>
      <dns:name/>
      <dns:country/>
    </dns:data>
  </dns:datas>
  <tns:trailer xmlns:tns="http://trailer">
    <tns:id/>
    <tns:name/>
  </tns:trailer>
</root>
```

ヘッダレコードに対応する要素

データレコードに対応する要素

トレーラレコードに対応する要素

＜cscFileOperation.properties（ファイル操作アダプタ定義ファイル）の設定例＞

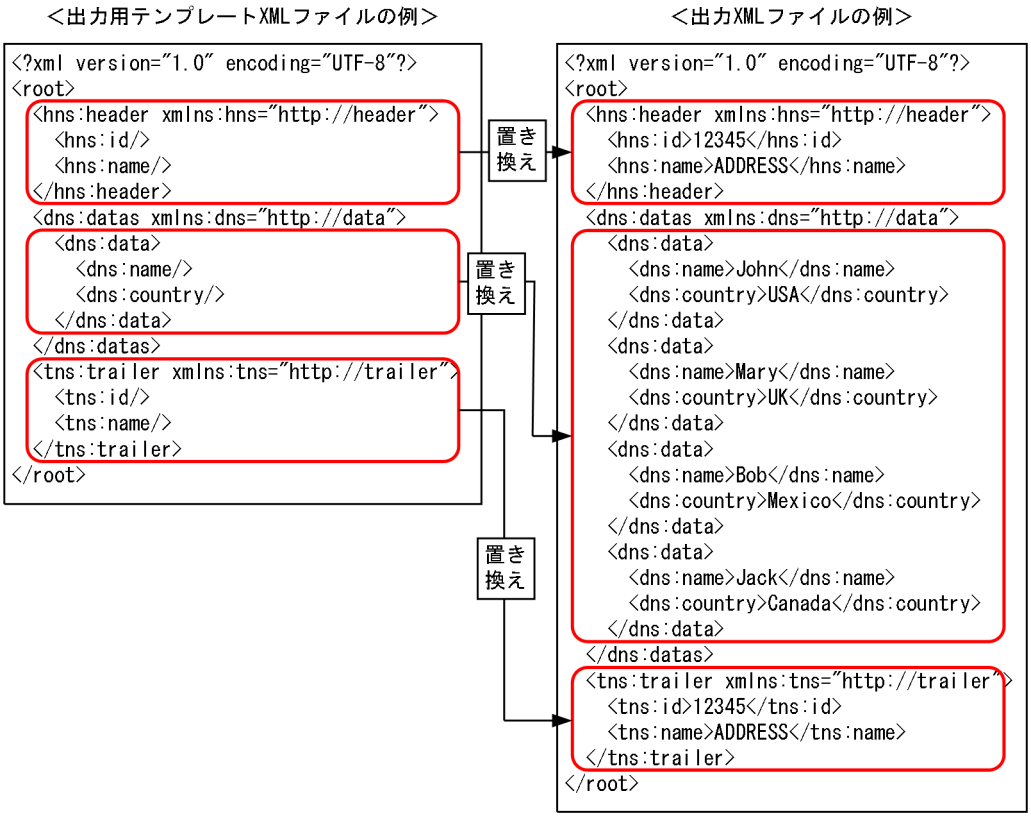
```
csc.adapter.fileOperation.transform.header.outElement =
root/{http://header}header

csc.adapter.fileOperation.transform.data.outElement =
root/{http://data}datas/{http://data}data

csc.adapter.fileOperation.transform.trailer.outElement =
root/{http://trailer}trailer
```

出力用テンプレート XML ファイルを出力 XML ファイルに変換した例を次の図に示します。なお，この例では出力 XML ファイルの改行や空白を表示していますが，実際の出力 XML ファイルでは，要素の間の改行や空白は削除されます。

図 8-48　XML ファイルの出力例



出力用テンプレート XML ファイルを使用した出力 XML ファイルは，次の表に示す形式で出力されます。

表 8-22　出力 XML ファイルの形式

項目	仕様
文字エンコーディング	UTF-8
要素の間のスペース	削除されます。
要素の間の改行	削除されます。
名前空間宣言	• 出力用テンプレート XML ファイルの名前空間宣言を引き継ぎます。 • 未使用の名前空間宣言は削除されません。 • 不足している名前空間宣言は，入力 XML の宣言で補われます。
ファイルの内容	出力 XML スキーマに沿った XML が出力されます。出力 XML スキーマとの整合性は検証されません。

注意事項

- XML Processor がサポートしていない文字エンコーディングが指定されている場合，解析処理でエラーとなります。XML Processor がサポートしている文字エンコーディングについては，マニュアル「XML Processor ユーザーズガイド」の「1.3.2　処理できる文字コード」を参照してください。

- 出力用テンプレート XML ファイルに，ヘッダ，データ，トレーラのどのレコードにも該当しない要素が定義されている場合，値も含めてそのまま出力されます。値の前後に空白がある場合も空白は取り除かれません。
- データレコードは，最初に出現した要素，および連続して出現する兄弟要素が 1 つのデータレコード群として扱われます。データレコード群が終了したあとに再びデータレコードが出現した場合，エラーとなります。
- 出力用テンプレート XML ファイルの内容が「(b) 対象とするデータフォーマット」で作成した出力 XML スキーマに沿った形式でない場合，レコードに置き換える要素に出力用テンプレート XML ファイルに存在する正しい値を指定しても，レコードの変換結果は置き換えられないで処理が終了します。この場合，実行時エラーとなりません。このため，入力データがデータ変換定義で出力されるように設定していても，レコードの変換結果が出力ファイルに出力されない場合は，出力 XML スキーマと出力用テンプレート XML ファイルの内容を見直してください。

(g)　複数のデータレコードの同時変換

ファイル変換オペレーションでは，分割処理方式を選択した場合に，複数のデータレコードをまとめて 1 要素として変換することができます。入力ファイルがバイナリ形式または XML 形式のどちらの場合でも変換できます。

入力ファイルがバイナリ形式，かつデータレコードの総数が 6 で，2 つのデータレコードを 1 要素として変換する場合の例を次に示します。

図 8-49　2 つのデータレコードを 1 要素として変換する例（固定長形式の場合）

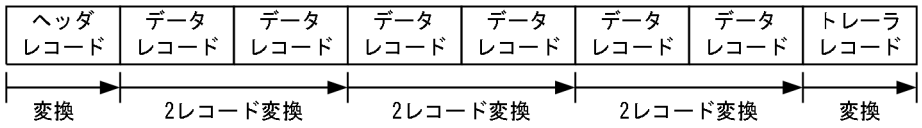
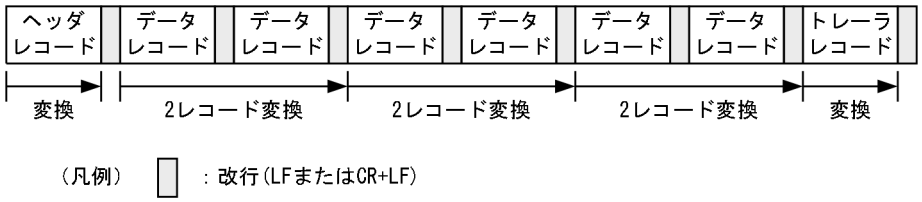


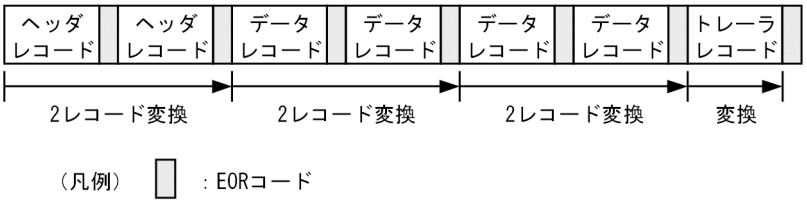
図 8-50　2 つのデータレコードを 1 要素として変換する例（改行セパレータ形式の場合）



参考

改行セパレータ形式の場合，データレコードの改行を含めて変換されます。

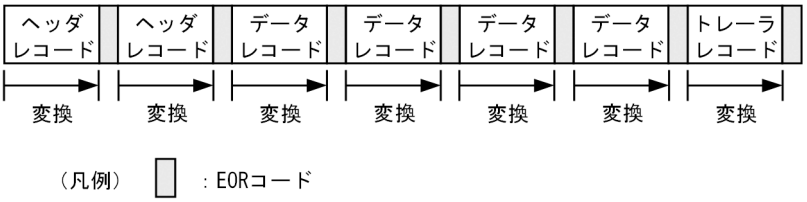
図 8-51　2 つのデータレコードを 1 要素として変換する例（EOR セパレータ形式の場合）



参考

EOR セパレータ形式の場合，データレコードの EOR コードを含めて変換されます。そのため，入力ファイルのバイナリフォーマット定義では，EOR コードを含めたデータ構造を定義する必要があります。なお，1 レコードずつ変換する場合は，変換対象のデータに EOR コードは含まれません。

図 8-52　1 レコードずつ変換する例（EOR セパレータ形式の場合）



1 要素（同時に変換するレコード数）の設定は，ファイル操作アダプタ定義ファイルの次のパラメタで指定します。

- csc.adapter.fileOperation.transform.data.batchSize

パラメタの内容については，マニュアル「サービスプラットフォーム リファレンス」の「3.5.2　ファイル操作アダプタ定義ファイル」を参照してください。

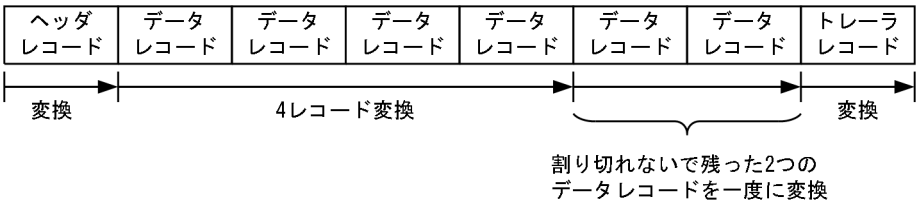
注意事項

複数のデータレコードを同時変換すると変換効率の向上が図れますが，1 要素を大きくしてしまうと処理性能が劣化することがあります。1 要素当たりのデータサイズの上限は 500MB を目安にして，同時に変換するレコード数を適切に設定してください。

データレコードの総数が，パラメタに設定した値で割り切れなかった場合は，余ったレコードを一度に変換します。

変換例として，データレコードの総数が 6 で，4 つのデータレコードを 1 要素として変換する場合の例を次に示します。

図 8-53　1 要素に指定した値で割り切れなかった場合の変換例



(h)　複数のヘッダレコードの同時変換

次の条件をすべて満たす場合，複数のヘッダレコードをまとめて 1 要素として変換することができます。

- 入力ファイルのデータ種別がバイナリ形式である
- 分割処理方式を指定している
- レコードの判定方式に改行セパレータ形式，または EOR セパレータ形式を指定している
- ヘッダレコードが存在する

変換するヘッダレコード数は，ファイル操作アダプタ定義ファイルの csc.adapter.fileOperation.transform.header.batchSize プロパティで指定できます。

次の条件では変換時にエラーになる場合があります。

- データ中のヘッダレコードが多い

  ファイル操作アダプタ定義ファイルの csc.adapter.fileOperation.transform.header.batchSize プロパティの指定より入力データのヘッダレコードの数が多い場合，ヘッダレコードの一部がデータレコードと見なされ，変換エラーになることがあります。
- データ中のヘッダレコードが少ない

  ファイル操作アダプタ定義ファイルの csc.adapter.fileOperation.transform.header.batchSize プロパティの指定より入力データのヘッダレコードの数が少ない場合，データレコードの一部がヘッダレコードと見なされ，変換エラーになることがあります。
- 入力データの全レコード数が指定したレコードに満たない

  入力データ中の全レコード数が，ファイル操作アダプタ定義ファイルの csc.adapter.fileOperation.transform.header.batchSize プロパティの指定に満たない場合は，エラー（KDEC80047-E）が発生します。

(5)　変換するファイル形式の指定方法

ファイル変換オペレーションで読み込むファイルや書き込むファイルの形式は，あらかじめバイナリフォーマット定義画面でバイナリフォーマット定義ファイルを作成したあと，ファイル操作アダプタの独自定義ファイルとして指定してください。

バイナリフォーマット定義画面の詳細は，マニュアル「サービスプラットフォーム リファレンス」の「1.2.1　バイナリフォーマット定義画面」を参照してください。

バイナリフォーマット定義ファイルの作成については，マニュアル「サービスプラットフォーム 開発ガイド 基本開発編」の「4.4　電文フォーマット（バイナリフォーマット定義ファイル）の作成方法」を参照してください。

ファイル操作アダプタの独自定義ファイルの指定については，マニュアル「サービスプラットフォーム リファレンス」の「3.5.2　ファイル操作アダプタ定義ファイル」を参照してください。

(6)　データ変換定義の指定方法

あらかじめ作成した入力ファイルから出力ファイルへ変換するデータ変換定義ファイル（xsl ファイル）を，ファイル操作アダプタの独自定義ファイルとして指定します。

開発環境のマッピング定義エディタでマッピング定義ファイルを作成するときに，データ変換定義を利用して xsl ファイルを作成してください。ファイル操作アダプタの独自定義ファイルの指定については，マニュアル「サービスプラットフォーム リファレンス」の「3.5.2　ファイル操作アダプタ定義ファイル」を参照してください。

(7)　文字コード変換 UOC を使用したファイルの変換

ファイル変換オペレーションでは，ファイルの変換に文字コード変換 UOC を使用できます。

文字コード変換 UOC を使用する場合は，サービスアダプタ定義画面（詳細）の独自定義ファイルに，次のファイルを設定する必要があります。

1. 入力ファイルと出力ファイルの変換で同じ独自定義ファイルを使用する場合
   - csc_owncodeconvert.properties
2. 入力ファイルと出力ファイルの変換で別々の独自定義ファイルを使用する場合
   - csc_owncodeconvert_in.properties
   - csc_owncodeconvert_out.properties

1.で示したファイルと 2.で示したファイルとを同時に指定した場合，1.で示したファイルが優先され，2.で示したファイルは無視されます。

サービスアダプタ定義画面（詳細）の設定方法については，マニュアル「サービスプラットフォーム 開発ガイド 受付・アダプタ定義編」の「3.3.11　ファイル操作アダプタを定義する」のファイル変換オペレーションについて説明している個所を参照してください。

文字コード変換 UOC の概要および実装方法については，マニュアル「サービスプラットフォーム 開発ガイド 基本開発編」の「付録 H　文字コード変換 UOC を利用した文字コード変換」を参照してください。

(8)　分割処理方式での独自定義ファイルの指定

ファイル変換オペレーションで分割処理方式の変換をする場合に必要な独自定義ファイル，および指定が必要なファイル操作アダプタ定義ファイルのプロパティの一覧を次の表に示します。ファイル操作アダプ

タ定義ファイルの詳細は，マニュアル「サービスプラットフォーム リファレンス」の「3.5.2　ファイル操作アダプタ定義ファイル」を参照してください。

表 8-23　分割処理方式で指定が必要な独自定義ファイルとファイル操作アダプタ定義ファイルのプロパティ

独自定義ファイルとして指定するファイル		入力ファイル		出力ファイル	
		バイナリ	XML	バイナリ	XML
ヘッダ用レイアウト変換用 XSLT ファイル		○※1	○※1	○※1	○※1
データ用レイアウト変換用 XSLT ファイル		○	○	○	○
トレーラ用レイアウト変換用 XSLT ファイル		○※2	○※2	○※2	○※2
入力ヘッダ用 FDX ファイル		○※1	－	－	－
入力データ用 FDX ファイル		○	－	－	－
入力トレーラ用 FDX ファイル		○※2	－	－	－
入力ヘッダ用 XML スキーマファイル		－	－	－	－
入力データ用 XML スキーマファイル		－	－	－	－
入力トレーラ用 XML スキーマファイル		－	－	－	－
ファイル操作アダプタ定義ファイルに指定するプロパティ	csc.adapter.fileOperation.transform.input （入力ファイル種別）	○	○	－	－
	csc.adapter.fileOperation.transform.output （出力ファイル種別）	－	－	○	○
	csc.adapter.fileOperation.transform.headerRecord （ヘッダレコード使用有無）	○※1	○※1	○※1	○※1
	csc.adapter.fileOperation.transform.trailerRecord （トレーラレコード使用有無）	○※2	○※2	○※2	○※2
	csc.adapter.fileOperation.transform.header.inElement （入力ヘッダ要素名）	－	○※1	－	－
	csc.adapter.fileOperation.transform.data.inElement （入力データ要素名）	－	○	－	－
	csc.adapter.fileOperation.transform.trailer.inElement （入力トレーラ要素名）	－	○※2	－	－
	csc.adapter.fileOperation.transform.header.outElement （出力ヘッダ要素名）	－	－	－	○※1
	csc.adapter.fileOperation.transform.data.outElement （出力データ要素名）	－	－	－	○
	csc.adapter.fileOperation.transform.trailer.outElement	－	－	－	○※2

独自定義ファイルとして指定するファイル		入力ファイル		出力ファイル	
		バイナリ	XML	バイナリ	XML
ファイル操作アダプタ定義ファイルに指定するプロパティ	（出力トレーラ要素名）	－	－	－	○※2
	csc.adapter.fileOperation.transform.outputTemplateXmlFile （出力用テンプレート XML ファイルのファイル名）	－	－	－	○
出力ヘッダ用 FDX ファイル		－	－	○※1	－
出力データ用 FDX ファイル		－	－	○	－
出力トレーラ用 FDX ファイル		－	－	○※2	－
出力ヘッダ用 XML スキーマファイル		－	－	－	－
出力データ用 XML スキーマファイル		－	－	－	－
出力トレーラ用 XML スキーマファイル		－	－	－	－
出力用テンプレート XML ファイル		－	－	－	○
文字コード変換 UOC 用入力プロパティ（入力・出力ファイル共通） csc_owncodeconvert.properties		○※3	－	○※3	－
文字コード変換 UOC 用入力プロパティ（入力ファイル変換用） csc_owncodeconvert_in.properties		○※3	－	－	－
文字コード変換 UOC 用入力プロパティ（出力ファイル変換用） csc_owncodeconvert_out.properties		－	－	○※3	－

（凡例）

○：必ず指定します。

－：指定は不要です。

注※1

ヘッダレコードを使用する場合だけ指定します。

注※2

トレーラレコードを使用する場合だけ指定します。

注※3

文字コード変換 UOC を使用する場合に指定します。

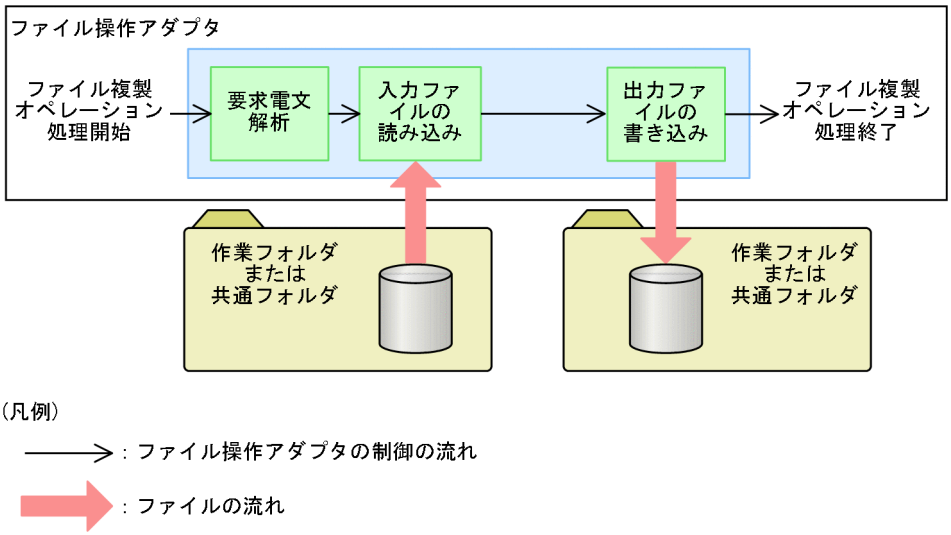
8.7.4 ファイル複製オペレーション

ここでは，ファイル複製オペレーションでサポートする機能やファイル形式について説明します。

(1)　ファイル複製オペレーションの機能概要

ファイル複製オペレーションは，作業フォルダまたは共通フォルダから複製元の入力ファイルを読み込み，複製後の出力ファイルを作業フォルダまたは共通フォルダに書き込みます。ファイル複製オペレーションの処理の概要について，次の図に示します。

図 8-54　ファイル複製オペレーションの処理



(2)　サポートするファイル形式

ファイル複製オペレーションの複製対象となる入力ファイル，および出力ファイルのファイル形式は問いません。作業フォルダまたは共通フォルダ内に存在するファイルはすべて複製できます。

8.7.5　ファイル移動オペレーション

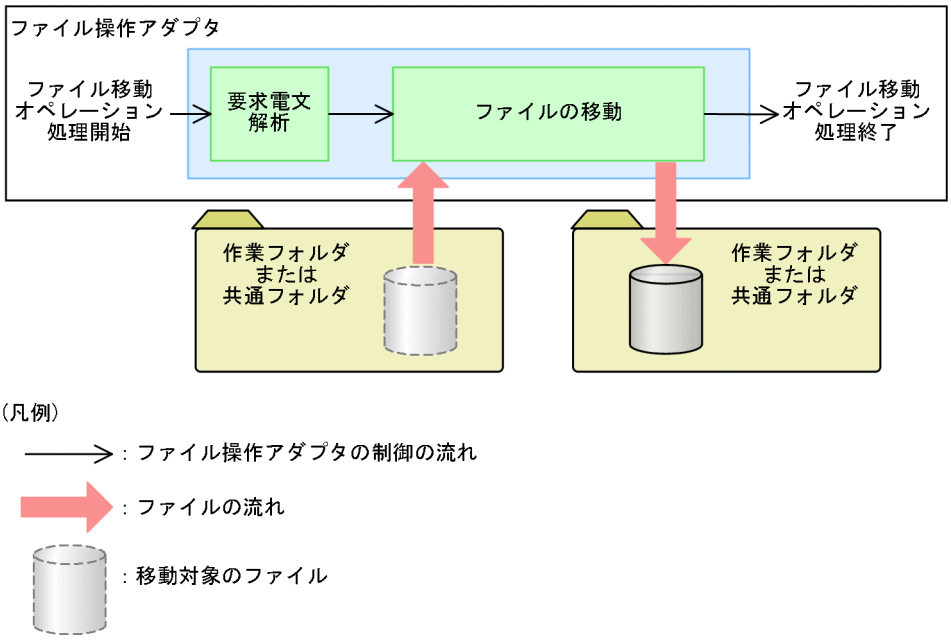
ここでは，ファイル移動オペレーションでサポートする機能やファイル形式について説明します。

(1)　ファイル移動オペレーションの機能概要

ファイル移動オペレーションは，作業フォルダまたは共通フォルダのファイルを，指定した作業フォルダまたは共通フォルダへ移動します。

ファイル移動オペレーションの処理の概要について，次の図に示します。

図 8-55　ファイル移動オペレーションの処理



（凡例）

：ファイル操作アダプタの制御の流れ

：ファイルの流れ

：移動対象のファイル

(2)　サポートするファイル形式

ファイル移動オペレーションの移動対象となるファイルのファイル形式は問いません。作業フォルダまたは共通フォルダ内に存在するファイルはすべて移動できます。

ディレクトリは移動できません。

(3)　ファイル移動オペレーションの注意事項

ファイル移動オペレーションの注意事項を次に示します。

- 移動先のファイルが共通フォルダにある場合でも，占有ロックは取得しません。
- 移動に成功したファイルは移動元の最終更新日時を保持します。
- 別ドライブ間の移動に要する処理時間はファイルシステムの性質上，コピー&削除になるため，ファイル複製オペレーションよりも長くなります。

8.7.6　ファイル・フォルダ削除オペレーション

ここでは，ファイル・フォルダ削除オペレーションでサポートする機能やファイル形式について説明します。

(1)　ファイル・フォルダ削除オペレーションの機能概要

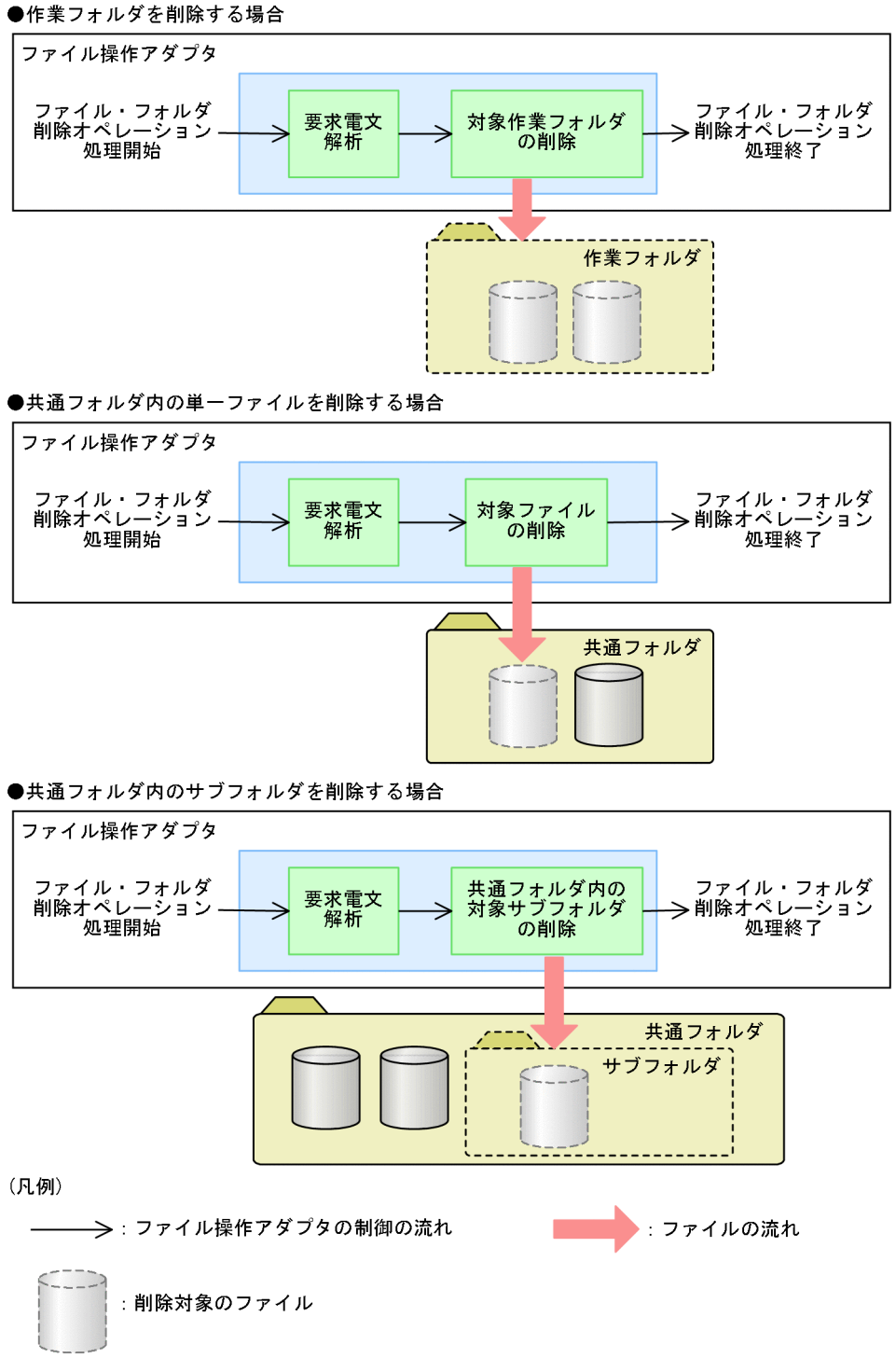
ファイル・フォルダ削除オペレーションは，次の処理を実施します。

- 作業フォルダの削除

- 共通フォルダ内の単一ファイルの削除
- 共通フォルダ内のサブフォルダの削除

ファイル・フォルダ削除オペレーションの処理の概要について，次の図に示します。

図 8-56　ファイル・フォルダ削除オペレーションの処理



(2) サポートするファイル形式

ファイル削除オペレーションの削除対象となるファイルの形式は問いません。共通フォルダ内に存在する任意のファイルを削除できます。

(3) 作業フォルダの削除

フォルダ削除オペレーションでは，要求電文で指定したリクエスト ID に基づいて生成される作業フォルダを削除します。対象の作業フォルダ以下のファイルおよびフォルダはすべて削除されます。

(4) 共通フォルダ内の単一ファイルの削除

要求電文で指定した共通フォルダ内のファイルを削除します。削除されるのは，対象の共通フォルダ内の指定したファイル 1 つだけです。

(5) 共通フォルダ内のサブフォルダの削除

要求電文で指定した共通フォルダ内のサブフォルダを削除します。削除されるのは，対象の共通フォルダ内の指定したサブフォルダ 1 つだけです。

(6) ファイル・フォルダ削除オペレーションの注意事項

ファイル・フォルダ削除オペレーションの注意事項を次に示します。

- フォルダを削除するオペレーションを，削除対象のフォルダ下のファイルをアクセス中に，実行しないでください。動作を保証しません。

8.7.7 ファイル圧縮オペレーション

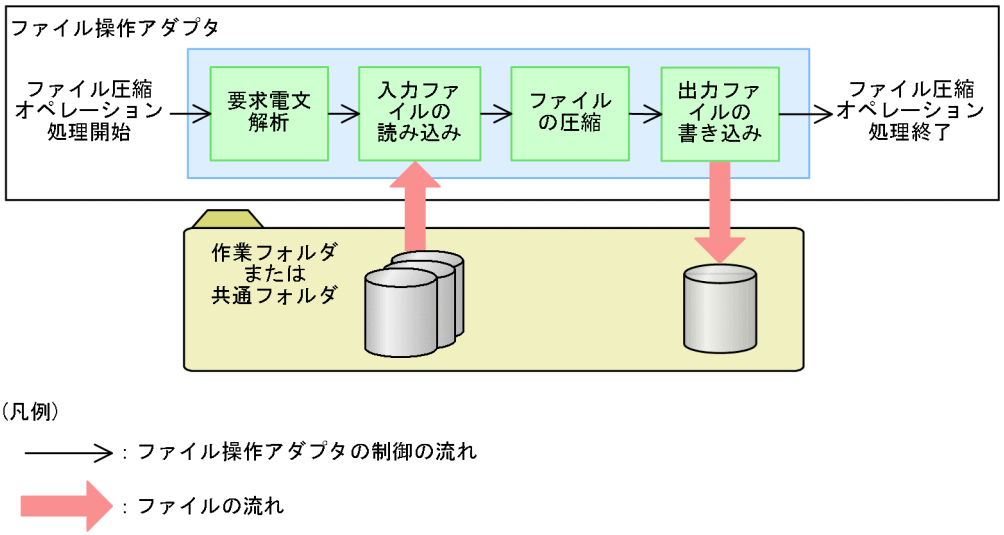
ここでは，ファイル圧縮オペレーションでサポートする機能やファイル形式について説明します。

(1) ファイル圧縮オペレーションの機能概要

ファイル圧縮オペレーションは，作業フォルダまたは共通フォルダから圧縮元の入力ファイルを読み込み，圧縮後の出力ファイルを作業フォルダまたは共通フォルダに書き込みます。

ファイル圧縮オペレーションの処理の概要について，次の図に示します。

図 8-57　ファイル圧縮オペレーションの処理



（凡例）

：ファイル操作アダプタの制御の流れ

：ファイルの流れ

(2)　サポートするファイル形式

ファイル圧縮オペレーションでサポートするファイル形式を次に示します。なお，ディレクトリの圧縮はサポートしていません。入力ファイルにディレクトリを指定した場合，エラーメッセージ（KDEC80035-E）が出力され，ファイル圧縮オペレーションが中止されます。

表 8-24　ファイル圧縮オペレーションでサポートするファイル形式

ファイル形式	入力ファイル	出力ファイル
ZIP 形式※	○	○
上記以外	○	×

（凡例）

○：サポートしています。

×：サポートしていません。

注※

DEFLATE 圧縮アルゴリズムを使用して圧縮された ZIP 形式だけをサポートします。

ファイル圧縮オペレーションを使用する際の，入出力ファイルに関する制限事項を次に示します。

表 8-25　ファイル圧縮オペレーション使用時の制限事項

項目	制限事項
入力ファイル名	次に示す半角の英数字と記号だけが使用できます。 • 英数字 英字（a～z，A～Z）と数字（0～9）を使用できます。 なお，サブフォルダ名については，上記の制限はありません。
出力ファイルのエントリ名	制限事項は入力ファイル名と同様です。

項目	制限事項
入力ファイル数	入力ファイル数は 1～999 の範囲で指定できます。これを満たさない場合の動作は保証しません。 単一の入力元フォルダに存在する複数の入力ファイルを圧縮できますが，複数の入力元フォルダを指定することはできません。
入力ファイルのファイルサイズ	入力ファイルの最大ファイルサイズの初期値は 4GB です。 入力ファイルの最大ファイルサイズを設定するには，ファイルの圧縮を打ち切る入力ファイルのサイズのパラメタ（ファイル操作アダプタ実行環境プロパティファイルの compress.inputfile.limitsize）を指定してください。
出力ファイルのファイルサイズ	出力ファイルの最大ファイルサイズの初期値は 4GB です。 出力ファイルの最大ファイルサイズを設定するには，ファイルの圧縮を打ち切る出力ファイルのサイズのパラメタ（ファイル操作アダプタ実行環境プロパティファイルの compress.outputfile.limitsize）を指定してください。

ファイル操作アダプタ実行環境プロパティファイルの詳細は，マニュアル「サービスプラットフォーム リファレンス」の「6.8.8　ファイル操作アダプタ実行環境プロパティファイル」を参照してください。

8.7.8　ファイル伸張オペレーション

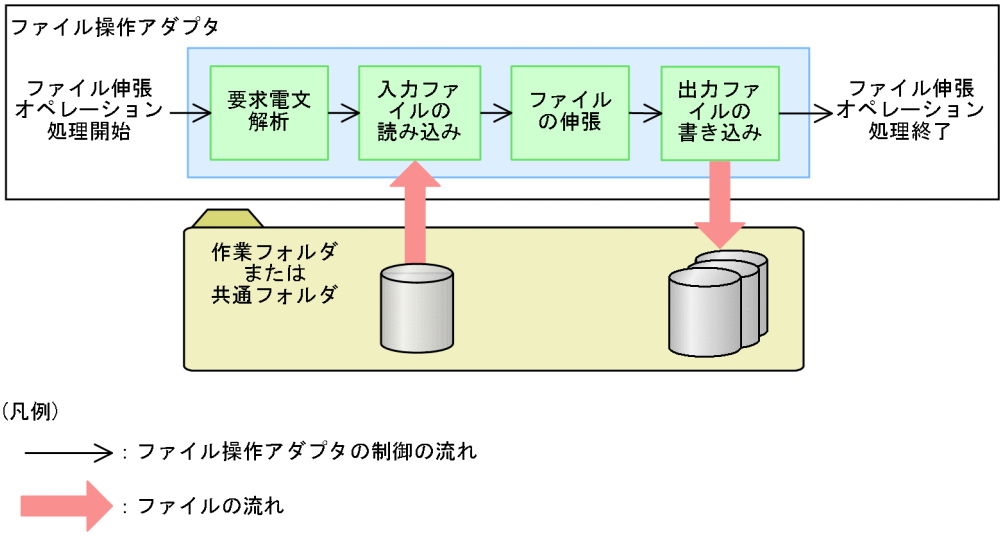
ここでは，ファイル伸張オペレーションでサポートする機能やファイル形式について説明します。

(1)　ファイル伸張オペレーションの機能概要

ファイル伸張オペレーションは，作業フォルダまたは共通フォルダから伸張元の入力ファイルを読み込み，伸張後の出力ファイルを作業フォルダまたは共通フォルダに書き込みます。

ファイル伸張オペレーションの処理の概要を次の図に示します。

図 8-58　ファイル伸張オペレーションの処理



(2) サポートするファイル形式

ファイル伸張オペレーションでサポートするファイル形式を次に示します。ディレクトリが格納された ZIP ファイルの伸張はサポートしていません。入力ファイルにディレクトリが格納された ZIP ファイルを指定した場合，エラーメッセージ（KDEC80123-E）を出力し，ファイル伸張オペレーションが中止されます。

表 8-26　ファイル伸張オペレーションでサポートするファイル形式

ファイル形式	入力ファイル	出力ファイル
ZIP 形式※	○	○
上記以外	×	○

（凡例）

○：サポートしています。

×：サポートしていません。

注※

DEFLATE 圧縮アルゴリズムを使用して圧縮された ZIP 形式だけをサポートします。

ファイル伸張オペレーションを使用する際の，入出力ファイルに関する制限事項を次に示します。

表 8-27　ファイル伸張オペレーション使用時の制限事項

項目	制限事項
入力ファイルのエントリ名	次に示す半角の英数字と記号だけが使用できます。 • 英数字 英字（a～z，A～Z）と数字（0～9）を使用できます。
出力ファイル名	出力ファイル名は，要求電文に定義された出力ファイル名の接頭辞を使用して自動的に付与されます。
出力ファイル数	伸張後の出力ファイル数は 1～999 を制限とします。これを満たさない場合，エラーメッセージ（KDEC80125-E）を出力し，ファイル伸張オペレーションを中止します。 複数の出力ファイルを単一の出力先フォルダへ伸張することはできますが，複数の出力先フォルダを指定することはできません。
入力ファイルのファイルサイズ	入力ファイルの最大ファイルサイズの初期値は 4GB です。 入力ファイルの最大ファイルサイズを設定するには，ファイルの伸張を打ち切る入力ファイルのサイズのパラメタ（ファイル操作アダプタ実行環境プロパティファイルの extract.inputfile.limitsize）を指定してください。
出力ファイルのファイルサイズ	出力ファイルの最大ファイルサイズの初期値は 4GB です。 出力ファイルの最大ファイルサイズを設定するには，ファイルの伸張を打ち切る出力ファイルのサイズのパラメタ（ファイル操作アダプタ実行環境プロパティファイルの extract.outputfile.limitsize）を指定してください。

ファイル操作アダプタ実行環境プロパティファイルの詳細は，マニュアル「サービスプラットフォーム リファレンス」の「6.8.8　ファイル操作アダプタ実行環境プロパティファイル」を参照してください。

8.7.9　ファイル一覧取得オペレーション

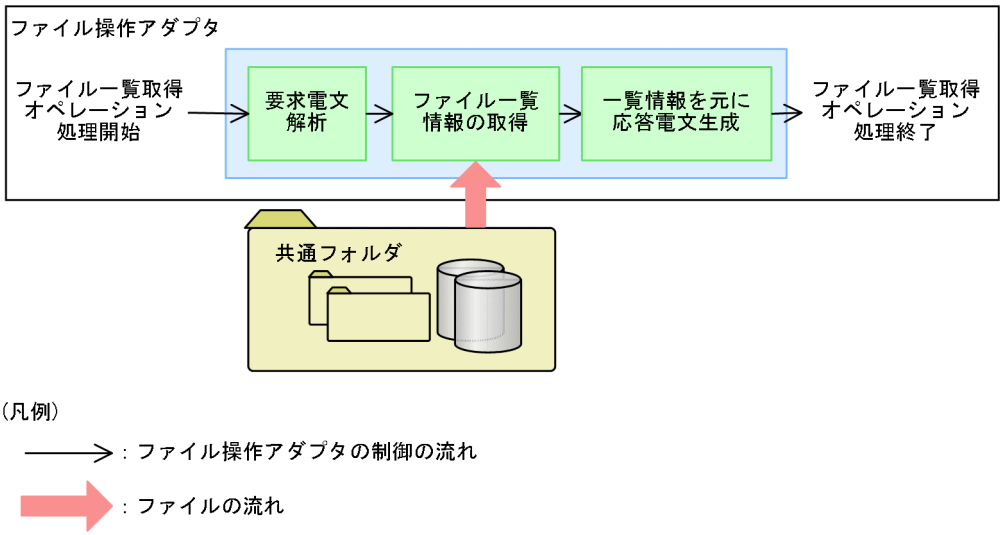
ここでは，ファイル一覧取得オペレーションでサポートする機能やファイル形式について説明します。

(1)　ファイル一覧取得オペレーションの機能概要

ファイル一覧取得オペレーションは，共通フォルダからファイル一覧情報を取得し，一覧情報を応答電文で返します。一覧取得元にファイルが指定された場合は，指定されたファイルだけを返します。一覧取得元にディレクトリが指定された場合は，指定されたディレクトリの直下のファイル・ディレクトリだけを返します。

ファイル一覧取得オペレーションの処理の概要について，次の図に示します。

図 8-59　ファイル一覧取得オペレーションの処理



（凡例）

：ファイル操作アダプタの制御の流れ

：ファイルの流れ

(2)　一覧取得元に指定可能な対象

ファイル一覧取得オペレーションで一覧取得元に指定可能な対象を次に示します。

表 8-28　一覧取得元に指定可能な対象

一覧取得元に指定可能な対象		サポート
作業フォルダ	ファイル	×
	作業フォルダ自体	×
共通フォルダ	ファイル	○
	共通フォルダ自体	○
	サブフォルダ	○

（凡例）

○：サポートしています。

×：サポートしていません。

(3)　一覧取得で取得可能な対象

一覧取得元の共通フォルダで取得可能な対象を次に示します。

表 8-29　一覧取得で取得可能な対象

一覧取得で取得可能な対象		サポート
直下のファイル	通常ファイル	○
	隠しファイル	○
	シンボリックリンク	○
直下のサブフォルダ※1	通常ディレクトリ	○
	隠しディレクトリ※2	○
	シンボリックリンク	○

（凡例）

○：サポートしています。

注※1

再帰的な取得はしません。

注※2

「.」（カレントディレクトリ），「..」（1 つ上のディレクトリ）は対象外です。

(4)　ファイル一覧取得オペレーションの注意事項

ファイル一覧取得オペレーションの注意事項を次に示します。

- 一覧取得元がディレクトリの場合，一覧取得元ディレクトリとその直下のファイル・ディレクトリのロックは取得しません。
- ファイル一覧の最終更新日時はローカルタイムとなります。
- 一覧取得元が存在しない場合，空の一覧情報を返します。

8.7.10　ファイル一覧書き込みオペレーション

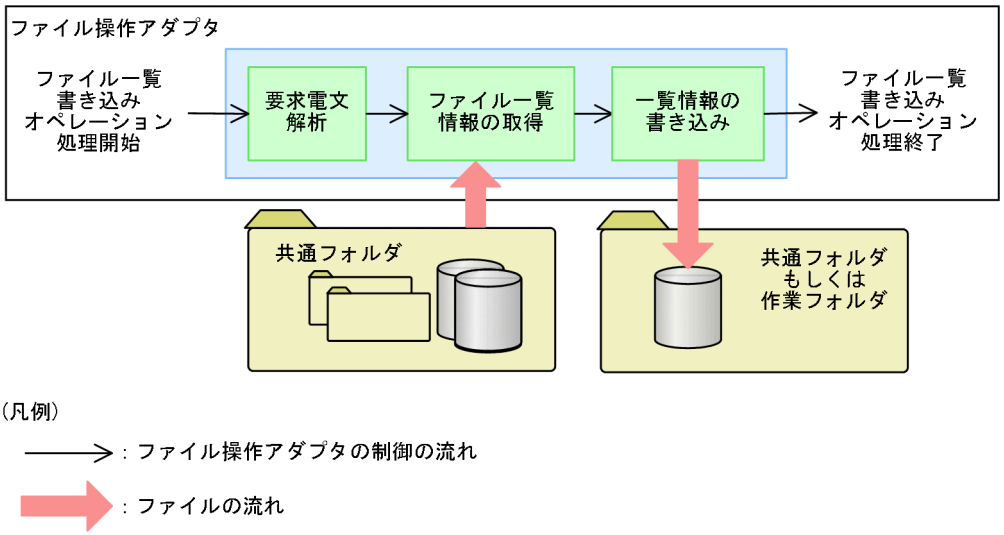
ここでは，ファイル一覧書き込みオペレーションでサポートする機能やファイル形式について説明します。

(1)　ファイル一覧書き込みオペレーションの機能概要

ファイル一覧書き込みオペレーションは，共通フォルダからファイル一覧情報を取得し，一覧情報をファイルに書き込みます。一覧取得元にファイルが指定された場合は，指定されたファイルだけを返します。一覧取得元にディレクトリが指定された場合は，指定されたディレクトリの直下のファイル・ディレクトリだけを返します。

ファイル一覧書き込みオペレーションの処理の概要について，次の図に示します。

図 8-60　ファイル一覧書き込みオペレーションの処理



（凡例）

：ファイル操作アダプタの制御の流れ

：ファイルの流れ

(2)　一覧取得元に指定可能な対象

ファイル一覧書き込みオペレーションで一覧取得元に指定可能な対象を次に示します。

表 8-30　一覧取得元に指定可能な対象

一覧取得元に指定可能な対象		サポート
作業フォルダ	ファイル	×
	作業フォルダ自体	×
共通フォルダ	ファイル	○
	共通フォルダ自体	○
	サブフォルダ	○

（凡例）

○：サポートしています。

×：サポートしていません。

(3)　一覧取得で取得可能な対象

一覧取得元の共通フォルダで取得可能な対象を次に示します。

表 8-31　一覧取得で取得可能な対象

一覧取得で取得可能な対象		サポート
直下のファイル	通常ファイル	○
	隠しファイル	○

一覧取得で取得可能な対象		サポート
直下のファイル	シンボリックリンク	○
直下のサブフォルダ※1	通常ディレクトリ	○
	隠しディレクトリ※2	○
	シンボリックリンク	○

（凡例）

○：サポートしています。

注※1

再帰的な取得はしません。

注※2

「.」（カレントディレクトリ），「..」（1 つ上のディレクトリ）は対象外です。

(4) 出力するファイル形式

出力するファイルの文字コードは，UTF-16（ビッグエンディアン，BOM あり）です。取得したファイル一覧が空の場合は，BOM だけを出力します。

全項目を出力する場合（要求電文 list-option で l が指定された場合）のファイル出力形式を次に示します。

表 8-32 ファイル出力形式（全項目）

列順	列名	列サイズ（バイト）	出力形式※1	左右寄せ
1	ファイル種別	2※2	f もしくは d※3	左
2	ファイルサイズ	40※2	ファイルサイズの数値文字列※4	右
3	最終更新日時	出力形式のバイト※2	ユーザ指定形式	左
4	ファイル名	可変長※5	ファイル名文字列	左

注※1

埋め時は半角スペース。

注※2

別途，末尾に半角スペース（2 バイト）が必ず 1 個付与されます。

注※3

f：ファイル，d：ディレクトリ。

注※4

ディレクトリの場合，値は 0 になります。

注※5

別途，末尾に OS 依存の改行コードが付与されます。

ファイル名だけを出力する場合（要求電文 list-option で l が指定された場合）のファイル出力形式を次に示します。

表 8-33　ファイル出力形式（ファイル名のみ）

列順	列名	列サイズ（バイト）	出力形式	左右寄せ
1	ファイル名	可変長※	ファイル名文字列	左

注※

別途，末尾に OS 依存の改行コードが付与されます。

(5)　ファイル一覧書き込みオペレーションの注意事項

ファイル一覧書き込みオペレーションの注意事項を次に示します。

- 一覧取得元がディレクトリの場合，一覧取得元ディレクトリとその直下のファイル・ディレクトリのロックは取得しません。
- ファイル一覧の最終更新日時はローカルタイムとなります。
- 一覧取得元が存在しない場合，空の一覧情報を出力します。
- 最終更新日時の列サイズは要求電文 date-time-pattern 要素で指定したパターン文字列とサーバロケールに依存します。そのため，長さが可変になるパターン文字列を指定した場合（H，EEEE，d，D など），ファイルごとに列サイズが異なる可能性があります。

8.7.11　ファイル読み込みオペレーション

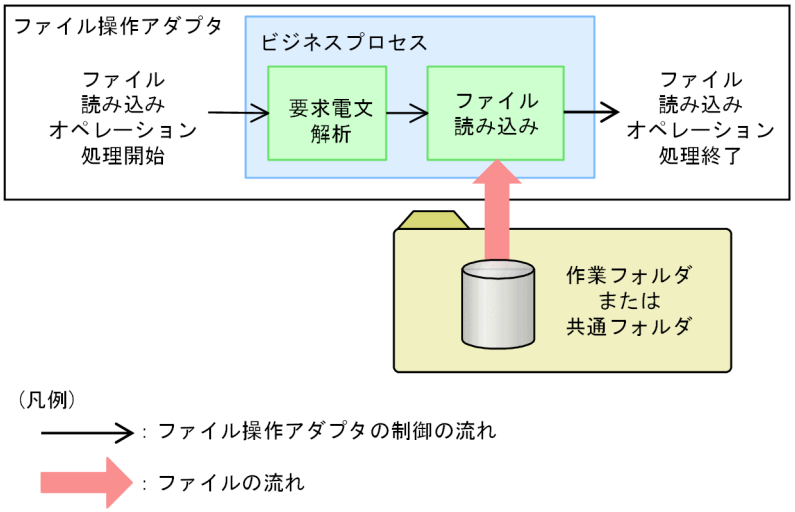
ここでは，ファイル読み込みオペレーションでサポートする機能について説明します。

(1)　ファイル読み込みオペレーションの機能概要

ファイル読み込みオペレーションは，作業フォルダ（または共通フォルダ）からファイルを読み込み，ほかのサービスに渡します。

ファイル読み込みオペレーションの処理の概要について，次の図に示します。

図 8-61　ファイル読み込みオペレーションの処理



(2)　ファイル読み込みオペレーションの注意事項

ファイル読み込みオペレーションの注意事項を次に示します。

- 入力ファイルが 0 バイトの場合は，0 バイトのデータを返します。
- 入力ファイルの内容がそのまま応答電文になるため，サービスプラットフォームではスキーマファイルを提供していません。スキーマファイルはユーザが作成してください。

8.7.12　ファイル書き込みオペレーション

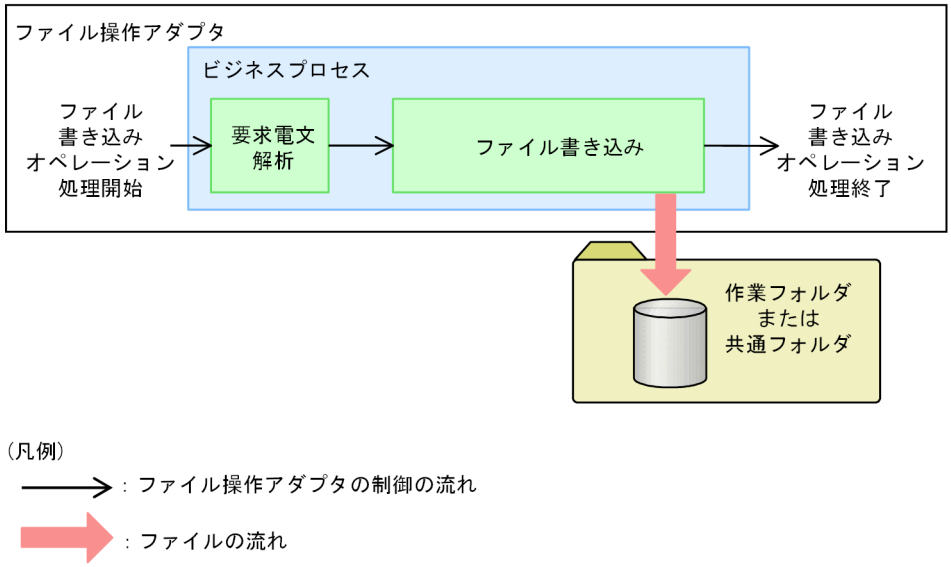
ここでは，ファイル書き込みオペレーションでサポートする機能やファイル形式について説明します。

(1)　ファイル書き込みオペレーションの機能概要

ファイル書き込みオペレーションは，他サービスから取得したデータを，作業フォルダ（または共通フォルダ）に書き込みます。

ファイル書き込みオペレーションの処理の概要について，次の図に示します。

図 8-62　ファイル書き込みオペレーションの処理



(2)　要求電文のヘッダを省略した場合の動作

要求電文のヘッダを省略した場合，ファイル操作アダプタ実行環境プロパティファイルの指定に従って，ファイルを出力します。ファイル操作アダプタ実行環境プロパティファイルには，共通フォルダだけ指定できます。

表 8-34　ファイル操作アダプタ実行環境プロパティファイルで設定するプロパティ

プロパティのキー名称	説明
write.output-folder-name	output-folder-name タグの値の代わりとなります。作業フォルダには出力できないため，common 属性の設定は指定できません。
write.output-file-name	output-file-name タグの値の代わりとなります。
write.output-mode	output-mode の値の代わりとなります。省略時は new を仮定します。
write.output-file-validation	output-file-validation タグの値の代わりとなります。省略時は OFF を仮定します。

ファイル操作アダプタ実行環境プロパティファイルの詳細は，マニュアル「サービスプラットフォーム リファレンス」の「6.8.8　ファイル操作アダプタ実行環境プロパティファイル」を参照してください。

なお，ヘッダが指定されている場合は，ファイル操作アダプタ実行環境プロパティファイルの指定は参照されません。

ファイル操作アダプタ実行環境プロパティファイルを参照した場合，次のどれかのときはエラーメッセージ（KDEC80161-E）が出力されます。

- ファイル操作アダプタ実行環境プロパティファイルがないとき
- 参照した実行環境プロパティに write.output-folder-name または write.output-file-name のどちらかがないとき

(3) ファイル書き込みオペレーションの注意事項

ファイル書き込みオペレーションの注意事項を次に示します。

- 出力データが 0 バイトの場合，0 バイトのファイルが出力されます。
- 出力モード（output-mode タグ）が new の場合，書き込み処理中にエラーが発生すると，エラーが発生する直前までのデータが書き込まれます。エラーが発生するタイミングによっては，空ファイルが作成されることがあります。
- 出力モード（output-mode タグ）が append の場合，書き込み処理中にエラーが発生すると，エラーが発生する直前までのデータが書き込まれます。ファイルが新規に作成されたときは，エラーが発生するタイミングによっては，空ファイルが作成されることがあります。

8.7.13 各オペレーションで使用するフォルダおよびファイルの管理

ファイル操作アダプタでは，作業フォルダ，または共通フォルダを指定してファイルの変換，複製，削除などを行います。作業フォルダおよび共通フォルダ同士を使用してファイルを入出力したり，作業フォルダからファイルを入力して共通フォルダへファイルを出力したりすることもできます。

(1) 同名のファイルが存在する場合の動作の指定

次のオペレーションでは，すでに同名のファイルが出力先のフォルダに存在する場合，ファイル操作アダプタ定義ファイルのプロパティで，ファイルを上書きするか，エラーメッセージを出力するかを指定できます。

- ファイル変換オペレーション
- ファイル複製オペレーション
- ファイル移動オペレーション
- ファイル圧縮オペレーション
- ファイル伸張オペレーション
- ファイル一覧書き込みオペレーション
- ファイル読み込みオペレーション
- ファイル書き込みオペレーション

ファイル操作アダプタ定義ファイルの詳細は，マニュアル「サービスプラットフォーム リファレンス」の「3.5.2 ファイル操作アダプタ定義ファイル」を参照してください。

(2) 排他取得タイムアウトの指定

読み込み対象のファイルが共通フォルダにあるとき，ファイルに対して共有ロックを取得します。また，書き込み対象，削除対象，移動対象のファイルが共通フォルダにあるとき，ファイルに対して占有ロックを取得します。

それぞれのロック取得に失敗したとき，指定間隔おきに指定回数まで排他取得処理をリトライします。リトライ間隔とリトライ回数の積に，各排他取得時のファイルアクセス時間を加えた時間がタイムアウト時間となります。タイムアウト時間を設定するには，ファイル操作アダプタ実行環境プロパティファイルにあるロックリトライ回数およびロックリトライ間隔のパラメタを指定してください。設定するパラメタについて，次の表に示します。

表 8-35 排他取得タイムアウトの設定

項番	値	プロパティのキー名称	デフォルト値
1	共有ロックリトライ回数（回）	readLock.retry.count	0
2	共有ロックリトライ間隔（秒）	readLock.retry.interval	1
3	占有ロックリトライ回数（回）	writeLock.retry.count	0
4	占有ロックリトライ間隔（秒）	writeLock.retry.interval	1

ファイル操作アダプタ実行環境プロパティファイルの詳細は，マニュアル「サービスプラットフォーム リファレンス」の「6.8.8 ファイル操作アダプタ実行環境プロパティファイル」を参照してください。

8.8　作業フォルダおよび共通フォルダを利用したファイルアクセス

FTP 連携では，サービスプラットフォームでファイル転送を中継するために作業フォルダおよび共通フォルダを利用します。

ここでは，作業フォルダおよび共通フォルダについて説明します。

8.8.1　作業フォルダを利用したファイルアクセス

作業フォルダとは，各ファイル転送パターンでサービスプラットフォームが管理する，一時的な中間ファイルを格納するためのフォルダです。

作業フォルダはリクエストごとに FTP 受付が生成します。作業フォルダの詳細は「8.4.5　作業フォルダと中間ファイルの作成」を参照してください。

8.8.2　共通フォルダを利用したファイルアクセス

共通フォルダとは，FTP クライアントから別の FTP クライアントへのファイル転送パターンなど，複数のビジネスプロセスがファイルを参照する場合に，共通して利用するファイルを格納するためのフォルダです。

共通フォルダを利用したファイルアクセスの詳細は，「10.3　共通フォルダを利用したファイルアクセス」を参照してください。

8.9　FTP 連携の適用範囲

ここでは，FTP 連携でサポートする FTP コマンド，FTP コマンドの指定方法，および FTP 連携を使用するときの注意事項について説明します。

8.9.1　FTP 連携でサポートする FTP コマンド

FTP 連携でサポートする FTP コマンドを次の表に示します。

表 8-36　サポートする FTP コマンド

分類	コマンド	オプション	コンポーネント				RFC
			FTP インバウンドアダプタ	FTP 受付	ビジネスプロセス	FTP アダプタ	
認証	ACCT	－	×	－	－	○	959
	USER	－	○	○	○	○	959
	PASS	－	○	×	×	○	959
制御	ABOR	－	×	－	－	○	959
	NOOP	－	○	－	－	○	959
	PASV	－	○	－	－	○	959
	PORT	－	○	－	－	○	959
	QUIT	－	○	－	－	○	959
	REIN	－	×	－	－	×	959
	REST	－	×	－	－	×	959
	EPRT	－	○	－	－	○	2428
	EPSV	－	○	－	－	○	2428
セッション	CDUP	－	△	△	△	○	959
	CWD	－	△	△	△	○	959
	MODE	－	○	－	－	○	959
		S	○	－	－	○	－
		B	×	－	－	×	－
		C	×	－	－	○	－
	LPRT	－	×	－	－	×	1639
	LPSV	－	×	－	－	×	1639

分類	コマンド	オプション	コンポーネント				RFC
			FTP インバウンドアダプタ	FTP 受付	ビジネスプロセス	FTP アダプタ	
セッション	ADAT	－	×	－	－	×	2228
	AUTH	－	○	－	－	○	2228
		SSL	○	－	－	○	－
		TLS	○	－	－	○	－
	CCC	－	×	－	－	×	2228
	CONF	－	×	－	－	×	2228
	ENC	－	×	－	－	×	2228
	MIC	－	×	－	－	×	2228
	PBSZ	－	○	－	－	○	2228
	PROT	－	○	－	－	○	2228
		C	○	－	－	○	－
		P	○	－	－	○	－
	OPTS	－	○	－	○	○	2389
		UTF8	○	－	○	○	2640
		MLST	×	－	×	×	3659
ファイル	TYPE	－	○	○	○	○	959
		A	○	○	○	○	－
		I	○	○	○	○	－
	LANG	－	×	－	－	×	2640
転送	APPE	－	○	○	○	○	959
	RETR	－	○	○	○	○	959
	STOU	－	×	－	－	×	959
	STRU	－	○	－	－	○	959
		F	○	－	－	○	－
		R※	○	×	×	○	－
		P	×	－	－	×	－
更新	ALLO	－	△	△	△	○	959
	DELE	－	△	△	△	○	959
	MKD	－	△	△	△	○	959

分類	コマンド	オプション	コンポーネント				RFC
			FTP インバウンドアダプタ	FTP 受付	ビジネスプロセス	FTP アダプタ	
更新	RMD	−	△	△	△	○	959
	RNFR	−	△	△	△	○	959
	RNTO	−	△	△	△	○	959
	SMNT	−	△	△	△	○	959
	STAT	−	△	△	△	○	959
参照	HELP	−	×	−	−	×	959
	LIST	−	○	○	○	○	959
	NLST	−	○	○	○	○	959
	PWD	−	△	△	△	○	959
	SITE	−	○	○	○	○	959
	STOR	−	○	○	○	○	959
	SYST	−	△	△	△	○	959
	FEAT	−	×	−	−	×	2389
	MDTM	−	△	−	△	○	3659
	MLSD	−	×	−	−	×	3659
	MLST	−	×	−	−	×	3659
	SIZE	−	△	−	△	○	3659

（凡例）

○：サポートしています。

△：SITE CSCTHR コマンドでサポートしています。

×：サポートしていません。

−：該当しません。

注※

FTP クライアントから STRU R コマンドを送信した場合，ファイルの変換処理は実施しません。また， FTP サーバへの中継は行いません。STRU F コマンドの場合と同じ動作となります。

表中の FTP コマンドの分類について説明します。

(1) 認証

セッションの接続に必要なコマンドです。HCSC サーバ内でのアクセス管理に使用されます。

USER コマンドで渡されるユーザ名は，送受信されるファイルの属性としてビジネスプロセスに引き継ぐ必要があります。

(2) 制御

セッションの接続・解放などの制御に関わるコマンドです。ビジネスプロセスを経由して FTP アダプタに情報を引き継ぐ必要はありません。

(3) セッション

1 セッション内で有効な情報を設定するためのコマンドです。ビジネスプロセスを経由して FTP アダプタに情報を引き継ぐ必要はありません。

(4) ファイル

送受信されるファイルの情報を設定するためのコマンドです。ビジネスプロセスを経由して FTP アダプタに情報を引き継ぐ必要があります。

(5) 転送

ファイルを転送するのためのコマンドです。

このコマンドを契機にビジネスプロセスが起動され，ビジネスプロセスを経由して FTP インバウンドアダプタから FTP アダプタに情報を引き継ぎます。

(6) 更新

FTP サーバ上のファイルおよびディレクトリを更新するコマンドです。

ファイル転送の前後でバッチ用途で実行することができます。FTP クライアントから SITE CSCTHR コマンドで指定します。

(7) 参照

FTP サーバ上のファイルおよびディレクトリに関する情報を参照し，FTP クライアントに渡すためのコマンドです。

LIST 系コマンドは，転送コマンドと同様にビジネスプロセスを経由して FTP インバウンドアダプタから FTP アダプタに情報を引き継ぎます。LIST 系以外のコマンドは，FTP クライアントから SITE CSCTHR コマンドで指定し，ビジネスプロセス上で結果を参照できます。

8.9.2 転送コマンドの指定方法

FTP クライアントからの転送コマンドの指定方法について説明します。

転送コマンドには，送受信するファイル名，および起動するビジネスプロセスを識別するための FTP 受付の受付 ID を指定します。

転送コマンドの指定形式と指定内容は次のとおりです。

指定形式

```
/<受付ID>/<サーバ上のパス>/<ファイル名>
```

指定内容

/

パスの区切り文字です。OS に関係なく固定です。

<受付 ID>

FTP 受付の受付 ID です。指定した受付 ID によって起動するビジネスプロセスが識別されます。

<サーバ上のパス>

接続先の FTP サーバ上のパスです。指定は省略できます。

<ファイル名>

送受信するファイル名です。

8.9.3 FTP 連携を利用するときの注意

FTP 連携の機能を利用するときの注意について説明します。

(1) ビジネスプロセスの作成に関する注意

- ビジネスプロセスはファイル転送の経路ごとに作成する必要があります。
- ビジネスプロセスは FTP 受付に応答を返すように作成する必要があります。
- 1 回のビジネスプロセスの処理で，複数ファイルを処理することはできません。
- FTP 連携では BPEL の並行処理（FLOW）を使用できません。

(2) サーバの運用で FTP を利用する場合の注意

サービスプラットフォームのサーバ自体の運用および保守で FTP を使用する場合，別途 FTP サーバを構築する必要があります。この場合，両方の FTP サーバのポート番号または IP アドレスは一意になるように設計してください。

(3) FTP コマンド実行時の注意

(a) ファイル転送要求を行う場合

FTP クライアントに対して，Linux 付属の ftp コマンドを使用してファイル転送要求を行った場合，FTP インバウンドアダプタのメッセージログに次の 2 種類のメッセージが出力されます。

- 未サポートの FTP プロトコルのコマンドである，SYST コマンドを受信したことを示すメッセージ（KDEC07531-E）
- エラー応答をクライアントへ返したことを示すメッセージ（KDEC07542-E）

また，FTP クライアントに「502 Command SYST not implemented.」のレスポンスが返りますが，ファイル転送要求に影響はありません。

(b) LIST コマンドおよび NLST コマンドを実行する場合

LIST コマンドおよび NLST コマンドを実行する場合，次の点に注意してください。

- LIST コマンドまたは NLST コマンドの受付時に，呼び出されるビジネスプロセスの名称は，固定です。そのため，ビジネスプロセスで宛先の振り分けが必要となります。
- FTP クライアントから，LIST コマンドに続いてファイル転送を行った場合，FTP クライアントとサービスプラットフォームの間では同一コネクションとなりますが，サービスプラットフォームと FTP サーバの間では，LIST コマンドとファイル転送は，別コネクションとなることがあります。

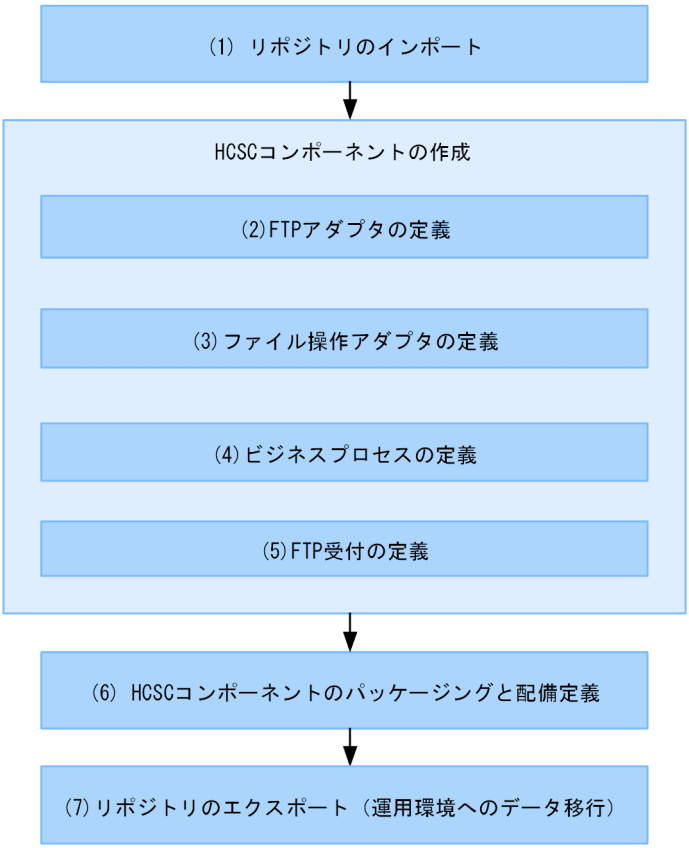
8.10　FTP 連携システムの開発の手順

ここでは，FTP 連携システムを開発する手順について説明します。

8.10.1　FTP 連携システムの開発の流れ

FTP 連携システムの開発の流れを次の図に示します。図中の項目は，すべて開発環境で実施する作業です。

図 8-63　FTP 連携システムの開発の流れ



(1)　リポジトリのインポート

運用環境でエクスポートしたリポジトリを開発環境にインポートします。

リポジトリのインポート方法の詳細については，マニュアル「サービスプラットフォーム 開発ガイド 基本開発編」の「3.2.3　リポジトリをインポートする」を参照してください。

(2)　FTP アダプタの定義

サービスアダプタ定義画面を使用して FTP アダプタを定義します。FTP アダプタは，接続対象の FTP サーバごとに 1 つ定義します。

FTP アダプタの定義については，マニュアル「サービスプラットフォーム 開発ガイド 受付・アダプタ定義編」の「3.　サービスアダプタの定義」を参照してください。

(3)　ファイル操作アダプタの定義

サービスアダプタ定義画面を使用してファイル操作アダプタを定義します。ファイル操作アダプタは，変換，複製，削除などのオペレーションごとに 1 つ定義します。

ファイル操作アダプタの定義については，マニュアル「サービスプラットフォーム 開発ガイド 受付・アダプタ定義編」の「3.　サービスアダプタの定義」を参照してください。

(4)　ビジネスプロセスの定義

ビジネスプロセス定義画面を使用してビジネスプロセスを定義します。ビジネスプロセスは，ファイル転送の経路ごとに 1 つ定義します。

ビジネスプロセスの定義については，マニュアル「サービスプラットフォーム 開発ガイド 基本開発編」の「5.　ビジネスプロセスの定義」を参照してください。

(5)　FTP 受付の定義

ユーザ定義受付定義画面を使用して FTP 受付を定義します。FTP 受付は，ビジネスプロセスごとに 1 つ定義します（必要に応じて複数定義します）。なお，1 つの FTP 受付を複数のビジネスプロセスに対応させることはできません。

FTP 受付の定義の概要については，「8.4　FTP 受付の機能」を参照してください。FTP 受付の定義については，マニュアル「サービスプラットフォーム 開発ガイド 受付・アダプタ定義編」の「2.4　FTP 受付の定義」を参照してください。

(6)　HCSC コンポーネントのパッケージングと配備定義

定義した HCSC コンポーネントの定義に関連する定義ファイル，および実行環境が提供するファイルをまとめて EAR ファイルにパッケージングします。また，HCSC コンポーネントを配備するクラスタ（または単一の HCSC サーバ）の情報を定義（配備定義）します。

パッケージングと配備定義については，マニュアル「サービスプラットフォーム 開発ガイド 基本開発編」の「8.　HCSC コンポーネントのパッケージングと配備定義」を参照してください。

(7)　リポジトリのエクスポート

HCSC コンポーネントの配備に必要な定義ファイルを格納したリポジトリをエクスポートします。開発環境からエクスポートした情報は，運用環境にインポートして利用されます。

リポジトリのエクスポートについては，マニュアル「サービスプラットフォーム 開発ガイド 基本開発編」の「3.2.2　リポジトリをエクスポートする」を参照してください。

運用環境にインポートしたあと，FTP 連携システムを運用する方法については，マニュアル「サービスプラットフォーム システム構築・運用ガイド」の「5.4　HCSC サーバのオペレーション運用（FTP 連携する場合)」を参照してください。

8.11　FTP 受付および FTP アダプタの実行環境に依存する設定の定義

FTP 受付および FTP アダプタでは，実行環境に依存する設定を次の方法で定義できます。実行環境に依存する設定とは，接続先 FTP サーバの IP アドレスやチューニングパラメタなどのことです。

- **共通定義ファイルによる定義（FTP アダプタ実行環境共通プロパティファイル）**

  要求電文および個別定義ファイルによる設定がない場合に有効になる定義です。設定値は HCSC コンポーネントの起動時に有効になります。

  次の場合に使用します。

  - すべての FTP 受付または FTP アダプタで設定を合わせたい場合
  - 多くの FTP 受付または FTP アダプタで同じ値を設定する場合で，かつ運用時に管理する定義ファイルの数を削減したい場合

  FTP アダプタで定義ファイルの数を削減するためには，要求電文による定義で接続先 FTP サーバに依存するパラメタを設定する必要があります。

- **個別定義ファイルによる定義（FTP アダプタ実行環境プロパティファイル）**

  要求電文による設定がない場合に有効になる定義です。設定値は HCSC コンポーネントの起動時に有効になります。

  次の場合に使用します。

  - それぞれの FTP 受付または FTP アダプタに異なる値を設定したい場合で，かつ HCSC コンポーネントの再起動だけで設定値を変更したい場合

- **要求電文による定義**

  すべての定義方法の中で最も優先度の高い定義です。設定値は 1 つのリクエスト処理でだけ有効になります。

  次の場合に使用します。

  - リクエストごとに異なる値を設定したい場合
  - それぞれの FTP アダプタで異なる値を設定したい場合で，かつ運用時に管理する定義ファイルの数を削減したい場合

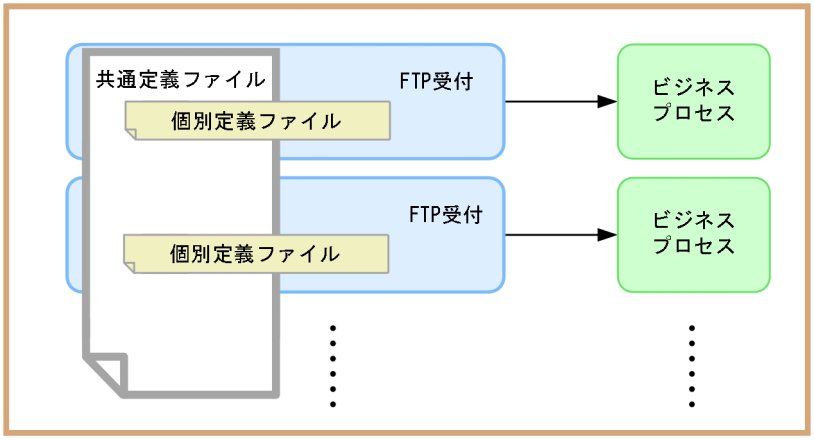
  ビジネスプロセスで固定値を設定した場合，設定を変更するためには開発環境でビジネスプロセスを変更する必要があります。

8.11.1　FTP 受付の実行環境に依存する設定の定義

FTP 受付では，実行環境に依存する設定を次の 2 つの方法で定義できます。

- 共通定義ファイルによる定義
- 個別定義ファイルによる定義

図 8-64　FTP 受付の実行環境に依存する設定の定義



FTP 受付で実行環境に依存する定義を設定する場合，FTP 受付コンフィグファイルを使用します。

FTP 受付コンフィグファイルには，FTP 受付ごとに設定する個別定義ファイルと，すべての FTP 受付に対して設定する共通定義ファイルがあります。両方に同じプロパティを設定した場合，個別定義ファイルの設定が有効になります。

詳細は，マニュアル「サービスプラットフォーム リファレンス」の「6.8.5　FTP 受付コンフィグファイル」を参照してください。

ここでは，FTP 受付で使用する，次の実行環境定義ファイルについて説明します。

- FTP 実行許可リスト定義ファイル
- FTP コマンド許可リスト定義ファイル（FTP 受付用）
- リストコマンドオプション定義ファイル

(1)　FTP 実行許可リスト定義ファイル

FTP 受付では実行を許可するユーザを制限できます。実行ユーザを制限する方法は次の 3 つです。

1. FTP インバウンドアダプタのユーザ認証情報
2. FTP 受付コンフィグファイルの urecp-ftp.permission-def.filepath プロパティに指定する FTP 実行許可リスト定義ファイル
3. 製品固定パス（<サービスプラットフォームのインストールディレクトリ>¥CSC¥config¥ftprecp¥common¥ftp_permission_allow.properties）の FTP 実行許可リスト定義ファイル

製品固定パスの定義は FTP 受付コンフィグファイルで，urecp-ftp.permission-def.filepath プロパティを定義していないすべての FTP 受付に対して適用されます。実行ユーザの制限の組み合わせを次の表に示します。なお，表中の 1.～3.は上記の実行ユーザを制限する方法と対応しています。

表 8-37　実行ユーザの制限の組み合わせ

実行ユーザを制限する方法			実行ユーザの制限
1.	2.	3.	
ユーザ情報なし	－	－	実行を拒否します。
ユーザ情報あり	拒否	－	実行を拒否します。
ユーザ情報あり	設定なし※1	拒否	実行を拒否します。
ユーザ情報あり	設定なし※1	設定なし※2	実行を許可します。
ユーザ情報あり	設定なし※1	許可	実行を許可します。
ユーザ情報あり	許可	－	実行を許可します。

（凡例）

－：設定は無視されます。

注※1

次のどれかを示します。

- FTP 受付コンフィグファイルを省略している
- FTP 受付コンフィグファイルの urecp-ftp.permission-def.filepath プロパティを設定していない
- FTP 受付コンフィグファイルの urecp-ftp.permission-def.filepath プロパティはあるが値を設定していない

注※2

次のどれかを示します。

- 製品固定パスに FTP 実行許可リスト定義ファイルを格納していない
- FTP 実行許可リスト定義ファイルでプロパティを設定していない
- FTP 実行許可リスト定義ファイルのプロパティはあるが値が指定されていない

FTP インバウンドアダプタのユーザ認証情報の詳細については，マニュアル「サービスプラットフォーム システム構築・運用ガイド」の「5.4.12　FTP インバウンドアダプタのユーザ情報を確認・変更する」を参照してください。

FTP 実行許可リスト定義ファイルの詳細については，マニュアル「サービスプラットフォーム リファレンス」の「6.8.7　FTP 実行許可リスト定義ファイル」を参照してください。

(2)　FTP コマンド許可リスト定義ファイル（FTP 受付用）

FTP 受付では，転送前，転送後，リストコマンド実行前，またはリストコマンド実行後に実行を許可するコマンドを制限できます。コマンドを制限する方法は次の 2 つです。

1. FTP 受付コンフィグファイルの urecp-ftp.ftp-command-allow.filepath プロパティに指定する FTP コマンド許可リスト定義ファイル

2. 製品固定パス（<サービスプラットフォームのインストールディレクトリ>¥CSC¥config¥ftprecp¥common¥ftp_command_allow.properties）の FTP コマンド許可リスト定義ファイル

製品固定パスの定義は FTP 受付コンフィグファイルで，urecp-ftp.ftp-command-allow.filepath プロパティを設定していないすべての FTP 受付に対して適用されます。コマンドの制限の組み合わせを次の表に示します。なお，表中の 1.～2.は上記のコマンドを制御する方法と対応しています。

表 8-38　コマンドを制御の組み合わせ（FTP 受付の場合）

実行を許可するコマンドを制限する方法		実行ユーザの制限
1.	2.	
拒否	－	実行を拒否します。
設定なし※1	拒否	実行を拒否します。
設定なし※1	設定なし※2	実行を拒否します。
設定なし※1	許可	実行を許可します。
許可	－	実行を許可します。

（凡例）

－：設定は無視されます。

注※1

次のどれかを示します。

- FTP 受付コンフィグファイルを省略している
- FTP 受付コンフィグファイルの urecp-ftp.ftp-command-allow.filepath プロパティを設定していない
- FTP 受付コンフィグファイルの urecp-ftp.ftp-command-allow.filepath プロパティはあるが値を設定していない

注※2

次のどれかを示します。

- 製品固定パスに FTP コマンド許可リスト定義ファイルを格納していない
- FTP コマンド許可リスト定義ファイルでプロパティを設定していない
- FTP コマンド許可リスト定義ファイルのプロパティはあるが値を設定していない

FTP コマンド許可リスト定義ファイルの詳細は，マニュアル「サービスプラットフォーム リファレンス」の「6.8.6　FTP コマンド許可リスト定義ファイル」を参照してください。

(3) リストコマンドオプション定義ファイル

FTP クライアントから LIST コマンドまたは NLST コマンドを実行するとき，コマンドのオプションを指定できます。リストコマンドの<オプション>は，GETINFO オペレーションの要求電文の<getinfo-option>に設定されます。リストコマンドの<パス>は，GETINFO オペレーションの要求電文の<getinfo-path>に設定されます。

リストコマンドの<オプション>と<パス>を適切に扱うためには，リストコマンドのオプションを実行環境の定義ファイルに設定しておく必要があります。リストコマンドオプションの定義方法は次の 2 つです。

1. FTP 受付コンフィグファイルの urecp-ftp.list-command-option.filepath プロパティに指定するリストコマンドオプション定義ファイル
2. 製品固定パス（<サービスプラットフォームのインストールディレクトリ>¥CSC¥config¥ftprecp¥common¥ftp_command_option.txt）のリストコマンドオプション定義ファイル

製品固定パスの定義は FTP 受付コンフィグファイルで，urecp-ftp.list-command-option.filepath プロパティを設定していないすべての FTP 受付に対して適用されます。コマンドオプション指定の組み合わせを次の表に示します。なお，表中の 1.～2.は上記のリストコマンドオプションの定義方法と対応しています。

表 8-39　コマンドオプション指定の組み合わせ

リストコマンドオプションの定義方法		<getinfo-option>	<getinfo-path>
1.	2.		
設定あり	－	LIST コマンドまたは NLST コマンドの直後の 1.で設定された<オプション文字列>が使用されます。 <オプション文字列>が一致しない場合は<getinfo-option>は出現しません。	LIST コマンドまたは NLST コマンドの直後の<getinfo-option>の値を除いた文字列が<パス>として使用されます。
設定なし※1	設定あり	LIST コマンドまたは NLST コマンドの直後の 2.で設定された<オプション文字列>が使用されます。 <オプション文字列>が一致しない場合は<getinfo-option>は出現しません。	LIST コマンドまたは NLST コマンドの直後の<getinfo-option>の値を除いた文字列が<パス>として使用されます。
設定なし※1	設定なし※2	<getinfo-option>は出現しません。	LIST コマンドまたは NLST コマンドの直後の文字列が<パス>として使用されます。

（凡例）

－：設定は無視されます。

注※1

次のどれかを示します。

- FTP 受付コンフィグファイルを省略している

- FTP 受付コンフィグファイルの urecp-ftp.list-command-option.filepath プロパティを設定していない
- FTP 受付コンフィグファイルの urecp-ftp.list-command-option.filepath プロパティはあるが値を設定していない

注※2

製品固定パスにリストコマンドオプション定義ファイルを格納していないことを示します。

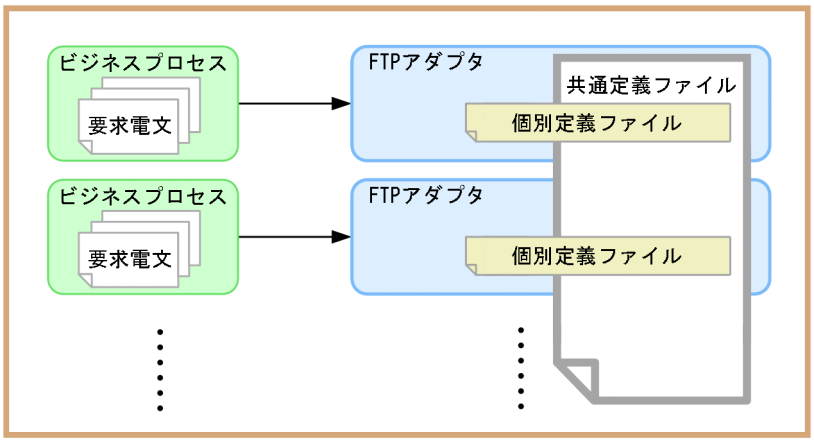
リストコマンドオプション定義ファイルの詳細は，マニュアル「サービスプラットフォーム リファレンス」の「6.8.9　リストコマンドオプション定義ファイル」を参照してください。

8.11.2　FTP アダプタの実行環境に依存する設定の定義

FTP アダプタでは，実行環境に依存する設定を 3 つの方法で定義できます。

- 共通定義ファイルによる定義（FTP アダプタ実行環境共通プロパティファイル）
- 個別定義ファイルによる定義（FTP アダプタ実行環境プロパティファイル）
- 要求電文による定義

図 8-65　FTP アダプタの実行環境に依存する設定の定義



FTP アダプタで実行環境に依存する設定を定義する場合，要求電文または FTP アダプタ実行環境プロパティファイルを使用します。

FTP アダプタ実行環境プロパティファイルは，次に示す 2 種類があります。

- FTP アダプタ実行環境プロパティファイル
  有効範囲：特定の FTP アダプタ
- FTP アダプタ実行環境共通プロパティファイル
  有効範囲：すべての FTP アダプタ

これらのファイルが両方設定されている場合，2 つのファイルの定義内容がマージされます。

また，1 つのマシンに複数の HCSC サーバをセットアップした場合，FTP アダプタ実行環境プロパティファイルを HCSC サーバごとに定義することもできます。

同じプロパティがプロパティファイル間で複数設定された場合，プロパティの優先度は次のようになります。

ファイルの種類	プロパティ重複時の優先度
FTP アダプタ実行環境プロパティファイル （HCSC サーバ単位の指定：あり）	高
FTP アダプタ実行環境共通プロパティファイル （HCSC サーバ単位の指定：あり）	｜
FTP アダプタ実行環境プロパティファイル （HCSC サーバ単位の指定：なし）	｜
FTP アダプタ実行環境共通プロパティファイル （HCSC サーバ単位の指定：なし）	低

FTP アダプタ実行環境プロパティファイルの詳細は，マニュアル「サービスプラットフォーム リファレンス」の「6.8.3　FTP アダプタ実行環境プロパティファイル」を参照してください。

FTP アダプタ実行環境共通プロパティファイルの詳細は，マニュアル「サービスプラットフォーム リファレンス」の「6.8.4　FTP アダプタ実行環境共通プロパティファイル」を参照してください。

ここでは，FTP アダプタで使用する，要求電文および次の実行環境定義ファイルについて説明します。

- FTP コマンド許可リスト定義ファイル（FTP アダプタ用）
- FTP アダプタアカウント定義ファイル

(1)　要求電文

FTP アダプタの電文フォーマットの定義ファイルはサービスプラットフォームが提供するスキーマを使用します。そのため，電文フォーマットの定義ファイルの作成は不要です。

FTP アダプタの要求電文フォーマットの内容については，マニュアル「サービスプラットフォーム 開発ガイド 受付・アダプタ定義編」の「3.3.10　FTP アダプタを定義する」を参照してください。

既存の FTP アダプタで，実行環境プロパティファイルの ftpadp.pasvport プロパティまたは ftpadp.file-size-check プロパティを要求電文で設定したい場合は，開発環境のサービスアダプタ定義画面で電文フォーマットを再設定してください。

サービスアダプタ定義画面の詳細は，マニュアル「サービスプラットフォーム リファレンス」の「1.2.2 サービスアダプタ定義画面」を参照してください。

(2) FTP コマンド許可リスト定義ファイル（FTP アダプタ用）

FTP アダプタでは，転送前，転送後，リストコマンド実行前，またはリストコマンド実行後に実行を許可するコマンドを制限できます。コマンドを制限する方法は次の 2 つです。

1. FTP アダプタ実行環境プロパティファイルの ftpadp.cscthrough-filepath プロパティに指定する FTP コマンド許可リスト定義ファイル
2. 製品固定パス（<サービスプラットフォームのインストールディレクトリ>¥CSC¥custom-adapter¥FTP¥config¥common）に格納された FTP コマンド許可リスト定義ファイル

製品固定パスの定義は FTP アダプタ実行環境プロパティファイルで ftpadp.cscthrough-filepath プロパティを設定していないすべての FTP アダプタに対して適用されます。コマンドの制限の組み合わせを次の表に示します。なお，表中の 1.～2.は上記のコマンドを制限する方法と対応しています。

表 8-40　コマンド制限の組み合わせ（FTP アダプタの場合）

実行を許可するコマンドを制限する方法		実行ユーザの制限
1.	2.	
拒否	－	実行を拒否します。
設定なし※1	拒否	実行を拒否します。
設定なし※1	設定なし※2	実行を拒否します。
設定なし※1	許可	実行を許可します。
許可	－	実行を許可します。

（凡例）

－：設定は無視されます。

注※1

次のどれかを示します。

- FTP アダプタ実行環境プロパティファイルを省略している
- FTP アダプタ実行環境プロパティファイルの ftpadp.cscthrough-filepath プロパティを設定していない
- FTP アダプタ実行環境プロパティファイルの ftpadp.cscthrough-filepath プロパティはあるが値を設定していない

注※2

次のどれかを示します。

- 製品固定パスに FTP コマンド許可リスト定義ファイルを格納していない
- FTP コマンド許可リスト定義ファイルでプロパティを定義していない
- FTP コマンド許可リスト定義ファイルのプロパティはあるが値を設定していない

FTP コマンド許可リスト定義ファイルの詳細は，マニュアル「サービスプラットフォーム リファレンス」の「6.8.6　FTP コマンド許可リスト定義ファイル」を参照してください。

(3)　FTP アダプタアカウント定義ファイル

FTP アダプタでは，FTP サーバに接続するためのユーザ認証情報を FTP アダプタアカウント定義ファイルで管理します。ユーザ認証情報を管理する方法は次の 2 つです。

1. ユーザ任意の単位に任意のパスのアカウント定義ファイルで管理する。
2. FTP アダプタ単位に製品固定パスのアカウント定義ファイルで管理する。

FTP アダプタのユーザ情報の確認，追加，変更，および削除は，運用環境で次のコマンドを実行します。

- ユーザ情報の確認：csaftpuserls コマンド
- ユーザ情報の追加：csaftpuseradd コマンド
- ユーザ情報の変更：csaftpuseradd コマンド
- ユーザ情報を削除：csaftpuserdel コマンド

これらのコマンドの使用方法については，マニュアル「サービスプラットフォーム システム構築・運用ガイド」の「5.4.14　FTP アダプタのユーザ情報を確認および変更する」を参照してください。

8.12　障害対策のための設計

FTP 連携で障害が発生した場合に備えた，システムのエラー設計の方針について説明します。

8.12.1　エラー設計の概要

FTP 連携のエラー設計で考慮するポイントを次に示します。

(1)　障害検知

FTP 連携の各コンポーネント間は同期型の呼び出しで連携しているため，呼び出し先からのリターンで障害発生を検知できます。しかし，呼び出し先が無応答となるおそれもあります。このため，構築時に適切なタイマ管理を設計し，ファイルを正常に転送できないといった障害の発生を検知する必要があります。

FTP 連携のタイマ管理については，「8.12.2　タイマ設計」を参照してください。

(2)　障害通知

障害が発生した場合，FTP 連携を実行した FTP クライアントおよびシステム運用者に対して，障害発生を確実に通知する必要があります。

FTP 連携を実行した FTP クライアントに対しては，FTP コマンドのリターンで障害発生を通知します。システム運用者に対しては，メッセージの出力で障害発生を通知します。障害発生のメッセージは，決められたメッセージ，メッセージログ，またはファイルに出力されます。そのため，JP1 などのモニタリング機能でエラーメッセージを監視することで，システム運用者に障害の発生をタイムリーに通知できます。

(3)　回復処理

障害が発生したあとの FTP クライアントおよびシステム運用者の回復処理を容易にするために，適切なトランザクション処理を設計します。

FTP 連携のトランザクションの設計については，「8.12.3　トランザクション設計」を参照してください。

(4)　原因調査

障害から回復し，かつ再発生を防ぐには，障害の原因を特定して取り除く必要があります。そのため，障害の原因を特定するための各種の障害情報を取得します。

FTP 連携の障害情報の取得については，「8.12.4　トラブルシュート設計」を参照してください。

8.12.2　タイマ設計

呼び出し先が無応答になったり，処理時間が想定を上回ったりした場合，その状況を検知して適切に対処することを目的として，タイマ管理を実施します。

FTP 連携で使用するタイマー覧を次の表に示します。

表 8-41　FTP 連携で使用するタイマー覧

コンポーネント	タイマ	タイマ管理の目的	タイムアウト時の動作
FTP インバウンドアダプタ※1	無通信監視タイマ	制御コネクション上で，コマンドの応答から次のコマンドまでの時間を監視します。	コネクションを切断し，FTP 通信をエラー終了します。
	クライアント接続待ちタイマ	データコネクションの接続方式がアクティブモードの場合に，FTP クライアントからコネクション接続要求が来るまでの時間を監視します。	コネクションを切断し，FTP 通信をエラー終了します。
	データ受信監視タイマ	データコネクション上で，1 回の受信データの read に要する時間を監視します。	コネクションを切断し，FTP 通信をエラー終了します。
	データ送信監視タイマ	データコネクション上で，1 回の送信データの write に要する時間を監視します。	コネクションを切断し，FTP 通信をエラー終了します。
FTP 受付※2	FTP 受付監視タイマ	FTP 受付が呼び出されてから応答するまでの時間を監視します。FTP 受付の呼び出し元が先にタイムアウトを検知した場合の予防措置として利用します。	FTP 受付が応答を返す時点でタイムアウトを検知し，システム例外を返して処理を終了します。
ビジネスプロセス※3	サービス呼び出し監視タイマ	サービス呼び出しから応答までの時間を監視します。ビジネスプロセス呼出し元が先にタイムアウトした場合の予防措置として利用します。	サービスから応答が返った時点でタイムアウトを検知し，システム例外を返して処理を終了します。
	ビジネスプロセス処理監視タイマ	サービス呼び出しから応答までのビジネスプロセス処理（データ変換など）の時間を監視します。ビジネスプロセス呼び出し元が先にタイムアウトした場合の予防措置として利用します。	次のサービス呼び出し時点でタイムアウトを検知し，システム例外を返して処理を終了します。
FTP アダプタ※4	応答監視タイマ	制御コネクション上で，コマンド送信から応答までの時間を監視します。	コネクションを切断し，FTP 通信をエラー終了します。そのあと，システム例外を返して処理を終了します。
	サーバ接続待ちタイマ	Active モードの場合に，FTP サーバからコネクション接続要求が返されるまでの時間を監視します。	コネクションを切断し，FTP 通信をエラー終了します。そのあと，システム例外を返して処理を終了します。

コンポーネント	タイマ	タイマ管理の目的	タイムアウト時の動作
FTP アダプタ※4	データ送受信監視タイマ	データ転送の開始から終了までの時間を監視します。	コネクションを切断し，FTP 通信をエラー終了します。そのあと，システム例外を返して処理を終了します。
	ログインセッションアイドルタイマ	ログインセッションが再利用のために維持されている時間を監視します。	ログインセッションを解放・切断します。
ファイル操作アダプタ※5	ファイル変換監視タイマ	大規模なファイルの分割読み込み時の変換処理時間を監視します。	コネクションを切断し，FTP 通信をエラー終了します。そのあと，システム例外を返して処理を終了します。
全体	リクエスト応答監視タイマ	FTP 受付が呼び出されてから応答を返すまでの時間を監視します。	FTP クライアントにエラー応答を返します。ただし，実行中のビジネスプロセス処理は終了するまで実行されます。

注※1

FTP インバウンドアダプタのタイマ管理機能の詳細は，「8.5.2　タイムアウトの監視」を参照してください。

注※2

FTP 受付のタイマ管理機能の詳細は，「8.4.9　タイムアウトの監視」を参照してください。

注※3

ビジネスプロセスおよび HCSC サーバ全体のタイマ管理機能の詳細は，「4.1　HCSC サーバに関連するタイマの設定」を参照してください。

注※4

FTP アダプタのタイマ管理機能の詳細は，「8.6.7　タイマ管理」を参照してください。

注※5

ファイル操作アダプタのタイマ管理機能は，ファイル操作アダプタ実行環境プロパティファイルで設定できます。

詳細は，マニュアル「サービスプラットフォーム リファレンス」の「6.8.8　ファイル操作アダプタ実行環境プロパティファイル」を参照してください。

8.12.3　トランザクション設計

障害からの回復を容易にするために，FTP 連携では次の方針でトランザクションを設計します。

- FTP クライアントからのデータ転送要求からデータ転送完了の応答までを 1 フェーズのトランザクションとします。
- ファイルやリソースの回復は，ファイルを取り扱う FTP 受付，FTP アダプタ，およびファイル操作アダプタが実行します。

(1)　トランザクションの範囲

正常終了や異常終了などの FTP クライアントへの応答内容と，ビジネスプロセスの処理ステータスと同期をとるために，EJB のトランザクションを利用したトランザクション管理をします。

トランザクションの範囲をファイル転送パターンごとに説明します。

(a)　同期型のビジネスプロセスの場合

同期型のビジネスプロセスの場合のトランザクション範囲を図に示します。

なお，同期型のビジネスプロセスを利用したファイル転送パターンについては，「8.3.1(1)　同期型ビジネスプロセスを利用したファイル転送の流れ」および「8.3.2　FTP サーバから FTP クライアントへのファイル転送」を参照してください。

図 8-66　同期型のビジネスプロセスの場合（FTP クライアントから FTP サーバへのファイル転送のとき）

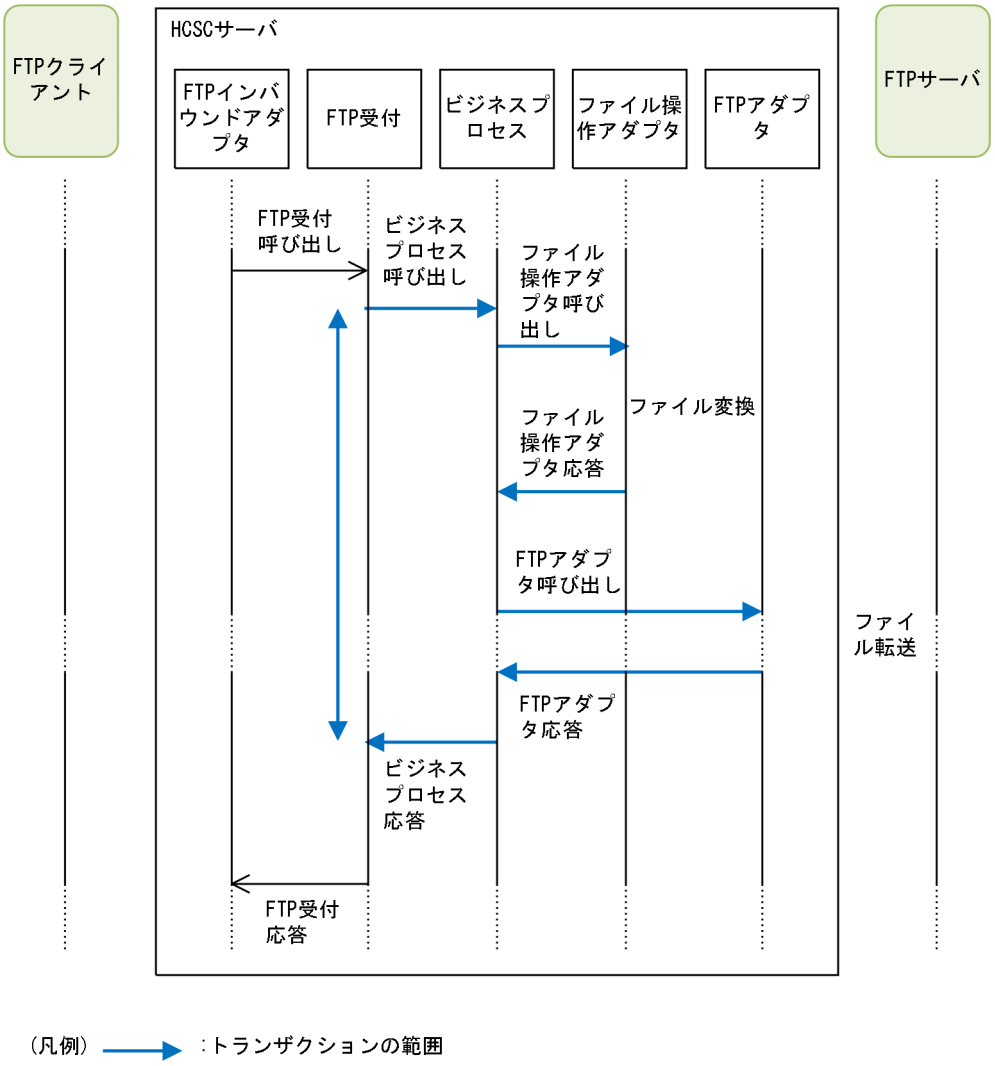
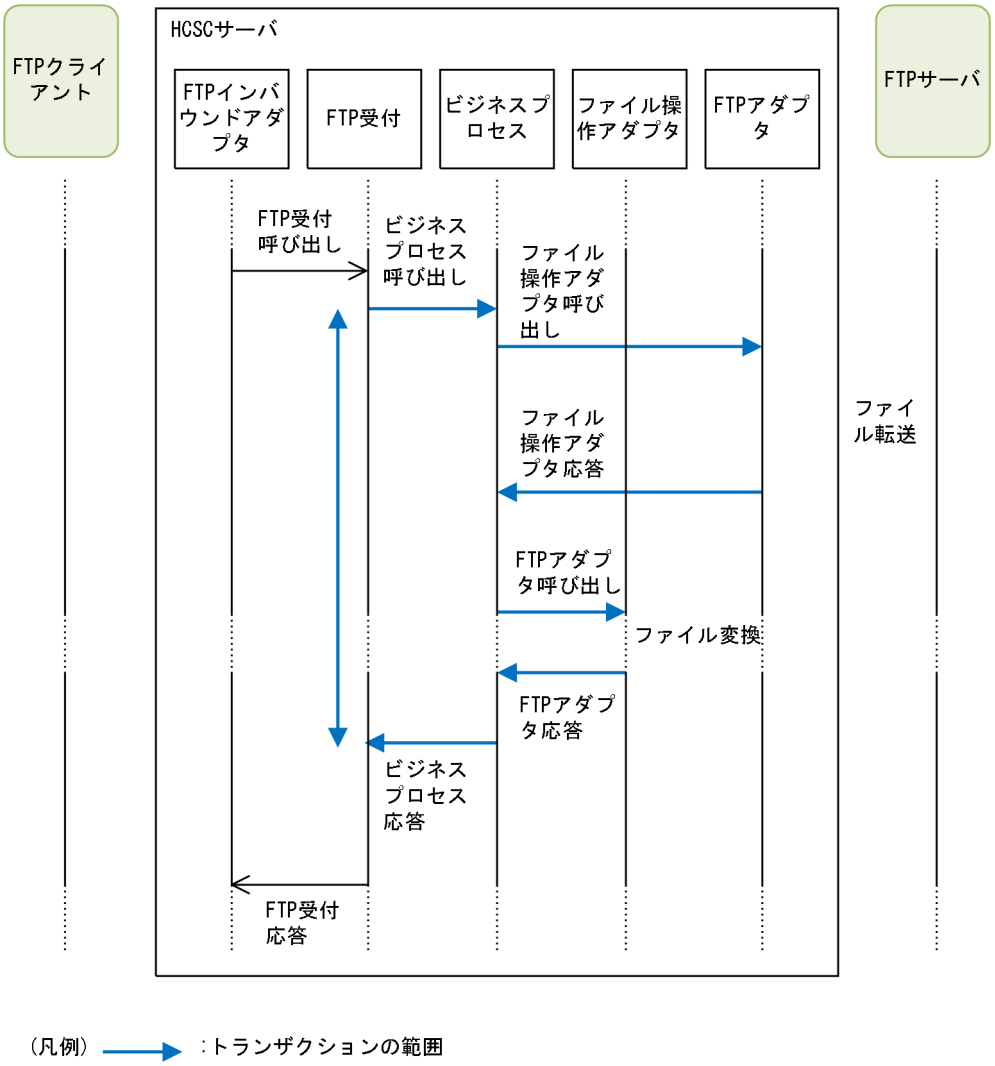


図 8-67　同期型のビジネスプロセスの場合（FTP サーバから FTP クライアントへのファイル転送のとき）

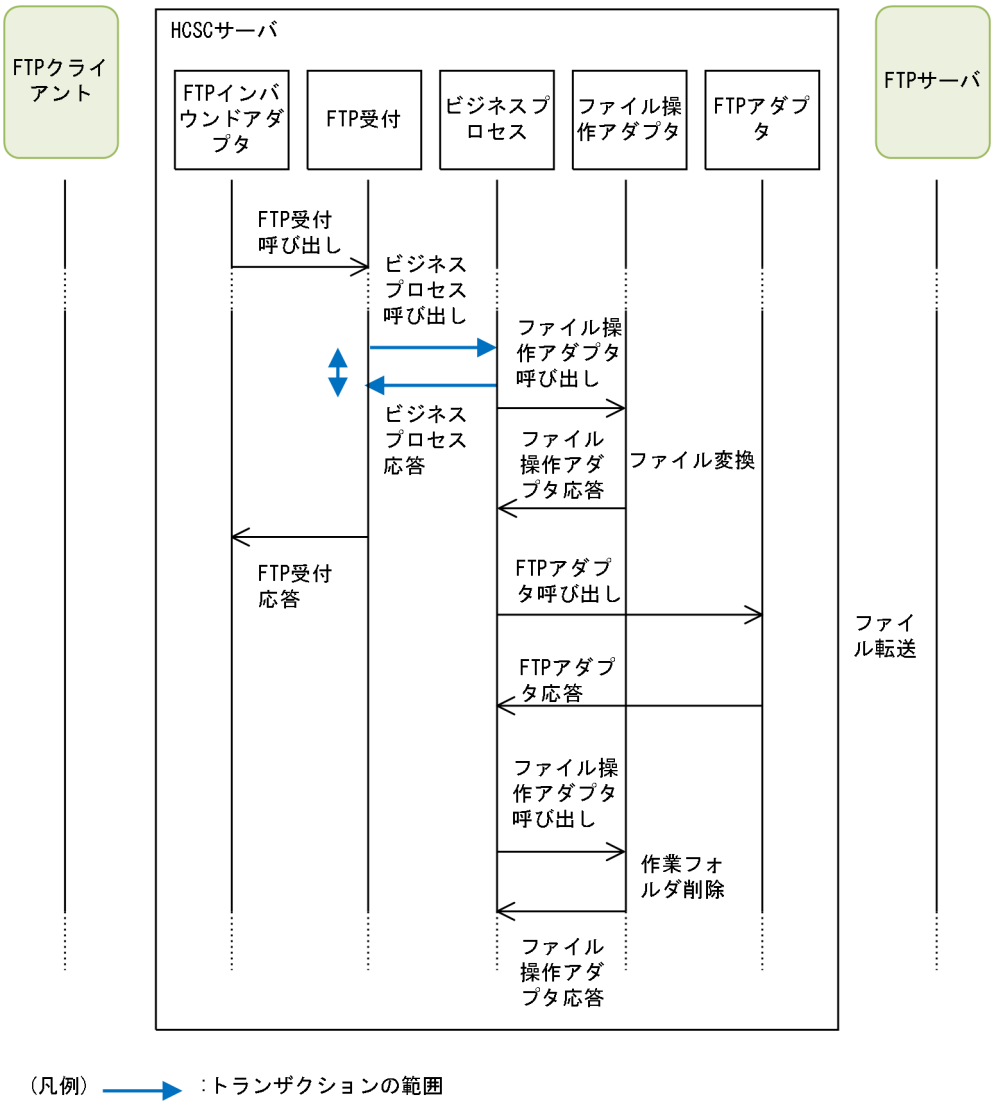


(b)　非同期型のビジネスプロセスの場合

非同期型のビジネスプロセスの場合のトランザクション範囲を図に示します。

なお，非同期型のビジネスプロセスを利用したファイル転送パターンについては，「8.3.1(2)　非同期型ビジネスプロセスを利用したファイル転送の流れ」を参照してください。

図 8-68　非同期型のビジネスプロセスの場合

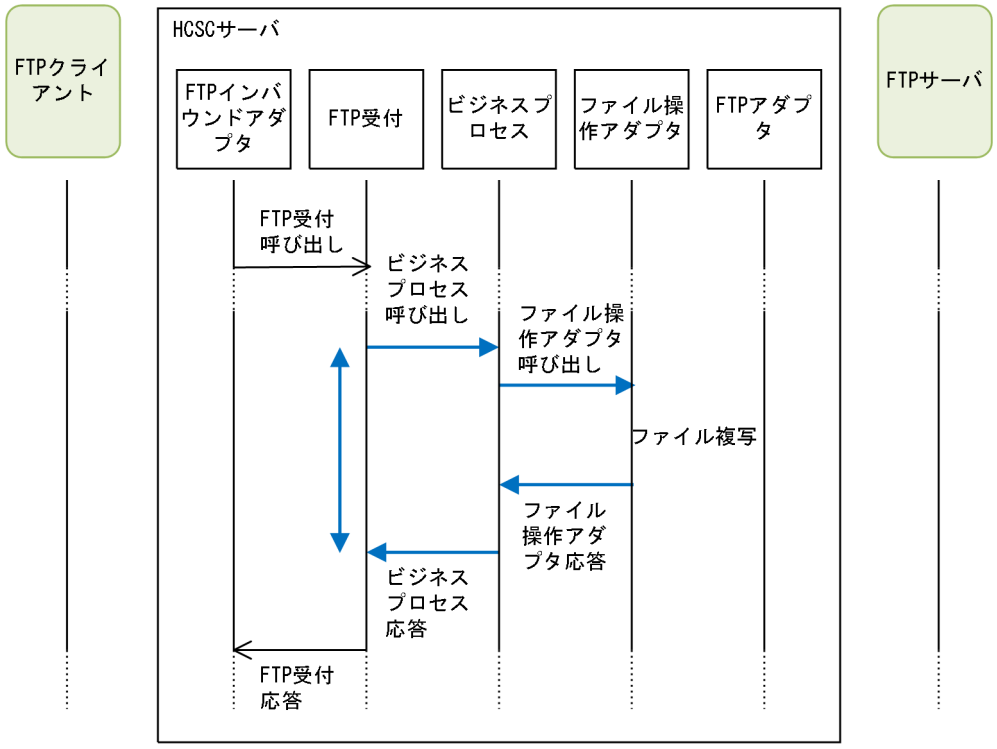


(c)　FTP クライアントから別の FTP クライアントへのファイル転送の場合

FTP クライアントから別の FTP クライアントへのファイル転送の場合の，トランザクション範囲を図に示します。

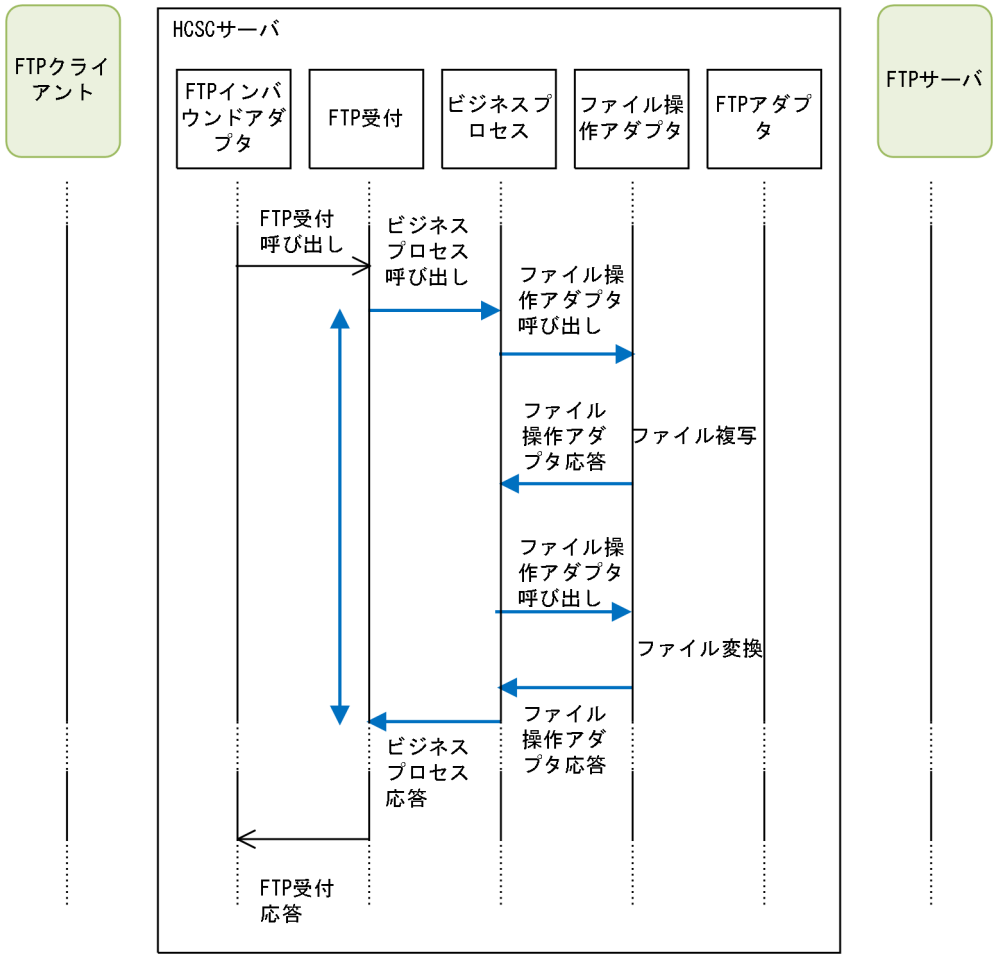
なお，FTP クライアントから別の FTP クライアントへのファイル転送パターンについては，「8.3.3　FTP クライアントから別の FTP クライアントへのファイル転送」を参照してください。

図 8-69　FTP クライアントから別の FTP クライアントへのファイル転送の場合（HCSC サーバ内の共通フォルダにファイルを転送するとき）



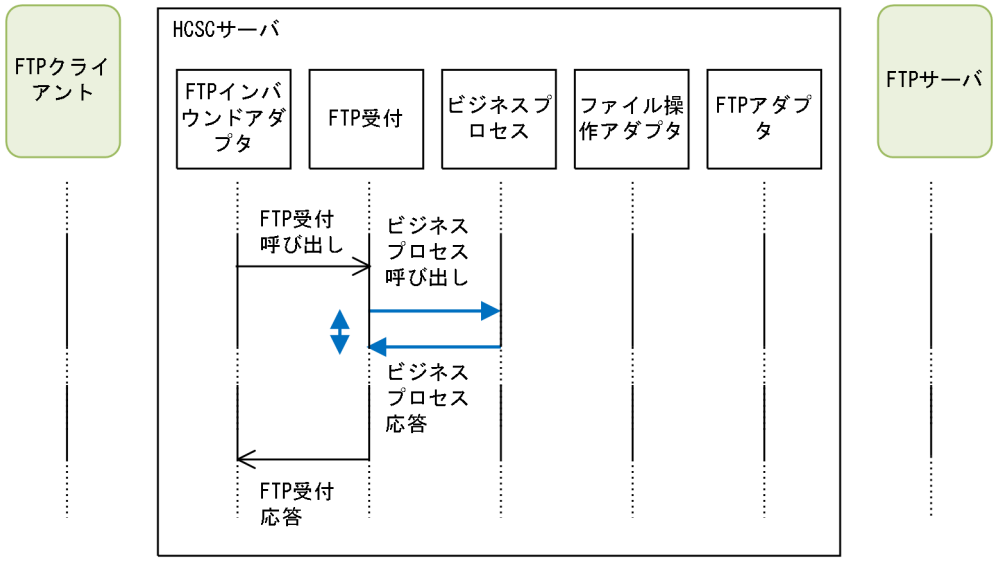
（凡例）　：トランザクションの範囲

図 8-70　FTP クライアントから別の FTP クライアントへのファイル転送の場合（HCSC サーバ内の共通フォルダからファイルを取得するとき）



（凡例）　：トランザクションの範囲

図 8-71　FTP クライアントから別の FTP クライアントへのファイル転送の場合（HCSC サーバ内の共通フォルダのファイルを直接参照するとき）



(2)　障害発生時のファイルの削除

FTP 連携では，障害が発生した場合，処理結果とファイルの状態を一致させるために各コンポーネントが出力ファイルを削除します。

FTP 連携の各処理と，対応するコンポーネントの処理を次の表に示します。

表 8-42　FTP 連携でのファイルの取り扱い

処理	コンポーネント	想定される障害要因	障害時のファイルの扱い	
			入力ファイル	出力ファイル
FTP クライアントからのファイル送信	FTP 受付	• ファイル転送中断 • ネットワーク障害 • ディスク容量不足 • ディスク障害	−	削除
FTP クライアントからのファイル受信	FTP 受付	• ファイル転送中断 • ネットワーク障害 • ディスク容量不足 • ディスク障害	処理前の状態を保証	−
FTP サーバへのファイル送信	FTP アダプタ	• ネットワーク障害 • ディスク容量不足 • ディスク障害	処理前の状態を保証	−
FTP サーバからのファイル受信	FTP アダプタ	• ネットワーク障害 • ディスク容量不足 • ディスク障害	−	削除

処理	コンポーネント	想定される障害要因	障害時のファイルの扱い	
			入力ファイル	出力ファイル
ファイル変換	ファイル操作アダプタ	• 変換エラー • ディスク容量不足 • ディスク障害	処理前の状態を保証	削除
ファイル複写	ファイル操作アダプタ	• ディスク容量不足 • ディスク障害	処理前の状態を保証	削除

（凡例）

－：該当しません。

なお，次の要因でファイルの削除が正常に終了しない場合は，システムを再起動してください。※1※2

- ファイルロックが解除できない
- ネットワーク障害やディスク障害などのハードウェア障害によって，ファイルにアクセスできない
- システムダウンによってファイルを削除できない

注※1

障害発生時に書き込み中のファイルが削除できないときは，そのファイルに対する排他ロックが解除されません。そのため，ほかのビジネスプロセスからのアクセスによって，削除対象のファイルに障害が及ぶことはありません。

注※2

共有フォルダ上の障害ファイルを削除する場合，クラスタを構成する全システムを停止する必要があります。

(3) ビジネスプロセス設計時の注意

1 つのファイル転送処理全体を 1 フェーズのトランザクションと考えるには，更新されるリソースは 1 つと見なす必要があります。FTP 連携では，複数のリソース更新の同期を合わせるための 2 フェーズ・コミットはできません。そのため，ビジネスプロセス設計時に，次のことを考慮する必要があります。

- 1 つのビジネスプロセス内で複数のリソースを更新する場合，リソース間の更新はビジネスプロセスが同期を合わせるように設計する必要があります。
- 非同期型のビジネスプロセスで，FTP クライアントに応答を返したあとに障害が発生した場合，ビジネスプロセスを再開することでロールフォワード処理をする必要があります。

8.12.4 トラブルシュート設計

障害発生時の原因調査に必要なログおよびトレースを取得できるようにシステムを設計します。

FTP 連携で取得できるログおよびトレースの種類を次の表に示します。

表 8-43　トラブルシュートのために取得する情報一覧

情報種別	取得内容	取得するコンポーネント			
		FTP インバウンドアダプタ	FTP 受付	FTP アダプタ	ファイル操作アダプタ
メッセージログ	HCSC サーバの操作時および稼働中のインフォメーション，警告，障害時のエラーなどのメッセージを取得します。	○	○	○	○
リクエストトレース	次のトレースを取得します。 • HCSC サーバのリクエスト受け付け／応答 • ビジネスプロセスの呼び出し／応答 • アダプタからのサービス呼び出し／応答	−	○	○	○
性能解析トレース	各コンポーネント間の処理の進行状況を示すトレースを取得します。	○	○	○	○
ユーザ電文トレース	決められた取得ポイントでのユーザ電文のトレースを取得します。	−	○	○	○
保守用ログ	障害の切り分け，原因調査を目的とした製品の内部処理の情報を取得します。	−	○	○	○
プロトコルトレース	HCSC サーバと外部システムとの通信プロトコルのトレースとして，FTP クライアントまたは FTP サーバとの制御コネクション上で送受信された FTP コマンドの情報を取得します。 必要に応じて，データコネクション上の情報も取得できます。	○	−	○	−

（凡例）

○：取得します。

−：取得しません。

FTP 連携で取得できるログおよびトレースの出力先や出力内容の詳細は，マニュアル「サービスプラットフォーム システム構築・運用ガイド」の「7.7.11　FTP 連携システムでの障害対策」を参照してください。

参考

FTP 連携を利用する場合，スローダウンによるファイル転送の時間ロスを低減するために，スローダウンが発生した場合の調査手段を準備しておくことを推奨します。調査方法を次の表に示します。

表 8-44　FTP 連携でのスローダウンの調査方法

想定される要因	調査方法
ネットワークの遅延	プロトコルトレース，および OS の稼働統計で調査します。
ディスク I/O の遅延	性能解析トレース，および OS の稼働統計で調査します。
排他待ち	保守用ログで調査します。
リトライ多発	保守用ログで調査します。

想定される要因	調査方法
CPU ネック	OS の稼働統計で調査します。

9 ファイルイベント連携を使用したファイルの監視

この章では，ファイルイベント連携を使用してファイルを監視する機能について説明します。

9.1　ファイル連携の機能の概要

監視対象のフォルダ内に存在するファイルを一定の間隔で監視することによって，ファイルイベント連携を実現します。監視対象のフォルダ内にファイルが作成された際のイベントを処理するために，ファイルイベント受付を使用します。ファイルイベント受付，および関連する HCSC サーバの機能を次に示します。

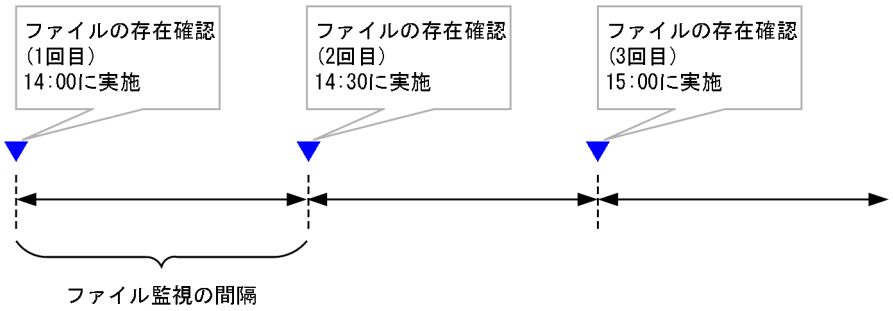
- ファイル監視機能
  監視対象のフォルダ内のファイルを監視します。
- 監視対象のファイルの状態管理機能
  監視対象のファイルのリネーム，および自動削除を実行し，監視対象のファイル状態を管理します。
- ファイルイベントトレース機能
  ファイルイベント連携の実行結果を確認するためのトレースファイルを出力します。

9.1.1　ファイル監視機能

ファイル監視機能とは，HCSC サーバからアクセスできる特定のフォルダ内での，ファイルの新規作成を監視する機能です。ファイル監視機能は，HCSC サーバランタイム定義ファイルで指定された fileevent-reception.monitor-interval プロパティの値を基準として，一定間隔でファイルの存在確認を実施します。

ファイル監視を 14:00 に開始し，fileevent-reception.monitor-interval プロパティの値を 30 分に設定した場合の例を次の図に示します。

図 9-1　ファイル監視の間隔



(1)　ファイル監視機能を有効にするための設定

ファイル監視機能は，HCSC サーバランタイム定義ファイルの fileevent-reception.monitored-file.enabled プロパティに true を設定することによって有効になります。ファイル監視機能は，ファイルイベント連携を使用する際に必要です。

注意事項

ファイルイベント受付を HCSC サーバに配備する場合は，fileevent-reception.monitored-file.enabled プロパティに必ず true を設定してください。false が設定されていると，ファイルイベント受付の起動に失敗します。

(2)　監視対象の設定

監視対象のフォルダは，ファイルイベント受付定義ファイルの urecp-fileevent.monitored.common-folder プロパティで設定します。次のどちらかを指定してください。

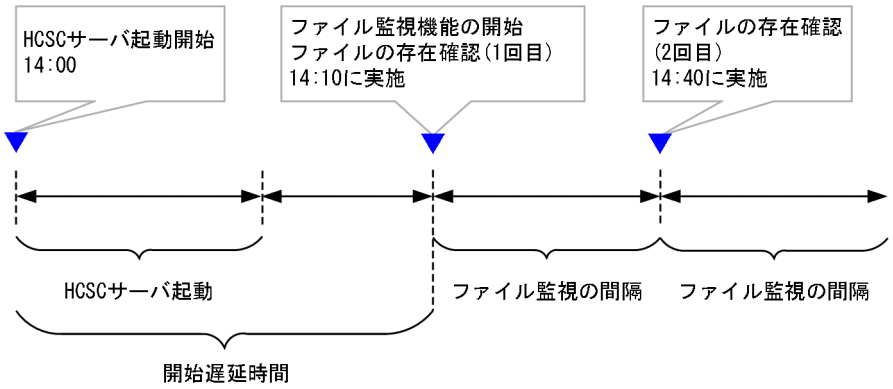
- false
  監視フォルダを監視対象とします。
- true
  共通フォルダを監視対象とします。

(3)　ファイル監視の開始

ファイル監視機能が有効になっている場合は，HCSC サーバの起動時にファイル監視が開始されます。ファイル監視は，HCSC サーバの起動コマンドが実行された直後から，HCSC サーバランタイム定義ファイルで指定された fileevent-reception.monitor-starting.delay プロパティの開始遅延時間が経過したあとに開始されます。それ以降は，HCSC サーバランタイム定義ファイルで指定された fileevent-reception.monitor-interval プロパティの値を基準として，一定間隔でファイル監視を実施します。

HCSC サーバの起動を 14:00 に開始し，開始遅延時間を 10 分，ファイル監視の間隔を 30 分で設定した場合の例を次の図に示します。

図 9-2　ファイル監視の開始



1 回目のファイルの存在確認は 14:10 に，2 回目は 14:40 に実施されます。

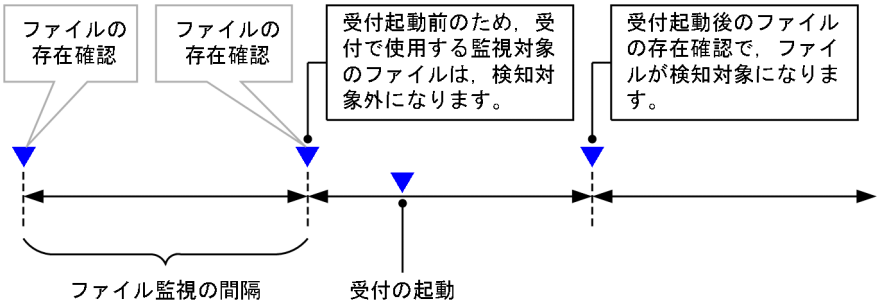
(4)　ファイル監視の終了

ファイル監視機能は，HCSC サーバを停止する際に自動的に停止します。ファイル監視機能を単独で停止することはできません。

(5)　監視対象のファイルの登録

監視対象となるフォルダ，およびファイル名などの条件は，ファイルイベント受付の起動時にファイル監視機能に登録されます。登録された内容は，次のファイルの存在確認時に反映されます。監視対象ファイルの登録の流れを次の図に示します。

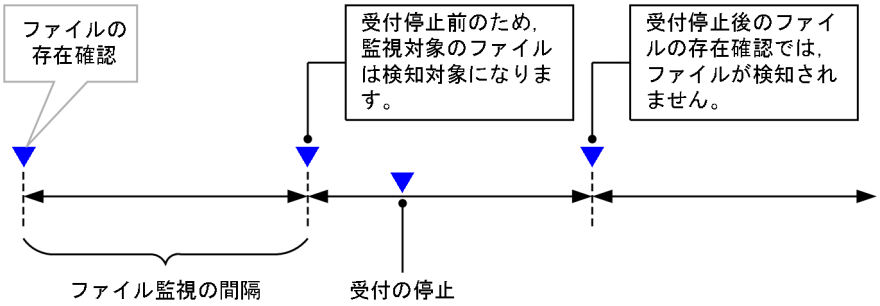
図 9-3　監視対象ファイルの登録



(6)　監視対象のファイルの登録解除

ファイル監視機能にファイル監視の対象として登録されたファイルの条件は，ファイルを登録したファイルイベント受付を停止したときにファイル監視機能から削除されます。登録が解除されたファイルの条件は，次のファイルの存在確認からファイル検知の対象外となります。

図 9-4　ファイル監視条件の登録解除



注意事項

ファイルイベント受付の停止の処理が異常終了した場合，登録されたファイルの条件がファイル監視機能から解除されずに，ファイルの検知を続けることがあります。この場合，異常停止したファイルイベント受付を起動するか，HCSC サーバを再起動してください。

(7)　監視対象のファイルと取得対象ファイル

ファイル監視機能は，監視対象のフォルダによって，監視対象ファイルと取得対象ファイルが異なります。

監視対象のフォルダによる違いを次に示します。

- 監視対象のフォルダが監視フォルダの場合

  監視フォルダ内の更新完了通知ファイルを監視します。取得対象ファイルの監視はできません。取得対象ファイルと更新完了通知ファイルの，両方を用意する必要があります。更新完了通知ファイルの詳細については「9.3.4(1)　更新完了通知ファイルの作成」を参照してください。

- 監視対象のフォルダが共通フォルダの場合

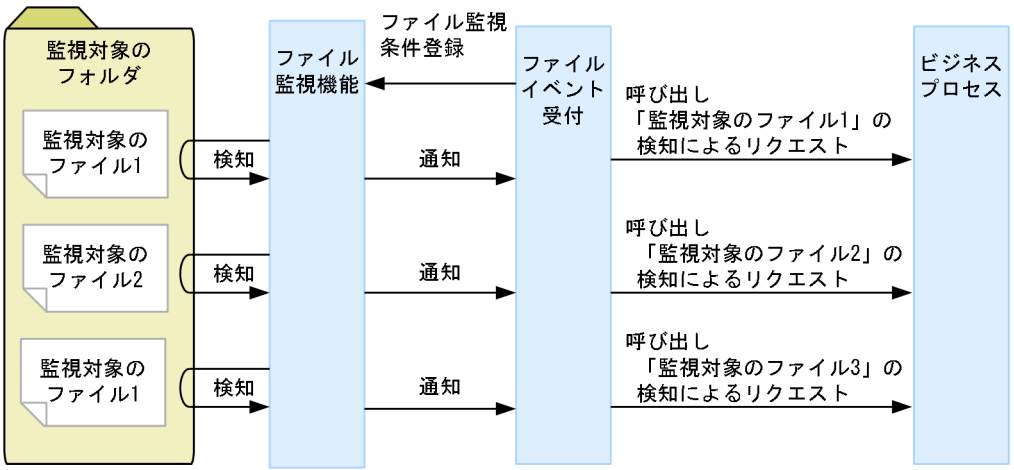
  共通フォルダ内の取得対象ファイルを監視します。監視フォルダおよび更新完了通知ファイルは不要です。取得対象ファイルの詳細については，「9.3.4(2)　取得対象ファイルの作成」を参照してください。

(8)　監視対象のファイルの数量とリクエスト回数の関係

ファイル監視機能によって複数の監視対象のファイルが検知された場合，ファイル監視機能はファイルイベント受付が検知したファイルと同じ数だけ，独立したリクエストとしてファイルイベントの検知を通知します。ただし，監視対象のファイルを通知する順番は指定できません。

監視対象のファイルが３つあり，リクエスト処理が３回実行される場合の動作を次の図に示します。

図 9-5　監視対象のファイルの数量とリクエスト回数の関係



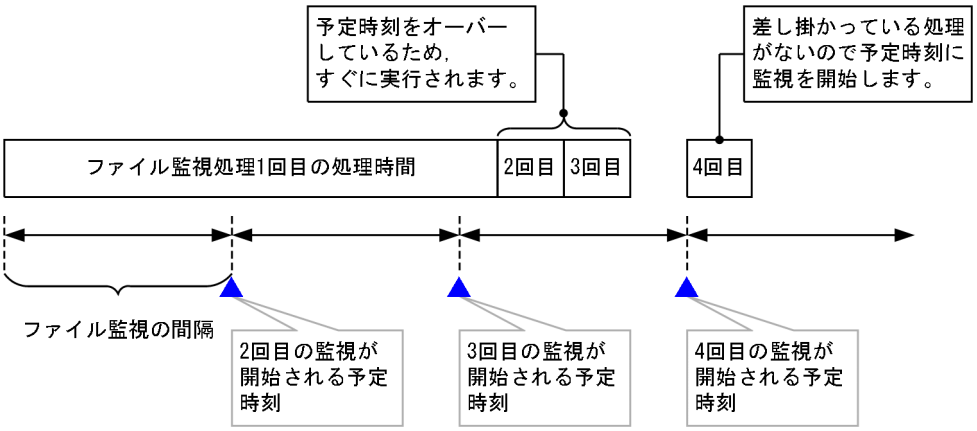
また，ファイル監視機能からファイルイベント受付へ通知する際にエラーが発生した場合は，エラーが発生した通知処理だけ終了し，そのほかに検知された処理は続行されます。

(9)　ファイル監視の処理時間がファイル監視の間隔を超えた場合の動作

処理の遅延などによって，ファイル監視処理に掛かる時間がファイル監視の間隔を超えた場合は，ファイル監視処理が終了したあと，ファイル監視の間隔を待たないで，すぐに次のファイル監視処理を開始します。

ファイル監視処理に掛かる時間がファイル監視の間隔を超えた場合の動作例を次の図に示します。

図 9-6　ファイル監視の処理時間がファイル監視間隔を超えた場合の動作



図の例では，ファイル監視 1 回目の処理時間が，2 回目と 3 回目のファイル監視予定時刻に差し掛かっています。この場合，1 回目の処理が終わった段階で 2 回目と 3 回目の予定時刻をオーバーしているため，すぐに実行されます。

9.1.2　監視対象のファイルの状態管理機能

ファイルイベント処理が不整合な状態になることを防ぐため，監視対象のファイルのリネーム，および監視対象のファイルの自動的な削除を実施し，監視対象のファイルの状態を管理します。

- 監視フォルダを監視する場合

  監視フォルダ内の更新完了通知ファイルをリネームします。また，リネーム後の更新完了通知ファイルを，ビジネスプロセスを呼び出す前に自動的に削除します。

- 共通フォルダを監視する場合

  共通フォルダ内の取得対象ファイルをリネームします。また，ファイルイベント受付定義ファイルの urecp-fileevent.file.auto-delete プロパティが true（デフォルト）の際は，ビジネスプロセスからの応答が正常のときだけ，リネーム後の取得対象ファイルを自動的に削除します。

注意事項

- 監視対象のファイルの状態管理機能では，ファイルイベントが発生した場合，検知した監視対象のファイルのファイル名をリネームします。リネームする際，ファイル名の最後に付与される文字列を次に示します。

  <監視対象のファイルの更新時刻>＋.csc_working

- 自動削除の対象になるのはリネームされた監視対象のファイルです。そのため，リネームされた監視対象のファイルに対する入出力エラーが発生した場合など，自動削除処理が終了していてもファイルが削除されない場合があります。この場合，ファイルは残り続けますが，リネームされたファイルは以降のファイル監視でファイルイベント検知の対象になりません。必要に応じて手動で削除してください。

- 共通フォルダを監視する場合に，取得対象ファイルを自動削除すると，ビジネスプロセスの応答アクティビティ後に取得対象ファイルを参照することができません。ビジネスプロセスの応答アクティビティ後に取得対象ファイルを参照する場合は，自動削除をしないでください。

9.1.3　ファイルイベントトレース機能

ファイルイベント連携の実行結果を確認するために，トレースファイルを出力します。ファイルイベント受付入口，およびファイルイベント受付出口のポイントでトレースします。トレースファイルの詳細については，マニュアル「サービスプラットフォーム システム構築・運用ガイド」の「7.7.12　ファイルイベント連携システムでの障害対策」を参照してください。

9.1.4　ファイルイベント連携使用時の注意事項

ファイルイベント連携を使用する場合の注意事項を示します。

(1)　取得対象ファイルについて

- リクエスト処理中は，外部のプロセスからリクエスト処理で参照する取得対象ファイルを更新しないでください。取得対象ファイルが不完全な状態で，ビジネスプロセス内で参照される可能性があります。
- 取得対象ファイルの転送が完了するまでは，取得対象ファイルに対してファイルロックを取得してください。ファイルロックを取得しない場合は，ファイルが更新中でもファイルイベントを検知することがあります。
- 取得対象ファイルのファイルサイズが 0 の場合はファイルイベント連携の対象になりません。ファイルサイズが 0 の取得対象ファイルをファイルイベント連携する場合は，監視フォルダを監視する方法を選択してください。

(2)　アクセス権限について

監視フォルダおよび共通フォルダ配下にあるフォルダとファイルには，HCSC サーバの管理者（Service Coordinator 管理者）がアクセスできるように適切なアクセス権限を設定してください。

(3)　監視対象のフォルダについて

次に示すフォルダおよびファイルが監視対象のフォルダ配下に作成されます。これらのフォルダおよびファイルはファイルイベント連携で使用するため，参照や削除などのファイル操作はしないでください。

- 拡張子が「.csc_lock」のファイル
- 「.csc_lock」フォルダ

また，HCSC サーバが冗長構成の場合に監視対象のフォルダを作成する際は，次の点に注意してください。

- HCSC サーバが Windows 上で動作する場合は，監視対象のフォルダは SMB によるファイル共有をしないでください。監視対象のフォルダは共有ディスク装置に作成してください。
- HCSC サーバが UNIX 系の OS 上で動作する場合に NFS によるファイル共有をする際は，NFS キャッシュ機能を使用しないでください。

9.2　ファイルイベント連携を使用するためのシステムの構成例

ファイルイベント連携を使用する際のシステムについて次に示します。

ファイル転送クライアント

　サービスプラットフォームが構築されているマシン上にファイルを転送します。

サービスプラットフォーム（ファイルイベント連携システム）

　次に示すサーバで構成されています。

- ファイル転送サーバ

  ファイル転送クライアントから転送されたファイルを受け取ります。

- ローカルディスク

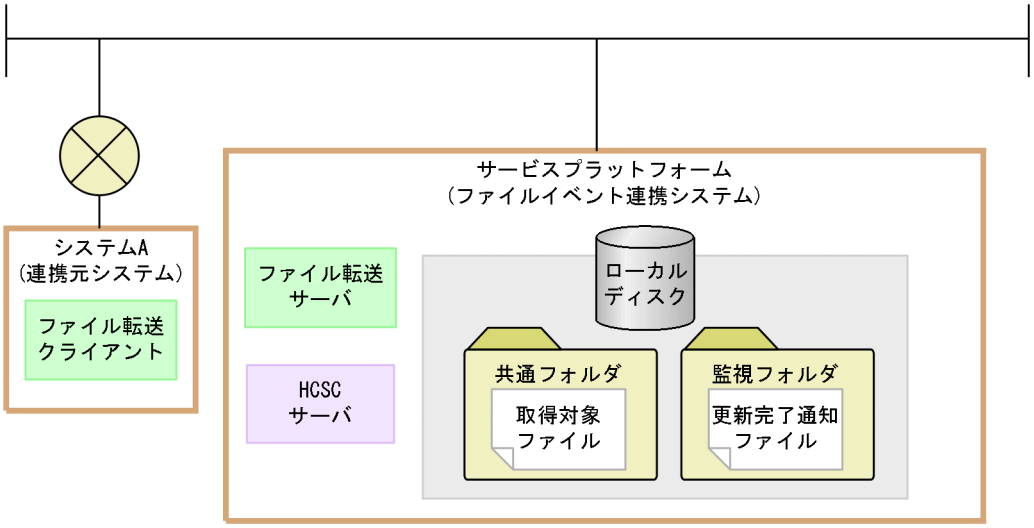
  サービスプラットフォームが存在するマシンのローカルディスクです。更新完了通知ファイルや取得対象ファイルを格納します。

なお，ファイル転送クライアントおよびファイル転送サーバは Service Coordinator の構成要素ではありません。

9.2.1　サービスプラットフォームのサーバマシンのローカルディスクを使用する構成例

1 台の HCSC サーバで，ファイルイベント連携を使用する場合の基本となるシステム構成例を次の図に示します。

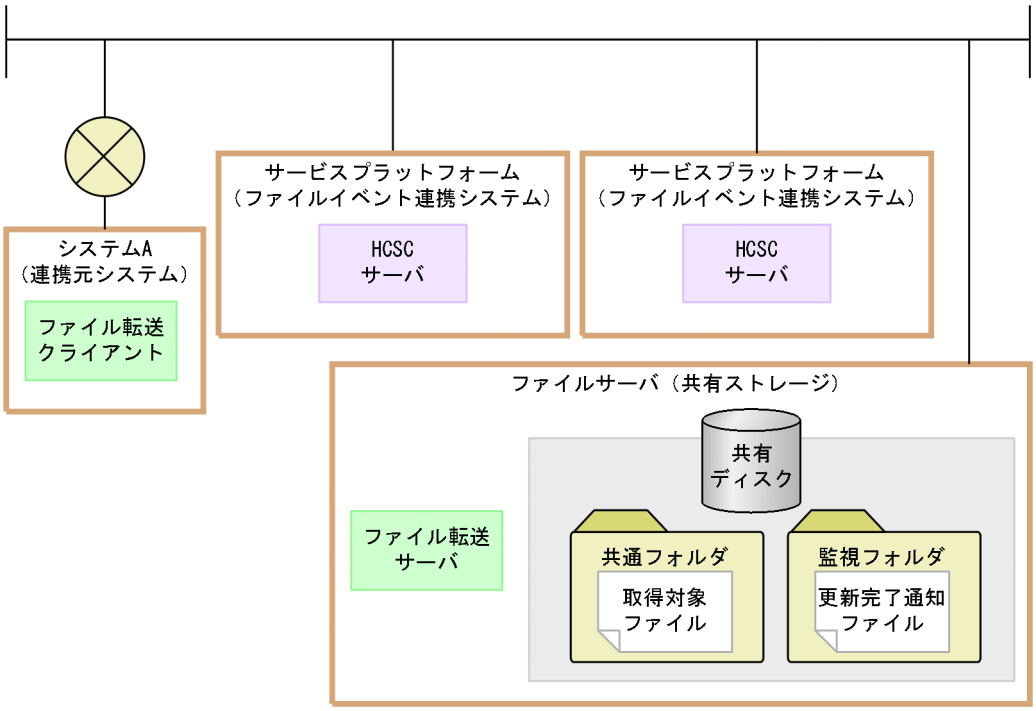
図 9-7　システム構成（サービスプラットフォームのサーバマシンのローカルディスクを使用する構成）



9.2.2　ファイルサーバの共有ディスクを使用する構成例

HCSC サーバが冗長構成の場合で，ファイルイベント連携を使用する際の基本となるシステム構成例を次の図に示します。冗長構成内の HCSC サーバが同一のファイルサーバを使用することで，複数の HCSC サーバでファイルイベントを処理できます。複数の HCSC サーバで同一のファイルイベントは処理しません。

図 9-8　システム構成（ファイルサーバの共有ディスクを使用する構成）

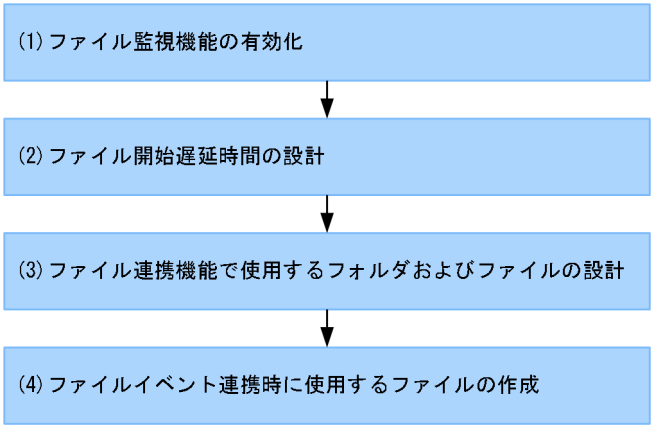


9.3　ファイルイベント連携で必要な設定

ファイルイベント連携で必要な設定について説明します。

9.3.1　ファイルイベント連携の設定の流れ

ファイルイベント連携を使用する場合の流れを次の図に示します。



(1)　ファイル監視機能の有効化

ファイル監視機能は，HCSC サーバランタイム定義ファイルの fileevent-reception.monitored-file.enabled プロパティに true を設定することによって有効になります。

サーバランタイム定義ファイルの詳細については，マニュアル「サービスプラットフォーム リファレンス」の「6.5.6　HCSC サーバランタイム定義ファイル」を参照してください。

(2)　ファイル開始遅延時間の設計

ファイル監視は，HCSC サーバランタイム定義ファイルの fileevent-reception.monitor-starting.delay プロパティに指定している開始遅延時間が経過したあとに，開始されます。そのため，fileevent-reception.monitor-starting.delay プロパティには，HCSC サーバの起動に掛かる時間（起動コマンドを実行してから返るまでの時間）より大きな値を設定してください。HCSC サーバの起動中に開始遅延時間が経過した場合は，条件に一致するファイルが検知されない場合があります。

サーバランタイム定義ファイルの詳細については，マニュアル「サービスプラットフォーム リファレンス」の「6.5.6　HCSC サーバランタイム定義ファイル」を参照してください。

(3)　ファイル連携機能で使用するフォルダおよびファイルの設計

ファイルイベント連携機能を使用する際に，設計が必要なフォルダおよびファイルがあります。

詳細については，「9.3.2　ファイルイベント連携機能で設計が必要な項目」を参照してください。

(4)　ファイルイベント連携時に使用するファイルの作成

ファイルイベント連携時は，次のファイルを作成する必要があります。

- 更新完了通知ファイル
  詳細については，「9.3.4(1)　更新完了通知ファイルの作成」を参照してください。
- 取得対象ファイル
  詳細については，「9.3.4(2)　取得対象ファイルの作成」を参照してください。

9.3.2　ファイルイベント連携機能で設計が必要な項目

ファイルイベント連携機能では，監視対象とするフォルダやファイル，および取得ファイルの扱いなどを設計し，必要に応じて定義ファイルやクライアントで設定を実施します。設計および設定が必要な項目を次に示します。

表 9-1　ファイルイベント連携機能で設計が必要な項目

設定項目	定義ファイル	プロパティ	説明
監視対象のフォルダ	ファイルイベント受付定義ファイル	urecp-fileevent.monitored.common-folder	ファイルイベント受付で監視の対象にするフォルダを指定します。 • 共通フォルダの場合 true • 監視フォルダの場合 false（デフォルト）
サブフォルダ		urecp-fileevent.sub-folder	ファイルイベント受付ごとに使用するサブフォルダを設定します。1 つの受付につき 1 個所だけ設定できます。
監視対象のファイル		urecp-fileevent.monitored-file-name.regex	監視対象のファイルと見なすファイル名を，正規表現で指定します。
取得対象ファイルの自動削除		urecp-fileevent.file.auto-delete	ファイルイベント受付で監視の対象を共通フォルダにしている場合，取得対象ファイルを自動的に削除するかどうかを指定します。 • 削除する場合 true（デフォルト） • 削除しない場合 false 監視対象のフォルダが監視フォルダの場合は無視されます。
共通フォルダ		urecp-fileevent.common-folder.definition-name	取得対象ファイルを格納する共通フォルダの，共通フォルダ定義名を指定します。

設定項目	定義ファイル	プロパティ	説明
共通フォルダ	ファイルイベント受付定義ファイル	urecp-fileevent.common-folder.definition-name	デフォルト値を次に示します。 default
監視フォルダ	HCSC サーバランタイム定義ファイル	fileevent-reception.monitored-folder	監視フォルダのルートフォルダを指定します。1 つの HCSC サーバに対して，1 個所だけ設定できます。 監視対象のフォルダが共通フォルダの場合は無視されます。 デフォルト値を次に示します。 <サービスプラットフォームのインストールディレクトリ>/CSC/spool/fileevent
取得対象ファイル	ファイルイベント連携クライアントが配置します。定義ファイルのプロパティでは設定しません。		処理対象となるファイルです。ファイルイベント受付で存在確認を実施します。

9.3.3　ファイルイベント受付が監視するフォルダの設定

ファイルイベント受付が監視するフォルダの設定について，次に示します。

監視フォルダ

取得対象ファイルの更新完了後に，更新完了通知ファイルを作成することによって，取得対象ファイルが完全な状態であることを保証したファイル連携ができます。取得対象ファイル以外に，更新完了通知ファイルの運用が必要になります。

共通フォルダ

更新完了通知ファイルが不要なため，よりシンプルにシステムを構成できます。取得対象ファイルの更新が完了するまで排他を掛ける必要があります。

詳細については「9.1.2　監視対象のファイルの状態管理機能」を参照してください。

(1)　監視フォルダの設定

監視フォルダのパスは，HCSC サーバランタイム定義ファイルの fileevent-reception.monitored-folder プロパティで設定できます。

デフォルトの監視フォルダは，製品のインストール時に作成される次のフォルダです。

```
<サービスプラットフォームのインストールディレクトリ>/CSC/spool/fileevent
```

デフォルト以外のフォルダを使用する場合は，監視フォルダとするフォルダを作成し，その絶対パスを HCSC サーバランタイム定義ファイルの fileevent-reception.monitored-folder プロパティに設定してください。

注意事項

- 監視フォルダおよび共通フォルダに同一のフォルダを指定しないでください。同一のフォルダを指定した場合の動作は保証されません。
- 監視フォルダのパスには ISO-8859-1 文字エンコーディングを使用してください。

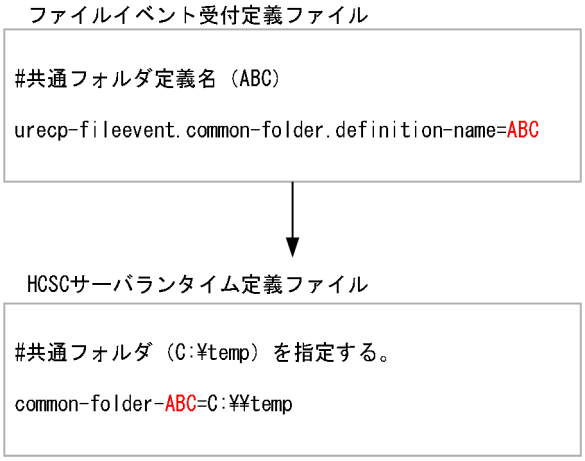
(2) 共通フォルダの設定

共通フォルダは，HCSC サーバランタイム定義ファイルの common-folder-<共通フォルダ定義名>プロパティで指定します。指定しない場合は，デフォルトの設定値のフォルダを共通フォルダとして使用します。

デフォルト以外のフォルダを使用する場合は，共通フォルダを作成する必要があります。この場合，HCSC サーバランタイム定義ファイルの common-folder-<共通フォルダ定義名>プロパティで設定してください。また，取得対象ファイルの格納場所を指定するために，共通フォルダ定義名をファイルイベント受付定義ファイルの urecp-fileevent.common-folder.definition-name プロパティに設定する必要があります。

共通フォルダ定義名を指定する例を次の図に示します。

図 9-9 共通フォルダ定義名の設定の例



注意事項

監視フォルダを監視する受付と，共通フォルダを監視する受付は，それぞれ異なる共通フォルダを指定してください。共通フォルダが同一の場合の動作は保証されません。

(3) サブフォルダの設定

サブフォルダは，監視対象のフォルダの配下に指定できます。監視対象のフォルダの配下にサブフォルダを手動で作成し，ファイルイベント受付定義ファイルの urecp-fileevent.sub-folder プロパティでサブフォルダのパスを設定してください。

なお，監視フォルダを監視する場合，監視フォルダ配下のフォルダ構成と共通フォルダ配下のフォルダ構成を一致させる必要があります。監視フォルダ配下にあるサブフォルダと同名のサブフォルダを，共通フォルダ配下に作成してください。

9.3.4　ファイルイベント連携時に作成するファイル

ファイルイベント連携時に作成するファイルについて説明します。

(1)　更新完了通知ファイルの作成

更新完了通知ファイルとは，クライアントが HCSC サーバにファイルイベントを通知するために作成するファイルです。クライアントは取得対象ファイルの転送が完了したあとで，このファイルを作成する必要があります。更新完了通知ファイルと取得対象ファイルは同じファイル名でなければなりませんが，同一のファイルでなくても構いません。

更新完了通知ファイルの格納先

更新完了通知ファイルは，処理したいファイルイベント受付に対応する監視フォルダの配下に格納してください。

更新完了通知ファイルの命名規則

更新完了通知ファイル名の命名規則を次に示します。

- 取得対象ファイルと同じファイル名にしてください。
- ファイルイベント受付定義ファイルで設定した urecp-fileevent.monitored-file-name.regex プロパティの設定値の正規表現に合致するファイル名にしてください。

ただし，「.csc」で始まる文字列はサービスプラットフォームで予約されているため，更新完了通知ファイルに「.csc」を含む文字列は指定できません。ファイル名に「.csc」を含むファイルは更新完了通知ファイルと見なしません。

(2)　取得対象ファイルの作成

取得対象ファイルは，ビジネスプロセスなどで処理の対象とするファイルです。このファイルは，システムを利用するユーザがファイルイベント連携処理を開始する際に，ファイル連携ソフトウェアなどを用いて格納します。

取得対象ファイルの格納

監視フォルダを監視する場合は，取得対象ファイルは更新完了通知ファイルを作成する前に格納してください。

取得対象ファイルの格納先

取得対象ファイルは，ファイルイベント受付定義ファイルで設定した urecp-fileevent.common-folder.definition-name プロパティの値に対応する共通フォルダ内に格納してください。

監視フォルダを監視する場合は，監視フォルダから見た更新完了通知ファイルの相対パスと，共通フォルダから見た取得対象ファイルの相対パスは一致している必要があります。相対パスが異なるときはエラーとなります。また，相対パスを比較する際の大文字・小文字による違いは，OS のファイルシステムに依存します。正常になるケースとエラーになるケースを次に示します。

正常なケース

```
更新完了通知ファイル格納先 → <監視フォルダパス>/sub_folder1/upload12345.txt
取得対象ファイル格納先 → <共通フォルダパス>/sub_folder1/upload12345.txt
```

エラーとなるケース（サブフォルダが異なる）

```
更新完了通知ファイル格納先 → <監視フォルダパス>/sub_folder1/upload12345.txt
取得対象ファイル格納先 → <共通フォルダパス>/sub_folder2/upload12345.txt
```

（凡例）

赤字：差異がある個所を示します。

エラーとなるケース（ファイル名が異なる）

```
更新完了通知ファイル格納先 → <監視フォルダパス>/sub_folder1/upload12345.txt
取得対象ファイル格納先 → <共通フォルダパス>/sub_folder1/upload54321.txt
```

（凡例）

赤字：差異がある個所を示します。

取得対象ファイルの命名規則

共通フォルダを監視する場合の取得対象ファイルのファイル名は，ファイルイベント受付定義ファイルで設定した，urecp-fileevent.monitored-file-name.regex プロパティの設定値の正規表現と合致したファイル名にする必要があります。ただし，「.csc」で始まる文字列はサービスプラットフォームで予約されているため，更新完了通知ファイルに「.csc」を含む文字列は指定できません。ファイル名に「.csc」を含むファイルは取得対象ファイルと見なしません。

9.3.5　ファイルイベント連携に関する実行環境の設定

実行環境での，ファイルイベント連携機能に関する設定について説明します。

(1)　ファイルイベント受付の定義ファイルの作成

ファイルイベント受付で，実行環境に依存する定義を設定する場合，ファイルイベント受付コンフィグファイル，およびファイルイベント受付共通コンフィグファイルを使用します。詳細を次の表に示します。

表 9–2　実行環境で設定するファイルイベント受付の定義ファイル

ファイルの種類	ファイル名	説明
ファイルイベント受付コンフィグファイル	<ファイルイベント受付の受付ID>.properties	ファイルイベント受付のコンフィグファイルです。ファイルイベントトレースの設定などを定義できます。
ファイルイベント受付共通コンフィグファイル	fileeventrecp_config.properties	すべてのファイルイベント受付で共通して定義できるファイルイベント受付コンフィグファイルです。

複数のファイルイベント受付で共通の定義値を設定する場合は，各ファイルイベント受付のファイルイベント受付コンフィグファイルではなく，ファイルイベント受付共通コンフィグファイルに設定します。これによって，すべてのファイルイベント受付に対してプロパティ値を有効にできます。

ファイルイベント受付コンフィグファイル，およびファイルイベント受付共通コンフィグファイルに同一のプロパティを設定している場合は，ファイルイベント受付コンフィグファイルのプロパティの設定が有効になります。有効になるプロパティは，プロパティ単位で判定されます。

ファイルイベント受付コンフィグファイルおよびファイルイベント受付共通コンフィグファイルの詳細については，マニュアル「サービスプラットフォーム リファレンス」の「6.9.1　ファイルイベント受付コンフィグファイル」，および「6.9.2　ファイルイベント受付共通コンフィグファイル」を参照してください。

(2)　同時実行スレッド数の制御の設定

HCSC サーバランタイム定義の fileevent-reception.monitored-file.enabled プロパティに true が設定され，ファイル監視機能が有効になっている場合，ファイル監視機能のためにスレッドが生成されます。ファイル監視機能のためのスレッドは，1 つだけ生成されます。同時実行数の設定はできません。

また，ファイルイベント受付単位でファイルイベントを処理するスレッドが生成されます。生成されたスレッドは，スレッドプールで管理されます。常駐するスレッドは，ファイルイベント受付起動時に生成されます。常駐するスレッドの数（以降，常駐スレッド数と呼びます）はファイルイベント受付定義ファイルの urecp-fileevent.thread-pool.minimum プロパティで設定できます。

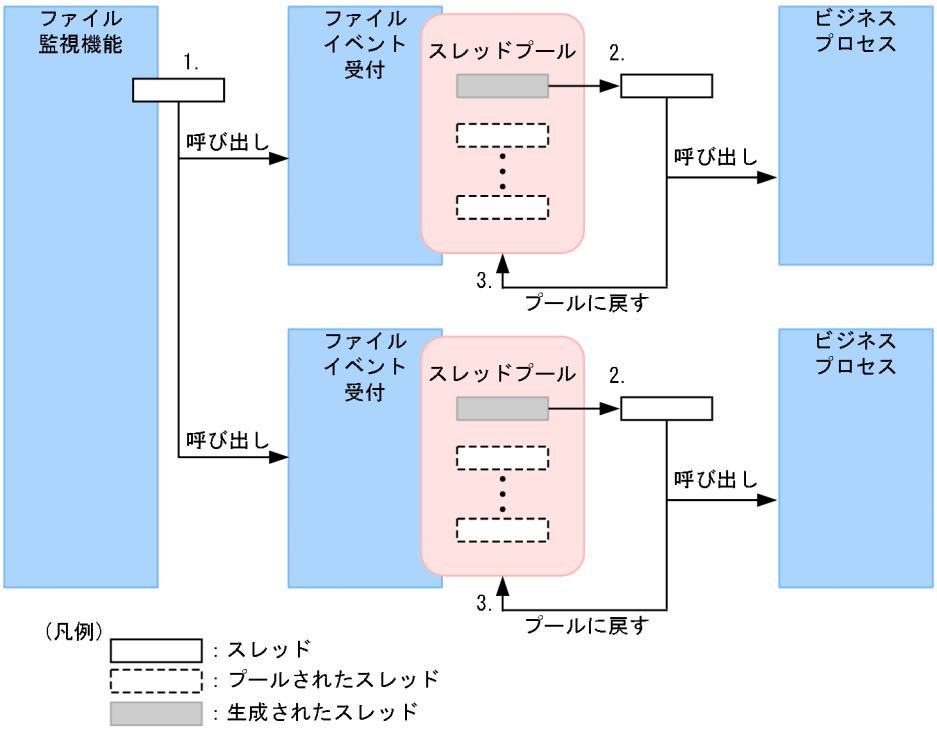
常駐スレッド数を超える数のファイルイベントが通知された場合は，ファイルイベントを処理するときにスレッドが一時的に追加生成されます。

ファイルイベント受付の処理のために生成されるスレッドの最大数（常駐スレッド数＋追加生成されたスレッド数）は，ファイルイベント受付定義ファイルの urecp-fileevent.thread-pool.maximum プロパティで設定できます。なお，常駐スレッド数と最大スレッド数を同じ値に設定した場合，スレッドは追加生成されません。

ファイルイベント受付定義ファイルの詳細については，マニュアル「サービスプラットフォーム リファレンス」の「3.12.1　ファイルイベント受付定義ファイル」を参照してください。

ファイルイベント連携機能で使用するスレッドの例を次の図に示します。

図 9-10　ファイルイベント連携機能で使用するスレッドの例



図中の 1.～3.について説明します。

1. ファイル監視機能のスレッドは，HCSC サーバの起動時に生成され，HCSC サーバ停止まで常駐します。
2. ファイルイベント受付の処理は，スレッドプール内のスレッドで実行されます。
   スレッドプールは，ファイルイベント受付単位に存在します。
3. ファイルイベント受付の処理終了後に，ファイルイベント受付が使用したスレッドがスレッドプールに戻されます。

9.3.6　ファイルイベント連携のエラー設計の概要

ファイルイベント連携のエラー設計で考慮するポイントを次に示します。

(1)　障害通知

ファイルイベント連携機能では，処理中に障害が発生した場合でもクライアントにエラー応答を返すことはできません。ただし，システム運用者に対しては，メッセージの出力で障害発生を通知します。障害発生のメッセージは，メッセージログに出力されます。これらの情報を JP1 などのモニタリング機能で監視することで，システム運用者に障害の発生を通知できます。

(2)　回復処理

ファイルイベント受付の処理中に障害が発生した場合，ビジネスプロセスの実行履歴を取得している場合は，ビジネスプロセスインスタンスの再実行によって処理を途中から再開できることがあります。ビジネ

スプロセスの実行履歴を取得していない場合は，再度監視対象のファイルを作成することで処理を最初から実行し直す必要があります。

注意事項

ビジネスプロセスの再実行時は，次の処理は行われません。

- ファイルイベントトレースの出力
- 共通フォルダを監視する場合の，取得対象ファイルの自動削除

10 作業フォルダおよび共通フォルダの機能

この章では，ファイル転送を中継するために利用する作業フォルダおよび共通フォルダについて説明します。

10.1　作業フォルダおよび共通フォルダにアクセスできるコンポーネント

作業フォルダおよび共通フォルダにアクセスできるコンポーネントを次の表に示します。

表 10–1　作業フォルダおよび共通フォルダにアクセスできる受付

受付種別	作業フォルダ	共通フォルダ
FTP 受付	○	○※
SOAP 受付（ユーザ定義受付）	○	×
HTTP 受付	○	×
TP1/RPC 受付	×	×
Message Queue 受付	×	×
ファイルイベント受付	×	○
Kafka 受付	×	×
gRPC 受付	×	×
スケジュール駆動受付	○	×

（凡例）

○：アクセスできます。

×：アクセスできません。

注※

GET オペレーションのみ共通フォルダ内のファイルへアクセス可能です。詳細は，「8.4.10　共通フォルダと中間ファイルの作成」を参照してください。

表 10–2　作業フォルダおよび共通フォルダにアクセスできるアダプタ

アダプタ種別	作業フォルダ	共通フォルダ
ファイル操作アダプタ	○	○
FTP アダプタ	○	○
SOAP アダプタ	○	○
DB アダプタ	×	×
TP1 アダプタ	×	×
ファイルアダプタ	×	×
Object Access アダプタ	×	×
Message Queue アダプタ	×	×
メールアダプタ	○	○
HTTP アダプタ	○	○
コマンドアダプタ	○	○

アダプタ種別	作業フォルダ	共通フォルダ
SFTP アダプタ	○	○
Kafka アダプタ	×	×
gRPC アダプタ	×	×

（凡例）

○：アクセスできます。

×：アクセスできません。

注意事項

作業フォルダおよび共通フォルダとして設定したフォルダには，HCSC サーバ管理者のオーナー，およびグループを設定してください。

Windows の場合は，作業フォルダおよび共通フォルダに対するアクセス許可をフルコントロールにしてください。

UNIX の場合は，アクセス権限に（7:rwx）を設定してください。

10.2　作業フォルダを利用したファイルアクセス

作業フォルダとは，各ファイル転送パターンでサービスプラットフォームが管理する，一時的な中間ファイルを格納するためのフォルダです。

10.2.1　作業フォルダルートの定義

作業フォルダルートは，HCSC サーバに対して 1 つだけ存在します。作業フォルダルートの定義は，HCSC サーバランタイム定義ファイルの work-folder プロパティに記述します。

デフォルトの作業フォルダルートは次のとおりです。

```
<インストールディレクトリ>/CSC/spool/ftp/work
```

HCSC サーバランタイム定義ファイルについては，マニュアル「サービスプラットフォーム リファレンス」の「6.5.6　HCSC サーバランタイム定義ファイル」を参照してください。

10.2.2　作業フォルダとリクエスト ID の生成

作業フォルダを使用する受付が，クライアントからリクエストを受け付けると，リクエストごとにリクエスト ID と作業フォルダが生成されます。

(1)　リクエスト ID の生成

リクエストごとに一意となるリクエスト ID が生成されます。リクエスト ID の形式は次のとおりです。

リクエスト ID の生成規則

```
リクエストID＝<受付ID><クライアントのIPアドレス><日付><時刻><追番><分割キー><HCSCサーバ名>
```

表 10-3　リクエスト ID の形式

項番	項目	桁数	内容	備考
1	<受付 ID>	8 桁	受付 ID です。 受付 ID が 8 桁に満たない場合は，受付 ID の右側に足りない分の"-"が付加されます。	－
2	<クライアントの IP アドレス>	12 桁（IPv4 の場合），または 32 桁（IPv6 の場合）	要求元のクライアントの IP アドレスです。 • IPv4 の場合 IP アドレスの各バイトを 10 進数 3 桁で表し，結合した文字列になります。	－

項番	項目	桁数	内容	備考
2	<クライアントの IP アドレス>	12 桁（IPv4 の場合），または 32 桁（IPv6 の場合）	IP アドレスのそれぞれのバイトが 3 桁に満たない場合は，各バイトの左側に足りない分の「0」が設定されます。 • IPv6 の場合 IP アドレスの各バイトを 16 進数 4 桁で表し，結合した文字列になります。 IP アドレスの各バイトが 4 桁に満たない場合は，各バイトの左側に足りない分の「0」が設定されます。 英字は小文字で表記されます。	－
3	<日付>	8 桁	受付がリクエストを受け付けた日付を表す文字列。"yyyyMMdd"形式の文字列となります。	－
4	<時刻>	9 桁	受付がリクエストを受け付けた時刻を表す文字列。ミリ秒単位までを示す"HHmmssSSS"形式の文字列となります。	－
5	<追番>	3 桁	ほかのリクエストの処理のために，項番 3，項番 4 の部分が全く同じリクエスト ID が現在使用されていない場合は，"000"となります。 ほかのリクエストの処理のために項番 3，項番 4 の部分が全く同じリクエスト ID が現在使用されている場合は，この追番を 1 ずつインクリメントして重複しないリクエスト ID となる 3 桁の数値が使用されます。	もし"999"まで試行しても重複しない追番が設定できなかった場合は，項番 3，項番 4 を変えて"000"で採番し直します。
6	<分割キー>	1 桁	HCSC サーバ名が付加されていることを示す分割キーワード。"/"を使用します。	－
7	<HCSC サーバ名>	8 桁	受付がデプロイされている HCSC サーバの名称。 HCSC サーバ名が 8 桁に満たない場合でも，"-"による文字埋めはしません。	－

（凡例）

－：該当する項目はありません。

リクエスト ID の例を次に示します。

1. IPv4 アドレスの場合

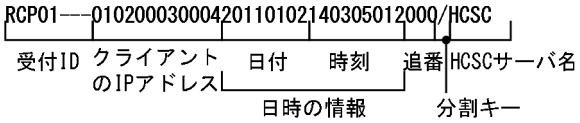
受付 ID：RCP01

クライアントの IP アドレス：10.200.30.4

実行日時：2011/1/2 14:03:05.012

HCSC サーバ名：HCSC

図 10–1　IPv4 アドレスの場合のリクエスト ID



2. IPv6 アドレスの場合

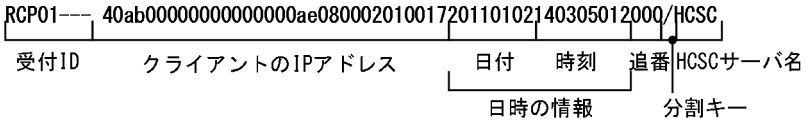
受付の受付 ID：RCP01

クライアントの IP アドレス：40AB:0:0:0:AE:800:201:17

実行日時：2011/1/2 14:03:05.012

HCSC サーバ名：HCSC

図 10–2　IPv6 アドレスの場合のリクエスト ID



(2)　作業フォルダの生成

次の規則に従って作業フォルダルート以下にフォルダが生成されます。

1. 作業フォルダルート以下に，HCSC サーバ名をフォルダ名とするフォルダが生成されます。このフォルダを **HCSC サーバフォルダ**と呼びます。
2. HCSC サーバフォルダ以下に，受付 ID をフォルダ名とするフォルダが生成されます。このフォルダを **受付 ID フォルダ**と呼びます。
3. 受付 ID フォルダ以下に，フォルダが生成されます。このフォルダを作業フォルダと呼びます。作業フォルダ名は，リクエスト ID の<クライアントの IP アドレス>から<追番>までの値が使用されます。

ただし，HTTP 受付では，例外として 32 桁すべて「0」で命名された特殊な作業フォルダも別途起動時に生成されます。これは，受信したデータをすべてメモリ上に展開することによるメモリ圧迫を避けるため，HTTP 受付処理前に一時的に受信データをすべてファイルに出力する際に使用するフォルダです。通常の作業フォルダとは用途が異なります。

10.2.3　作業フォルダ内のファイルの生成

各コンポーネントによって作業フォルダ内に生成されるファイルは，次の規則に従ったファイル名になります。また，ファイルは，HCSC サーバの管理者の権限で生成されます。

作業フォルダ内ファイル名の生成規則

```
ファイル名＝csc_<ユニークID>
```

ユニーク ID：フォルダ内に存在しないファイル名になるような文字列。

10.2.4　作業フォルダ内のファイルの排他制御

作業フォルダ内のファイルへのアクセスには，読み込み，書き込みともにファイルロックは取得されません。

10.2.5　作業フォルダおよび作業フォルダに格納された中間ファイルの削除

作業フォルダおよび作業フォルダに格納された中間ファイルは，次の方法で削除できます。

- FTP 受付定義ファイルの「urecp-ftp.work-dir.auto-delete」
- HTTP 受付定義ファイルの「httprecp.work-dir.auto-delete」
- スケジュール駆動受付定義ファイルの「scdrecp.work-dir.auto-delete」
- ファイル操作アダプタの削除オペレーション
- フォルダ運用管理コマンド

ただし，受付による自動削除の方法では，プロセスが異常終了した場合は自動削除が実行されないで作業フォルダおよび中間ファイルが残ります。また，ビジネスプロセスを再実行した場合も受付が呼び出されないため，自動削除は実行されません。その場合は，フォルダ運用管理コマンドで別途削除する必要があります。

フォルダ運用管理コマンドによる削除方法は，マニュアル「サービスプラットフォーム システム構築・運用ガイド」の「5.3.32　作業フォルダを削除する」を参照してください。

10.3　共通フォルダを利用したファイルアクセス

共通フォルダとは，複数のビジネスプロセスがファイルを参照する場合に，共通して利用するファイルを格納するためのフォルダです。共通フォルダはすべてのビジネスプロセスからアクセスできます。複数のビジネスプロセスがファイルを参照する場合とは，FTP クライアントから別の FTP クライアントへのファイル転送パターンなどを指します。

参考

- 共通フォルダ内のサブフォルダへは，共通フォルダからの相対パスを指定した場合だけアクセスできます。
- 共通フォルダ内にあるファイルの読み込み，および書き込みは HCSC サーバの管理者（Service Coordinator 管理者）の権限で実行されます。

10.3.1　共通フォルダ定義名と対応するパスの定義

共通フォルダは HCSC サーバに対して複数定義できます。共通フォルダ定義名は共通フォルダを識別するために使用します。共通フォルダ 1 つに対して，共通フォルダ定義名が 1 つ存在する必要があります。

共通フォルダはシステム構築時にユーザが作成します。作成したフォルダのフルパスを HCSC サーバランタイム定義の common-folder-<共通フォルダ定義名>プロパティに記述することで，共通フォルダとして利用できます。

定義例を次に示します。

```
common-folder-FOLDER1=/usr/share/store/share1
common-folder-FOLDER2=/usr/share/store/share2
```

デフォルトの共通フォルダ定義名は default です。対応する共通フォルダのパスは次のとおりです。

```
<インストールディレクトリ>/CSC/spool/ftp/com
```

HCSC サーバランタイム定義ファイルについては，マニュアル「サービスプラットフォーム リファレンス」の「6.5.6　HCSC サーバランタイム定義ファイル」を参照してください。

10.3.2　共通フォルダ内のファイルの生成

共通フォルダ内にファイルを生成する際，次のファイル名が使用できます。

- ユーザ指定
- 製品が自動生成した一意な ID

共通フォルダにファイルを生成するコンポーネントとファイル名の関係を次の表に示します。

表 10-4　共通フォルダにファイルを生成するコンポーネントとファイル名の関係

種別	ユーザ指定	自動生成
HTTP アダプタ	×	○
SOAP アダプタ	×	○
上記以外	○	×

（凡例）

○：ファイル名に使用できます。

×：ファイル名に使用できません。

(1)　ユーザが指定するファイル名

共通フォルダ内にファイルを生成する場合，そのファイル名は要求電文で指定します。

ファイル名は，共通フォルダを起点とした相対パスを含むファイルパスで指定できます。指定したファイルパスに対するサブフォルダが共通フォルダ内に存在しない場合は，ファイルを生成するコンポーネントがサブフォルダを生成します。

- 注意事項
  - 必ず共通フォルダからの相対パスとして扱われるため，絶対パスは指定しないでください。
  - 親ディレクトリ表現「../」「..¥」やカレントディレクトリを含む表現は，共通フォルダの配下とならない場合にエラーとなります。

(2)　製品が自動生成するファイル名

システム内で一意の ID を動的に採番し，それを出力ファイル名として使用します。一意の ID の生成にはタイムスタンプを使用しているため，時刻補正をする場合には注意してください。

SOAP アダプタの場合は，自動生成するファイルの格納先に共通フォルダを起点とした相対パスを指定できます。指定したファイルパスに対応するサブフォルダが共通フォルダ内に存在しない場合は，ファイルを生成するコンポーネントがサブフォルダを生成します。

- 注意事項
  - 必ず共通フォルダからの相対パスとして扱われるため，絶対パスは指定しないでください。
  - 親ディレクトリ表現「../」「..¥」やカレントディレクトリを含む表現は，共通フォルダの配下とならない場合にエラーとなります。

10.3.3　共通フォルダ内のファイルの排他制御

共通フォルダ内のファイルへのアクセス時は，読み込み，書き込みともにファイルロックを取得します。

(1)　ロック種別

共通フォルダ内のファイルへのアクセスでは，読み込み時には共有ロック，書き込み時には占有ロックをそれぞれ取得します。現在のロックが共有ロックまたは占有ロックのファイルに対するロックの可否を次の表に示します。なお，他プロセスとは，ロックしたプロセス以外に排他管理の API を使用してアクセスするプロセスのことを指します。

表 10–5　共有ロックと占有ロック

現在のロック	他プロセス／スレッドからの再ロック	
	共有ロック	占有ロック
共有ロック	○	×
占有ロック	×	×

（凡例）

○：ロックできます。

×：ロックできません。

(2)　ロック取得時のパラメタ

ファイルロックの取得時に使用する次のパラメタについて説明します。

- ロックリトライ回数
- ロックリトライ間隔

ロックリトライ回数

すでに別のプロセスまたはスレッドによってファイルのロックが取得されている場合，ファイルのロック取得が失敗します。このとき，与えられたロックリトライ回数だけ再度ロックを試みます。

ロックリトライ間隔

ロック失敗から次のロック取得を試みるまでの時間間隔（秒）です。

10.3.4　共通フォルダ内のファイルの削除

共通フォルダ自体はユーザが削除します。共通フォルダ内のファイルは次の方法で削除できます。

- ファイル操作アダプタの削除オペレーションを利用した削除

  ファイル操作アダプタの削除オペレーションを利用した方法については，「8.7.1　ファイル操作アダプタでサポートするオペレーション」を参照してください。

- フォルダ運用管理コマンドを利用した削除

  フォルダ運用コマンドを利用した方法については，マニュアル「サービスプラットフォーム システム構築・運用ガイド」の「5.4.22　共通フォルダ内のファイルを削除する」を参照してください。

- FTP 受付による自動削除

  共通フォルダ内のファイルを FTP 受付で削除する方法については，「8.4.10(4)　共通フォルダ内のファイルの削除」を参照してください。

  ただし，FTP 受付による自動削除の方法では，プロセスが異常終了した場合は自動削除が実行されないで，共通フォルダ内のファイルが残ります。また，ビジネスプロセスを再実行した場合も受付が呼び出されないため，自動削除は実行されません。その場合は，フォルダ運用管理コマンドで別途削除する必要があります。

- ファイルイベント受付による自動削除

  共通フォルダ内の取得対象ファイルの削除については，「9.1.2　監視対象のファイルの状態管理機能」を参照してください。

付録

付録 A　TP1 アダプタと TP1/Client/J の関係

TP1 アダプタは，TP1/Client/J を使用して，OpenTP1 またはメインフレーム（TMS-4V や DCCM3 など）上で動作するアプリケーションを呼び出します。ここでは，TP1 アダプタで利用する TP1/Client/J について説明します。

付録 A.1　TP1/Client/J の機能

TP1 アダプタでは，次に示す TP1/Client/J の機能を使用できます。

表 A-1　TP1 アダプタで使用できる TP1/Client/J の機能

項番	TP1/Client/J の機能	TP1 アダプタの対応可否
1	常設コネクション	○
2	リモートプロシジャコール	△
3	トランザクション制御	×
4	TCP/IP 通信機能	△
5	サーバからの一方通知受信機能	×
6	TP1/Web 接続機能	×
7	動的定義変更機能	×
8	TCP/IP コネクションの確立の監視機能	○
9	DCCM3 との接続機能	○
10	XA リソースサービス機能	×
11	トラブルシュート機能	○
12	データ圧縮機能	×
13	送信元ホスト指定機能	×
14	受信ポート固定機能	×
15	ホスト切り替え機能	○
16	サービス要求のスケジュールプライオリティ設定機能	×

（凡例）

○：対応しています。

△：一部対応しています。

×：対応していません。

以降で，TP1/Client/J の機能について説明します。

(1) 常設コネクション

TP1/Client/J では，CUP（TP1/Client/J を使用した Java アプレット，Java アプリケーション，または Java サーブレット）と rap サーバの間のコネクションを確立したままでメッセージを送受信できます。このようなコネクションを**常設コネクション**といいます。なお，リモート API 機能を使用した RPC を実行するとき，連鎖型 RPC を使用するとき，およびトランザクション機能を使用するときは，必ず常設コネクションを確立してください。

常設コネクションを確立すると，コネクション確立・解放のための制御用パケットを減らせるため，通信の効率が向上します。

(a) 常設コネクションの確立・解放

TP1/Client/J が同時に確立できる常設コネクションは，1CUP で 1 コネクションだけです。TP1/Client/J の常設コネクションには，管理方式によって次のモードがあります。

非オートコネクトモード

非オートコネクトモードは，CUP で明示的に openConnection メソッドを呼び出して TP1/Server の rap リスナー，rap サーバとのコネクションを確立します。コネクションを解放するには，CUP で closeConnection メソッドを明示的に呼び出します。

オートコネクトモード

オートコネクトモードは，TP1/Client/J でコネクションを管理します。TP1/Client/J は，最初に呼び出されたときにコネクションが確立されていないと判断すると，自動的に rap リスナー，rap サーバとの間に常設コネクションを確立します。TP1/Client/J で管理しているコネクションを強制的に解放したい場合は，closeConnection メソッドを呼び出します。

非オートコネクトモードを使用するか，オートコネクトモードを使用するかは，TP1/Client/J 環境定義の dcrapautoconnect オペランドで指定するか，または setRpcextend メソッドで指定します。

closeConnection メソッドを呼び出して常設コネクションを解放するまで，メッセージの送受信には常設コネクションが使用されます。ただし，何らかの障害が発生した場合，常設コネクションは解放されます。

(2) リモートプロシジャコール

CUP から SPP へのリモートプロシジャコール（RPC）について説明します。

RPC の詳細については，マニュアル「OpenTP1 プログラム作成の手引」を参照してください。

(a) RPC の形態

TP1/Client/J で実行できる RPC の形態は，同期応答型 RPC，非応答型 RPC，および連鎖型 RPC です。

●同期応答型 RPC

CUP から SPP へ問い合わせメッセージを送信し，応答メッセージを受信する形態です。CUP は，SPP から処理結果が返ってくるまで，次の処理を待ちます。同一サービスを複数回呼び出した場合，サーバプロセスはそのたびにスケジュールされます。

●非応答型 RPC

CUP から SPP へ問い合わせメッセージを送信し，応答メッセージを受信しない形態です。CUP は，SPP へ問い合わせメッセージを送信したあと，SPP から処理結果を受け取らないで，すぐに次の処理を実行します。

非応答型 RPC では，何らかの通信障害が発生した場合や呼び出したサービスが誤っていた場合でもエラーを受け取れません。

●連鎖型 RPC

TP1/Client/J のリモート API 機能を使用した CUP から SPP へ問い合わせメッセージを送信し，問い合わせた応答メッセージを受信する形態です。CUP は，SPP から処理結果が返ってくるまで，次の処理を待ちます。同期応答型 RPC との違いは，サーバプロセスを固定できることです。連鎖型 RPC の場合，同一サービスに複数回問い合わせメッセージを送信しても，サーバプロセスは再スケジュールされません。なお，TP1 アダプタでは，この機能を使用できません。

(b) スケジュール機能

TP1/Server のスケジュール機能は，CUP から SPP に対するサービス要求でも有効です。TP1/Server は，SPP のサービスグループごとにスケジュールキューを作成し，サービス要求をスケジュールします。

(c) ノード間負荷バランス機能

OpenTP1 では，RPC による要求が特定のノードに集中しないようにノード間で負荷を分散する機能があります。これを**ノード間負荷バランス機能**といいます。

ノード間負荷バランス機能を使用するためには，負荷分散の前提として次の条件を満たしている必要があります。

- 複数のノードに同一のサービスを提供するユーザサーバが起動されていること。
- 各 OpenTP1 ノードはシステム共通定義の all_node オペランドに自分以外のノードを定義して，お互いの OpenTP1 ノードで起動されているユーザサーバの情報（ネーム情報）をやり取りしていること。

ここでは，OpenTP1 のノード間負荷バランス機能を使用する場合の TP1/Client/J 側，TP1/Server 側の関連する定義，処理，および RPC の処理を説明します。なお，TP1/Client/J では TP1/Server 側の判断による負荷分散機能だけを提供します。

●TP1/Server 側の判断で負荷分散を行う場合

TP1/Server のスケジュールサービスが，ノードのスケジュール状態に応じて，より効率的に処理できるノードへ負荷を分散させます。

TP1/Client/J 側の定義

TP1/Client/J 環境定義に dcscddirect=Y を指定します。

これによって，TP1/Client/J 側は TP1/Server のスケジュールサービスに負荷分散を依頼できます。TP1/Client/J 側の定義では，どの OpenTP1 ノードのスケジュールサービスに判断を依頼するかを指定します。

この場合，スケジュールを依頼する OpenTP1 ノードが複数あると，dchost オペランドに指定された順番にスケジュールを依頼します。dchost オペランドに指定された順番ではなく，スケジュールを依頼する TP1/Server をランダムに選択するには，TP1/Client/J 環境定義に dchostselect=Y を指定します。

TP1/Server 側の定義

TP1/Server 側の定義では，次のどちらかの設定をする必要があります。

- スケジュールサービス定義のオペランドに次の指定をする。
  scd_this_node_first ＝ N （デフォルト）
  scd_announce_server_status ＝ Y （デフォルト）
- スケジュールサービス定義を省略する。

(d) 時間監視

同期応答型 RPC を行う場合，応答メッセージを受信するまでの時間を監視できます。

監視時間は，TP1/Client/J 環境定義の dcwatchtim オペランドで指定します。

(e) リモート API 機能を使用した RPC

TP1/Client/J では，CUP と TP1/Server 間に常設コネクションを確立して，CUP が発行した API を，TP1/Server 側に転送して TP1/Server 側のプロセスで実行できます。このような機能を**リモート API 機能**といいます。リモート API 機能を使うと，ファイアウォールの内側にある UAP に対しても，サービスを要求できます。なお，リモート API 機能を使用した RPC は，TP1/Server のほかに，DCCM3 および TMS-4V/SP を使用できます。

(f) ネームサービスを使用した RPC

ネームサービスを使用した RPC は，Java アプリケーションおよび Java サーブレットから実行できるサービス要求です。Java アプレットのセキュリティの制約があるため，Java アプレットからは実行できません。

●ネームサービスを使用した RPC の機能

ネームサービスを使用した RPC の使用時は，ソケット受信型 SPP に対して RPC を発行できません。また，TP1/Client/J 環境定義の dccltrpcmaxmsgsize オペランドに 2 以上を指定した場合，ネームサービスを使用した RPC は，システム共通定義の rpc_max_message_size オペランドをサポートする TP1/Server 上で稼働しているサービスの情報だけを取得します。サービスの入力パラメタ長が，サービス要求先のシステム共通定義の rpc_max_message_size オペランドの値を超える場合もサービス要求は実行されます。システム共通定義の rpc_max_message_size オペランドをサポートしない TP1/Server 上の SPP に対しては RPC を発行できません。

●RPC ごとにサービス要求先スケジューラを分散させる場合の定義

ネームサービスを使用した RPC では，サービス要求先スケジューラの情報をネームサーバからキャッシュに格納し，キャッシュの情報を参照して RPC ごとにサービス要求先スケジューラを分散させることができます。RPC ごとにサービス要求先スケジューラを分散させる場合の TP1/Client/J 側，TP1/Server 側の関連する定義について説明します。

●TP1/Client/J 側の定義

RPC ごとにサービス要求先スケジューラを分散させる場合は，TP1/Client/J 環境定義に dccltloadbalance=Y を指定します。

これによって，TP1/Client/J は，ネームサーバから複数のサービス要求先スケジューラの情報を取得し，負荷レベルが最も低いサービス要求先スケジューラの情報をキャッシュに格納します。キャッシュに格納されるサービス要求先スケジューラは，1 つだけではなく，複数の場合もあります。

キャッシュに格納されたサービス要求先スケジューラが複数ある場合，最初のサービス要求先スケジューラは，ランダムに選択されます。RPC でのサービス要求が 2 度目以降の場合は，サービス要求先スケジューラの情報を取得するためにキャッシュを参照し，RPC ごとにラウンドロビン方式でサービス要求先スケジューラを切り替え，分散させます。

- キャッシュの有効期限の指定

  TP1/Client/J 環境定義の dccltcachetim オペランドで，サービス要求先スケジューラの情報を保持する，キャッシュの有効期限を指定できます。

  キャッシュにサービス要求先スケジューラの情報を格納したあとで，キャッシュの有効期限を満了した場合，そのサービス要求先スケジューラの情報を破棄します。その後，サービス要求先スケジューラの情報を再度取得して，その情報をキャッシュに格納することで，キャッシュを更新します。

  TP1/Client/J 環境定義の dccltcachetim オペランドは，TP1/Client/J 環境定義に dccltloadbalance=Y を指定した場合だけ有効になります。

●TP1/Server 側の定義

次に示す TP1/Server 側の定義のオペランドを指定すると，ネームサーバからサービス要求先スケジューラの情報を取得するときに，スケジューラの負荷情報も同時に取得します。

- スケジュールサービス定義

  scd_announce_server_status=Y（デフォルト）

  このオペランドに N を指定した場合，スケジューラの負荷レベルは常に LEVEL0（負荷が低い状態）となります。負荷レベルが常に LEVEL0 のスケジューラが，TP1/Client/J のサービス要求先スケジューラとなった場合，実際の負荷状態に関係なく，常にサービス要求先スケジューラとして選択されます。

- ユーザサービス定義，またはユーザサービスデフォルト定義

  loadcheck_interval

  levelup_queue_count

  leveldown_queue_count

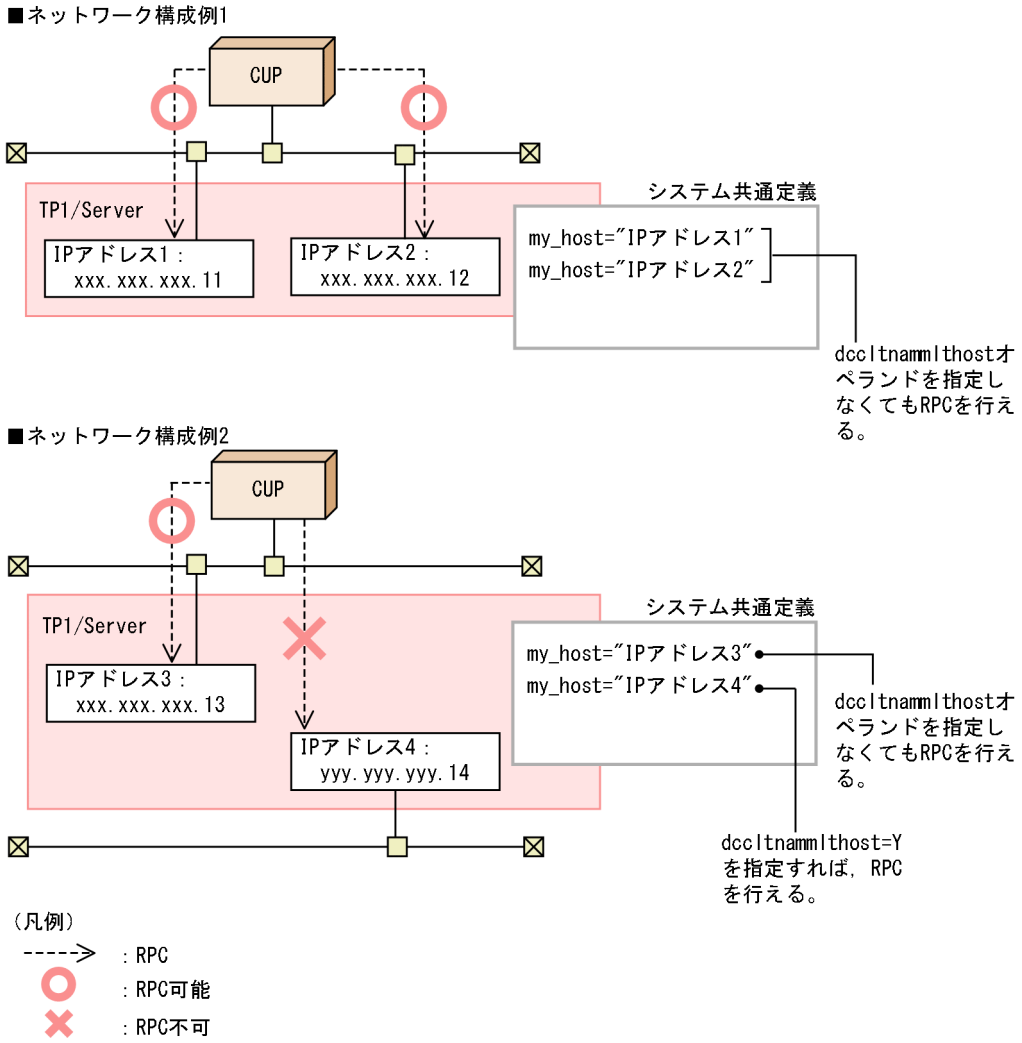
「levelup_queue_count」および「leveldown_queue_count」については，スケジューラの負荷レベルがサービス要求の滞留数で決まる方式とする場合に指定します。

●マルチホームドホスト形態の TP1/Server に対して RPC を行う場合の定義

ネームサービスを使用した RPC では，通信先 TP1/Server がマルチホームドホスト形態の場合に通信障害が発生することがあります。これは，CUP が存在するネットワークから，通信先 TP1/Server のシステム共通定義の my_host オペランドで指定したホスト名（IP アドレス）に対応するネットワークアダプタに通信できないことが原因です。

これを回避するためには，TP1/Client/J 環境定義に dccltnammlthost=Y を指定します。ネットワークの構成例を次の図に 2 つ示し，それぞれのネットワーク構成での TP1/Client/J 環境定義の dccltnammlthost オペランドの要否を示します。

図 A-1　マルチホームドホスト形態の TP1/Server に対して RPC を行う場合の例



ネットワーク構成例 2 の CUP が IP アドレス 4 と通信するには，TP1/Client/J 環境定義に dccltnammlthost=Y の指定が必要です。この指定がない場合，CUP と IP アドレス 4 との間では通信障害が発生します。

通信先 TP1/Server がマルチホームドホスト形態で，かつ同じ TP1/Server 上にサービス要求先の SPP が存在しているときは，TP1/Client/J 環境定義に dccltnammlthost=Y を指定することで RPC が行えるようになります。通信先 TP1/Server のシステム共通定義の all_node オペランドに，CUP が接

続されていないネットワーク上に存在する TP1/Server が指定されている場合，この TP1/Server 上の SPP に対して RPC を行うと通信障害が発生します。なお，通信先 TP1/Server がマルチホームドホスト形態でない場合，TP1/Client/J 環境定義の dccltnammlthost オペランドの指定に関係なく RPC を行えます。

(g) 通信先を指定した RPC

通信先を指定した RPC を使用すると，サービス要求を実行するサーバ（通信先の OpenTP1 ノード）を明示的に指定できます。

通信先を指定した RPC を使用した場合，サービス要求を実行するサーバに指定した通信先の OpenTP1 ノードでサービス要求が実行されます。指定した OpenTP1 ノードが停止，または閉塞しているときでも，他 OpenTP1 ノードへの転送は行われません。また，ノード間負荷バランス機能は使用できません。

(h) 同期応答型 RPC タイムアウト時のサーバ負荷軽減

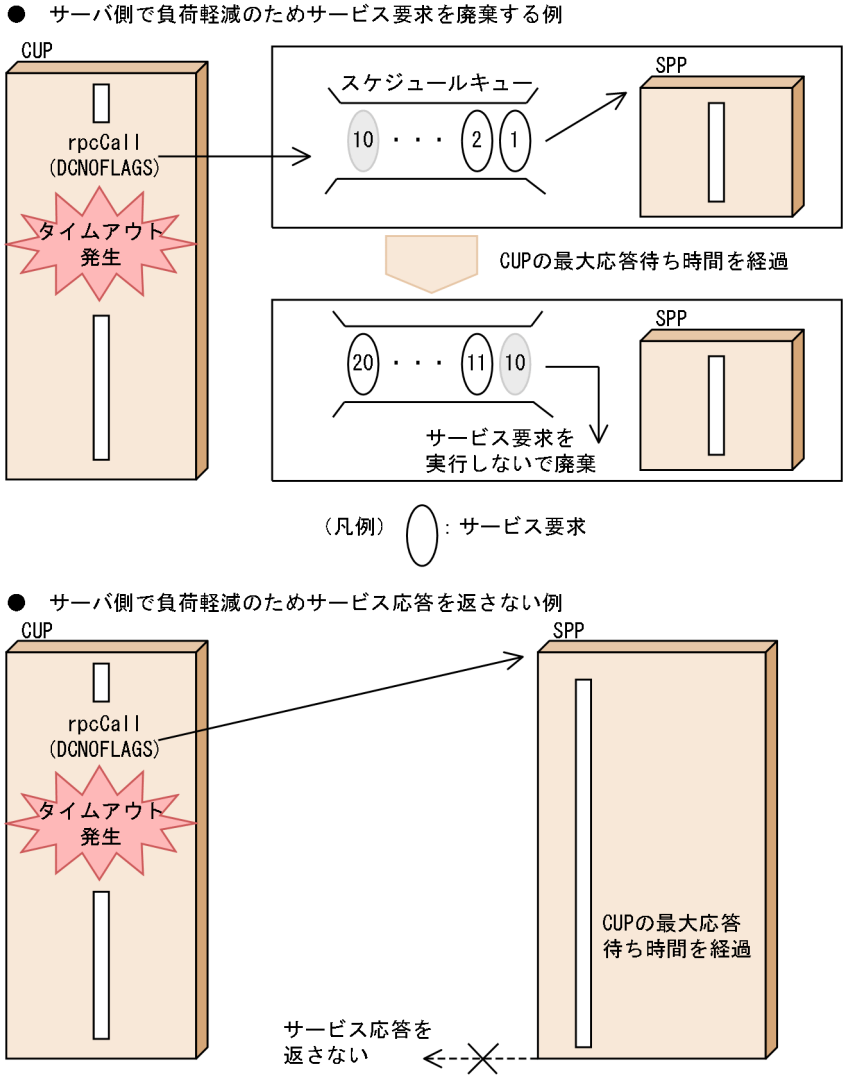
TP1/Client/J の CUP からサーバを呼び出すと，TP1/Server ではサービス要求を受け付けます。

この要求は SPP の実行待ち時間，実行時間，通信障害などによって，遅れるおそれがあるため，TP1/Client/J 側では応答待ち時間に上限を設けることで異常を監視しています。

一方 TP1/Server 側では，TP1/Client/J の最大応答待ち時間を認識していないため，TP1/Client/J 側でタイムアウトを検出していても，TP1/Server 側ではサービスを処理し続けている場合があります。

同期応答型 RPC タイムアウト時のサーバ負荷軽減機能を使用すると，TP1/Server 側での上記のような不要な処理を軽減できます。同期応答型 RPC タイムアウト時のサーバ負荷軽減機能を使用するかどうかは，TP1/Client/J 環境定義の dcwatchtimrpcinherit オペランドで指定します。同期応答型 RPC タイムアウト時のサーバ負荷軽減機能の処理概要を次の図に示します。

図 A-2　同期応答型 RPC タイムアウト時のサーバ負荷軽減機能の処理概要



(i)　ServerSocket を使用する RPC

TP1/Client/J で，スケジューラダイレクト機能を使用した RPC，ネームサービスを使用した RPC，および通信先を指定した RPC を行う場合，ServerSocket を使用します。TP1/Client/J を使用するクライアントプログラムを動作させる Java VM でセキュリティマネジャを適用する場合，TP1/Client/J のクラスライブラリが ServerSocket の機能（connect，listen，および accept）を使用できるよう，適切なパーミッションをセキュリティポリシファイルに記述してください。Java サーブレット，または EJB の場合，これらコンテナを実装しているアプリケーションサーバ製品で，セキュリティの制約のためデフォルトでセキュリティマネジャを使用していることがあります。使用するアプリケーションサーバの仕様をご確認の上，適切に設定してください。

(j)　マルチスケジューラ機能を使用した RPC

CUP からスケジュールキューを使う SPP（キュー受信型サーバ）にサービスを要求した場合，要求先 SPP があるノードのスケジューラデーモンが，いったんサービス要求メッセージを受信し，該当する SPP のスケジュールキューに格納します。スケジューラデーモンとは，スケジュールサービスを提供するシステムデーモンのことです。

長大なサービス要求メッセージは，一定の長さに分割してスケジューラデーモンに送信します。スケジューラデーモンは，サービス要求メッセージを組み立ててキュー受信型サーバのスケジュールキューに格納します。スケジューラデーモンは，OpenTP1 システムごとに 1 プロセスです。そのため，分割されたサービス要求メッセージの受信処理が完了するまで，スケジューラデーモンはほかのサービス要求メッセージを受信できません。通信速度が遅い回線を使用して，長大なサービス要求メッセージを送信した場合，ほかのサービス要求のスケジューリングが遅延することがあります。また，システムの大規模化，マシンやネットワークの高性能化などに伴って，効率良くスケジューリングできないことがあります。この場合，従来のスケジューラデーモンとは別に，サービス要求受信専用デーモンを複数プロセス起動し，サービス要求メッセージ受信処理を並行動作させることによって，スケジューリング遅延を回避できます。この機能を**マルチスケジューラ機能**といいます。以降，従来のスケジューラデーモンを**マスタスケジューラデーモン**，サービス要求受信専用デーモンを**マルチスケジューラデーモン**と呼びます。

マルチスケジューラ機能の検討が必要なシステム構成については，マニュアル「OpenTP1 プログラム作成の手引」を参照してください。

●マルチスケジューラデーモンをランダムに選択する方法

マルチスケジューラ機能を使用することで，複数起動されているマルチスケジューラデーモンの中から，利用できるマルチスケジューラデーモンをランダムに選択してサービス要求を送信できます。マルチスケジューラデーモンをランダムに選択して，スケジューラダイレクト機能，またはネームサービスを使用した RPC を実行できます。

●スケジューラダイレクト機能を使用した RPC

TP1/Client/J 環境定義 dcscddirect オペランドに Y を指定して，スケジューラダイレクト機能を使用した RPC を行う場合について説明します。

TP1/Client/J では，窓口となる TP1/Server のネームサービスへ問い合わせないで，マルチスケジューラデーモンをランダムに選択できます。これによって通信回数が削減され，ネームサービスの負荷を軽減できます。

マルチスケジューラデーモンをランダムに選択するためには，TP1/Client/J 環境定義の dchost オペランド，または dcscdport オペランドに次のポート番号を指定します。

- スケジュールサービス定義の scd_port オペランドに指定された，スケジュールサービスのポート番号
- スケジュールサービス定義の scdmulti の-p オプションに指定されたポート番号

また，あらかじめ TP1/Server で起動しているマルチスケジューラデーモンのプロセス数を，TP1/Client/J 環境定義の dcscdmulticount オペランドに指定しておく必要があります。サービス要求を送信するマルチスケジューラデーモンのポート番号は，次に示す範囲の値からランダムに選択されます。

- 下限値：TP1/Client/J 環境定義の dchost オペランド，または dcscdport オペランドに指定したポート番号の値
- 上限値：下限値 + TP1/Client/J 環境定義の dcscdmulticount オペランドに指定したプロセス数 - 1

ただし，TP1/Client/J 環境定義の dchost オペランドで指定した窓口となる TP1/Server 間で，スケジュールサービス定義の scdmulti で指定した値を統一する必要があります。

●ネームサービスを使用した RPC

マルチスケジューラ機能を使用して，ネームサービスを使用した RPC を行う場合について説明します。サービス情報を一時的に格納する領域に，該当するサービス情報がないときはネームサービスにサービス情報を問い合わせます。サービス情報を基にマルチスケジューラデーモンをランダムに選択してサービス要求を送信します。

●TP1/Client/J 環境定義とサービス要求を送信するスケジューラデーモンの関連

マルチスケジューラ機能を使用した場合，サービス要求を送信するスケジューラデーモンは TP1/Client/J 環境定義の指定によって異なります。

スケジューラダイレクト機能を使用した RPC の場合の，TP1/Client/J 環境定義のオペランドの指定とスケジューラデーモンの関連を次の表に示します。

表 A–2　TP1/Client/J 環境定義のオペランドの指定とスケジューラデーモンの関連（スケジューラダイレクト機能を使用した RPC）

TP1/Client/J 環境定義のオペランドの指定			サービス要求を送信するスケジューラデーモン
dcscddirect	dcscdmulti	dcscdmulticount	
Y	Y	○	ランダムに選択したマルチスケジューラデーモン※
		－	TP1/Client/J 環境定義の dchost オペランド，または dcscdport オペランドに指定したポート番号で起動されているスケジューラデーモン
	N	無効	TP1/Client/J 環境定義の dchost オペランド，または dcscdport オペランドに指定したポート番号で起動されているスケジューラデーモン

（凡例）

Y：オペランドの指定値が Y です。

N：オペランドの指定値が N です。

○：オペランドに値を指定します。

－：オペランドに値を指定しません。

注※

マルチスケジューラデーモンのポート番号は，次に示す範囲の値から選択されます。

下限値：TP1/Client/J 環境定義の dchost オペランド，または dcscdport オペランドに指定したポート番号の値

上限値：下限値 ＋ TP1/Client/J 環境定義の dcscdmulticount オペランドに指定したプロセス数 - 1

ネームサービスを使用した RPC の場合の，TP1/Client/J 環境定義のオペランドの指定とスケジューラデーモンの関連を次の表に示します。

表 A-3　TP1/Client/J 環境定義のオペランドの指定とスケジューラデーモンの関連（ネームサービスを使用した RPC）

TP1/Client/J 環境定義のオペランドの指定			サービス要求を送信するスケジューラデーモン
dcnamuse	dcscdmulti	dcscdmulticount	
Y	Y	無効	サービス情報を基にランダムに選択したマルチスケジューラデーモン※
	N		マスタスケジューラデーモン※

（凡例）

Y：オペランドの指定値が Y です。

N：オペランドの指定値が N です。

注※

ネームサービスへの問い合わせが発生します。

(3)　トランザクション制御

この機能は，リモート API 機能を使用する場合で，要求先 TP1/Server Base のバージョンが 07-50 以降のときだけ使用できます。

TP1 アダプタでは，この機能を使用できません。

(4)　TCP/IP 通信機能

TP1/Client/J の TCP/IP 通信機能は，CUP がクライアント型かサーバ型かによって通信方法が異なります。CUP がクライアント型の場合は，CUP から定義や引数で指定した通信相手に対してコネクションを確立して通信します。CUP がサーバ型の場合は，定義で指定したポートで通信相手がコネクションを確立してくるのを待って通信します。コネクション確立後は，引数で指定した領域に対してデータの送受信を行います。

(5)　サーバからの一方通知受信機能

一方通知受信機能を使用すると，オンライン開始の合図をクライアント側に一斉配布する，従来，メインフレームの OLTP での端末一斉起動と同様な運用ができるようになります。

TP1 アダプタでは，この機能を使用できません。

(6)　TP1/Web 接続機能

TP1/Web 接続機能は，Java アプレットとして作成した CUP と TP1/Web を接続して，HTTP プロトコルに基づいた通信サービスをするための機能です。

TP1 アダプタでは，この機能を使用できません。

(7)　動的定義変更機能

TP1/Client/J は，TP1/Client/J 環境定義を rpcOpen メソッド内で解析し，CUP に一意の情報として格納しています。

動的定義変更機能を使用すると，TP1/Client/J 環境定義で定義しなかったオペランドの値を設定したり，TP1/Client/J 環境定義で定義したオペランドの値を動的に変更したりできます。

(8)　TCP/IP コネクションの確立の監視機能

TCP/IP コネクションの確立の監視機能では，データ送信時の TCP/IP コネクションの確立処理に対する最大監視時間を指定し，コネクションの確立処理を監視できます。

(9)　DCCM3 との接続機能

TP1/Client/J では，RPC や TCP/IP 通信機能を使用して，VOS3 や VOS1 上の DCCM3 と接続できます。

(a)　DCCM3 との RPC

TP1/Client/J では，OpenTP1 のサーバだけでなく，DCCM3 のサーバとも RPC を使用した通信ができます。DCCM3 と RPC を行うには，DCCM3 側に，OpenTP1 の RPC 要求を解釈する機能が組み込まれていることが前提です。次に示す製品を DCCM3 側に組み込むことで，DCCM3 とも RPC を使用した通信ができます。

DCCM3 側の適用 OS が VOS3 の場合

DCCM3/Internet

DCCM3 側の適用 OS が VOS1 の場合

DCCM3/SERVER/TP1

相手サーバが DCCM3 の場合の特徴は次のとおりです。

- 使用できる RPC の形態は，同期応答型 RPC と非応答型 RPC です。連鎖型 RPC は使用できません。
- トランザクション制御機能は使用できません。
- 常設コネクションを使用して DCCM3 論理端末へ RPC を行う場合，負荷分散機能を使用できます。詳細については，「●DCCM3 論理端末に対して RPC を行う場合の負荷分散」を参照してください。
- DCCM3 に対して RPC を行う場合，サービス名がトランザクション名称と評価されます。

●相手サーバの指定方法

DCCM3 のサーバと RPC を使用した通信をする場合，相手サーバをサービスグループ名とサービス名で指定します。この指定方法は OpenTP1 のサーバに対して RPC を行う場合と同じです。

- サービスグループ名

サービスグループ名として不正でないダミーの文字列を指定します。1～31 文字の識別子を指定してください。

- サービス名

  DCCM3 側のトランザクション名称を指定します。使用できる文字は，アルファベット（A～Z，a～z）と数字（0～9）です。文字数の合計が 1～8 文字になるように指定してください。

●相手サーバのアドレス定義

DCCM3 のサーバと RPC を使用した通信をする場合，OpenTP1 のネームサービスの管理外にあるサーバを呼び出すため，TP1/Client/J 側でサービス名ごとに定義を分けて，サーバのアドレスを定義する必要があります。サーバのアドレスを定義するには，TP1/Client/J 環境定義の dchost オペランドに，それぞれ RPC 受け付け窓口のホスト名およびポート番号を指定します。

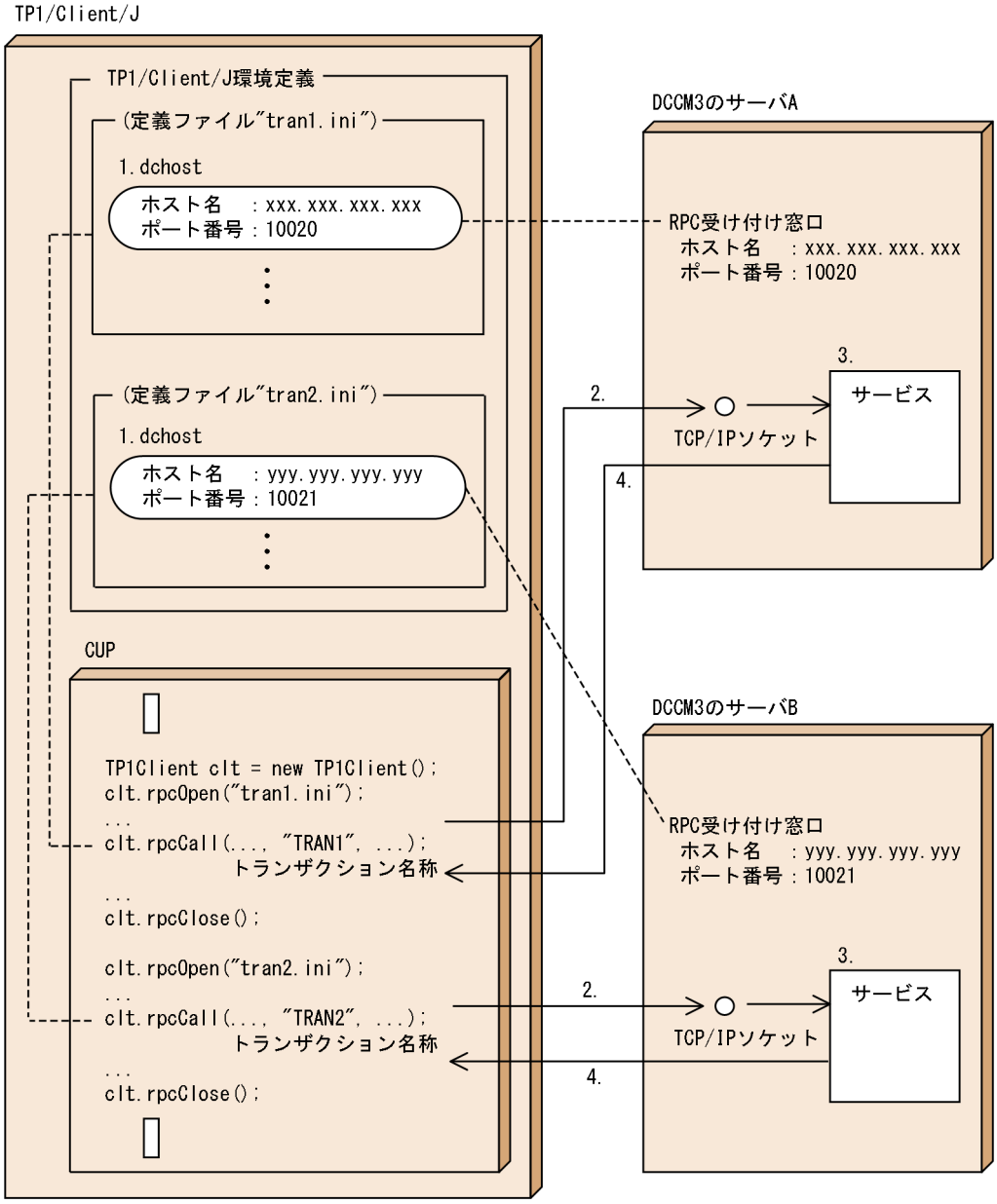
●DCCM3 論理端末に対して RPC を行う場合の負荷分散

TP1/Client/J と DCCM3 論理端末が常設コネクションを使用して RPC を行う場合，コネクション確立時に接続先を複数の DCCM3 に振り分けて負荷を分散できます。TP1/Client/J は，TP1/Client/J 環境定義の dchost オペランドに指定された複数の DCCM3 論理端末のホスト名およびポート番号の中から，接続先をランダムに選択し，接続を試みます。ある DCCM3 論理端末との接続に失敗すると，それ以外の DCCM3 論理端末から再びランダムに選択し，接続を試みます。この処理を繰り返し，TP1/Client/J 環境定義の dchost オペランドに指定されたすべての DCCM3 論理端末との接続にすべて失敗したときに，初めてエラーを検知します。

●TP1/Client/J 環境定義の定義例

次に示す図のように動作する，DCCM3 接続時の TP1/Client/J 環境定義の定義例を示します。

図 A-3　DCCM3 のサーバとの RPC



1. TP1/Client/J 環境定義で，各トランザクションを処理するサーバのアドレス（RPC 受け付け窓口）を，定義ファイルを分けて dchost オペランドに定義します。
2. RPC を行うとき，定義ファイルから RPC 受け付け窓口のアドレスを求め，RPC のメッセージを送信します。
3. RPC のメッセージを解釈し，要求されたサービスを実行します。
4. 同期応答型 RPC の場合，サーバからの応答メッセージを受信します。

定義ファイル"tran1.ini"の定義例

```
dcrapdirect=Y
dcwatchtim=180
dcrapautoconnect=Y
dchostselect=Y
```

```
#"TRAN1"のトランザクションを処理できるサーバのアドレスを定義
dchost=xxx.xxx.xxx.xxx:10020,zzz.zzz.zzz.zzz:10022
```

定義ファイル"tran2.ini"の定義例

```
dcrapdirect=Y
dcwatchtim=180
dcrapautoconnect=Y
dchostselect=Y
#"TRAN2"のトランザクションを処理できるサーバのアドレスを定義
dchost=yyy.yyy.yyy.yyy:10021,zzz.zzz.zzz.zzz:10022
```

●DCCM3 論理端末に対して RPC を行う場合の注意事項

- リモート API 機能を使用した RPC だけ使用できます。
- 連鎖型 RPC は使用できません。
- トランザクション制御機能は使用できません。
- TP1/Client/J 環境定義の dccltinquiretime オペランドを指定しても無効になります。CUP からサーバに対する問い合わせ間隔最大時間は，DCCM3 の「端末放置監視時間」で指定してください。
- DCCM3 側の注意事項については，マニュアル「VOS3 データマネジメントシステム XDM E2 系概説」およびマニュアル「VOS1 データコミュニケーションマネジメントシステム DCCM3 解説」を参照してください。

(b) DCCM3 論理端末への端末識別情報の通知

常設コネクションを使用して DCCM3 論理端末と通信する場合，端末識別情報を DCCM3 論理端末に通知し，CUP に割り当てられる DCCM3 の論理端末を固定することができます。

●DCCM3 論理端末でのメッセージ送受信方法

DCCM3 論理端末では，通信相手となる TP1/Client/J を IP アドレスと DCCM3 論理端末のポート番号で区別する論理端末として定義し，論理端末ごとにメッセージの送受信を行っています。

したがって，複数の CUP を同一マシンから起動した場合，どの CUP からの要求でも，DCCM3 論理端末から見ると IP アドレスが同じになってしまいます。複数の CUP から同じ DCCM3 論理端末のポートにサービスを要求すると，DCCM3 の定義では区別が付きません。そのため，該当する DCCM3 の論理端末が複数定義されていた場合，DCCM3 のどの論理端末に CUP が割り当てられるのかが不定になってしまいます。サービス要求を受け付ける DCCM3 の論理端末が異なると，DCCM3 側のサーバ処理の順番が保証されなくなるため，業務によっては問題となることがあります。

●端末識別情報の通知

CUP が DCCM3 論理端末と常設コネクションを確立するときに，端末識別情報を DCCM3 論理端末に通知することで，CUP に割り当てられる DCCM3 の論理端末を固定することができます。これを**端末識別情報設定機能**といいます。この機能を使用することで，CUP を常に同じ DCCM3 の論理端末に割り当てることができます。なお，DCCM3 側では，この機能を**端末固定割り当て機能**といいます。

端末識別情報設定機能を使用していない場合と使用している場合の CUP と DCCM3 論理端末の関係を，次の図にそれぞれ示します。

図 A-4　CUP と DCCM3 論理端末の関係（端末識別情報設定機能を使用していない場合）

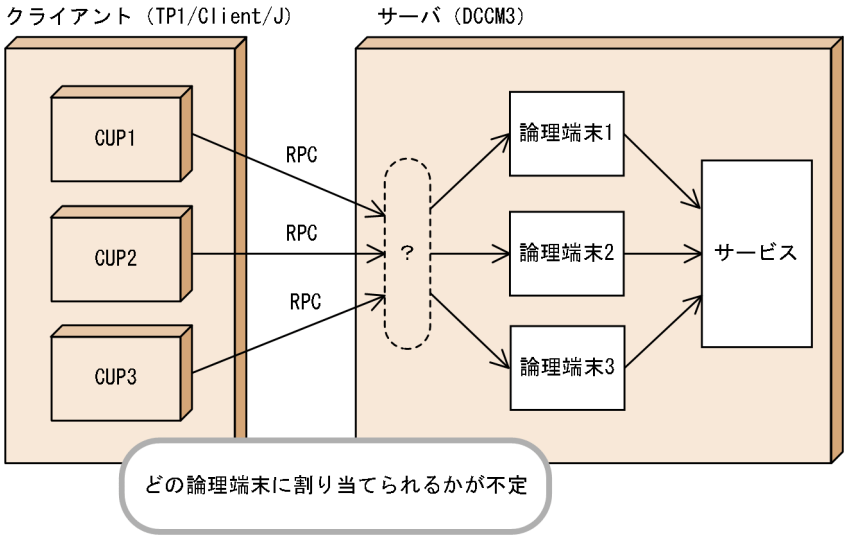
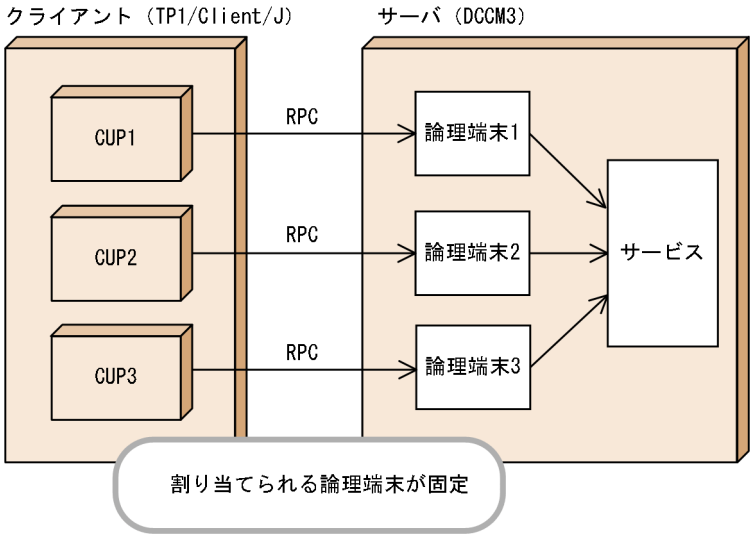


図 A-5　CUP と DCCM3 論理端末の関係（端末識別情報設定機能を使用している場合）



端末識別情報設定機能は，次の方法で使用できます。

1. TP1/Client/J 環境定義の dchost オペランドに DCCM3 論理端末のホスト名を指定します。
2. TP1/Client/J 環境定義の dchost オペランドまたは dcrapport オペランドに DCCM3 論理端末のポート番号を指定します。
3. TP1/Client/J 環境定義の dccltconnectinf オペランドに端末識別情報を設定します。
4. 次の方法で DCCM3 論理端末との常設コネクションを確立します。

   TP1/Client/J 環境定義の dcrapautoconnect オペランドに Y を指定します。

●DCCM3 論理端末に端末識別情報を通知する場合の注意事項

- 端末固定割り当て機能を使用していない DCCM3 の論理端末に対して，TP1/Client/J から端末識別情報を設定して常設コネクションの確立を要求した場合，DCCM3 は TP1/Client/J が設定した端末識別情報の設定内容を無視します。

- TP1/Client/J から端末識別情報を設定して，TP1/Server の rap サーバと常設コネクションを確立する場合，rap サーバは TP1/Client/J が設定した端末識別情報の設定内容を無視します。また，TP1/Client/J が rap サーバを介して DCCM3 へ RPC を発行する場合，TP1/Client/J で設定した端末識別情報は DCCM3 に伝達されません。
- 端末識別情報は，リモート API 機能を使用した RPC の場合だけ有効となります。ネームサービスを使用した RPC，またはスケジューラダイレクト機能を使用した RPC の場合は，端末識別情報を設定しても無視されます。

(10) XA リソースサービス機能

XA リソースサービス機能を使用する場合，TP1 Connector が必要です。

TP1 アダプタでは，この機能を使用できません。

(11) トラブルシュート機能

TP1/Client/J では，トラブルシュート機能として，UAP トレース，データトレース，エラートレース，メモリトレース，メソッドトレース，デバッグトレース，性能解析トレース，および性能検証用トレースを取得できます。

詳細については，「付録 A.3　TP1/Client/J の障害対策」を参照してください。

(12) データ圧縮機能

データ圧縮機能を使用すると，RPC によってネットワーク上に送り出されるユーザデータを圧縮できます。これによってネットワーク上に送り出されるパケット数を削減し，ネットワークの混雑を緩和できます。

データ圧縮機能を使用するかどうかは，TP1/Client/J 環境定義の dccltdatacomp オペランドで指定します。

TP1 アダプタでは，この機能を使用できません。

(13) 送信元ホスト指定機能

TP1/Client/J では，RPC や TCP/IP 通信機能でのコネクション確立要求時に，CUP の送信元ホストを指定できます。この機能を，**送信元ホスト指定機能**といいます。

CUP が動作するホスト上に，複数のネットワークアダプタが接続されている場合，CUP がこれらのコネクション確立要求時に使用する送信元ホストは，TCP/IP の制御で任意に決まります。しかし，送信元ホスト指定機能を使用すると，次のコネクション確立要求時の送信元ホストを固定できます。

- スケジューラダイレクト機能を使用した RPC
- ネームサービスを使用した RPC
- リモート API 機能を使用した RPC

- 通信先を指定した RPC
- TCP/IP 通信機能

CUP の送信元ホストは，TP1/Client/J 環境定義の dccltcupsndhost オペランドで指定します。

(14)　受信ポート固定機能

TP1/Client/J では，スケジューラダイレクト機能を使用した RPC の受信ポート，およびネームサービスを使用した RPC の受信ポートを固定できます。この機能を，**受信ポート固定機能**といいます。

この機能は，TP1/Server から TP1/Client/J への RPC の応答通信をする場合で，TP1/Server と TP1/Client/J との間に設置したファイアウォールで TP1/Client/J の受信ポートにだけ通知を許可するようにフィルタリングしたいときに使用します。

この機能を使用する場合は，TP1/Client/J 環境定義の dccltcuprcvport オペランドを指定します。

TP1 アダプタでは，この機能を使用できません。

(15)　ホスト切り替え機能

サーバダウンなどの理由によって，指定したホストとの通信が一時的に不可となった場合，通信先ホストを 1 つしか指定していないと，TP1/Client/J のメソッドはエラーリターンし，通信先ホストの指定を変更したあと，CUP を再起動する必要があります。

通信先ホストを複数指定しておくと，TP1/Client/J はホストを切り替えて，別のホストに通信を試みます。これによって，ユーザは通信先ホストの切り替えを意識することなく業務を継続することができます。

なお，障害の内容によっては，ホストを切り替えない場合があります。

(16)　サービス要求のスケジュールプライオリティ設定機能

CUP からのサービス要求に，優先順位（スケジュールプライオリティ）を付けることができます。1 つの処理から呼び出す複数のサービス要求に優先順位を付けたい場合，サービス要求ごとのプライオリティを設定できます。

TP1 アダプタでは，この機能を使用できません。

付録 A.2　TP1/Client/J の定義

TP1/Client/J 環境定義ファイルについて説明します。

(1) TP1/Client/J の定義の概要

ここでは，TP1/Client/J 環境定義ファイルのオペランド一覧，定義の規則，およびパス名の形式について説明します。

(a) TP1/Client/J 環境定義の一覧

TP1/Client/J 環境定義ファイルのオペランド一覧を，次の表に示します。

項番	オペランド	定義内容	指定値※1
1	dcnamuse	ネームサービスを使用した RPC を使用するかどうかを指定	Y ｜《N》
2	dcnamport※2	ネームサービスのポート番号	〈符号なし整数〉((5001～65535))《10000》
3	dchost※2	窓口となる TP1/Server のホスト名	〈文字列〉
4	dcwatchtim※2	最大応答待ち時間	〈符号なし整数〉((0～65535))《180》（単位：秒）
5	dccltinquiretime※2	常設コネクション問い合わせ間隔最大時間	〈符号なし整数〉((0～1048575))（単位：秒）
6	dcwatchtiminherit※2	リモート API 機能を使用した RPC を行う場合に，CUP の最大応答待ち時間を rap サーバ側に引き継ぐかどうかを指定	Y ｜《N》
7	dcwatchtimrpcinherit※2	CUP の最大応答待ち時間を TP1/Server 側に引き継ぐかどうかを指定	Y ｜《N》
8	dccltdelay※2	最大通信遅延時間	〈符号なし整数〉((0～65535))《0》（単位：秒）
9	dcselint	応答電文監視時間間隔	〈符号なし整数〉((1～65535))《100》（単位：ミリ秒）
10	dccltextend※2	TP1/Client/J の機能拡張レベルの設定を指定	00000000 ｜ 00000001
11	dccache	ネームキャッシュの最大エントリ数	〈符号なし整数〉((2～256))《8》
12	dchostselect	窓口となる TP1/Server をランダムに選択するかどうかを指定	Y ｜《N》
13	dcscddirect	TP1/Server のネームサービスにサービス情報を問い合わせないで，スケジューラダイレクト機能を使用した RPC を使用するかどうかを指定	Y ｜《N》
14	dcscdport※2	スケジュールサービスのポート番号	〈符号なし整数〉((5001～65535))
15	dcscdloadpriority※2	サービス要求を受け付けた窓口となる TP1/Server を優先して負荷分散するかどうかを指定	Y ｜《N》

項番	オペランド	定義内容	指定値※1
16	dcrapdirect	TP1/Server のネームサービスにサービス情報を問い合わせないで，直接 rap リスナーに問い合わせる機能，すなわちリモート API 機能を使用するかどうかを指定	《Y》｜N
17	dcrapport※2	rap リスナーのポート番号	〈符号なし整数〉((5001～65535))
18	dcrapautoconnect	CUP と rap サーバとの間の常設コネクションを自動的に確立させるかどうかを指定	Y｜《N》
19	dcerrtrace	CUP でエラートレースを取得するかどうかを指定	Y｜《N》
20	dcerrtracepath	エラートレースファイル作成ディレクトリ	〈文字列〉
21	dcerrtracesize	エラートレースファイルサイズ	〈符号なし整数〉((4096～1048576))《4096》（単位：バイト）
22	dcmethodtrace	CUP でメソッドトレースを取得するかどうかを指定	Y｜《N》
23	dcmethodtracepath	メソッドトレースファイル作成ディレクトリ	〈文字列〉
24	dcmethodtracesize	メソッドトレースファイルサイズ	〈符号なし整数〉((4096～1048576))《4096》（単位：バイト）
25	dcuaptrace	CUP で UAP トレースを取得するかどうかを指定	Y｜《N》
26	dcuaptracepath	UAP トレースファイル作成ディレクトリ	〈文字列〉
27	dcuaptracesize	UAP トレースファイルサイズ	〈符号なし整数〉((4096～1048576))《4096》（単位：バイト）
28	dcdatatrace	CUP でデータトレースを取得するかどうかを指定	Y｜《N》
29	dcdatatracepath	データトレースファイル作成ディレクトリ	〈文字列〉
30	dcdatatracesize	データトレースファイルサイズ	〈符号なし整数〉((4096～1048576))《4096》（単位：バイト）
31	dcdatatracemaxsize	データトレースの最大データ長	〈符号なし整数〉((16～1048576))《128》（単位：バイト）
32	dccllttrstatisitem	トランザクションブランチの統計情報を取得する項目を指定	統計情報項目〔,統計情報項目〕…

項番	オペランド	定義内容	指定値※1
33	dcclttroptiitem	複数のユーザサーバで構成されるグローバルトランザクションの性能を向上させるための最適化項目を指定	トランザクション最適化項目〔,トランザクション最適化項目〕…
34	dcclttrwatchtime	トランザクション同期点処理時の最大通信待ち時間	〈符号なし整数〉((1～65535))（単位：秒）
35	dcclttrrbinfo	トランザクションブランチがロールバックした場合に，ロールバック要因に関する情報をログに取得するかどうかを指定	no ｜ self ｜ remote ｜ all
36	dcclttrlimittime	トランザクションブランチ最大実行可能時間	〈符号なし整数〉((0～65535))（単位：秒）
37	dcclttrrbrcv	RPC 先トランザクションブランチにロールバック指示を送信したあと，ロールバック完了通知を受信するかどうかを指定	Y ｜ N
38	dcclttrrecoverytype	UAP 障害時のトランザクション同期点処理方式を指定	type1 ｜ type2 ｜ type3
39	dcclttrexptm	トランザクションブランチ限界経過時間	〈符号なし整数〉((0～65535))（単位：秒）
40	dcclttrcputm	トランザクションブランチ CPU 監視時間	〈符号なし整数〉((0～65535))（単位：秒）
41	dcclttrexpsp	トランザクションブランチの処理を監視するとき，次の処理時間も監視時間に含むかどうかを指定 • 監視対象のトランザクションブランチが，RPC 機能を使ってほかのトランザクションブランチを呼び出し，その処理が終わるのを待つ時間 • 連鎖型 RPC で呼び出されたサーバ UAP が次のサービス要求を待つ時間 • 監視対象のトランザクションブランチが，非応答型 RPC を使用してほかのトランザクションブランチを呼び出したあと，処理結果受信処理を行っている時間	Y ｜ N ｜ F
42	dcsndrcvtype	TCP/IP 通信機能の使用時に，初期化する環境を指定	DCCLT_ONEWAY_SND ｜ DCCLT_ONEWAY_RCV ｜ DCCLT_SNDRCV
43	dcrcvport	受信用の CUP のポート番号	〈符号なし整数〉((1～65535))《11000》
44	dcsndhost※2	接続する MHP が存在するノードのホスト名	〈文字列〉
45	dcsndport※2	接続する MHP のポート番号	〈符号なし整数〉((1～65535))《12000》

項番	オペランド	定義内容	指定値※1
46	dcsockopenatrcv	TCP/IP 通信機能の使用時に，1 コネクションで送受信する場合の，受信用ソケットを開設する契機（送信相手からの接続を待ち受け始める契機）を指定	Y ｜《N》
47	dcweburl	サービス要求先の TP1/Web の URL	〈パス名〉
48	dccltdbgtrcfilecount	デバッグトレースファイルの最大ファイル数	〈符号なし整数〉((0～256))《0》
49	dccltrpcmaxmsgsize	RPC 送受信電文の最大長	〈符号なし整数〉((1～8))《1》（単位：メガバイト）
50	dcscdhostchange	TP1/Client/J 環境定義の dchost オペランドに指定したサービス要求先スケジューラを RPC ごとに分散させるかどうかを指定	Y ｜《N》
51	dccltloadbalance	ネームサーバから得た複数のサービス要求先スケジューラの情報から，負荷レベルが最も低いサービス要求先スケジューラの情報をキャッシュに格納するかどうかを指定	Y ｜《N》
52	dccltcachetim	キャッシュの有効期限	〈符号なし整数〉((0～65535))《30》（単位：秒）
53	dccltconnecttimeout	コネクション確立最大監視時間	〈符号なし整数〉((0～65535))《0》（単位：秒）
54	dccltprftrace	TP1/Client/J をサービスプラットフォーム上で動作させた場合に，サービスプラットフォームの性能解析トレースを取得するかどうかを指定	《Y》｜N
55	dccltprfinfosend	TP1/Server に対して，ネームサービスを使用した RPC，およびスケジューラダイレクト機能を使用した RPC を行う場合に，OpenTP1 の性能検証用トレースに出力する情報として TP1/Client/J 内部で識別情報を付加するかどうかを指定	《Y》｜N
56	dccltnammlthost	マルチホームドホスト形態の TP1/Server に対してネームサービスを使用した RPC を行うかどうかを指定	Y ｜《N》
57	dccltdatacomp	データ圧縮機能を使用するかどうかを指定	Y ｜《N》
58	dccltconnectinf※2	端末識別情報として，DCCM3 論理端末の論理端末名称を EBCDIK コードで指定	端末識別情報
59	dcscdmulti	マルチスケジューラ機能を使用するかどうかを指定	Y ｜《N》
60	dcscdmulticount	マルチスケジューラデーモンのプロセス数	〈符号なし整数〉((1～4096))《1》
61	dccltcupsndhost	CUP の送信元ホスト	〈文字列〉

項番	オペランド	定義内容	指定値※1
62	dccltcuprcvport	サーバからのメッセージを受信する CUP のポート番号	〈符号なし整数〉((5001～65535))
63	dcclttrcmplmttm	トランザクション完了限界時間	〈符号なし整数〉((0～65535))（単位：秒）
64	dccltsendbuffsize	TCP/IP の送信バッファサイズ	〈符号なし整数〉((8192～2147483647))（単位：バイト）
65	dccltrecvbuffsize	TCP/IP の受信バッファサイズ	〈符号なし整数〉((8192～2147483647))（単位：バイト）
66	dcinquiretimecheck	常設コネクションを使用し，かつオートコネクトモードの場合に，CUP 実行プロセスで問い合わせ間隔最大時間を監視するかどうかを指定	《Y》｜N
67	dcnotifyreshost	サービス要求元が応答電文を受信するために使用する IP アドレスを通知するかどうかを指定	Y｜《N》
68	dcresponsehost	サービス要求元が応答電文を受信するために使用する IP アドレスを通知する場合のホスト名	〈文字列〉

注※1

指定値の記号については，「(b)　定義の規則」を参照してください。

注※2

TP1 アダプタの接続先動的変更を使用して，通信条件パラメタを変更できます。TP1 アダプタの接続先動的変更については，「2.7.6　TP1 アダプタの接続先動的変更」を参照してください。

(b)　定義の規則

定義の説明に使用する各種の記号を説明します。

ここで述べる文法記述記号，属性記述記号，および構文要素記号は実際の定義には記述しません。

●文法記述記号

文法の記述について説明する記号です。

文法記述記号	意味
〔　　〕	この記号で囲まれている項目は省略できることを示します。 （例）ホスト名〔:ポート番号〕 これは，「ホスト名」と指定するか，または「ホスト名:ポート番号」と指定することを示します。
｜	この記号で仕切られた項目は選択できることを示します。 （例）dcnamuse=Y｜N これは，dcnamuse=Y と指定するか，または dcnamuse=N と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。この記号の直前に示された項目を繰り返し複数個指定できます。 （例）ホスト名〔:ポート番号〕〔,ホスト名〔:ポート番号〕,…〕 ホスト名〔:ポート番号〕を続けて指定できることを示します。

●属性記述記号

ユーザ指定値の範囲などを説明する記号です。

属性記述記号	意味
～	この記号のあとにユーザ指定値の属性を示します。
《 》	ユーザ指定値の省略値を示します。
〈 〉	ユーザ指定値の構文要素記号を示します。
(( ))	ユーザ指定値の指定範囲を示します。

●構文要素記号

ユーザ指定値の内容を説明する記号です。

構文要素記号	意味
〈英字〉	アルファベット（A～Z，a～z），および_（アンダースコア）
〈英数字〉	英字と数字（0～9）
〈英字記号〉	アルファベット（A～Z，a～z），#，@，および¥
〈符号なし整数〉	数字（0～9）
〈符号なし 16 進整数〉	数字（0～9），A～F，a～f
〈記号名称〉	英字記号と数字の並び（先頭は英字記号）
〈文字列〉	任意の文字の配列
〈パス名〉	記号名称，/，および .（ピリオド） （ただし，パス名は使用する OS に依存）

(c) パス名の記述形式

次のオペランドにパス名を指定する場合，ファイル区切り文字に「¥」を使用するときは，「¥」を「¥¥」と記述してください。

- dcerrtracepath
- dcmethodtracepath
- dcuaptracepath
- dcdatatracepath

例　"C:¥Clt4J¥trace"にエラートレースを取得する場合

```
dcerrtracepath = C:¥¥Clt4J¥¥trace
```

または，

```
dcerrtracepath = C:/Clt4J/trace
```

(2)　TP1/Client/J 環境定義の詳細

Java アプリケーション，または Java サーブレットでは，TP1/Client/J 環境定義を任意のファイルに格納して rpcOpen メソッドの引数でこのファイル名を指定します。また，rpcOpen メソッドを引数なしで呼び出した場合は，システムプロパティとして定義することもできます。

Java アプレットでは，param タグで TP1/Client/J 環境定義を定義します。

TP1 アダプタから TP1/Client/J を使用する場合，通信方式によって必要な定義と省略できる定義があります。

詳細については，「サービスプラットフォーム 開発ガイド 受付・アダプタ定義編」の「3.3.6　TP1 アダプタを定義する」の「TP1/Client/J 環境定義の設定」の説明を参照してください。

TP1/Client/J 環境定義を次に示します。

(a)　形式

```
〔dcnamuse=Y|N〕
〔dcnamport=ネームサービスのポート番号〕
〔dchost=窓口となるTP1/Serverのホスト名〕
〔dcwatchtim=最大応答待ち時間〕
〔dccltinquiretime=常設コネクション問い合わせ間隔最大時間〕
〔dcwatchtiminherit=Y|N〕
〔dcwatchtimrpcinherit=Y|N〕
〔dccltdelay=最大通信遅延時間〕
〔dcselint=応答電文監視時間間隔〕
〔dccltextend=機能拡張レベル〕
〔dccache=ネームキャッシュの最大エントリ数〕
〔dchostselect=Y|N〕
〔dcscddirect=Y|N〕
〔dcscdport=スケジュールサービスのポート番号〕
〔dcscdloadpriority=Y|N〕
〔dcrapdirect=Y|N〕
〔dcrapport=rapリスナーのポート番号〕
〔dcrapautoconnect=Y|N〕
〔dcerrtrace=Y|N〕
〔dcerrtracepath=エラートレースファイル作成ディレクトリ〕
〔dcerrtracesize=エラートレースファイルサイズ〕
〔dcmethodtrace=Y|N〕
〔dcmethodtracepath=メソッドトレースファイル作成ディレクトリ〕
〔dcmethodtracesize=メソッドトレースファイルサイズ〕
〔dcuaptrace=Y|N〕
〔dcuaptracepath=UAPトレースファイル作成ディレクトリ〕
〔dcuaptracesize=UAPトレースファイルサイズ〕
〔dcdatatrace=Y|N〕
〔dcdatatracepath=データトレースファイル作成ディレクトリ〕
〔dcdatatracesize=データトレースファイルサイズ〕
〔dcdatatracemaxsize=データトレースの最大データ長〕
〔dcclttrstatisitem=統計情報項目〕
〔dcclttroptiitem=トランザクション最適化項目〕
〔dcclttrwatchtime=トランザクション同期点処理時の最大通信待ち時間〕
〔dcclttrrbinfo=no|self|remote|all〕
〔dcclttrlimittime=トランザクションブランチ最大実行可能時間〕
```

```
〔dcclttrrbrcv=Y|N〕
〔dcclttrrecoverytype=type1|type2|type3〕
〔dcclttrexptm=トランザクションブランチ限界経過時間〕
〔dcclttrcputm=トランザクションブランチCPU監視時間〕
〔dcclttrexpsp=Y|N|F〕
〔dcsndrcvtype=DCCLT_ONEWAY_SND｜DCCLT_ONEWAY_RCV｜DCCLT_SNDRCV〕
〔dcrcvport=受信用のCUPのポート番号〕
〔dcsndhost=接続するMHPが存在するノードのホスト名〕
〔dcsndport=接続するMHPのポート番号〕
〔dcsockopenatrcv=Y|N〕
〔dcweburl=サービス要求先のTP1/WebのURL〕
〔dccltdbgtrcfilecount=デバッグトレースファイルの最大ファイル数〕
〔dccltrpcmaxmsgsize=RPC送受信電文の最大長〕
〔dcscdhostchange=Y|N〕
〔dccltloadbalance=Y|N〕
〔dccltcachetim=キャッシュの有効期限〕
〔dccltconnecttimeout=コネクション確立最大監視時間〕
〔dccltprftrace=Y|N〕
〔dccltprfinfosend=Y|N〕
〔dccltnammlthost=Y|N〕
〔dccltdatacomp=Y|N〕
〔dccltconnectinf=端末識別情報〕
〔dcscdmulti=Y|N〕
〔dcscdmulticount=マルチスケジューラデーモンのプロセス数〕
〔dccltcupsndhost=送信元ホスト〕
〔dccltcuprcvport=CUPの受信で使用するポート番号〕
〔dcclttrcmplmttm=トランザクション完了限界時間〕
〔dccltsendbuffsize=TCP/IPの送信バッファサイズ〕
〔dccltrecvbuffsize=TCP/IPの受信バッファサイズ〕
〔dcinquiretimecheck=Y|N〕
〔dcnotifyreshost=Y|N〕
〔dcresponsehost=ホスト名〕
```

(b) オペランド

dcnamuse=Y ｜ N ～《N》

ネームサービスを使用した RPC を使用するかどうかを指定します。

Y

ネームサービスを使用した RPC を使用します。

N

ネームサービスを使用した RPC を使用しません。

dcnamuse，dcrapdirect，および dcscddirect オペランドは排他関係にあるため，同時に Y を指定できません。これらのオペランドの 2 つ以上に Y を指定した場合，定義エラーとなります。

TP1/Web 接続機能は，dcrapdirect オペランドに Y を指定した場合にだけ使用できます。dcweburl オペランドを定義して，かつ dcscddirect=Y および dcrapdirect=Y の場合は，定義エラーとなります。

dcnamport=ネームサービスのポート番号

～〈符号なし整数〉((5001～65535))《10000》

クライアントが属している OpenTP1 のシステム共通定義の name_port オペランドで指定したネームサービスのポート番号を指定します。dchost オペランドでポート番号を指定していないホストに対して，この定義の値が有効になります。

dchost=窓口となる TP1/Server のホスト名　～〈文字列〉

窓口となる TP1/Server のホスト名と，rap リスナーのポート番号，スケジュールサービスのポート番号，またはネームサービスのポート番号を指定します。区切り文字「,」を使用して，複数の TP1/Server を指定できます。RPC を rap リスナー経由で行う場合は，rap リスナーのホスト名，またはファイアウォールのホスト名を指定します。

dchost オペランドと dcweburl オペランドの両方を指定した場合は，dcweburl オペランドの指定が有効になります。

形式

dcrapdirect=Y の場合

ホスト名〔:rap リスナーのポート番号〕〔,ホスト名〔:rap リスナーのポート番号〕,…〕

dcscddirect=Y の場合

ホスト名〔:スケジュールサービスのポート番号〕〔,ホスト名〔:スケジュールサービスのポート番号〕,…〕

dcnamuse=Y の場合

ホスト名〔:ネームサービスのポート番号〕〔,ホスト名〔:ネームサービスのポート番号〕,…〕

- ホスト名　～〈文字列〉

  TP1/Server のホスト名を 2 つ以上指定した場合，窓口となる TP1/Server の障害を検出したとき，dchost オペランドに指定されている次の TP1/Server を参照して切り替えを試みます。ただし，RPC を rap リスナー経由で行う場合は，TP1/Client/J 環境定義で dcrapautoconnect=Y，または setDccltextend メソッドで DCRPC_RAP_AUTOCONNECT を指定した場合だけ，dchost オペランドの次の TP1/Server を参照して切り替えを試みます。

- ポート番号　～〈符号なし整数〉((5001～65535))

  ポート番号は，dcrapdirect=Y の場合，rap リスナーのポート番号を指定します。dcscddirect=Y の場合，スケジュールサービスのポート番号を指定します。dcnamuse=Y の場合，ネームサービスのポート番号を指定します。

  ポート番号を省略すると，dcrapdirect=Y の場合，dcrapport オペランドの値を仮定します。dcscddirect=Y の場合，dcscdport オペランドの値を仮定します。dcnamuse=Y の場合，dcnamport オペランドの値を仮定します。

dcwatchtim=最大応答待ち時間

～〈符号なし整数〉((0～65535))《180》(単位：秒)

CUP から TP1/Server に対しての要求を開始してから応答が返るまでの待ち時間の最大値を指定します。

指定時間を過ぎても応答が返らない場合は，CUP へエラーリターンします。0 を指定した場合は，応答を受信するまで無限に待ち続けます。

dcwatchtim オペランドの指定が適用される API（メソッド）を次に示します。

- closeConnection メソッド
- openConnection メソッド
- rpcCall メソッド（スケジューラダイレクト機能を使用した非応答型 RPC の場合を除く）
- rpcCallTo メソッド（同期応答型 RPC の場合）
- trnBegin メソッド
- trnChainedCommit メソッド
- trnChainedRollback メソッド
- trnUnchainedCommit メソッド
- trnUnchainedRollback メソッド

dccltinquiretime=常設コネクション問い合わせ間隔最大時間

～〈符号なし整数〉((0～1048575))（単位：秒）

CUP がサーバに対して問い合わせを行ってから，次の問い合わせをするまでの間隔の最大時間を指定します。常設コネクション問い合わせ間隔最大時間は rap サーバで監視するタイマです。指定時間を超えても問い合わせがない場合，rap サーバ側で強制的に常設コネクションを解放します。また，トランザクション内で常設コネクション問い合わせ間隔最大時間に達したことを検知した場合は，該当するトランザクションを強制的にロールバックします。

この定義に 0 を指定した場合は，CUP からの問い合わせを無限に待ちます。この指定を省略した場合は，rap リスナーサービス定義の rap_inquire_time オペランドの指定に従います。

dcwatchtiminherit=Y | N ～《N》

リモート API 機能を使用した RPC を行う場合に，CUP の最大応答待ち時間を rap サーバ側に引き継ぐかどうかを指定します。

Y

rap サーバ側に CUP の最大応答待ち時間を引き継ぎます。

N

rap サーバ側に CUP の最大応答待ち時間を引き継ぎません。

dcwatchtimrpcinherit=Y | N ～《N》

CUP の最大応答待ち時間を TP1/Server 側に引き継ぐかどうかを指定します。CUP の最大応答待ち時間を引き継ぐと，CUP がタイムアウトしているのに TP1/Server 側でサービスを実行することを防止できます。

Y

TP1/Server 側に CUP の最大応答待ち時間を引き継ぎます。

N

TP1/Server 側に CUP の最大応答待ち時間を引き継ぎません。

dccltdelay=最大通信遅延時間

～〈符号なし整数〉((0～65535))《0》（単位：秒）

CUP と rap サーバとの間の通信オーバヘッドを考慮し，rap サーバ側の応答監視を TP1/Client/J 側よりも早く終わらせる場合に指定します。この定義を指定すると，rap サーバ側の監視を指定した時間分だけ早く終了させ，TP1/Client/J 側の監視時間のタイムアウトによるメッセージのすれ違いを防ぎます。

この定義は，TP1/Client/J 環境定義に dcrapdirect=Y および dcwatchtiminherit=Y を指定した場合にだけ有効です。

TP1/Client/J 環境定義に dcwatchtim=0 を指定した場合または setDcwatchtim メソッドで 0 を指定した場合，および TP1/Client/J 環境定義に dcwatchtiminherit=N を指定した場合は，dccltdelay オペランドの指定は無視されます。dcwatchtim オペランドに指定した値から dccltdelay オペランドに指定した値を引いたときに，0 または負の値になった場合も，dccltdelay オペランドの指定は無視され，1 が仮定されます。

dcselint=応答電文監視時間間隔　～〈符号なし整数〉((1～65535))《100》(単位：ミリ秒)

このオペランドは，旧バージョンとの互換性のためだけに存在します。現在はこのオペランドを指定しても無視されます。

dccltextend=00000000 | 00000001

TP1/Client/J の機能拡張レベルの設定を指定します。

00000000

TP1/Client/J の機能を拡張しません。

00000001

rpcCall メソッド呼び出し時，自 CUP の IP アドレスをサービスに連絡します。呼び出したサービスで dc_rpc_get_callers_address()関数を実行し，CUP のアドレスを求める必要がある場合に指定します。

dccache=ネームキャッシュの最大エントリ数　～〈符号なし整数〉((2～256))《8》

TP1/Client/J で使用するサービス情報をキャッシュするためのキャッシュの最大エントリ数を指定します。LRU 方式で管理します。キャッシュのエントリ数が指定値を超える場合は，最も過去に呼び出されたサービス情報を削除します。

dchostselect=Y | N　～《N》

rpcOpen メソッド実行時，窓口となる TP1/Server をランダムに選択するかどうかを指定します。この定義は，dchost オペランドに窓口となる TP1/Server を複数指定した場合にだけ有効です。

Y

窓口となる TP1/Server をランダムに選択します。

N

窓口となる TP1/Server をランダムに選択しません。

窓口となる TP1/Server へ問い合わせ中に障害を検出した場合，障害を検出した TP1/Server を除いて，dchost オペランドに指定された次の TP1/Server に切り替えを試みます。詳細については，「付録 A.1(15)　ホスト切り替え機能」を参照してください。

dcscddirect=Y | N　～《N》

TP1/Server のネームサービスにサービス情報を問い合わせないで，スケジューラダイレクト機能を使用した RPC を使用するかどうかを指定します。

Y

スケジューラダイレクト機能を使用した RPC を使用します。

N

スケジューラダイレクト機能を使用した RPC を使用しません。

TP1/Client/J 環境定義の dcscdport オペランドにスケジュールサービスのポート番号を指定したときは，そのポート番号を使用して問い合わせを行います。dcrapdirect，dcscddirect，および dcnamuse オペランドは排他関係にあるため，同時に Y を指定できません。これらのオペランドの 2 つ以上に Y を指定した場合，定義エラーとなります。

TP1/Web 接続機能は，dcrapdirect オペランドに Y を指定した場合にだけ使用できます。dcweburl オペランドを定義して，かつ dcscddirect=Y および dcrapdirect=Y の場合は，定義エラーとなります。

dcscdport=スケジュールサービスのポート番号

～〈符号なし整数〉((5001～65535))

クライアントが属している OpenTP1 のスケジュールサービス定義の scd_port オペランドで指定したスケジュールサービスのポート番号を指定します。dchost オペランドでポート番号を指定していないホストに対して，この定義の値が有効になります。このオペランドは TP1/Client/J 環境定義に dcscddirect=Y を指定したときだけ有効です。詳しくは，TP1/Client/J 環境定義の dcscddirect オペランドを参照してください。

dcscdloadpriority=Y | N　～《N》

サービス要求を受け付けた窓口となる TP1/Server を優先して負荷分散するかどうか指定します。

Y

サービス要求を受け付けた窓口となる TP1/Server を優先して負荷分散します。

N

サービス要求を受け付けた窓口となる TP1/Server を優先しません。TP1/Server は，ノード間の負荷を分散します。

この定義は，スケジューラダイレクト機能を使用した RPC を行う場合（TP1/Client/J 環境定義に dcscddirect=Y を指定）にだけ，有効です。

dcrapdirect=Y | N　～《Y》

TP1/Server のネームサービスにサービス情報を問い合わせないで，直接 rap リスナーに問い合わせる機能，すなわちリモート API 機能を使用するかどうかを指定します。このオペランドの指定を有効にするには，TP1/Client/J 環境定義の dcrapport オペランドに rap リスナーのポート番号を指定します。

dcrapdirect，dcscddirect，および dcnamuse オペランドは排他関係にあるため，同時に Y を指定することはできません。これらのオペランドの 2 つ以上に Y を指定した場合，定義エラーとなります。

dcrapport=rap リスナーのポート番号　～〈符号なし整数〉((5001～65535))

クライアントが属している TP1/Server の rap リスナーのポート番号，またはファイアウォールのポート番号を指定します。dchost オペランドでポート番号を指定していないホストに対して，このオペランドの値が有効になります。

このオペランドは，TP1/Client/J 環境定義に dcrapdirect=Y を指定したときだけ有効です。詳しくは，TP1/Client/J 環境定義の dcrapdirect オペランドを参照してください。

dcrapautoconnect=Y | N　～《N》

CUP と rap サーバとの間の常設コネクションを自動的に確立させるかどうかを指定します。

Y

次回メソッド実行時に自動的に常設コネクションを確立します。常設コネクション確立要求先は，TP1/Client/J 環境定義の dchost オペランドに指定した rap リスナーです。

N

常設コネクションを自動的に確立させません。

コネクション確立モードの違いによる各メソッドの動作

非オートコネクトモードの場合およびオートコネクトモードの場合のコネクション確立処理は，次の表のようになります。

表 A-4　コネクション確立モードの違いによる各メソッドの動作

dcrapautoconnect オペランドの指定値	openConnection()メソッドを発行	openConnection(host, port)メソッドを発行	openConnection(url, flags)メソッドを発行	rpcCall()メソッドを発行
N	• dchost オペランドが指定されている場合，dchost オペランドに指定された rap サーバにコネクションを確立する。 • dcweburl オペランドが指定されている場合，dcweburl オペランドに指定された URL にコネクションを確立する。 • dchost オペランドおよび dcweburl オペランドが指定されていない場合，ErrHostUndefException が返される。	host 引数および port 引数に指定された rap サーバにコネクションを確立する。	url 引数に指定された URL にコネクションを確立する。	ErrProtoException が返される。
Y	ErrProtoException が返される。	ErrProtoException が返される。	ErrProtoException が返される。	• dchost オペランドが指定されている場合，dchost オペランドに指定された rap サーバにコネクションを確立する。

dcrapautoconnect オペランドの指定値	openConnection()メソッドを発行	openConnection(host, port)メソッドを発行	openConnection(url, flags)メソッドを発行	rpcCall()メソッドを発行
Y	ErrProtoException が返される。	ErrProtoException が返される。	ErrProtoException が返される。	• dcweburl オペランドが指定されている場合，dcweburl オペランドに指定された URL にコネクションを確立する。 • dchost オペランドおよび dcweburl オペランドが指定されていない場合，ErrHostUndefException が返される。

注

- setRpcextend メソッドによって dcrapautoconnect オペランドの値を動的に変更できます。
- rpcOpen メソッドで定義を読み込みます。
- rpcClose メソッドで読み込んだ定義情報を破棄します。
- 引数に URL を指定した openConnection メソッドを実行した場合，指定された URL は，その URL を指定した openConnection メソッドだけで採用します。このメソッドの実行後，closeConnection メソッドでコネクションを切断し，次に引数指定なしの openConnection メソッドを実行しても，前回の引数に指定した URL は採用されません。

dcerrtrace=Y | N ～《N》

CUP でエラートレースを取得するかどうかを指定します。

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。

CUP を Java アプレットとして動作させる場合，Y を指定しないでください。Y を指定した場合の動作は保証されません。

dcerrtracepath=エラートレースファイル作成ディレクトリ ～〈文字列〉

エラートレースファイルを作成するディレクトリのパス名を指定します。

このオペランドは，TP1/Client/J 環境定義に dcerrtrace=Y を指定した場合にだけ有効です。

dcerrtracesize=エラートレースファイルサイズ ～〈符号なし整数〉((4096～1048576))《4096》(単位：バイト)

エラートレースファイルのサイズを指定します。

このオペランドは，TP1/Client/J 環境定義に dcerrtrace=Y を指定した場合にだけ有効です。

dcmethodtrace=Y | N ～《N》

CUP でメソッドトレースを取得するかどうかを指定します。

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。

CUP を Java アプレットとして動作させる場合，Y を指定しないでください。Y を指定した場合の動作は保証されません。

dcmethodtracepath=メソッドトレースファイル作成ディレクトリ　～〈文字列〉

メソッドトレースファイルを作成するディレクトリのパス名を指定します。

このオペランドは，TP1/Client/J 環境定義に dcmethodtrace=Y を指定した場合にだけ有効です。

dcmethodtracesize=メソッドトレースファイルサイズ　～〈符号なし整数〉((4096～1048576))《4096》(単位：バイト)

メソッドトレースファイルのサイズを指定します。

このオペランドは，TP1/Client/J 環境定義に dcmethodtrace=Y を指定した場合にだけ有効です。

dcuaptrace=Y | N　～《N》

CUP で UAP トレースを取得するかどうかを指定します。

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。

CUP を Java アプレットとして動作させる場合，Y を指定しないでください。Y を指定した場合の動作は保証されません。

dcuaptracepath=UAP トレースファイル作成ディレクトリ　～〈文字列〉

UAP トレースファイルを作成するディレクトリのパス名を指定します。

このオペランドは，TP1/Client/J 環境定義に dcuaptrace=Y を指定した場合にだけ有効です。

dcuaptracesize=UAP トレースファイルサイズ　～〈符号なし整数〉((4096～1048576))《4096》(単位：バイト)

UAP トレースファイルのサイズを指定します。

このオペランドは，TP1/Client/J 環境定義に dcuaptrace=Y を指定した場合にだけ有効です。

dcdatatrace=Y | N　～《N》

CUP でデータトレースを取得するかどうかを指定します。

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。

CUP を Java アプレットとして動作させる場合，Y を指定しないでください。Y を指定した場合の動作は保証されません。

dcdatatracepath=データトレースファイル作成ディレクトリ　～〈文字列〉

データトレースファイルを作成するディレクトリのパス名を指定します。

このオペランドは，TP1/Client/J 環境定義に dcdatatrace=Y を指定した場合にだけ有効です。

dcdatatracesize=データトレースファイルサイズ　～〈符号なし整数〉((4096～1048576))《4096》(単位：バイト)

データトレースファイルのサイズを指定します。

このオペランドは，TP1/Client/J 環境定義に dcdatatrace=Y を指定した場合にだけ有効です。

dcdatatracemaxsize=データトレースの最大データ長　～〈符号なし整数〉((16～1048576))《128》(単位：バイト)

1 件のデータトレースのデータ長の最大値を指定します。

このオペランドは，TP1/Client/J 環境定義に dcdatatrace=Y を指定した場合にだけ有効です。

dcclttrstatisitem=統計情報項目〔,統計情報項目〕…

トランザクションブランチの統計情報を取得する項目を，次の文字列で指定します。CUP からトランザクションを開始する場合にだけ有効です。

nothing

統計情報を取得しません。

base

基本情報として，次の情報を取得します。

- トランザクションブランチの識別子
- トランザクションブランチの決着結果
- トランザクションブランチの実行プロセス種別
- トランザクションブランチの実行サーバ名
- トランザクションブランチの実行サービス名

executiontime

基本情報とトランザクションブランチの実行時間情報を取得します。

cputime

基本情報とトランザクションブランチの CPU 時間情報を取得します。

nothing の指定は，1 つしかできません。また，nothing とほかの統計情報項目を同時に指定した場合，nothing の指定は無効になります。

トランザクションに関する統計情報を取得する場合は，次のどちらかを指定してください。

- トランザクションサービス定義に trn_tran_statistics=Y を指定
- trnstics コマンドで-s オプションを指定

この定義を省略した場合は，rap リスナーサービス定義の trn_statistics_item オペランドの指定に従います。

dccltttroptiitem=トランザクション最適化項目〔,トランザクション最適化項目〕…

複数のユーザサーバで構成されるグローバルトランザクションの性能を向上させるための最適化項目を，次の文字列で指定します。CUP からトランザクションを開始する場合にだけ有効です。

base

同期点取得処理全体（プリペア処理，コミット処理，およびロールバック処理）を最適化します。OpenTP1 のトランザクション制御は 2 相コミット方式で実行しているため，2 つのトランザクションブランチ間のコミット制御には，4 回のプロセス間通信が必要となります。

次の条件をすべて満たす場合，親トランザクションブランチが子トランザクションブランチのコミット処理を代わりに実行することで，コミット制御で必要な 4 回のプロセス間通信を削減します。

- 親トランザクションブランチと，子トランザクションブランチが同一 OpenTP1 下にあること。
- 親トランザクションブランチが，子トランザクションブランチを同期応答型 RPC で呼び出していること。
- 子トランザクションブランチでアクセスしたリソースマネジャの XA インターフェース用オブジェクトが，親トランザクションブランチにもリンケージされていること。

asyncprepare

base の指定条件を満たしていないため同期点取得処理全体の最適化ができない場合に，プリペア処理を最適化します。

次の条件をすべて満たす場合，親トランザクションブランチから発行された RPC によって子トランザクションブランチがサービス要求を実行したときに，RPC が返される前にプリペア処理を実行することで，2 回のプロセス間通信を削減します。

- base を指定した最適化ができないこと。
- 親トランザクションブランチが，子トランザクションブランチを同期応答型 RPC で呼び出していること。

ただし，この最適化を実行した場合，親トランザクションブランチが発行した同期応答型 RPC の応答時間が遅くなります。また，子トランザクションブランチは，プリペア処理からコミット処理までの間隔（親トランザクションブランチからの指示がないとトランザクションを決着できない状態）が大きくなります。そのため，親トランザクションブランチの OpenTP1 がシステムダウンし，トランザクションブランチ間の連絡ができなくなると，ジャーナルファイルのスワップやチェックポイントダンプファイルの有効化が遅れ，子トランザクションブランチの OpenTP1 もシステムダウンする場合があります。

トランザクション最適化項目は，重複して指定できます。ただし，優先順位は次のようになります。

1 ＞ 2

1：base

2：asyncprepare

この指定を省略した場合は，rap リスナーサービス定義の trn_optimum_item オペランドの指定に従います。

dccltttrwatchtime=トランザクション同期点処理時の最大通信待ち時間

～〈符号なし整数〉((1～65535))（単位：秒）

トランザクションの同期点処理で，トランザクションブランチ間で行う通信（プリペア，コミット，ロールバック指示，または応答など）の受信待ち時間の最大値を指定します。CUP からトランザクションを開始する場合にだけ有効です。

指定時間を過ぎても指示または応答がない場合は，該当するトランザクションブランチが 2 相コミットの 1 相目であればロールバックさせ，1 相目完了後であればトランザクションサービスのシステムプロセスでトランザクション決着処理を再試行します。

この定義を省略した場合は，rap リスナーサービス定義の trn_watch_time オペランドの指定に従います。

dccltttrrbinfo=no | self | remote | all

トランザクションブランチがロールバックした場合に，ロールバック要因に関する情報をログに取得するかどうかを指定します。CUP からトランザクションを開始する場合にだけ有効です。

no

ロールバック情報を取得しません。

self

ロールバック要因が発生したトランザクションブランチでだけ，ログにロールバック情報を取得します。

remote

self に加え，他ノードのトランザクションブランチからロールバック要求されたトランザクションブランチでも，ログにロールバック情報を取得します。

all

remote に加え，自ノードのトランザクションブランチからロールバック要求されたトランザクションブランチでも，ログにロールバック情報を取得します。

この指定を省略した場合は，rap リスナーサービス定義の trn_rollback_information_put オペランドの指定に従います。

dccltttrlimittime=トランザクションブランチ最大実行可能時間

～〈符号なし整数〉((0～65535))（単位：秒）

トランザクションブランチの最大実行可能時間を指定します。CUP からトランザクションを開始する場合だけ有効です。

トランザクションブランチを開始してから同期点処理が終了するまでの時間がこのオペランドの指定時間を超えないように，rpcCall メソッドおよび同期点処理内で行う通信のタイムアウト時間を，次のように自動設定します。

- rpcCall メソッドのタイムアウト時間

「K≧このオペランドの指定時間」の場合：要求処理を実行しないで，タイムアウトでエラーリターンします。

「K ＜このオペランドの指定時間」および「(このオペランドの指定時間－K）≧W」の場合：W をタイムアウト時間とします。

「K ＜このオペランドの指定時間」および「(このオペランドの指定時間－K）＜ W」の場合：(このオペランドの指定時間－K）をタイムアウト時間とします。

「K」「W」の内容は次のとおりです。

K：現時刻－トランザクションブランチ開始時刻

W：dcwatchtim オペランド指定時間

- 同期点処理内で行う通信のタイムアウト時間

  「K≧このオペランドの指定時間」の場合：タイムアウト時間を 1 秒とします。

  「K ＜このオペランドの指定時間」および「(このオペランドの指定時間－K）≧W」の場合：W をタイムアウト時間とします。

  「K ＜このオペランドの指定時間」および「(このオペランドの指定時間－K）＜ W」の場合：(このオペランドの指定時間－K）をタイムアウト時間とします。

  「K」「W」の内容は次のとおりです。

  K：現時刻－トランザクションブランチ開始時刻

  W：dcclttrwatchtime オペランド指定時間（dcclttrwatchtime オペランドを省略した場合は dcwatchtim オペランド指定時間）

上記の受信待ち以外の処理で時間が掛かった場合は，このオペランドの指定時間内にトランザクションブランチが終了しないことがあります。

同期点処理開始前にこのオペランドの指定時間が経過した場合，そのトランザクションはロールバックされます。

0 を指定した場合は，時間監視を行いません。

この定義を省略した場合は，rap リスナーサービス定義の trn_limit_time オペランドの指定に従います。

dcclttrrbrcv=Y | N

RPC 先トランザクションブランチにロールバック指示を送信したあと，ロールバック完了通知を受信するかどうかを指定します。CUP からトランザクションを開始する場合にだけ有効です。

Y

ロールバック完了通知を受信します。

N

ロールバック完了通知を受信しません。

N を指定した場合，RPC 先トランザクションブランチからのロールバック完了通知を受信しないで（RPC 先トランザクションブランチのロールバック処理の完了を待たないで）自トランザクションブランチを終了します。

この定義を省略した場合は，rap リスナーサービス定義の trn_rollback_response_receive オペランドの指定に従います。

dccltrrecoverytype=type1 | type2 | type3

UAP 障害時のトランザクション同期点処理方式を指定します。CUP からトランザクションを開始する場合にだけ有効です。

RPC がタイムアウトし，RPC 発行先プロセスのアドレスが未解決の場合およびトランザクション実行中の UAP がダウンした場合に，トランザクションブランチ間の連絡がスムーズにできないため，トランザクションの決着に時間が掛かることがあります。

このオペランドでは，次に示す障害が発生した場合のトランザクション同期点処理方式を，指定値に示す 3 つの方式から選択して指定します。

（障害 1）RPC がタイムアウトした場合

この場合，RPC 発行元トランザクションブランチは，サービス要求がどのプロセスで実行されているかがわからないため，RPC 発行先トランザクションブランチにトランザクション同期点メッセージを送信できません。そのため，RPC 発行元トランザクションブランチおよび RPC 発行先トランザクションブランチはトランザクション同期点メッセージ待ちとなり，トランザクションの決着に時間が掛かります。

（障害 2）rap サーバが RPC の応答受信前にダウンした場合

この場合，RPC 発行元トランザクションブランチは，サービス要求がどのプロセスで実行されているかがわからないため，RPC 発行先トランザクションブランチにトランザクション同期点メッセージを送信できません。そのため，RPC 発行先トランザクションブランチはトランザクション同期点メッセージ待ちとなり，トランザクションの決着に時間が掛かります。

（障害 3）RPC 発行先 UAP からの応答受信後に rap サーバと RPC 発行先 UAP がほぼ同時にダウンした場合

この場合，それぞれのトランザクションブランチを引き継いだトランザクション回復プロセスは，相手プロセスのダウンを知らないため，すでに存在しないプロセスにトランザクション同期点メッセージを送信してしまい，トランザクションの決着に時間が掛かることがあります。

それぞれの指定値が示す処理方式について説明します。

type1

（障害 1）が発生した場合，RPC 発行元トランザクションブランチおよび RPC 発行先トランザクションブランチは，トランザクション同期点メッセージ受信処理がタイムアウトすることによって，トランザクションを決着します。

（障害 2）が発生した場合，RPC 発行元トランザクションブランチは，RPC 発行先トランザクションブランチにトランザクション同期点メッセージを送信しないでトランザクションを決着します。RPC 発行先トランザクションブランチは，トランザクション同期点メッセージ受信処理がタイムアウトすることによって，トランザクションを決着します。

（障害 3）が発生した場合，RPC 発行元トランザクションブランチおよび RPC 発行先トランザクションブランチは，トランザクション同期点メッセージ受信処理がタイムアウトすることによって，トランザクションを決着します。

type2

（障害 1）が発生してトランザクションをコミットする場合は type1 と同じです。

（障害 1）が発生してトランザクションをロールバックする場合，または（障害 2）が発生した場合は，RPC 発行元トランザクションブランチは，RPC 発行先トランザクションブランチが存在するノードのトランザクションサービスプロセスにトランザクション同期点メッセージを送信後，トランザクションを決着します。トランザクション同期点メッセージを受信したトランザクションサービスプロセスは，該当するトランザクションブランチを処理中のプロセスに，トランザクション同期点指示を送信します。

（障害 3）が発生した場合，RPC 発行元トランザクションブランチおよび RPC 発行先トランザクションブランチは，トランザクション同期点メッセージ受信処理がタイムアウトすることによって，トランザクションを決着します。

type3

（障害 1）が発生してトランザクションをコミットする場合は，type1 と同じです。

（障害 1）が発生してトランザクションをロールバックする場合，（障害 2）が発生した場合，または（障害 3）が発生した場合，相手トランザクションブランチが存在するノードのトランザクションサービスプロセスに，トランザクション同期点メッセージを送信します。トランザクション同期点メッセージを受信したトランザクションサービスプロセスは，該当するトランザクションブランチを処理中のプロセスに，トランザクション同期点指示を送信します。

次に示す場合，このオペランドに type2 または type3 を指定しても，トランザクションの決着に時間が掛かることがあります。

- RPC 実行中に，RPC 発行先 UAP の状態が変更となり（負荷増加，UAP 終了，UAP 閉塞など），ほかのノードの同一 UAP にサービス要求が再転送された場合
- 相手先の OpenTP1 がこのオプションをサポートしていないバージョンの場合
- 相手先トランザクションブランチがトランザクション同期点メッセージ受信処理以外で時間が掛かっている場合

この定義を省略した場合は，rap リスナーサービス定義の trn_partial_recovery_type オペランドの指定に従います。

dccltrexptm=トランザクションブランチ限界経過時間

～〈符号なし整数〉((0～65535))（単位：秒）

トランザクションブランチの処理時間の最大値を指定します。CUP からトランザクションを開始する場合にだけ有効です。

指定時間を超えてもトランザクションブランチが完了しないとき，そのトランザクションブランチのプロセスを異常終了させて，ロールバックします。0 を指定した場合は，時間監視をしません。

この指定を省略した場合，rap リスナーサービス定義の trn_expiration_time オペランドの指定に従います。

なお，RPC 機能を使用した場合に，他プロセスで実行するトランザクションブランチの処理時間も監視時間に含むかどうかは，TP1/Client/J 環境定義の dccltrexpsp オペランドで指定してください。

dccltrcputm=トランザクションブランチ CPU 監視時間

～〈符号なし整数〉((0～65535))（単位：秒）

トランザクションブランチが同期点処理までに使用できる CPU 時間を指定します。CUP からトランザクションを開始する場合にだけ有効です。

0 を指定した場合，CPU 時間を監視しません。

指定時間を超えた場合は，そのトランザクションブランチのプロセスを異常終了させ，ロールバックします。

この指定を省略した場合は，rap リスナーサービス定義の trn_cpu_time オペランドの指定に従います。

dccltrexpsp=Y | N | F

トランザクションブランチの処理を監視するとき，次の処理時間も監視時間に含むかどうかを指定します。

1. 監視対象のトランザクションブランチが，RPC 機能を使ってほかのトランザクションブランチを呼び出し，その処理が終わるのを待つ時間
2. 連鎖型 RPC で呼び出されたサーバ UAP が次のサービス要求を待つ時間
3. 監視対象のトランザクションブランチが，非応答型 RPC を使用してほかのトランザクションブランチを呼び出したあと，処理結果受信処理を行っている時間

それぞれの指定値について説明します。

Y

1，2，3 すべてを監視時間に含みます。

N

3 だけを監視時間に含みます。

F

1，2，3 のどれも監視時間に含みません。

この指定を省略した場合は，rap リスナーサービス定義の trn_expiration_time_suspend オペランドの指定に従います。

dcsndrcvtype=DCCLT_ONEWAY_SND | DCCLT_ONEWAY_RCV | DCCLT_SNDRCV

TCP/IP 通信機能の使用時に初期化する環境を指定します。

このオペランドを指定してない場合は，TCP/IP 通信機能を使用できません。

DCCLT_ONEWAY_SND

メッセージを一方送信するための環境

DCCLT_ONEWAY_RCV

メッセージを一方受信するための環境

DCCLT_SNDRCV

メッセージを送受信するための環境

dcrcvport=受信用の CUP のポート番号

～〈符号なし整数〉((1～65535))《11000》

MHP から送信されたメッセージを受信する CUP のポート番号を指定します。MHP がメッセージを送信するために使用するポート番号と同じ値を指定してください。同一マシン内で，複数のプロセスまたはスレッドを同時に実行する場合は，それぞれ異なるポート番号を指定してください。

指定できるポート番号でも，OS またはほかのプログラムで使用するポート番号は指定しないでください。指定した場合，応答データを正しく受信できないことがあります。なお，OS が使用するポート番号は，OS ごとに異なります。OS のマニュアルなどを参照してください。

dcsndhost=接続する MHP が存在するノードのホスト名　～〈文字列〉

CUP からメッセージを送信する場合，コネクションを確立して接続する MHP が存在するノードのホスト名を指定します。

また，ホスト名として，10 進ドット記法の IP アドレスを指定することもできます。

dcsndport=接続する MHP のポート番号

～〈符号なし整数〉((1～65535))《12000》

CUP からメッセージを送信する場合，コネクションを確立して接続する MHP のポート番号を指定します。dcnamport オペランドに指定するポート番号と重複しないように指定してください。

dcsockopenatrcv=Y | N　～《N》

TCP/IP 通信機能の使用時に，1 コネクションで送受信する場合の，受信用ソケットを開設する契機（送信相手からの接続を待ち受け始める契機）を指定します。

なお，このオペランドの指定を省略した場合は，rpcOpen メソッド実行時に受信用ソケットを開設します。

Y

cltReceive メソッド実行時に，コネクションが確立されていない場合は受信用ソケットを開設します。

N

rpcOpen メソッド実行時に，受信用ソケットを開設します。

dcweburl=サービス要求先の TP1/Web の URL　～〈パス名〉

サービス要求先の TP1/Web の URL を指定します。

プロトコル，WWW サーバ，プロンプターの CGI 名称，TP1/Web のサービス名などの情報を URL 形式で指定します。

dchost オペランドと dcweburl オペランドの両方を指定した場合は，dcweburl オペランドの指定が有効になります。

また，TP1/Web 接続機能を使用する場合は，dcscddirect オペランドおよび dcnamuse オペランドを使用できません。dcweburl オペランドを定義して，かつ dcscddirect=Y および dcnamuse=Y の場合は，定義エラーとなります。

dccltdbgtrcfilecount=デバッグトレースファイルの最大ファイル数

～〈符号なし整数〉((0～256))《0》

デバッグトレースファイルの最大ファイル数を指定します。0 を指定した場合，ファイル数の制限はなくなります。デバッグトレースファイルの総数が，このオペランドの指定値を超えた場合，更新日時の最も古いファイルが削除されます。

dccltdbgtrcfilecount オペランドで異なる値を指定した複数の TP1Client インスタンスを同時に使用した場合，実際の最大ファイル数は保証できません。

rpcOpen メソッド未発行の状態で TP1Client インスタンスを作成した場合，dccltdbgtrcfilecount オペランドは，システムプロパティとしても定義できます。この場合，TP1Client インスタンスの作成後に rpcOpen メソッドを発行すると，rpcOpen メソッドの内容を解析した結果を上書きして動作します。rpcOpen メソッドを発行するとシステムプロパティに定義された dccltdbgtrcfilecount オペランドの指定は無効になります。

dccltdbgtrcfilecount オペランドの指定が不正な場合，エラーは通知されません。この場合，dccltdbgtrcfilecount オペランドの指定はデフォルト値が有効になります。

dccltrpcmaxmsgsize=RPC 送受信電文の最大長

～〈符号なし整数〉((1～8))《1》（単位：メガバイト）

RPC で受け渡しできる電文の最大長を指定します。

rpcCall メソッドで送受信できる RPC 電文の最大長は，このオペランドを省略した場合，またはデフォルト値（1）を指定した場合は，1 メガバイト（1048576 バイト）です。このオペランドに指定できる最大値（8）を指定した場合は，8 メガバイト（8388608 バイト）です。このオペランドの指定よりも大きいサイズの送信メッセージ長および受信メッセージ長を，rpcCall メソッドの引数に指定することはできません。送信メッセージ長または受信メッセージ長がこのオペランドの指定値よりも大きい場合，rpcCall メソッドは ErrInvalidArgsException を返します。

dccltrpcmaxmsgsize オペランドを指定した場合，次の機能は使用しないでください。通信先の TP1/Server ノードでエラーが発生します。

- スケジューラダイレクト機能を使用した RPC
- 通信先を指定した RPC
- TP1/Web 接続機能

システム共通定義の rpc_max_message_size オペランドをサポートしている TP1/Server（バージョン 06-02 以降）以外の TP1/Server にサービス要求した場合の動作については次の表を参照してください。

表 A-5　dccltrpcmaxmsgsize オペランドに 2 以上を指定した場合の rpcCall メソッドの動作

メソッド	RPC の種類を指定する定義	窓口となる TP1/Server のバージョン	ノードのバージョン※1	
			06-02 より前の場合	06-02 以降の場合※2
rpcCall	dcrapdirect=Y	06-02 より前	×※3	×※3
		06-02 以降	×※4	○
	dcnamuse=Y	06-02 より前	×※4	○
		06-02 以降	×※4	○

（凡例）

○：要求できます。

×：要求できません。

注※1

窓口となる TP1/Server に指定されている all_node オペランドで定義された，SPP を起動しているノードのバージョンです。

注※2

システム共通定義に rpc_max_message_size オペランドを指定している場合です。

注※3

入力データ長が 1 メガバイト以上の場合は，ErrMessageTooBigException が発生し，入力データ長が 1 メガバイト以下で出力領域長が 1 メガバイト以上の場合は ErrInvalidArgsException が発生します。

注※4

ErrNoSuchServiceGroupException が発生します。

dcscdhostchange=Y | N ～《N》

TP1/Client/J 環境定義の dchost オペランドに指定したサービス要求先スケジューラを RPC ごとに分散させるかどうかを指定します。

Y

サービス要求先スケジューラを RPC ごとに分散させます。

N

サービス要求先スケジューラを RPC ごとに分散させません。

Y を指定した場合の処理を説明します。

RPC でのサービス要求が 1 度目の場合，TP1/Client/J 環境定義に dchostselect=Y を指定したときは，TP1/Client/J 環境定義の dchost オペランドに指定したサービス要求先スケジューラがランダムに選択されます。TP1/Client/J 環境定義に dchostselect=N を指定したときは，TP1/Client/J 環境定義の dchost オペランドの先頭に指定したサービス要求先スケジューラが選択されます。

RPC でのサービス要求が 2 度目以降の場合，前回のサービス要求先スケジューラから dchost オペランドで指定した順にラウンドロビン方式でサービス要求先スケジューラが選択されます。

dccltloadbalance=Y | N ～《N》

ネームサーバから得た複数のサービス要求先スケジューラの情報から，負荷レベルが最も低いサービス要求先スケジューラの情報をキャッシュに格納するかどうかを指定します。

Y

負荷レベルが最も低いサービス要求先スケジューラの情報をキャッシュに格納します。

N

負荷レベルが最も低いサービス要求先スケジューラの情報をキャッシュに格納しません。

Y を指定した場合の処理を説明します。

キャッシュに格納されたサービス要求先スケジューラが複数ある場合，最初のサービス要求先スケジューラは，ランダムに選択されます。RPC でのサービス要求が 2 度目以降の場合，キャッシュに格納されたサービス要求先スケジューラが，ラウンドロビン方式で選択されます。

負荷レベルの詳細については，マニュアル「OpenTP1 解説」を参照してください。

dccltcachetim=キャッシュの有効期限

～〈符号なし整数〉((0～65535))《30》(単位：秒)

サービス要求先スケジューラの情報をキャッシュに格納した時点からの，キャッシュの有効期限を指定します。キャッシュの有効期限を満了したサービス要求先スケジューラの情報は，キャッシュから削除されます。その後，サービス要求先スケジューラの情報を再度取得して，その情報をキャッシュに格納することで，キャッシュを更新します。

このオペランドは，TP1/Client/J 環境定義に dccltloadbalance=Y を指定したときだけ有効です。

dccltconnecttimeout=コネクション確立最大監視時間

～〈符号なし整数〉((0～65535))《0》（単位：秒）

データ送信時のコネクション確立処理に対する最大監視時間を指定します。

このオペランドで指定する値は，java.net.Socket.connect メソッド（以降「connect メソッド」と呼びます）の処理時間ではなく，connect メソッドで実行されるコネクションの確立処理に掛かる最大監視時間です。

相手システムが起動していないなどの要因でコネクションを確立できない場合，このオペランドで指定した監視時間が経過する前に CUP から発行したメソッドがエラーリターンすることがあります。これは，このオペランドで指定した最大監視時間よりも，OS によるコネクション確立処理の監視時間が優先されるためです。OS によるコネクション確立処理の監視時間，コネクション確立要求の再送回数および再送処理の間隔は，プラットフォームによって異なります。また，使用するメソッドや機能によっては，connect メソッドの処理時間がこのオペランドで指定した値よりも大きくなる場合があります。

0 を指定した場合，またはこのオペランドの指定を省略した場合は，OS がコネクションの確立処理を監視します。

dccltprftrace=Y | N ～《Y》

TP1/Client/J をサービスプラットフォーム上で動作させた場合に，サービスプラットフォームの性能解析トレースを取得するかどうかを指定します。

Y

性能解析トレースを取得します。ただし，PRF デーモンが起動していない場合は性能解析トレースを取得しません。

N

性能解析トレースを取得しません。

dccltprfinfosend=Y | N ～《Y》

TP1/Server に対して，ネームサービスを使用した RPC，およびスケジューラダイレクト機能を使用した RPC を行う場合に，OpenTP1 の性能検証用トレースに出力する情報として TP1/Client/J 内部で識別情報（IP アドレスなど）を付加するかどうかを指定します。

Y

TP1/Client/J 内部で識別情報を付加します。

N

TP1/Client/J 内部で識別情報を付加しません。

dccltnammlthost=Y | N ～《N》

マルチホームドホスト形態の TP1/Server に対してネームサービスを使用した RPC を行うかどうかを指定します。

Y

マルチホームドホスト形態の TP1/Server に対してネームサービスを使用した RPC を行います。RPC の要求先を複数定義している場合，またはマルチホームドホスト形態の TP1/Server が 1 つ以上ある場合に指定してください。

N

マルチホームドホスト形態の TP1/Server に対してネームサービスを使用した RPC を行いません。マルチホームドホスト形態の TP1/Server に対してネームサービスを使用した RPC を行うと，エラーが発生する場合があります。

詳細については，「●マルチホームドホスト形態の TP1/Server に対して RPC を行う場合の定義」を参照してください。

dccltdatacomp=Y | N ～《N》

データ圧縮機能を使用するかどうかを指定します。

Y

データ圧縮機能を使用します。

N

データ圧縮機能を使用しません。

dccltconnectinf=端末識別情報

端末識別情報として，DCCM3 論理端末の論理端末名称を EBCDIK コードで指定します。端末識別情報は，先頭に 0x を付け，0x の後ろに 128 文字までの 16 進数表記で指定します。先頭の 0x は 128 文字に含まれません。2 文字で 1 バイトの値として，64 バイト（128 文字）まで指定できます。ただし，DCCM3 側では先頭 8 バイト目までに指定した値だけが有効になり，9 バイト目以降に指定した値は無視されます。

このオペランドに指定した値は，setConnectInformation メソッドを呼び出すことで，動的に変更できます。このオペランドに指定した値は，setConnectInformation メソッドを呼び出したあと，再び rpcOpen メソッドを呼び出すまで無視されます。

このオペランドに指定した値は，TP1/Client/J 環境定義の dchost オペランドに DCCM3 論理端末のホスト名を，dchost オペランドまたは dcrapport オペランドに DCCM3 論理端末のポート番号を指定し，次のどちらかの方法で DCCM3 論理端末との常設コネクションを確立した場合に有効となります。

- openConnection メソッドを呼び出します。引数ありの openConnection メソッドの場合，引数 host に DCCM3 論理端末のホスト名，引数 port に DCCM3 論理端末のポート番号を指定します。
- TP1/Client/J 環境定義の dcrapautoconnect オペランドに Y を指定し，rpcCall メソッドを呼び出します。

このオペランドの指定を省略した場合，DCCM3 論理端末に端末識別情報は通知されません。ただし，setConnectInformation メソッドを呼び出した場合は，setConnectInformation メソッドに指定した端末識別情報が，DCCM3 論理端末との常設コネクション確立時に DCCM3 論理端末に通知されます。

dcscdmulti=Y | N ～《N》

マルチスケジューラ機能を使用するかどうかを指定します。

Y

マルチスケジューラ機能を使用します。

N

マルチスケジューラ機能を使用しません。

マルチスケジューラ機能を使用すると，複数起動されたマルチスケジューラデーモンの中から，1 つをランダムに選択することで，スケジューリングの負荷を軽減できます。

この定義に Y を指定した場合，TP1/Client/J 環境定義 dchost，dcscdport，dcscdmulticount も併せて参照してください。また，この定義は，rpcCallTo メソッド実行時は無効です。

dcscdmulticount=マルチスケジューラデーモンのプロセス数　～〈符号なし整数〉((1～4096))《1》

マルチスケジューラデーモンのプロセス数を指定します。スケジュールサービス定義 scdmulti の-m オプションに指定したプロセス数か，またはそれ以下の値を指定します。

この定義は，TP1/Client/J 環境定義に dcscdmulti=Y および dcscddirect=Y を指定した場合に有効です。なお，この場合，ポート番号は，次に示す範囲の値からランダムに選択されます。

- 下限値：dchost または dcscdport に指定したポート番号の値
- 上限値：下限値 + dcscdmulticount に指定したプロセス数 - 1

このポート番号の最大値が 65535 を超える場合は，定義解析で ErrFatalException，および setDchost で ErrInvalidPortException が発生します。

scdmulti の-m オプションに指定したプロセス数よりも大きい値を指定した場合，マルチスケジューラが起動していないポートに接続を試み，ErrNetDownAtClientException が発生する場合があります。

dccltcupsndhost=送信元ホスト　～〈文字列〉

次に示すコネクション確立要求時の，送信元ホストを指定します。

- スケジューラダイレクト機能を使用した RPC
- ネームサービスを使用した RPC
- リモート API 機能を使用した RPC
- 通信先を指定した RPC
- TCP/IP 通信機能

ホスト名として，10 進ドット記法の IP アドレスを指定することもできます。

なお，次の場合は，発行したメソッドで例外が発生します。

- ホスト名として localhost を指定した場合
- 127 で始まる IP アドレスを指定した場合
- CUP を実行するマシン上に存在しないホストを指定した場合

この定義を省略すると，送信元ホストは任意に割り当てられます。

dccltcuprcvport=CUP の受信で使用するポート番号　～〈符号なし整数〉((5001～65535))

サーバからのメッセージを受信する CUP のポート番号を指定します。

この定義で指定したポート番号は，次の機能を使用する場合に有効となります。

- スケジューラダイレクト機能を使用した RPC の受信時（受信ポート）
- ネームサービスを使用した RPC の受信時（受信ポート）

この定義を省略すると，システムが任意に割り当てたポート番号を使用します。

同一マシン内で，複数のプロセス，または複数のスレッドを同時に実行する場合は，それぞれ異なるポート番号を指定してください。

指定できるポート番号でも，OS またはほかのプログラムで使用するポート番号は指定しないでください。指定した場合，応答データを正しく受信できないことがあります。なお，OS が使用するポート番号は，OS ごとに異なります。OS のマニュアルなどを参照してください。

dccltrcmplmttm=トランザクション完了限界時間　～〈符号なし整数〉((0～65535))（単位：秒）

rap サーバで代理実行するトランザクションブランチの，開始から終了までの最大実行時間を指定します。この指定時間を超えた場合，rap サーバのプロセスが異常終了したあとに，トランザクションブランチが回復プロセスでコミット，またはロールバックのどちらかに決着して終了します。

0 を指定した場合は，トランザクションブランチの最大実行時間を監視しません。

この指定を省略した場合は，rap リスナーサービス定義の trn_completion_limit_time オペランドの指定に従います。

この機能をバージョン 07-01 以前の TP1/Server に対して使用した場合，機能が無効の状態で動作します。

dccltsendbuffsize=TCP/IP の送信バッファサイズ　～〈符号なし整数〉((8192～2147483647))（単位：バイト）

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指定します。

高速な通信媒体や MTU の大きな通信媒体を使用している場合，この値を大きくすることによって性能向上を図れます。

このオペランドを省略した場合は，JavaVM の送信バッファサイズを適用します。

このオペランドの値は，JavaVM で使用できる TCP/IP 送信バッファの上限値以下を指定してください。TCP/IP 送信バッファのデフォルト値，変更方法，変更できるサイズの範囲は，OS によって異なります。詳しくは，各 OS のマニュアルを参照してください。また，CUP と通信するすべてのノードで同じ値を指定してください。同じ値を指定しない場合，通信するノードとバッファサイズに差異が生じ，通信性能が劣化するおそれがあります。

dccltrecvbuffsize=TCP/IP の受信バッファサイズ　～〈符号なし整数〉((8192～2147483647))（単位：バイト）

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指定します。

高速な通信媒体や MTU の大きな通信媒体を使用している場合，この値を大きくすることによって性能向上を図れます。

このオペランドを省略した場合は，JavaVM の受信バッファサイズを適用します。

このオペランドの値は，JavaVM で使用できる TCP/IP 受信バッファの上限値以下を指定してください。TCP/IP 受信バッファのデフォルト値，変更方法，変更できるサイズの範囲は，OS によって異なります。詳しくは，各 OS のマニュアルを参照してください。また，CUP と通信するすべてのノード

で同じ値を指定してください。同じ値を指定しない場合，通信するノードとバッファサイズに差異が生じ，通信性能が劣化するおそれがあります。

dcinquiretimecheck=Y | N ～《Y》

常設コネクションを使用し，かつオートコネクトモード※1 の場合に，CUP 実行プロセスで「問い合わせ間隔最大時間」をチェックするかどうかを指定します。

rap サーバが常設コネクションを解放するタイミングと，CUP 実行プロセスが RPC を実行するタイミングが重なった場合，RPC の実行に失敗して ErrNetDownAtClientException 例外を返すことがあります。この現象を回避するためには，このオペランドを指定します。

このオペランドを省略，またはこのオペランドに Y を指定した場合，「問い合わせ間隔最大時間」には 0 以外を指定※2 してください。

このオペランドは省略できます。

Y

CUP 実行プロセスで，「問い合わせ間隔最大時間」をチェックします。

N

CUP 実行プロセスで，「問い合わせ間隔最大時間」をチェックしません。

省略した場合は，CUP 実行プロセスでも「問い合わせ間隔最大時間」をチェックします。

CUP 実行プロセスは，トランザクション処理の処理区間外※3，かつ連鎖型 RPC の処理区間外※4 で rpcCall メソッドを使用して RPC を実行するタイミングで問い合わせ間隔をチェックします。

問い合わせ間隔のチェックは次の計算式で求めた時間を使用します。この時間を超過した場合は，常設コネクションを解放して再接続し RPC を実行します。

```
チェック時間 = 問い合わせ間隔最大時間 - T
```

T：問い合わせ間隔最大時間の 2%の値

ただし，250 ミリ秒 ≦ T ≦ 3 秒となります。

注※1

次のどちらかの場合が，オートコネクトモードに該当します。

- TP1/Client/J 環境定義の dcrapautoconnect オペランドに Y を指定している。
- CUP 内で，SetRpcextend メソッドの extendoption パラメタに DCRPC_RAP_AUTOCONNECT を指定して実行している。

注※2

TP1/Client/J 環境定義の dccltinquiretime オペランドに 0 以外を指定してください。または CUP 内で，setDccltinquiretime メソッドの sec パラメタに 0 以外を指定して実行してください。

注※3

次の両方の条件が重なった場合がトランザクション処理の処理区間外となります。

- XA リソースサービス機能を使用していない（CUP が TP1 Connector 上で動作していない，または CUP が TP1 Connector 上で動作しているが Connector 属性ファイルの transaction-support タグに XATransaction を指定していない）。

- trnBegin メソッド実行から trnUnchainedCommit メソッドまたは trnUnchainedRollback メソッドを実行するまでの処理区間外である。

注※4

連鎖型 RPC でも，初回の rpcCall メソッドでは問い合わせ間隔をチェックします。

dcnotifyreshost=Y | N　～《N》

OpenTP1 システム内に Kubernetes ノードで起動する TP1/Server が存在する場合，TP1/Client/J でこのオペランドに Y を指定してください。

このオペランドに Y を指定することで，サービス要求先の TP1/Server に，dcresponsehost で指定した応答電文受信用 IP アドレスを通知します。

これによって，Kubernetes ノードを含む OpenTP1 システム内で通信が可能となります。

Y

サービス要求先の TP1/Server に応答電文受信用 IP アドレスを通知します。次の RPC 通信に対して有効になります。

- スケジューラダイレクト機能を使用した RPC（rpcCall メソッド）
- 通信先を指定した RPC（rpcCallTo メソッド）
- ネームサービスを使用した RPC（rpcCall メソッド）

N

サービス要求先の TP1/Server に，応答電文受信用 IP アドレスを通知しません。

注意事項

この定義で Y を指定した場合，dcresponsehost の指定が必要です。

dcresponsehost=ホスト名　～〈文字列〉

OpenTP1 システム内に Kubernetes ノードで起動する TP1/Server が存在する場合，このオペランドの指定が必要です。TP1/Client/J が応答電文を受信するために使用する，ホスト名または IP アドレスを指定します。

このオペランドを指定することで，サービス要求先の TP1/Server に応答電文受信用 IP アドレスが通知され，Kubernetes ノードを含む OpenTP1 システム内で通信が可能となります。

ホスト名に使用できる文字は先頭が英数字または-（ハイフン）で，先頭以外が英数字，-（ハイフン），または.（ピリオド）です。

ホスト名は，/etc/hosts ファイルまたは DNS などで，IP アドレスとのマッピングができなければなりません。なお，localhost または名前解決した結果が 127 で始まる IP アドレス（例：127.0.0.1）になるホスト名は指定しないでください。

ホスト名は，1 個だけ指定できます。

ホスト名または IP アドレスは，TP1/Client/J の環境によって次を指定してください。

- TP1/Client/J が Kubernetes クラスタ外の場合

  TP1/Client/J が起動するマシンのホスト名または IP アドレス
- TP1/Client/J が Kubernetes クラスタ内の場合

Kubernetes クラスタ内のどれかの Kubernetes ノードのホスト名または IP アドレス

注意事項

- ホスト名に 256 文字以上指定した場合，指定した文字列の先頭から 255 文字までをノード名として扱います。
- この指定は，dcnotifyreshost オペランドに Y を指定した場合に有効です。
- dcnotifyreshost オペランドに Y を指定した場合，このオペランドは省略できません。
- 指定したホスト名の名前解決は rpcOpen メソッド発行時に実施します。そのためホスト名にマッピングしている IP アドレスの変更は，CUP 実行プロセスを正常停止したあとに実施してください。

ホスト名または IP アドレスの変更手順

ホスト名または IP アドレスを変更する場合，CUP 実行プロセスを正常停止したあとに実施してください。CUP が Java サーブレットで動作している場合は Web サーバを停止したあとに実施してください。

(c) TP1/Client/J 環境定義を指定するときの注意事項

- TP1/Client/J 環境定義のオペランドを記述した行に，設定値以外の文字を記述しないでください。設定値以外の文字を記述した場合，rpcOpen メソッドが ErrFatalException を返す場合があります。
- TP1/Client/J は，TP1/Client/J 環境定義のオペランドで始まる行（行頭の空白およびタブは無視されます）を定義解析対象とします。TP1/Client/J 環境定義のオペランド以外の文字列が指定された行は，すべてコメント行と見なされます。

付録 A.3 TP1/Client/J の障害対策

TP1/Client/J のトラブルシュート機能で取得できるトレースの種類を次の表に示します。

トレースの種類	出力先
UAP トレース	ファイルに出力されます。
データトレース	ファイルに出力されます。
エラートレース	ファイルに出力されます。
メモリトレース	CUP で用意した String 型の配列に格納されます。
メソッドトレース	ファイルに出力されます。
デバッグトレース	TP1/Client/J のメモリ内に取得されます。 なお，TP1/Client/J の提供するメソッドが例外を返した場合は，ファイルまたは標準出力に出力されます。
性能解析トレース	サービスプラットフォーム上に出力されます。
性能検証用トレース	TP1/Server 上に出力されます。

トレースを出力する場合，環境変数 TZ にタイムゾーンを設定してください。設定していないと，トレース情報の時刻が正しく出力されません。

(1) トレースファイルの出力内容

ファイルに出力されるトレースは，出力先ディレクトリなどを変更できます。トレースファイルに指定できるオプションと出力ファイル名を次の表に示します。また，ファイル出力ディレクトリを変更するための TP1/Client/J 環境定義のオペランド名も次の表に示します。

表 A-6 トレースファイルのオプションとファイル名

トレースファイル	オプション	ファイル出力先指定方法※1	出力ファイル名※2
UAP トレース	• ファイル出力ディレクトリ • トレースファイルのサイズ	dcuaptracepath オペランド	DCUAP1.TRC DCUAP2.TRC
データトレース	• ファイル出力ディレクトリ • トレースファイルのサイズ • データの最大サイズ	dcdatatracepath オペランド	DCDAT1.TRC DCDAT2.TRC
エラートレース	• ファイル出力ディレクトリ • トレースファイルのサイズ	dcerrtracepath オペランド	DCERR1.TRC DCERR2.TRC
メソッドトレース	• ファイル出力ディレクトリ • トレースファイルのサイズ	dcmethodtracepath オペランド	DCMTD1.TRC DCMTD2.TRC

注※1

指定したディレクトリがない場合，および指定したディレクトリに書き込み権限がない場合は，トレースファイルを出力できません。

注※2

指定したファイル出力先ディレクトリに出力ファイル名のファイルがない場合は，該当するディレクトリにトレースファイルを作成します。

ファイルへの書き込みは追加モードで実行されます。

2 つのトレースファイルはラウンドロビン方式で使用されます。指定したトレースファイルのサイズを超えた書き込みが発生すると，ファイルの切り替えを行い，次のファイルの先頭から書き込みを再開します。

複数の CUP のトレースファイルの出力先に同じディレクトリを指定すると，複数の CUP のトレース情報が混在し，トラブルシュートが困難になります。そのため，トレースファイル出力先ディレクトリはそれぞれの CUP で別々にすることをお勧めします。

また，1 つのトレースデータの出力サイズによっては，指定したトレースファイルのサイズよりファイルサイズが大きくなる場合があります。

(2) UAP トレース

UAP トレース情報は，TP1/Client/J 環境定義の dcuaptracepath オペランドで指定したディレクトリに，DCUAP1.TRC および DCUAP2.TRC というファイル名で出力されます。ファイルのサイズは，TP1/Client/J 環境定義の dcuaptracesize オペランドで指定します。

UAP トレースは，TP1/Client/J が提供するメソッドの開始時と終了時に次の情報を取得します。

- 日付
- 時刻
- 出入口種別
- 実行スレッド名
- 呼び出したメソッド名
- 発生した例外
- 指定した引数の情報
- 呼び出したメソッドに関係する情報
- 送受信したデータの内容

なお，送受信データ長が 60 バイトを超える場合は先頭の 60 バイトを取得します。

UAP トレースファイルの出力形式を次に示します。

```
yyyy/mm/dd hh:mm:ss.uuu Location = lll ThreadName = ttt
MethodName = nnnnnnnnnnnnnnnnnnn
Exception = eeeeeeeeeeeeeeeeeee
ADDRESS +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +a +b +c +d +e +f 0123456789abcdef
00000000 73 61 69 73 70 70 00 00 00 00 00 00 00 00 00 00 saispp..........
00000010 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ................
00000020 73 61 69 73 76 72 30 31 00 00 00 00 00 00 00 00 saisvr01........
:         :         :
```

- yyyy/mm/dd hh:mm:ss.uuu：UAP トレースを取得した日時
- lll：出入口種別
- ttt：実行スレッド名
- nnnnnnnnnnnnnnnnnnn：呼び出したメソッド名
- eeeeeeeeeeeeeeeeeee：発生した例外
- ADDRESS 以降の文字列：各種情報（指定した引数の情報，呼び出したメソッドに関係する情報，および送受信したデータの内容）

各種情報に出力する情報は，メソッドおよび取得するタイミングによって異なります。

(3) データトレース

データトレース情報は，TP1/Client/J 環境定義の dcdatatracepath オペランドで指定したディレクトリに，DCDAT1.TRC および DCDAT2.TRC というファイル名で出力されます。ファイルのサイズは，TP1/Client/J 環境定義の dcdatatracesize オペランドで指定します。1 トレースのデータサイズは，TP1/Client/J 環境定義の dcdatatracemaxsize オペランドで指定します。

データトレースは，CUP と TP1/Server 間の送受信メッセージの内容を取得します。

データトレースファイルへの出力形式を次に示します。

```
yyyy/mm/dd hh:mm:ss.uuu Event = eeee ThreadName = ttt
ADDRESS +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +a +b +c +d +e +f 0123456789abcdef
00000000 07 70 c0 00 00 00 00 5c 00 00 00 00 00 06 00 05 .p.....¥........
00000010 00 00 00 10 00 00 00 00 00 00 00 00 15 bf 00 00 ................
00000020 00 00 00 00 00 00 00 00 00 00 00 00 00 01 01 00 ................
:         :         :
```

- yyyy/mm/dd hh:mm:ss.uuu：データトレースを取得した日時
- eeee：送受信種別
- ttt：実行スレッド名
- ADDRESS 以降の文字列：データ

(4) エラートレース

エラートレース情報は，TP1/Client/J 環境定義の dcerrtracepath オペランドで指定したディレクトリに，DCERR1.TRC および DCERR2.TRC というファイル名で出力されます。ファイルのサイズは，TP1/Client/J 環境定義の dcerrtracesize オペランドで指定します。

エラートレースは，TP1/Client/J が検知した障害をメッセージ形式でファイルに記録します。メソッド実行中に障害が発生した場合，その原因は例外として報告されますが，それだけでは原因を特定できない場合があります。エラートレースを取得すると，詳細なエラー情報がファイルに出力され，原因調査の資料として使用できます。

エラートレースファイルの出力形式を次に示します。

```
(ttt)yyyy/mm/dd hh:mm:ss.uuu eeeeeeeeeeee
```

- ttt：実行スレッド名
- yyyy/mm/dd hh:mm:ss.uuu：エラートレースまたはメモリトレースを取得した日時
- eeeeeeeeeeee：メッセージ

表 A-7　エラートレース，メモリトレースで取得するメッセージ

メッセージの内容	意味
Invalid message received. method=aaaaaaaa	TP1/Server から不正なメッセージを受信しました。 aaaaaaaa：このメッセージを出力したメソッド名
Error reply received. inf=aaaaaaaa, method=bbbbbbbb	TP1/Server からエラー応答を受け取りました。 aaaaaaaa：受け取ったエラーコード bbbbbbbb：このメッセージを出力したメソッド名

メッセージの内容	意味
Exception occured. inf=aaaaaaaa, exception=bbbbbbbb(cc...cc), method=dddddddd	TP1Client クラス内で Java システムから例外を受け取りました。または TP1Client クラスから Java アプレット，Java アプリケーション，または Java サーブレットに例外を返しました。 aaaaaaaa：例外が発生したときの保守情報 bbbbbbbb：Java から受け取った例外名，または Java アプレット，Java アプリケーション，または Java サーブレットに返した例外名 cc...cc：例外の詳細メッセージ 詳細メッセージがある場合だけ出力します。 dddddddd：このメッセージを出力したメソッド名
Invalid data received. (aa...aa), method=bbbbbbbb	cltAssemReceive メソッドで相手システムから不正なデータを受信しました。 aa...aa：不正なデータ メッセージ長が不正な場合 receive message length=メッセージ長（10 進数） bbbbbbbb：このメッセージを出力したメソッド名
Receiving message was canceled. aaaaaaaa　(bb...bb) method=cccccccc	cltAssemReceive メソッドで相手システムから受信したメッセージを破棄しました。 aaaaaaaa：メッセージを破棄した原因 bb...bb：データの内容 受信バッファオーバーフローの場合 receive buffer overflowed. (receive buffer size=受信バッファの大きさ（10 進数）, receive message body length=受信メッセージ本体の長さ（10 進数)) cccccccc：このメッセージを出力したメソッド名
User data did not compress, group=aa...aa, service=bb...bb, reason= cc...cc	ユーザデータを圧縮しませんでした。データを圧縮しないでサービスを要求します。 aa...aa：要求先サービスグループ名 bb...bb：要求先サービス名 cc...cc：ユーザデータを圧縮しなかった理由 NO EFFECT：ユーザデータに対する圧縮効果がありません。 NOT SUPPORT VERSION：サービス要求先の TP1/Server がデータ圧縮機能をサポートしていないバージョンです。 「NO EFFECT」の場合，圧縮前より圧縮後のデータの方が大きくなります。そのため，同一の CUP でほかにもこのメッセージが出力されていないか確認し，CUP 単位でデータ圧縮機能を使用するかどうかを再度検討してください。 「NOT SUPPORT VERSION」の場合，サービス要求先の TP1/Server が，データ圧縮機能を使用できるバージョンかどうかを確認してください。

(5)　メソッドトレース

メソッドトレース情報は，TP1/Client/J 環境定義の dcmethodtracepath オペランドで指定したディレクトリに，DCMTD1.TRC および DCMTD2.TRC というファイル名で出力されます。ファイルのサイズは，TP1/Client/J 環境定義の dcmethodtracesize オペランドで指定します。

TP1Client クラス内では多くの内部メソッドが実行されます。メソッドトレースは，これらの内部メソッドの実行順序や実行時刻をファイルに出力します。Java 環境では障害時に取得できる情報が少ないため，どのような内部処理の延長で障害が発生したかを原因調査の材料にします。

メソッドトレースファイルの出力形式を次に示します。

```
yyyy/mm/dd hh:mm:ss.uuu Location = lll ThreadName = ttt
MethodName = nnnnnnnnnnnnnnnnnnn
ADDRESS +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +a +b +c +d +e +f 0123456789abcdef
00000000 00 00 00 00 c7 0b 49 17 00 00 00 00 00 06 00 05 ......|.........
```

- yyyy/mm/dd hh:mm:ss.uuu：メソッドトレースを取得した日時
- lll：出入口種別
- ttt：実行スレッド名
- nnnnnnnnnnnnnnnnnnn：呼び出したメソッド名
- ADDRESS 以降の文字列：メソッドトレース情報

(6) デバッグトレース

デバッグトレース情報は，通常 TP1/Client/J 内のメモリバッファに取得され，例外を返した場合だけデバッグトレースファイルに出力されます。デバッグトレースファイルは，Java VM を実行するユーザのホームディレクトリ下の TP1clientJ ディレクトリに，dcCltXXXXXXXXXXXX.dmp（XXXXXXXXXXXX：タイムスタンプ）というファイル名で出力します。

デバッグトレースファイルを出力する時点で，デバッグトレースファイルの総数が，TP1/Client/J 環境定義の dccltdbgtrcfilecount オペランドで指定したファイル数を超える場合は，更新日時の最も古いファイルを削除します。デバッグトレースファイルの総数が，（dccltdbgtrcfilecount オペランドで指定したファイル数－1）となるまで既存のデバッグトレースファイルを削除します。

CUP を Java アプレットとして動作させる場合，古いデバッグトレースファイルの削除に失敗した場合など，デバッグトレースをファイルに出力できないときは，標準出力に出力します。標準出力に出力された場合，ブラウザなどの機能で Java コンソールを開けば，デバッグトレース情報を参照できます。

デバッグトレースの出力形式を次に示します。

```
(ttt)yyyy/mm/dd hh:mm:ss.uuu MethodName = nnnnnnnnnnnnnnnnnnn(ii) Location = lll
ADDRESS +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +a +b +c +d +e +f 0123456789abcdef
00000000 00 00 00 00 c7 0b 49 17 00 00 00 00 00 06 00 05 ......|.........
```

- ttt：スレッド ID
- yyyy/mm/dd hh:mm:ss.uuu：デバッグトレースを取得した日時
- nnnnnnnnnnnnnnnnnnn：呼び出したメソッド名
- ii：インスタンス ID

- lll：出入口種別
- ADDRESS 以降の文字列：デバッグトレース情報

(7) 性能解析トレース

性能解析トレース（PRF トレース）情報とは，サービスプラットフォームのトレース情報です。性能解析トレースを取得すると，UAP 実行時の，一連の処理の流れおよび処理に掛かった時間を知ることができ，性能解析に必要な情報を採取できます。また，障害が発生したときに，処理がどこまで到達したかを知ることができます。

性能解析トレースの概要および使用方法の詳細については，マニュアル「アプリケーションサーバ 機能解説 保守／移行編」の「7. 性能解析トレースを使用した性能解析」を参照してください。ここでは，次に示す事項について説明します。

- サービスプラットフォームでの性能解析トレースの出力
- サービスプラットフォームと OpenTP1 とのトレースの照合
- 性能解析トレースの取得情報
- 性能解析トレースのトレース取得ポイント
- UAP トレースに出力される性能解析トレースの識別情報

(a) サービスプラットフォームでの性能解析トレースの出力

TP1/Client/J では，TP1/Client/J をサービスプラットフォーム上で動作させた場合に，サービスプラットフォーム上で性能解析トレースを出力できます。性能解析トレースは，TP1/Client/J のトレース取得ポイントで取得されます。性能解析トレースを取得する場合は，TP1/Client/J 環境定義に dccltprftrace=Y を指定します。

サービスプラットフォーム上で性能解析トレースを出力する場合，TP1/Client/J が取得する情報の分だけ，出力される性能解析トレースのサイズが大きくなることを考慮して，性能解析トレースのサイズを指定してください。

(b) サービスプラットフォームと OpenTP1 とのトレースの照合

TP1/Server に対して，ネームサービスを使用した RPC，およびスケジューラダイレクト機能を使用した RPC を行う場合は，スケジューラに送信する RPC 電文中に，TP1/Client/J のインスタンスごとにほぼ一意となる識別情報（IP アドレスなど）を付加できます。付加した情報は，OpenTP1 の性能検証用トレースで出力します。この OpenTP1 の性能検証用トレースと，サービスプラットフォームの性能解析トレースとを照合することによって，サービスプラットフォームと OpenTP1 との間の，一連の処理の流れを知ることができます。また，バージョン 07-02 以降の TP1/Server に対して，リモート API 機能を使用した RPC を行う場合は，スケジューラに送信する RPC 電文中への識別情報（IP アドレスなど）を付加できます。

OpenTP1 の性能検証用トレースに識別情報を付加する場合は，TP1/Client/J 環境定義に dccltprfinfosend=Y を指定します。OpenTP1 の性能検証用トレースの詳細については，マニュアル「OpenTP1 解説」を参照してください。

(c) 性能解析トレースの取得情報

TP1/Client/J で取得した情報は，性能解析トレースの付加情報部分に出力されます。性能解析トレースに取得する情報，およびトレースの出力例を次に示します。

性能解析トレースに取得する情報

ノード ID：_Jaa（4 バイトの英数字）

aa：ランダムな 2 文字の英数字（数字（0～9）またはアルファベット（A～Z，a～z））

ルート通信通番部分：bbbbbbbb（4 バイトの 16 進数字）

bbbbbbbb：IP アドレス

RPC 通信通番：ccccdddd（4 バイトの 16 進数字）

cccc：ランダムな 2 バイトの 16 進数字

dddd：通信通番

サービスプラットフォームで性能解析トレースを出力する場合の出力例

```
...     OPT       ASCII
...     16進表示  _Jaa/0xbbbbbbbb/0xccccdddd...
```

OpenTP1 の性能検証用トレースに識別情報を付加する場合の出力例

```
PRF: Rec Node: smpl Run-ID: 0x416f809d Process: 2612 Trace: 11
Event: 0x2002 Time: 2004/10/15 17:17:07 195.000.000 Server-name: _scd
Rc: ********** Client:  - 0xccccdddd     Server: **** Root: _Jaa - 0xbbbbbbbb
Svc-Grp: tst_svg                        Svc: echo
Trn: *
```

(d) 性能解析トレースのトレース取得ポイント

TP1/Client/J の性能解析トレースの，トレース取得ポイントの詳細を，処理ごとに説明します。

●rap リスナーとのコネクション確立，切断処理

rap リスナーとのコネクション確立，切断処理でのイベント ID，トレース取得ポイント，付加情報および性能解析トレース取得レベルについて，次の表に示します。

表 A-8 rap リスナーとのコネクション確立，切断処理でのトレース取得ポイントの詳細

イベント ID	図中の番号※1	トレース取得ポイント	付加情報※2	レベル
0x9180	1	rap リスナーへのコネクション確立処理前	要求先の情報（ホスト名，ポート番）	A
0x9181	2	rap リスナーへのコネクション確立処理後（エラー時だけ取得）	内部リターンコード	A

イベント ID	図中の番号※1	トレース取得ポイント	付加情報※2	レベル
0x9182※3	3	rap リスナーへの接続要求データの送受信後	内部リターンコード	A
0x9183	4	rap リスナーへの切断要求データの送受信前	－	A
0x9184	5	rap リスナーへの切断要求データの送受信後	内部リターンコード	A

（凡例）

A：「標準」であることを示します。

－：該当しません。

注※1

「図 A-6　rap リスナーとのコネクション確立，切断処理でのトレース取得ポイント」中の番号と対応しています。

注※2

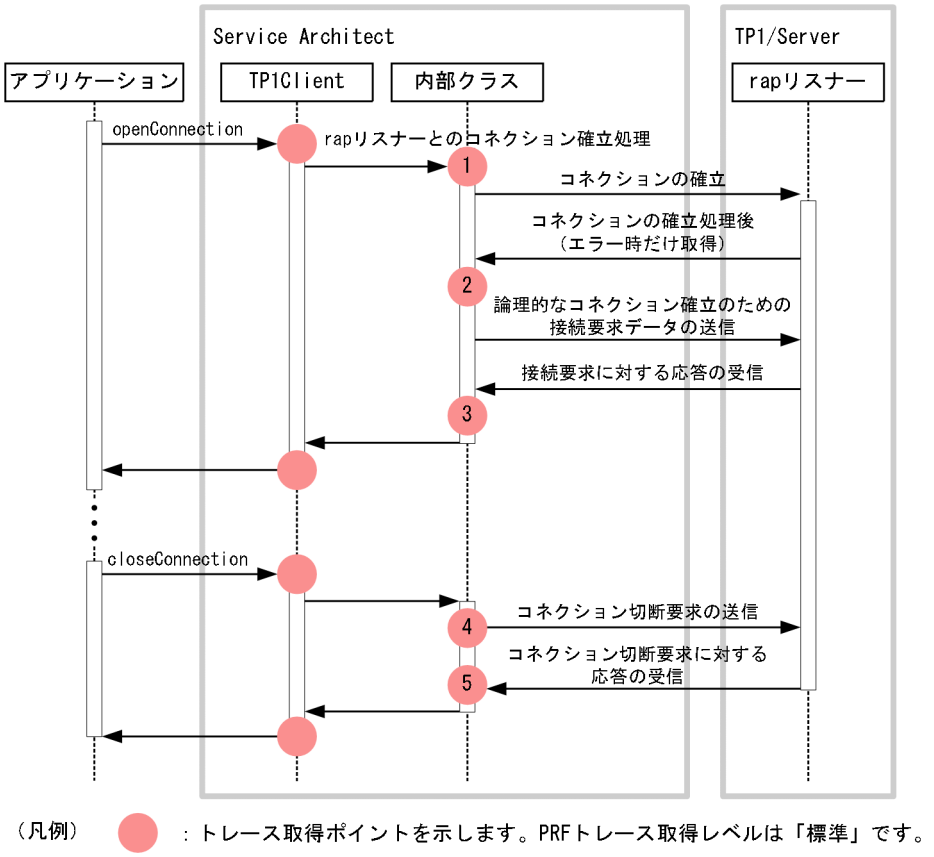
サービスプラットフォームの性能解析トレースに付加する情報です。

注※3

コネクション確立処理がエラーとなってイベント ID0x9181 が取得された場合，このイベント ID のトレースは取得されません。

rap リスナーとのコネクション確立，切断処理でのトレース取得ポイントを次の図に示します。

図 A-6　rap リスナーとのコネクション確立，切断処理でのトレース取得ポイント



（凡例）　：トレース取得ポイントを示します。PRFトレース取得レベルは「標準」です。

●リモート API 機能での API 代理実行要求

リモート API 機能での API 代理実行要求でのイベント ID，トレース取得ポイント，付加情報および性能解析トレース取得レベルについて，次の表に示します。

表 A-9　リモート API 機能での API 代理実行要求でのトレース取得ポイントの詳細

イベント ID	図中の番号※1	トレース取得ポイント	付加情報※2	レベル
0x9190	1	dc_rpc_call の代理実行の要求処理前	－	A
0x9191	2	dc_rpc_call の代理実行の要求処理後	内部リターンコード	A
0x9192	1	dc_trn_begin の代理実行の要求処理前	－	A
0x9193	2	dc_trn_begin の代理実行の要求処理後	内部リターンコード	A
0x9194	1	dc_trn_chained_commit の代理実行の要求処理前	－	A
0x9195	2	dc_trn_chained_commit の代理実行の要求処理後	内部リターンコード	A
0x9196	1	dc_trn_chained_rollback の代理実行の要求処理前	－	A
0x9197	2	dc_trn_chained_rollback の代理実行の要求処理後	内部リターンコード	A
0x9198	1	dc_trn_unchained_commit の代理実行の要求処理前	－	A
0x9199	2	dc_trn_unchained_commit の代理実行の要求処理後	内部リターンコード	A
0x919A	1	dc_trn_unchained_rollback の代理実行の要求処理前	－	A
0x919B	2	dc_trn_unchained_rollback の代理実行の要求処理後	内部リターンコード	A

（凡例）

A：「標準」であることを示します。

－：該当しません。

注※1

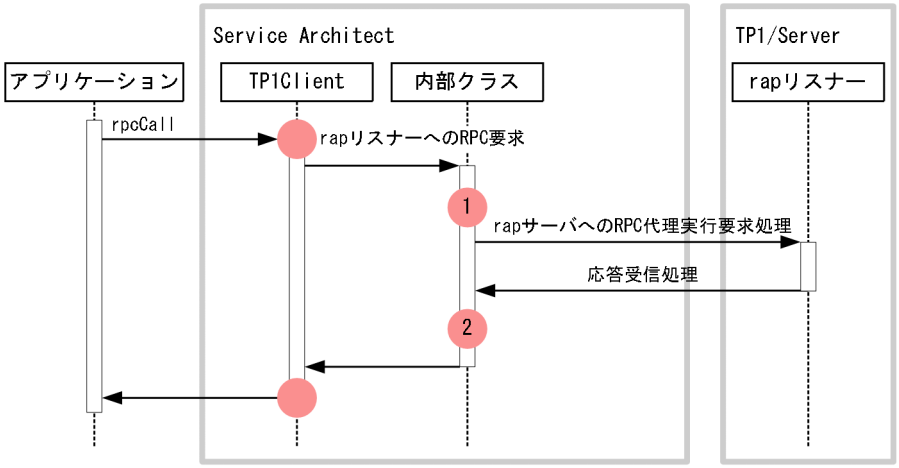
「図 A-7　リモート API 機能での API 代理実行要求でのトレース取得ポイント」中の番号と対応しています。

注※2

サービスプラットフォームの性能解析トレースに付加する情報です。

リモート API 機能での API 代理実行要求でのトレース取得ポイントを次の図に示します。

図 A-7　リモート API 機能での API 代理実行要求でのトレース取得ポイント



（凡例）　：トレース取得ポイントを示します。PRFトレース取得レベルは「標準」です。

●スケジュールサーバへの RPC

スケジュールサーバへの RPC でのイベント ID，トレース取得ポイント，付加情報および性能解析トレース取得レベルについて，次の表に示します。

表 A-10　スケジュールサーバへの RPC でのトレース取得ポイントの詳細

イベント ID	図中の番号※1	トレース取得ポイント	付加情報※2	レベル
0x91C0	1	スケジュールサーバへのコネクション確立処理前	サービス要求先ホストの情報（ホスト名，ポート番号）	A
0x91C1	2	スケジュールサーバへの RPC のサービス要求処理の実行後	内部リターンコード	A
0x91C2※3	3	SPP またはスケジュールサーバからの応答データを受け取るコネクションの確立要求を受け付け後	内部リターンコード※4	A
0x91C3	4	ネームサーバへのコネクション確立処理前	サービス要求先ホストの情報（ホスト名，ポート番号）	B
0x91C4	5	ネームサーバへのサービス情報の問い合わせ処理の実行後	内部リターンコード	B
0x91C5※3	6	ネームサーバからの応答データを受け取るコネクションの確立要求を受け付け後	内部リターンコード※5	B

（凡例）

A：「標準」であることを示します。

B：「詳細」であることを示します。

注※1

「図 A-8　スケジュールサーバへの RPC 要求でのトレース取得ポイント」および「図 A-9　ネームサーバへのサービス情報の問い合わせでのトレース取得ポイント」中の番号と対応しています。

注※2

サービスプラットフォームの性能解析トレースに付加する情報です。

注※3

要求データの送信が正常終了した場合に，このイベント ID のトレースは取得されます。

注※4

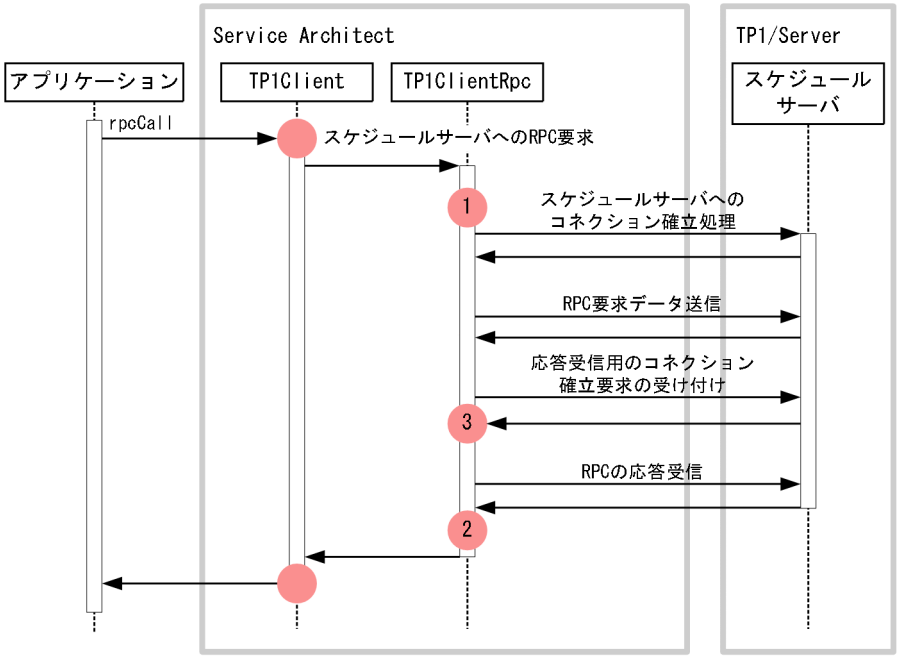
SPP またはスケジュールサーバからの応答データを受け取るコネクションの確立要求の受け付けが正常終了した場合は，接続元情報（IP アドレス，ポート番号）も付加します。

注※5

ネームサーバからの応答データを受け取るコネクションの確立要求の受け付けが正常終了した場合は，接続元情報（IP アドレス，ポート番号）も付加します。

スケジュールサーバへの RPC 要求でのトレース取得ポイントを次の図に示します。

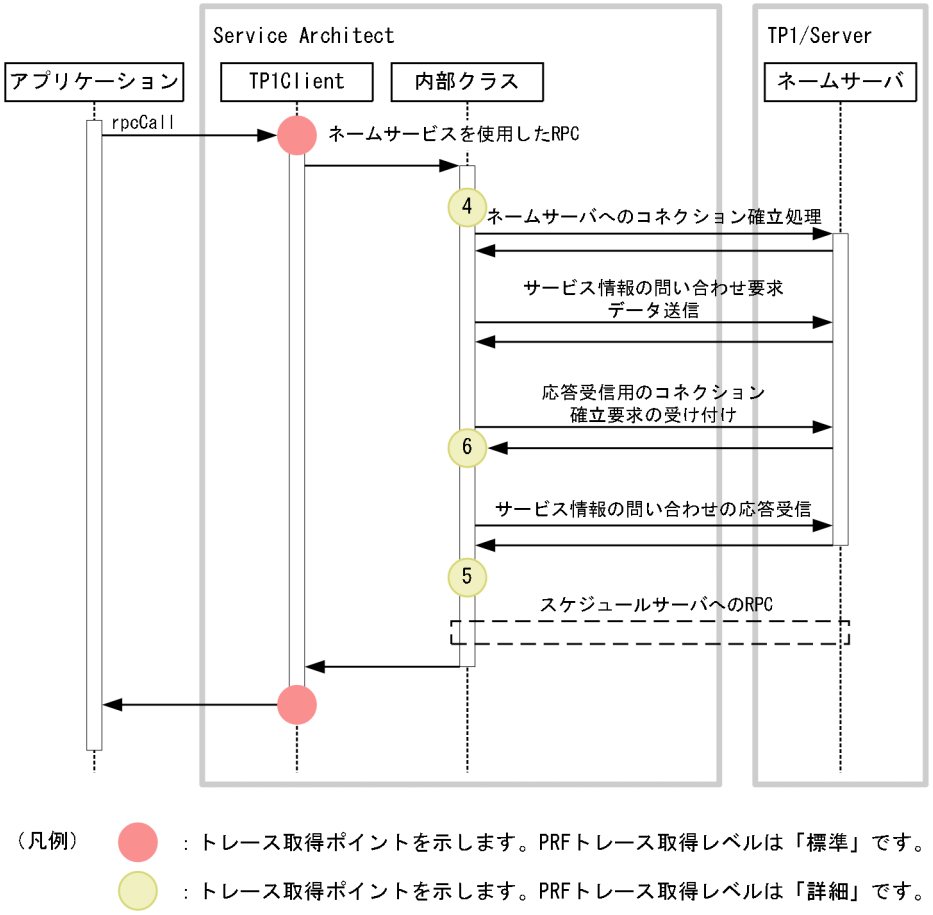
図 A-8　スケジュールサーバへの RPC 要求でのトレース取得ポイント



（凡例）　：トレース取得ポイントを示します。PRFトレース取得レベルは「標準」です。

ネームサーバへのサービス情報の問い合わせでのトレース取得ポイントを次の図に示します。

図 A-9　ネームサーバへのサービス情報の問い合わせでのトレース取得ポイント



●API 実行処理

API 実行処理でのイベント ID，トレース取得ポイント，付加情報および性能解析トレース取得レベルについて，次の表に示します。

表 A-11　API 実行処理でのトレース取得ポイントの詳細

イベント ID	トレース取得ポイント	付加情報※1	レベル
0x91D0	openConnection メソッドの入口	−	A
0x91D1	openConnection メソッドの出口	内部リターンコード	A
0x91D2	closeConnection メソッドの入口	−	A
0x91D3	closeConnection メソッドの出口	内部リターンコード	A
0x91D4	rpcCall メソッドの入口	inlen	A
0x91D5	rpcCall メソッドの出口	outlen，内部リターンコード	A
0x91D6	rpcCallTo メソッドの入口	inlen	A
0x91D7	rpcCallTo メソッドの出口	outlen，内部リターンコード	A
0x91D8	trnBegin メソッドの入口	−	A

イベント ID	トレース取得ポイント	付加情報※1	レベル
0x91D9	trnBegin メソッドの出口	内部リターンコード※2	A
0x91DA	trnChainedCommit メソッドの入口	−	A
0x91DB	trnChainedCommit メソッドの出口	内部リターンコード※2	A
0x91DC	trnChainedRollback メソッドの入口	−	A
0x91DD	trnChainedRollback メソッドの出口	内部リターンコード※2	A
0x91DE	trnUnchainedCommit メソッドの入口	−	A
0x91DF	trnUnchainedCommit メソッドの出口	内部リターンコード	A
0x91E0	trnUnchainedRollback メソッドの入口	−	A
0x91E1	trnUnchainedRollback メソッドの出口	内部リターンコード	A

（凡例）

A：「標準」であることを示します。

−：該当しません。

注※1

サービスプラットフォームの性能解析トレースに付加する情報です。

注※2

トランザクションの開始時は，トランザクショングローバル識別子，およびトランザクションブランチ識別子も付加します。

●XA リソースサービス機能

XA リソースサービス機能でのイベント ID，トレース取得ポイント，付加情報および PRF トレース取得レベルについて，次の表に示します。

表 A–12　XA リソースサービス機能でのトレース取得ポイントの詳細

イベント ID	トレース取得ポイント	付加情報	レベル
0x919C	TP1/Client/J からの XA トランザクションの start 要求の代理実行要求処理前	内部コード	A
0x919D	TP1/Client/J からの XA トランザクションの start 要求の代理実行要求処理後	内部リターンコード	A
0x919E	TP1/Client/J からの XA トランザクションの end 要求の代理実行要求処理前	−	A
0x919F	TP1/Client/J からの XA トランザクションの end 要求の代理実行要求処理後	内部リターンコード	A
0x91A0	TP1/Client/J からの XA トランザクションの prepare 要求の代理実行要求処理前	−	A
0x91A1	TP1/Client/J からの XA トランザクションの prepare 要求の代理実行要求処理後	内部リターンコード	A
0x91A2	TP1/Client/J からの XA トランザクションの commit 要求の代理実行要求処理前	−	A

イベント ID	トレース取得ポイント	付加情報	レベル
0x91A3	TP1/Client/J からの XA トランザクションの commit 要求の代理実行要求処理後	内部リターンコード	A
0x91A4	TP1/Client/J からの XA トランザクションの rollback 要求の代理実行要求処理前	−	A
0x91A5	TP1/Client/J からの XA トランザクションの rollback 要求の代理実行要求処理後	内部リターンコード	A
0x91A6	TP1/Client/J からの XA トランザクションの forget 要求の代理実行要求処理前	−	A
0x91A7	TP1/Client/J からの XA トランザクションの forget 要求の代理実行要求処理後	内部リターンコード	A
0x91A8	TP1/Client/J からの XA トランザクションの recover 要求の代理実行要求処理前	−	A
0x91A9	TP1/Client/J からの XA トランザクションの recover 要求の代理実行要求処理後	内部リターンコード	A
0x91AA	TP1/Client/J からの XA トランザクションの RPC 要求の代理実行要求処理前	−	A
0x91AB	TP1/Client/J からの XA トランザクションの RPC 要求の代理実行要求処理後	内部リターンコード	A
0x91E2	TP1 Connector からの XA トランザクションの start 要求受け付け直後	内部コード	A
0x91E3	TP1 Connector からの XA トランザクションの start 要求応答直前	内部リターンコード	A
0x91E4	TP1 Connector からの XA トランザクションの end 要求受け付け直後	内部コード	A
0x91E5	TP1 Connector からの XA トランザクションの end 要求応答直前	内部リターンコード	A
0x91E6	TP1 Connector からの XA トランザクションの prepare 要求受け付け直後	内部コード	A
0x91E7	TP1 Connector からの XA トランザクションの prepare 要求応答直前	内部リターンコード	A
0x91E8	TP1 Connector からの XA トランザクションの commit 要求受け付け直後	内部コード	A
0x91E9	TP1 Connector からの XA トランザクションの commit 要求応答直前	内部リターンコード	A
0x91EA	TP1 Connector からの XA トランザクションの rollback 要求受け付け直後	内部コード	A
0x91EB	TP1 Connector からの XA トランザクションの rollback 要求応答直前	内部リターンコード	A

イベント ID	トレース取得ポイント	付加情報	レベル
0x91EC	TP1 Connector からの XA トランザクションの forget 要求受け付け直後	内部コード	A
0x91ED	TP1 Connector からの XA トランザクションの forget 要求応答直前	内部リターンコード	A
0x91EE	TP1 Connector からの XA トランザクションの recover 要求受け付け直後	内部コード	A
0x91EF	TP1 Connector からの XA トランザクションの recover 要求応答直前	内部リターンコード	A
0x91F0	TP1 Connector からの XA トランザクションの RPC 要求受け付け直後	内部コード	A
0x91F1	TP1 Connector からの XA トランザクションの RPC 要求応答直前	内部リターンコード	A

（凡例）

A：「標準」であることを示します。

－：該当しません。

(e) UAP トレースに出力される性能解析トレースの識別情報

TP1/Client/J では，TP1/Server やサービスプラットフォームの性能解析トレースの情報と，TP1/Client/J のトレースの情報を結び付けるために，UAP トレースの出口情報の末尾に，性能解析トレースの識別情報を出力します。

UAP トレースに性能解析トレースの識別情報を出力するメソッドを次に示します。

- openConnection()
- openConnection(String host, int port)
- closeConnection()
- rpcCall(String group, String service, byte[] in_data, int[] in_len, byte[] out_data, int[] out_len, int flags)
- rpcCall(String group, String service, byte[] in_data, byte[] out_data, int flags)
- rpcCallTo(DCRpcBindTbl direction, String group, String service, byte[] in_data, int[] in_len, byte[] out_data, int[] out_len, int flags)
- trnBegin()
- trnChainedCommit()
- trnUnchainedCommit()
- trnChainedRollback()
- trnUnchainedRollback()

なお，性能解析トレースの識別情報は，TP1/Client/J 環境定義の dccltprfinfosend オペランドの指定値に関係なく出力されます。

性能解析トレースの識別情報を出力した，UAP トレースの出力例を次に示します。太字で示す部分が，性能解析トレースの識別情報です。

```
2008/09/01 16:37:24.698  Location = Out  ThreadName = main
MethodName = TP1Client.rpcCall
Exception  =
ADDRESS   +0 +1 +2 +3 +4 +5 +6 +7 +8 +9 +a +b +c +d +e +f  0123456789abcdef
00000000  00 00 00 1c 53 75 7a 75 6b 69 00 00 00 00 00 00  ....Suzuki......
00000010  00 00 00 00 00 00 00 00 00 00 00 00 51 00 00 00  ............Q...
00000020  5f 4a 30 6d 2f 30 78 30 61 64 31 30 66 37 63 2f  _J0m/0x0ad10f7c/
00000030  30 78 32 62 35 64 30 30 30 31                      0x2b5d0001
```

付録 B　データ変換機能の処理概要

ここでは，サービスプラットフォームが提供するデータ変換機能の処理概要について説明します。

データ変換機能に関する詳細については，マニュアル「サービスプラットフォーム 開発ガイド 基本開発編」の「4.　電文フォーマットの作成」および「7.　データ変換の定義」を参照してください。

付録 B.1　データ変換に必要な定義ファイルと作成方法

データ変換に必要な定義ファイルと，作成方法を示します。

データ変換の実行に必要な定義ファイル一覧を次の表に示します。

表 B-1　データ変換の実行に必要な定義ファイル

ファイル名	拡張子	用途	作成方法
XML スキーマ	xsd	XML 形式の構造を持つデータフォーマットを定義するファイルです。	Eclipse やエディタなどの開発ツール，または製品提供ツールで作成します。
バイナリフォーマット定義ファイル	fdx	バイナリ形式（XML 形式以外）のデータフォーマットを定義するファイルです。	バイナリフォーマット定義エディタで生成します。 COBOL の構文から作成することもできます。
マッピング定義ファイル	mdo	データ変換処理の変換パターンを定義するファイルです。	マッピング定義エディタで作成します。 マッピングの定義は Excel で生成した表形式の XML ファイルから自動生成することもできます。
データ変換定義ファイル	xsl	データ変換処理の実行形式のファイルです。	マッピング定義エディタから生成します。エディタ上のマッピングの定義に対応した内容になります。

注意事項

バイナリフォーマット定義ファイル，およびマッピング定義ファイルを，開発環境が提供するエディタ以外で変更した場合，動作を保証しません。また，データ変換定義ファイルを開いて変更した場合，動作を保証しません。

以降で，各ファイルの特長を説明します。

(1)　XML スキーマ

XML 形式の構造を持つデータのフォーマットを定義するファイルです。

XML スキーマの作成方法は，使用するデータや，ユーザ定義受付およびサービスアダプタの種類によって異なります。

(a) ユーザが定義するデータ構造の場合

データの構造に合わせて，Eclipse やエディタなどの開発ツールで作成します。JSON 形式のデータを扱うときは，JSON 形式データのファイルから cscjson2xsd コマンドを利用して作成します。

(b) SOAP 受付および SOAP アダプタの場合

WSDL ファイルから自動的に生成されます。

(c) そのほかのユーザ定義受付およびサービスアダプタの場合

ユーザ定義受付およびサービスアダプタの場合，データの構造に合わせて，Eclipse やエディタなどの開発ツールで作成します。なお，ユーザ定義受付およびサービスアダプタによっては，サービスプラットフォームで XML スキーマを提供するケースや，テンプレートファイルを提供するケースがあります。

ユーザ定義受付の定義については，マニュアル「サービスプラットフォーム 開発ガイド 受付・アダプタ定義編」の「2. ユーザ定義受付の定義」を参照してください。

サービスアダプタの定義については，マニュアル「サービスプラットフォーム 開発ガイド 受付・アダプタ定義編」の「3. サービスアダプタの定義」を参照してください。

(2) バイナリフォーマット定義ファイル

バイナリ形式（XML 形式以外）のデータフォーマットを定義するファイルです。

バイナリフォーマット定義ファイルは，開発環境のバイナリフォーマット定義エディタで作成します。また，開発環境の COBOL2FDX コンバータを利用して COBOL 登録集原文ファイルから生成することもできます。

(3) マッピング定義ファイル

変換元入力データから変換先出力データへのデータ変換パターンを定義するファイルです。

マッピング定義ファイルは，開発環境のマッピング定義エディタで作成します。マッピング情報の作成には，入力データおよび出力データでそれぞれに対応するデータフォーマットのファイル（XML スキーマまたはバイナリフォーマット定義ファイル）が必要です。また，Excel を利用して生成した定義ファイル（表形式の XML ファイル）をマッピング情報としてインポートできます。

(4) データ変換定義ファイル

データ変換処理の実行形式のファイルです。

データ変換定義ファイルは，マッピング定義ファイルを基に，開発環境のマッピング定義エディタで生成します。

データ変換定義ファイルは，データ変換機能を次の用途で利用する場合に必要となります。

- ファイル操作アダプタの変換オペレーション
- データ変換 API
- csctransform コマンド

付録 B.2　データ変換処理の概要

データ変換定義ファイルの内容に従ってデータを変換することをサービスプラットフォームのデータ変換処理と呼びます。以降，サービスプラットフォームのデータ変換処理をデータ変換処理と表記します。

データ変換処理では，変換元および変換先のデータ形式によって中間形式化（DOM 化）することで変換を実現しています。

ここでは，次のデータ変換処理の流れを説明します。

- XML 形式データの変換
- バイナリ形式データの変換
- JSON 形式データの変換
- gRPC 電文（Protocol Buffers 形式）の変換

(1)　XML 形式データの変換

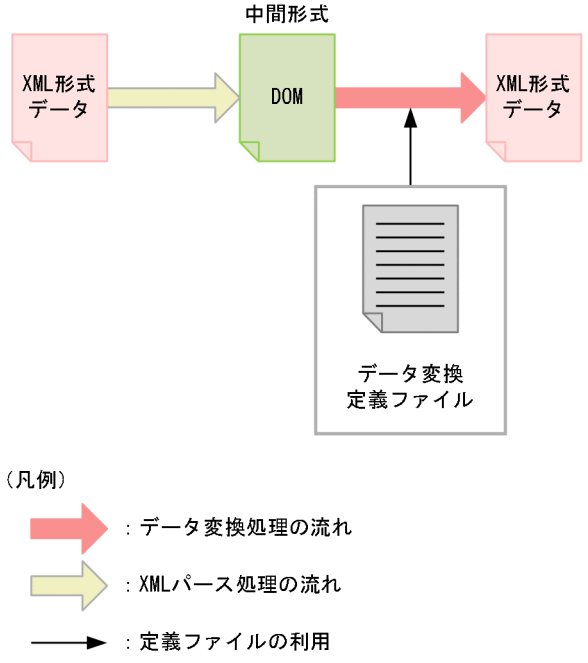
XML 形式データの変換では，XML 形式データを中間形式（DOM）に変換します。この変換を XML パース処理と呼びます。

XML パース処理で中間形式化したデータに対して，データ変換定義ファイルの内容に沿った変換処理が行われます。

なお，XML パース処理は，分岐アクティビティでの条件評価処理など，XML 形式データを解析するタイミングでも行われます。

XML 形式データの変換の流れを次の図に示します。

図 B-1　XML 形式データから XML 形式データへのデータ変換



(2)　バイナリ形式データの変換

バイナリ形式データの変換では，バイナリ形式データと中間形式（DOM）を相互に変換します。この変換をバイナリパース処理と呼びます。

バイナリ形式データを中間形式に変換する際，または中間形式からバイナリ形式データに変換する際に，バイナリフォーマット定義ファイルを使用します。

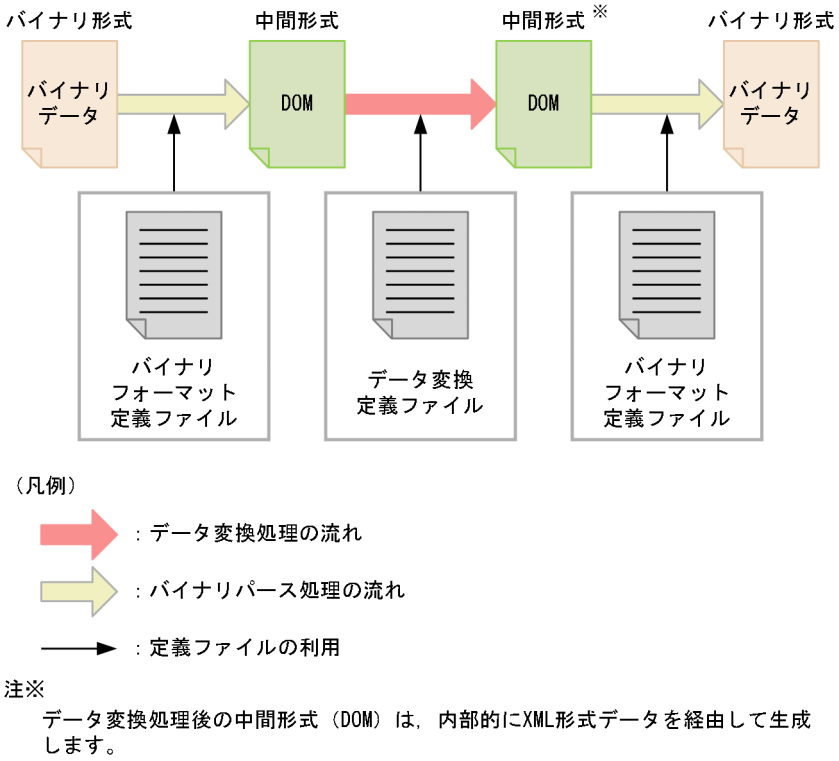
文字列データの型（文字列，整数，実数，固定小数部数値，日付時刻）については，変換時にバイナリフォーマット定義ファイルに指定した文字コードで文字コード変換が行われます。

バイナリパース処理で中間形式化したデータに対して，データ変換定義ファイルの内容に沿った変換処理が行われます。

なお，バイナリパース処理は，分岐アクティビティでの条件評価処理など，バイナリ形式データを解析するタイミングでも行われます。

バイナリ形式データの変換の流れを図に示します。

図 B-2　バイナリ形式データからバイナリ形式データへのデータ変換



注※
データ変換処理後の中間形式（DOM）は，内部的にXML形式データを経由して生成します。

注意事項

中間形式（DOM）の文字コードは Unicode になります。このため，Code Converter のユーザマッピングテーブルを使用する場合は，バイナリフォーマット定義ファイルに指定した文字コードと，Unicode（UCS-2）とのユーザマッピングテーブルを定義する必要があります。

例えば，MS932 のバイナリ形式のデータから，UTF-8 のバイナリ形式のデータへとデータ変換する場合は，MS932→Unicode（UCS-2）のユーザマッピングテーブルを定義してください。

(3)　JSON 形式データの変換

JSON 形式データは，HTTP 受付，HTTP アダプタ，Kafka 受付，および Kafka アダプタの処理で，XML 形式データとの変換が行われます。

JSON 形式データと XML 形式データの変換では，中間形式（DOM）化をしないため，実行環境のビジネスプロセス上では XML 形式データとして扱います。

詳細については，次の個所を参照してください。

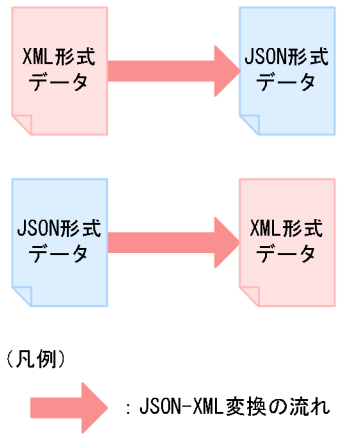
- 「2.15.10　HTTP 受付の JSON-XML 変換機能」
- 「2.16.15　HTTP アダプタの JSON-XML 変換機能」
- 「2.12.2　Kafka 受付の JSON-XML 変換機能」
- 「2.12.7　Kafka アダプタの JSON-XML 変換機能」

JSON 形式データと XML 形式データの変換は，サービスプラットフォームの変換規則に従います。JSON の形式に従っていない場合や，変換規則に従っていない場合はエラーとなります。

具体的な変換規則については，マニュアル「サービスプラットフォーム 開発ガイド 基本開発編」の「11.2 JSON-XML 変換の規則」を参照してください。

JSON 形式データと XML 形式データの変換の流れを次の図に示します。

図 B-3　JSON 形式データと XML 形式データの変換



(4)　gRPC 電文（Protocol Buffers 形式）の変換

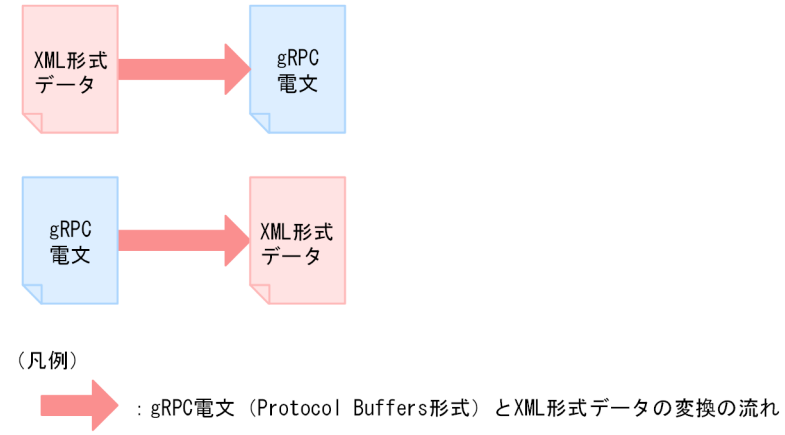
gRPC 電文（Protocol Buffers 形式）の変換では，gRPC 電文と XML 形式データとの変換が行われます。

詳細については，次の個所を参照してください。

- 「2.13.2　gRPC 電文（Protocol Buffers 形式）と XML 形式の変換」

gRPC 電文（Protocol Buffers 形式）と XML 形式データの変換の流れを次の図に示します。

図 B-4　gRPC 電文（Protocol Buffers 形式）と XML 形式データの変換



付録 B.3　データ変換機能のエミュレーション

サービスプラットフォームでは，HCSC サーバおよびアプリケーションサーバを構築しなくても，データ変換を実行できるエミュレーション機能を提供しています。

ここでは，エミュレーション機能として提供している次のコマンドについて説明します。

- csctransform コマンド
- cscbinaryparse コマンド
- cscjsonxmlconv コマンド
- cscjsonxmlconvadv コマンド

(1)　csctransform コマンド

サービスプラットフォームのデータ変換処理をエミュレーションするコマンドです。

ビジネスプロセスのデータ変換アクティビティおよび，データ変換 API の処理と同等の機能を実現します。入出力データがバイナリ形式データの場合，バイナリパース処理も同時に行います。

csctransform コマンドの使い方については，マニュアル「サービスプラットフォーム リファレンス」の「csctransform（データ変換のテスト）」を参照してください。

(2)　cscbinaryparse コマンド

バイナリパース処理をエミュレーションするコマンドです。

バイナリ形式データと，中間形式のデータとの相互変換を行います。データ変換処理内で行われるバイナリパース処理と同等の機能を実現します。

cscbinaryparse コマンドの使い方については，マニュアル「サービスプラットフォーム リファレンス」の「cscbinaryparse（DOM とバイナリデータの相互変換）」を参照してください。

(3)　cscjsonxmlconv コマンド

JSON 形式のデータと XML 形式のデータの変換をエミュレーションするコマンドです。

次の受付やアダプタの JSON-XML 変換処理，および JSON-XML 変換 API と同等の機能を実現します。

- HTTP 受付
- HTTP アダプタ
- Kafka 受付
- Kafka アダプタ

cscjsonxmlconv コマンドの使い方については，マニュアル「サービスプラットフォーム リファレンス」の「cscjsonxmlconv（JSON 形式データと XML 形式データの相互変換（標準モード））」を参照してください。

(4) cscjsonxmlconvadv コマンド

JSON 形式のデータと XML 形式のデータの変換をエミュレーションするコマンドです。

HTTP 受付の JSON-XML 変換処理，および JSON-XML 変換 API と同等の機能を実現します。

cscjsonxmlconvadv コマンドの使い方については，マニュアル「サービスプラットフォーム リファレンス」の「cscjsonxmlconvadv（JSON 形式データと XML 形式データの相互変換（拡張モード））」を参照してください。

付録 C　IPv6 対応

ここでは，IPv6 のサポート範囲と，このサポートに伴って変更された機能について説明します。

付録 C.1　サポート範囲

IPv6 のサポート範囲を，次に示します。

(1)　前提条件

HCSC サーバ稼働マシンは，IPv4/IPv6 デュアルスタック環境でだけ対応します。

(2)　データベース接続

IPv6 のデータベース接続はできません。

(3)　SOAP/HTTP のインバウンド

SOAP 受付と HTTP 受付は，HTTP Server を前提とします。

(4)　受付・アダプタ

受付，およびアダプタのサポート可否について，次の表に示します。

表 C-1　受付・アダプタのサポート可否

項番	種別	受付	アダプタ	備考
1	SOAP	○	○	SOAP1.1 モード，SOAP1.1/1.2 併用モード共にサポートします。
2	SessionBean	×	×	−
3	MDB（WS-R）	×	×	−
4	MDB（DB キュー）	×	×	−
5	DB アダプタ	−	×	−
6	FTP	○	○	−
7	HTTP	○	○	−
8	Message Queue	×	×	−
9	Object Access	×	×	−
10	TP1	×	×	−
11	メール	−	×	−

項番	種別	受付	アダプタ	備考
12	SFTP アダプタ	−	○	−
13	Kafka	○	○	−
14	スケジュール駆動	−	−	−
15	gRPC	○	○	−

(凡例)

○：対応しています。

×：対応していません。

−：該当しません。

付録 C.2　制限事項

負荷分散装置と IPv6 の併用はサポートしていません。

付録 D　用語解説

マニュアルで使用する用語について

マニュアル「アプリケーションサーバ & BPM/ESB 基盤 用語解説」を参照してください。

索引

数字

1:1 系切り替えシステム　41

A

API を使用したデータ変換　625

C

Cookie 情報〔Web サービス（SOAP 通信）〕　73
CTM〔HCSC サーバの冗長構成〕　40
CUP の最大応答待ち時間を rap サーバ側に引き継ぐ〔定義〕　864
CUP の最大応答待ち時間を TP1/Server 側に引き継ぐ〔定義〕　864
CUP の受信で使用するポート番号〔定義〕　882

D

DB アダプタが使用する DB Connector の設定　132
DB アダプタの接続先情報の動的変更　135
DB アダプタを使ったデータベースへのアクセス　129
DB キューを使用したサービスとの接続　119
dccache　865
dccltcachetim　879
dccltconnectinf　881
dccltconnecttimeout　880
dccltcuprcvport　882
dccltcupsndhost　882
dccltdatacomp　881
dccltdbgtrcfilecount　877
dccltdelay　864
dccltextend　865
dccltinquiretime　864
dccltloadbalance　879
dccltnammlthost　880
dccltprfinfosend　880
dccltprftrace　880
dccltrecvbuffsize　883
dccltrpcmaxmsgsize　878
dccltsendbuffsize　883
dcclttrcmplmttm　883
dcclttrcputm　875
dcclttrexpsp　876
dcclttrexptm　875
dcclttrlimittime　872
dcclttroptiitem　871
dcclttrrbinfo　872
dcclttrrbrcv　873
dcclttrrecoverytype　874
dcclttrstatisitem　870
dcclttrwatchtime　872
DCCM3 との接続機能〔TP1/Client/J〕　848
dcdatatrace　869
dcdatatracemaxsize　870
dcdatatracepath　870
dcdatatracesize　870
dcerrtrace　868
dcerrtracepath　868
dcerrtracesize　868
dchost　863
dchostselect　865
dcinquiretimecheck　884
dcmethodtrace　868
dcmethodtracepath　869
dcmethodtracesize　869
dcnamport　862
dcnamuse　862
dcnotifyreshost　885
dcrapautoconnect　867
dcrapdirect　866
dcrapport　867
dcrcvport　876
dcresponsehost　885
dcscddirect　866
dcscdhostchange　879
dcscdloadpriority　866

dcscdmulti 881
dcscdmulticount 882
dcscdport 866
dcselint 865
dcsndhost 877
dcsndport 877
dcsndrcvtype 876
dcsockopenatrcv 877
dcuaptrace 869
dcuaptracepath 869
dcuaptracesize 869
dcwatchtim 863
dcwatchtiminherit 864
dcwatchtimrpcinherit 864
dcweburl 877

F

FTP アダプタ 677
FTP アダプタアカウント定義ファイル 793
FTP アダプタの機能 714
FTP アダプタの実行環境に依存する設定の定義 785, 790
FTP アダプタの処理の流れ 716
FTP インバウンドアダプタ 677
FTP インバウンドアダプタの機能 705
FTP 受付 677
FTP 受付使用時のトランザクション 692
FTP 受付でサポートするオペレーションと FTP コマンド 690
FTP 受付で使用できる通信モデルおよびビジネスプロセス 692
FTP 受付の機能 690
FTP 受付の実行環境に依存する設定の定義 785
FTP クライアントおよび FTP サーバとの接続 208
FTP クライアントから FTP サーバへのファイル転送 678
FTP クライアントから FTP サーバへのファイル転送の流れ（同期型ビジネスプロセスを利用する場合） 679
FTP クライアントから FTP サーバへのファイル転送の流れ（非同期型ビジネスプロセスを利用する場合） 680
FTP クライアントから別の FTP クライアントへのファイル転送 684
FTP コマンド許可リスト定義ファイル（FTP アダプタ用） 792
FTP コマンド許可リスト定義ファイル（FTP 受付用） 787
FTP コマンドの実行 717
FTP コマンドリプライタイムアウト（FTP アダプタ） 728
FTP サーバから FTP クライアントへのファイル転送 681
FTP サーバのファイルリストの取得 687
FTP 実行許可リスト定義ファイル 786
FTP と連携してファイルを転送する機能（FTP 連携） 674
FTP 連携システムの開発の流れ 782
FTP 連携でサポートする FTP コマンド 776
FTP 連携の概要 675
FTP 連携の適用範囲 776
FTP 連携を利用したファイル転送パターンの例 678
FTP 連携を利用するためのシステムの構成 677
FTP 連携を利用するときの注意 780

G

GET メソッドのリクエストを送信する例〔HTTP 受付〕 294
gRPC アダプタの機能 272
gRPC インバウンドアダプタの機能 270
gRPC 受付の機能 266
gRPC 電文（Protocol Buffers 形式）と XML 形式の変換 254
gRPC プロトコルを利用したシステム 248
gRPC プロトコルを利用した接続 248

H

HA モニタ〔HCSC サーバの冗長構成〕 41
HCSC コンポーネントの検証 19
HCSC コンポーネントの配備 27, 36

HCSC サーバから TP1 アダプタを使った TP1 サービス部品の呼び出し 147
HCSC サーバからの MDB（DB キュー）サービスアダプタを使った TP1 サービス部品の呼び出し 165
HCSC サーバで設定できるタイマ 581
HCSC サーバ内の多重度 638
HCSC サーバに関連するタイマの設定 581
HCSC サーバのクラスタ構成 40
HCSC サーバの冗長構成〔N 対 1 クラスタ構成〕 43
HCSC サーバの冗長構成〔クラスタソフトウェア〕 41
HCSC サーバの冗長構成〔ロードバランス機能〕 40
HCSC サーバのセットアップ〔実行環境のセットアップ機能〕 35
HCSC サーバフォルダ 830
HiRDB と連携した実行履歴の管理 671
HTTP アダプタの JSON-XML 変換機能 413
HTTP アダプタのタイマ管理 396
HTTP アダプタのリダイレクト 398
HTTP アダプタを使用した Cookie 情報の引き継ぎ 388
HTTP アダプタを使用したサービス部品の呼び出し 354
HTTP アダプタを利用したシステムの例 404
HTTP 受付使用時のトランザクション 326
HTTP 受付でサポートする Web サーバおよび HTTP クライアント 293
HTTP 受付の JSON-XML 変換機能 327
HTTP 受付の構成例 299
HTTP 受付のタイマ管理 327
HTTP 受付の電文自動変換機能 339
HTTP 受付のファイル管理 322
HTTP 受付の呼び出しサービス選択機能 335
HTTP 受付のリクエスト処理 304
HTTP 受付のルーティング機能 336
HTTP 受付のレスポンス処理 312
HTTP 受付を使用したオンラインシステムの例 293
HTTP 受付を使用したサービス部品の呼び出し 292
HTTP 受付を使用するときの注意 351
HTTP 通信を使用したクライアントとの接続 292
HTTP 通信を使用したサービスとの接続 354
HTTP ヘッダ引き継ぎ〔Web サービス（SOAP 通信）〕 77
HTTP ボディの圧縮／解凍 395
HTTP リクエストと HTTP レスポンスのエンコーディング/デコーディング〔HTTP 受付〕 325
HTTP リクエストと要求電文の関係 355
HTTP リクエストヘッダの処理〔HTTP 受付〕 307
HTTP リクエストボディの処理〔HTTP 受付〕 308
HTTP レスポンスと応答電文の関係 379
HTTP レスポンスヘッダの処理〔HTTP 受付〕 312
HTTP レスポンスボディの処理〔HTTP 受付〕 317

I

IBM MQ の概要 188
IPv6 対応 911

J

Java 呼び出し〔ビジネスプロセス〕 491
JP1/IM との連携による障害検知 665
JP1 との連携 665
JP1 によるプロセス監視 666
JSON 変換ツールを使用した変換 629
JSON-XML 変換 API を使用した変換 626

K

Kafka アダプタの JSON-XML 変換機能 246
Kafka アダプタの機能 238
Kafka アダプタのタイムアウト監視 243
Kafka インバウンドアダプタの機能 217
Kafka インバウンドアダプタのタイムアウト監視 232
Kafka 受付の JSON-XML 変換機能 216
Kafka 受付の機能 211
Kafka との接続 211

L

Listen タイムアウト（FTP アダプタ） 733

M

Message Queue アダプタを使ったメッセージキューへのアクセス 202

Message Queue 受付使用時のトランザクション 198
Message Queue 受付使用時のメッセージの二重配信防止 199
Message Queue 受付と MQ リソースアダプタとの関係 191
Message Queue 受付とキューとの関係 192
Message Queue 受付とビジネスプロセスとの関係 194
Message Queue 受付の構成例 196
Message Queue 受付の送信オプション 188
Message Queue 受付を使用したサービス部品の呼び出し 187
MHP を接続先とした TCP/IP 通信 161

O

OAuth 2.0 に対応したサービスへのアクセス 419
Object Access アダプタ 181
Object Wrapper システムとの接続 181
OpenAPI 仕様書インポート機能 340
OpenAPI 仕様書から HCSC コンポーネントを作成する OpenAPI 仕様書インポート機能 340
OpenTP1 との接続 141

P

POST メソッドのリクエストを送信する例〔HTTP 受付〕 295
PRF トレース 892
protoc コマンドの取得 251
proto ファイルの出力 251

R

rap リスナーのポート番号〔定義〕 867
Reliable Messaging との関係〔MDB（DB キュー）〕 119
Reliable Messaging との関係〔MDB（WS-R）〕 107
RPC 838
RPC 送受信電文の最大長〔定義〕 878
RPC のサービス要求方式 150
RPC の送受信バッファの生成 160
RPC の通信形態 158

S

SessionBean を使用したサービスとの接続 102
SFTP アダプタが実行するコマンド 464
SFTP アダプタが使用する IP アドレスおよびポート番号の指定 469
SFTP アダプタからファイルを送信するときの認証 470
SFTP アダプタで設定できるファイル転送の機能 456
SFTP アダプタと SFTP サーバ間の通信 459
SFTP アダプタのタイムアウトを監視するポイント 466
SFTP アダプタの排他管理 468
SFTP アダプタのファイルアクセス方式 458
SFTP アダプタの要求電文に指定できるリモートのファイルパス 472
SFTP アダプタを使用したシステム 453
SFTP クライアントで使用する IP アドレス 469
SFTP クライアントで使用するポート番号 470
SFTP サーバとの接続 453, 459
SFTP サーバとの接続の切断 466
SOAP Fault 動作定義ファイルによるフォルトのキャッチ 594
SOAP Fault の faultcode によるフォルト判別 593
SOAP Fault の詳細項目によるフォルト判別 592
SOAP アダプタのフォルト判別の概要 590
SOAP アダプタを利用したフォルトのキャッチ 590
SOAP メッセージの構成〔Web サービス（SOAP 通信）〕 81
SQL 命令の INSERT 文，UPDATE 文および DELETE 文を使用する場合の注意事項 132

T

TCP/IP コネクションの確立の監視機能〔TP1/Client/J〕 848
TCP/IP 通信機能〔TP1/Client/J〕 847
TCP/IP 通信機能の使用時に初期化する環境〔定義〕 876
TCP/IP の受信バッファサイズ〔定義〕 883
TCP/IP の送信バッファサイズ〔定義〕 883
TP1/Client/J 環境定義の詳細 861
TP1/Client/J の機能 837
TP1/Client/J の障害対策 886

TP1/Client/J の定義　854
TP1/Client/J の定義の概要　855
TP1/RPC 受付使用時のトランザクション　142
TP1/RPC 受付でのデータの流れとデータ変換の関係　143
TP1/RPC 受付のデータ変換　143
TP1/Web 接続機能〔TP1/Client/J〕　847
TP1 アダプタおよび Message Queue アダプタのオペレーション動的変更機能　484
TP1 アダプタの運用例　149
TP1 アダプタのオペレーション動的変更　165
TP1 アダプタの接続先情報の動的変更　169
TP1 アダプタの接続先動的変更　165
TP1 アダプタを使用したシステムの構成例　150
TP1 から TP1/RPC 受付を使った HCSC サーバのサービス部品の呼び出し　141
TP1 から非同期の標準受付を使った HCSC サーバのサービス部品の呼び出し　141

U

UAP 障害時のトランザクション同期点処理方式〔定義〕　874
UAP トレース〔TP1/Client/J〕　887
UAP トレースファイルサイズ〔定義〕　869
UAP トレースファイル作成ディレクトリ〔定義〕　869
UAP トレースを取得する〔定義〕　869
UOC を利用したユーザ電文の値の変更　613

W

Web サービス（SOAP 通信）に関する多重度　644
Web サービス（SOAP 通信）の場合の接続先動的変更　84
Web サービス（SOAP 通信）の場合の添付ファイル　91
Web サービス（SOAP 通信）を使用したサービスとの接続　64
Web サービスの通信基盤との関係　64
Windows Server Failover Cluster〔HCSC サーバの冗長構成〕　41
WS-R（WS-Reliability）を使用したサービスとの接続　107
WWW 認証を使用したセキュアな接続　418

X

XA リソースサービス機能〔TP1/Client/J〕　853
XML 解析処理に関する多重度　647
XML 形式　178
XML 電文内の属性値に指定した名前空間接頭辞の補完　600
XML 電文のキャッシュ機能　649

あ

アイドルタイムアウト（FTP インバウンドアダプタ）　708
アクティビティ　505
アクティビティトレース機能　663
アクティビティ〔ビジネスプロセスの作成機能〕　24
アプリケーションからの稼働状態の確認　597
アプリケーションサーバ（EJB コンテナ）との関係　102

い

移行元サーバから取得したファイル情報一覧を基にファイルを取得して移行先サーバへ送信する（SFTP アダプタの場合）　454
一括〔読み込みモード〕　179

う

受付 ID フォルダ　830
受付・応答〔ビジネスプロセス〕　489
運用環境の機能概要　35

え

永続化しないビジネスプロセス　510
永続化するビジネスプロセス　510
永続化するビジネスプロセスと永続化しないビジネスプロセス　510
エラー設計の概要　794
エラートレース〔TP1/Client/J〕　889
エラートレースファイルサイズ〔定義〕　868
エラートレースファイル作成ディレクトリ〔定義〕　868

エラートレースを取得する〔定義〕 868

お

応答アクティビティのあとに定義されたアクティビティの非同期実行 570
応答アクティビティのトランザクション 542
応答電文受信用 IP アドレスを通知〔定義〕 885
応答電文で取得できる外部コマンドプロセスの終了情報 447
オペレーション動的変更機能を使ったサービス定義 486
オペレーション動的変更機能を利用した処理の流れ 484

か

開発環境の機能概要 19
外部コマンドの実行によるシステムとの接続 444
外部コマンドプロセスに関する動作オプションの設定 447
書き込みモード 179
各オペレーションで使用するフォルダおよびファイルの管理 773
各オペレーションの要求電文および応答電文 739
各種サービス呼び出し機能 33
カスタム受付を使用した場合のシステムとの接続 477
稼働状態の確認時の注意事項 598
稼働状態の確認の概要 597
稼働状態の確認の流れ 597
監視対象の設定 808
監視対象のファイルと取得対象ファイル 810
監視対象のファイルの状態管理機能 811
監視対象のファイルの数量とリクエスト回数の関係 810
監視対象のファイルの登録 809
監視対象のファイルの登録解除 809
監視フォルダの設定〔ファイルイベント連携機能〕 818

き

機能拡張レベルの設定〔定義〕 865
基本的な構造〔サービス部品呼び出し〕 51
キャッシュ機能を利用したオブジェクトの再利用 654
キャッシュの登録および削除のタイミング 654
キャッシュの有効期限〔定義〕 879
キューの名称と個数〔MDB（DB キュー）〕 128
キューの名称と個数〔MDB（WS-R）〕 118
共通フォルダ定義名と対応するパスの定義 832
共通フォルダと中間ファイルの作成〔FTP 受付〕 700
共通フォルダ内のファイルの削除 834
共通フォルダ内のファイルの生成 832
共通フォルダ内のファイルの排他制御 834
共通フォルダの設定〔ファイルイベント連携機能〕 819
共通フォルダを利用したファイルアクセス 775, 832

く

クライアント側から転送されたファイルを変換して目的のサーバへ送信する（SFTP アダプタの場合） 455
クラスタ構成における運用環境の構成パターン 44
クラスタソフトウェア 41
クラスタのセットアップ 35
繰り返し〔ビジネスプロセス〕 496

け

厳密な鍵交換 466

こ

更新完了通知ファイルの作成〔ファイルイベント連携機能〕 820
更新排他 180
固定長形式 178
コネクション確立最大監視時間〔定義〕 880
コネクション継続（Keep-Alive）を使用した通信 393
コマンドアダプタを使用した外部コマンドの実行 444
コンポーネントインポート機能を利用した分散開発の流れ 47
コンポーネント共通 UOC の概要 613
コンポーネント共通 UOC の使用例 617

さ

サーバからの一方通知受信機能〔TP1/Client/J〕 847

サービスアダプタ　22
サービスアダプタでのキャッシュの注意事項　655
サービスアダプタの作成〔サービスアダプタの作成機能〕　22
サービスのエミュレーション機能の概要　633
サービスのエミュレーション機能の設定　633
サービス部品側と異なる電文フォーマットでサービス部品を呼び出す場合の流れ　59
サービス部品の接続先変更　589
サービス部品のタイムアウト値の変更　588
サービス部品呼び出し〔基本的な流れ〕　51
サービス部品呼び出し〔メッセージの流れ（MDB(DB キュー)）〕　120
サービス部品呼び出し〔メッセージの流れ（MDB(WS-R)）〕　107
サービス部品呼び出し〔ユーザ電文と WSDL の関係〕　65
サービス部品呼び出し〔ユーザ電文とサービス部品のスタブとの関係〕　102
サービス部品を呼び出す〔応答電文の扱い（MDB(DB キュー)）〕　125
サービス部品を呼び出す〔応答電文の扱い（MDB(WS-R)）〕　111
サービス部品を呼び出す流れ　54
サービスプラットフォームの機能〔機能概要〕　18
サービスプラットフォームのサーバマシンのローカルディスクを使用する構成例　814
サービスプラットフォームの性能解析トレースを取得する〔定義〕　880
サービス振り分け実行機能　33
サービス名の設定方法（TP1/RPC 受付）　147
サービス要求先スケジューラを RPC ごとに分散させる〔定義〕　879
サービス要求先の TP1/Web の URL〔定義〕　877
サービス呼出アクティビティのトランザクション　533
サービス呼び出し〔ビジネスプロセス〕　490
サービスリクエスタ〔サービスリクエスタの作成機能〕　30
サービスリクエスタの作成〔サービスリクエスタの作成機能〕　30
最大応答待ち時間〔定義〕　863
最大通信遅延時間〔定義〕　864
作業フォルダおよび共通フォルダにアクセスできるコンポーネント　826
作業フォルダおよび共通フォルダの機能　825
作業フォルダおよび共通フォルダを利用したファイルアクセス　775
作業フォルダおよび作業フォルダに格納された中間ファイルの削除　831
作業フォルダと中間ファイルの作成　695
作業フォルダとリクエスト ID の生成　828
作業フォルダ内のファイルの生成　830
作業フォルダ内のファイルの排他制御　831
作業フォルダルートの定義　828
作業フォルダを利用したファイルアクセス　775, 828
サブフォルダの設定〔ファイルイベント連携機能〕　819
さまざまなシステムと接続するための機能　50
参照排他　180

し

時間監視　163
識別情報　61
システム動作を円滑にするための機能　580
システム例外のフォルト変換　594
システム例外をフォルトに変換する汎用フォルト　608
実行環境での運用を管理する　37
実行環境の機能概要　33
実行環境のセットアップ　35
実行許可コマンド定義ファイルによる実行制限　451
実行履歴の管理　39
実行履歴の検索〔プロセスインスタンス〕　39
実行履歴の削除〔プロセスインスタンス〕　39
実行履歴の詳細情報の参照〔プロセスインスタンス〕39
実行履歴の分割格納　671
受信ポート固定機能〔TP1/Client/J〕　854
受信用ソケットを開設する契機〔定義〕　877
受信用の CUP のポート番号〔定義〕　876
取得対象ファイルの作成〔ファイルイベント連携機能〕　820
障害解析機能　657

障害回復機能　657
障害情報の取得機能　657
障害情報の取得・障害回復・障害解析機能　657
障害対策のための設計　794
障害調査のための機能　656
障害発生時の系の切り替え　42
常設コネクション　838
常設コネクション問い合わせ間隔最大時間〔定義〕　864
常設コネクションを確立した場合の処理概要（オートコネクトモード）　153
常設コネクションを確立した場合の処理概要（非オートコネクトモード）　154
常設コネクションを自動的に確立させる〔定義〕　867
署名付き AWS API リクエストの署名情報付与　435
処理に使用するフォルダの指定　720
新規〔書き込みモード〕　179

す

スケジューラダイレクト機能　157
スケジューラダイレクト機能を使用した RPC〔定義〕　866
スケジューラダイレクト機能を使用した場合の処理の概要　157
スケジュール駆動受付でのスケジュールの設定　554
スケジュール駆動受付でのタスク情報管理機能　558
スケジュール駆動受付でのタスクの呼び出し機能 559
スケジュール駆動受付の作業フォルダ　564
スケジュール駆動受付のタイマ監視　565
スケジュール駆動受付のトランザクション　565
スケジュールサービスのポート番号〔定義〕　866
スコープアクティビティのトランザクション　541
スコープの開始時および終了時にコミットする設定を選択した場合のトランザクション　535
ステータスコード〔HTTP 受付〕　319
ステータスラインの処理〔HTTP 受付〕　312

せ

制御コネクション接続タイムアウト（FTP アダプタ）　728
性能解析トレース〔TP1/Client/J〕　892
性能解析トレースファイル　657
性能解析トレースを出力する場合の出力例　893
性能検証用トレース　892
性能検証用トレースに識別情報を付加する場合の出力例　893
性能検証用トレースに出力する情報として TP1/Client/J 内部で識別情報（IP アドレスなど）を付加する〔定義〕　880
性能向上のための機能　637
セッション接続タイムアウト（SFTP アダプタの場合）　467
接続する MHP が存在するノードのホスト名〔定義〕　877
接続する MHP のポート番号〔定義〕　877
接続できるデータベース　670
設定に応じたログファイルの出力内容　573
セパレータ形式　178
前提条件〔HTTP 通信を使用したサービスとの接続〕　354
前提条件〔外部コマンドの実行によるシステムとの接続〕　444
前提条件〔メールサーバとの接続〕　281

そ

相関セットの役割　509
相関セット〔ビジネスプロセスの作成機能〕　24
送受信後のファイルサイズ確認　705, 721
送信元ホスト指定機能〔TP1/Client/J〕　853
送信元ホスト〔定義〕　882

た

待機処理〔ビジネスプロセス〕　500
代入〔ビジネスプロセス〕　495
タイマ管理　726
タイマ設計　795
タイマ設定時の注意事項　588
タイマの設定　581
タイムアウト時間の設定方法（TP1/RPC 受付）　146

タイムアウトの監視〔FTP インバウンドアダプタ〕 706
タイムアウトの監視〔FTP 受付〕 700
多重度 638
他製品と連携して利用する機能 664
端末固定割り当て機能 851
端末識別情報設定機能 851
端末識別情報〔定義〕 881

ち

チャンク送信を使用した通信 394
注意事項〔SessionBean アダプタ使用時〕 106
注意事項〔TP1 アダプタ使用時〕 165

つ

追加〔書き込みモード〕 179
通信先を指定した接続 158
通信先を指定した場合の接続処理の概要 158
通信処理の運用中に監視時間に適用される値 164
通信処理の開始中および停止中に監視時間として適用される値 164

て

定義例外 594
データ圧縮機能〔TP1/Client/J〕 853
データ圧縮機能〔定義〕 881
データ検証機能 658
データ検証機能の概要 658
データ検証機能の設定 659
データ検証機能の有効範囲 660
データコネクション接続タイムアウト（FTP アダプタ） 729
データコネクション接続のタイムアウト（FTP インバウンドアダプタ） 710
データ受信のタイムアウト（FTP インバウンドアダプタ） 711
データ送信のタイムアウト（FTP インバウンドアダプタ） 712
データ転送・取得タイムアウト（FTP アダプタ） 730
データトレース〔TP1/Client/J〕 888
データトレースの最大データ長〔定義〕 870
データトレースファイルサイズ〔定義〕 870
データトレースファイル作成ディレクトリ〔定義〕 870
データトレースを取得する〔定義〕 869
データベースとの接続 129
データベースへのアクセスに関する多重度 641
データ変換 API を使用したデータ変換 625
データ変換機能のエミュレーション 909
データ変換機能の処理概要 903
データ変換時の構造変換スキップ機能の指定 144
データ変換処理の XML パーサプール機能 651
データ変換処理の概要 905
データ変換定義〔データ変換定義の作成機能〕 26
データ変換定義の作成〔データ変換定義の作成機能〕 26
データ変換に必要な定義ファイルと作成方法 903
データ変換のキャッシュ機能 654
データ変換〔ビジネスプロセス〕 492
データ変換をしない場合の変数（メッセージ型）の組み合わせ 602
データ読み込みタイムアウト 733
データ読み込みタイムアウト（SFTP アダプタの場合） 468
デッドメッセージキュー〔MDB（DB キュー）〕 127
デッドメッセージキュー〔MDB（WS-R）〕 118
デバッグ情報出力機能の概要 621
デバッグ情報の出力例 621
デバッグトレース〔TP1/Client/J〕 891
デバッグトレースファイルの最大ファイル数〔定義〕 877
転送コマンドの指定方法 779
転送データの圧縮（SFTP アダプタの場合） 465
転送前／転送後またはリスト前／リスト後に実行する FTP コマンドの管理 698
電文フォーマット〔電文フォーマットの作成機能〕 19
電文フォーマットの作成〔電文フォーマットの作成機能〕 19

と

「問い合わせ間隔最大時間」をチェックする〔定義〕 884
同期応答型 RPC 通信 159
同期応答型 RPC 通信をする場合の処理の概要 159
統計情報項目〔定義〕 870
同時実行数の設定方法（TP1/RPC 受付） 145
同時実行数の設定〔Message Queue 受付〕 201
同時実行スレッド数の制御の設定 822
動的定義変更機能〔TP1/Client/J〕 848
途中終了アクティビティのトランザクション 549
トラブルシュート機能〔TP1/Client/J〕 853
トラブルシュート設計 803
トランザクション〔MDB（DB キュー）〕 127
トランザクション〔MDB（WS-R）〕 113
トランザクション〔SessionBean〕 104
トランザクション〔Web サービス（SOAP 通信）〕 73
トランザクション完了限界時間〔定義〕 883
トランザクション最適化項目〔定義〕 871
トランザクション制御〔TP1/Client/J〕 847
トランザクション設計 796
トランザクションタイムアウト時間の設定方法（TP1/RPC 受付） 146
トランザクション同期点処理時の最大通信待ち時間〔定義〕 872
トランザクションブランチ CPU 監視時間〔定義〕 875
トランザクションブランチ限界経過時間〔定義〕 875
トランザクションブランチ最大実行可能時間〔定義〕 872
トランザクションブランチの処理を監視〔定義〕 876
トランザクションブランチの統計情報を取得する項目〔定義〕 870
トレースファイルの出力内容〔TP1/Client/J〕 887

に

任意形式（any 形式）の電文フォーマットの定義 601
任意形式（any 形式）を設定した場合の処理の流れ 601

ね

ネームキャッシュの最大エントリ数〔定義〕 865
ネームサービス機能 156
ネームサービス機能を使用した場合の処理の概要 157
ネームサービスのポート番号〔定義〕 862
ネームサービスを使用した RPC 840
ネームサービスを使用した RPC〔定義〕 862

は

排他取得タイムアウト（FTP アダプタ） 728
バイナリデータの変換エラー時のデバッグ情報の出力 621
配備から開始または停止から削除の一括実行機能 29
配備定義〔HCSC コンポーネントの配備機能〕 28
配備定義機能 27
パッケージング機能 28
パッケージング〔パッケージング機能〕 28
汎用カスタムアダプタ 22
汎用カスタムアダプタを使用した場合のシステムとの接続 480
汎用フォルト電文とメッセージログのマッピング 611
汎用フォルトを使用してシステム例外発生個所を特定する場合の流れ 610
汎用ロードバランサ 40

ひ

非応答型 RPC 通信 159
非応答型 RPC 通信をする場合の処理の概要 159
ビジネスプロセス実行機能 34
ビジネスプロセスのアクセス制御 696
ビジネスプロセスの階層化 25
ビジネスプロセスの概要 489
ビジネスプロセスの機能 488
ビジネスプロセスの再実行 551
ビジネスプロセスの再実行〔運用環境画面〕 551
ビジネスプロセスの再実行〔コマンド〕 552
ビジネスプロセスの再実行〔サービスリクエスタ（SessionBean）〕 553
ビジネスプロセスの再実行〔サービスリクエスタ（SOAP 通信）〕 552

ビジネスプロセスの作成〔ビジネスプロセスの作成機能〕 23
ビジネスプロセスの仕組み 501
ビジネスプロセスの事前キャッシュ機能 650
ビジネスプロセスの実行状態の管理 24
ビジネスプロセスの状態の互換性の指定を OFF，サービス呼出アクティビティの状態の互換性の指定を OFF にした場合のトランザクション 529
ビジネスプロセスの状態の互換性の指定を OFF，サービス呼出アクティビティの状態の互換性の指定を ON にした場合のトランザクション 521
ビジネスプロセスの状態の互換性の指定を ON にした場合のトランザクション 513
ビジネスプロセスの処理の流れ 501
ビジネスプロセスのスケジュール設定 554
ビジネスプロセスの通信モデル 501
ビジネスプロセスのデバッグ機能 30
ビジネスプロセスのトランザクション 512
ビジネスプロセス〔ビジネスプロセスの作成機能〕 23
ビジネスプロセスを使用した待機処理 567
非常設コネクションを確立した場合の処理の概要（非オートコネクトモード） 155

ふ

ファイアウォール 840
ファイルアダプタで XML データとして扱われるファイル形式 178
ファイルアダプタで使用できるファイル形式 178
ファイルアダプタでバイナリデータとして扱われるファイル形式 178
ファイルアダプタの運用例 177
ファイルアダプタを使ったファイルの読み込みと書き込み 176
ファイル圧縮オペレーション 762
ファイル一覧書き込みオペレーション 767
ファイル一覧取得オペレーション 766
ファイル移動オペレーション 759
ファイルイベント受付が監視するフォルダの設定 818
ファイルイベント受付の定義ファイルの作成 821
ファイルイベントトレース機能 812
ファイルイベント連携機能で設計が必要な項目 817
ファイルイベント連携時に作成するファイル 820
ファイルイベント連携時に使用するファイルの作成 817
ファイルイベント連携使用時の注意事項 812
ファイルイベント連携で必要な設定 816
ファイルイベント連携に関する実行環境の設定 821
ファイルイベント連携のエラー設計の概要 823
ファイルイベント連携の設定の流れ 816
ファイルイベント連携を使用したファイルの監視 474, 806
ファイルイベント連携を使用するためのシステムの構成例 814
ファイル開始遅延時間の設計 816
ファイル書き込みオペレーション 771
ファイル監視機能 807
ファイル監視機能の有効化 816
ファイル監視機能を有効にするための設定 807
ファイル監視の開始 808
ファイル監視の終了 809
ファイル監視の処理時間がファイル監視の間隔を超えた場合の動作 810
ファイルサーバの共有ディスクを使用する構成例 815
ファイル伸張オペレーション 764
ファイル操作アダプタ 677
ファイル操作アダプタでサポートするオペレーション 735
ファイル操作アダプタの機能 735
ファイル操作が行われるときのモード 178
ファイル転送のパターン（SFTP アダプタの場合） 462
ファイルの読み込みおよび書き込み時の排他制御 179
ファイルのリネーム 722
ファイル複製オペレーション 758
ファイル変換オペレーション 740
ファイル読み込みオペレーション 770
ファイル連携機能で使用するフォルダおよびファイルの設計 816
ファイル連携の機能概要 807
ファイルを扱うシステムとの接続 175
ファイル・フォルダ削除オペレーション 760
ファンクション〔データ変換定義の作成機能〕 26

負荷レベルが最も低いサービス要求先スケジューラの情報をキャッシュに格納する〔定義〕 879
複数のコネクションが確立された場合の処理の概要（オートコネクトモード） 156
複数のデータレコードの同時変換 753
プロキシサーバを経由した通信 392
プロセスインスタンスの再実行 39
プロセスインスタンスの実行履歴の管理 39
プロセスインスタンス〔ビジネスプロセスを使った運用〕 502
分岐〔ビジネスプロセス〕 498
分散開発時の注意事項 49
分散開発時のリポジトリのインポート 47

へ

並列処理〔ビジネスプロセス〕 499
ベーシック認証機能を使用した接続 399
ヘッダ割当変数の定義 485
変数 24
変数の種類 509
変数の役割 509

ほ

ホスト切り替え機能〔TP1/Client/J〕 854
ホスト名〔定義〕 885

ま

マスタスケジューラデーモン 845
マッピング〔データ変換定義の作成機能〕 26
窓口となる TP1/Server のホスト名〔定義〕 863
窓口となる TP1/Server を優先して負荷分散する〔定義〕 866
窓口となる TP1/Server をランダムに選択する〔定義〕 865
マルチスケジューラ機能 845
マルチスケジューラ機能〔定義〕 881
マルチスケジューラデーモン 845
マルチスケジューラデーモンのプロセス数〔定義〕 882
マルチホームドホスト形態の TP1/Server に対してネームサービスを使用した RPC を行う〔定義〕 880

め

メールアダプタからメールを送信するときの認証 290
メールアダプタ使用時のトランザクション 287
メールアダプタで設定できるメール情報 287
メールアダプタのタイマ管理 289
メールアダプタを使用したシステム 281
メールサーバとの接続 281
メソッドトレース〔TP1/Client/J〕 890
メソッドトレースファイルサイズ〔定義〕 869
メソッドトレースファイル作成ディレクトリ〔定義〕 869
メソッドトレースを取得する〔定義〕 868
メッセージキューとの接続 187
メッセージトラッキング機能 657
メッセージの二重配信防止 199
メッセージのロールバック回数の設定〔Message Queue 受付〕 200
メッセージログファイル 657

も

文字コード変換 UOC を使用したファイルの変換 756

ゆ

ユーザ定義受付の作成〔ユーザ定義受付の作成機能〕 20
ユーザ定義受付〔ユーザ定義受付の作成機能〕 20
ユーザ電文トレース機能 661
ユーザ電文トレース機能使用時の注意事項 662
ユーザ電文トレース機能の概要 661
ユーザ電文トレース機能の設定 661

よ

要求電文（ヘッダ）と HTTP アダプタ実行環境（共通）プロパティファイルのパラメタの対応関係 406
要求電文を利用した外部コマンド情報の指定 445
用語解説 913
要素 20
読み込みモード 178

り

リクエスト ID の生成　694
リクエストトレースファイル　657
リクエストラインの処理〔HTTP 受付〕　305
リストコマンドオプション定義ファイル　789
リストコマンドオプションの設定による LIST／NLST コマンドの実行　699
リストコマンドパスのデータ抽出機能〔FTP 受付〕　702
リモート API 機能　151
リモート API 機能〔TP1/Client/J〕　840
リモート API 機能〔定義〕　866
リモートプロシジャコール〔TP1/Client/J〕　838

れ

例外発生時にサービスアダプタが取得するエラーの選択　612
レングスタグ形式　178

ろ

ロードバランス機能〔HCSC サーバの冗長構成〕　40
ロールバック完了通知を受信する〔定義〕　873
ロールバック要因に関する情報をログに取得する〔定義〕　872
ログインセッション維持のアイドルタイムアウト　734
ログインセッションの維持　722
ログ出力アクティビティでのログ出力　573
ログ出力アクティビティのログファイル　576
ログ出力アクティビティのログファイルの出力　578