HITACHI
Inspire the Next

uCosminexus Service Coordinator Interactive Workflow
BPMN 連携機能　使用の手引

3020-3-M86-70

前書き

■ 対象製品

- P-2955-CJ34 uCosminexus Business Process Developer 03-11 （適用 OS：Windows Server 2016，Windows Server 2019，Windows Server 2022，Windows 10 x64，Windows 11）
- P-2943-CG34 uCosminexus Service Coordinator Interactive Workflow 03-30 （適用 OS：Windows Server 2019，Windows Server 2022）
- P-2955-CG34 uCosminexus Service Coordinator Interactive Workflow 03-30 （適用 OS：Windows Server 2019，Windows Server 2022，Windows 10 x64，Windows 11）
- P-9W43-CG31 uCosminexus Service Coordinator Interactive Workflow 03-30 （適用 OS：Red Hat Enterprise Linux Server 7（64-bit x86_64），Red Hat Enterprise Linux Server 8（64-bit x86_64），Red Hat Enterprise Linux Server 9（64-bit x86_64））

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

本製品では日立ネットワークオブジェクトプラザトレース共通ライブラリをインストールします。

■ 輸出時の注意

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

なお、不明な場合は、弊社担当営業にお問い合わせください。

■ 商標類

HITACHI，Cosminexus，HiRDB および uCosminexus は，株式会社 日立製作所の商標または登録商標です。

Excel は，マイクロソフト 企業グループの商標です。

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

Microsoft は，マイクロソフト 企業グループの商標です。

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

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

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

SQL Server は，マイクロソフト 企業グループの商標です。

■ 特記事項

マイクロソフト製品のスクリーンショットは，マイクロソフトの許可を得て使用しています。

■ 発行

2024 年 3 月

■ 著作権

変更内容

変更内容(3020-3-M86-70)　uCosminexus Service Coordinator Interactive Workflow 03-30

追加・変更内容	変更個所
使用できる DBMS として PostgreSQL をサポートした。	1.1.1，1.1.2，7.1，7.4，7.7，7.9.4(2)，8.15，8.18，10.6，11.1，17.1，17.1.3(3)，17.2，17.2.3(3)，17.2.5(3)，付録 A.1
アプリケーション呼び出しサービスが，呼び出し先の REST アプリケーションと CSCIW の案件，作業を紐づけできるように，リクエスト時の HTTP ヘッダとして案件 ID，作業 ID を送信できるようにした。	1.8.12
BPMN 要素ごとの，配置先への配置可否の説明を追加した。	2.4.1(1)
実行中の作業が存在する状況で，ビジネスプロセス定義を変更した場合に，変更が反映されないときの説明を追加した。	2.6
BPMN 連携機能に使用するワーク管理データベース容量を見積もる手順を変更した。	3.1.1
クラスパスに，日立ネットワークオブジェクトプラザトレース共通ライブラリ（HNTRLib2）の jar ファイルを追加した。	7.9.1
CIWBPMNLib インタフェースに次のメソッドを追加した。 • getFlowNodeInstancesListWithChildPIByPIID また，これに伴い，次のメソッドに getFlowNodeInstancesListWithChildPIByPIID メソッドに関する説明を追加した。 • getFlowNodeInstancesListByPDName • getFlowNodeInstancesListByPIID	12.4，12.4.26，12.4.27，12.4.28
共通設定ファイルに指定するパラメタに次のパラメタを追加した。 • AppCallServiceSendID	15.2.5，15.2.5(30)
ワーク管理データベースを移行する手順について，CSCIW03-20 から CSCIW03-30 へ移行する場合の手順を追加した。	17.1.3，17.2.5
ワーク管理データベースへのアクセス権の付与についての記述を修正した。	17.2.3(1)，17.2.3(2)，17.2.5(1)，17.2.5(2)

単なる誤字・脱字などはお断りなく訂正しました。

OS，ブラウザ，およびデータベースに関して，新しいバージョンの追加，および古いバージョンの削除については記載していません。サポートしているバージョンの詳細については「リリースノート」でご確認ください。

はじめに

このマニュアルは，uCosminexus Service Coordinator Interactive Workflow でのワークフローシステムの設計方法，ビジネスプロセスや業務アプリケーションの開発方法，およびワークフローシステムの構築方法などについて説明したものです。

■ 対象読者

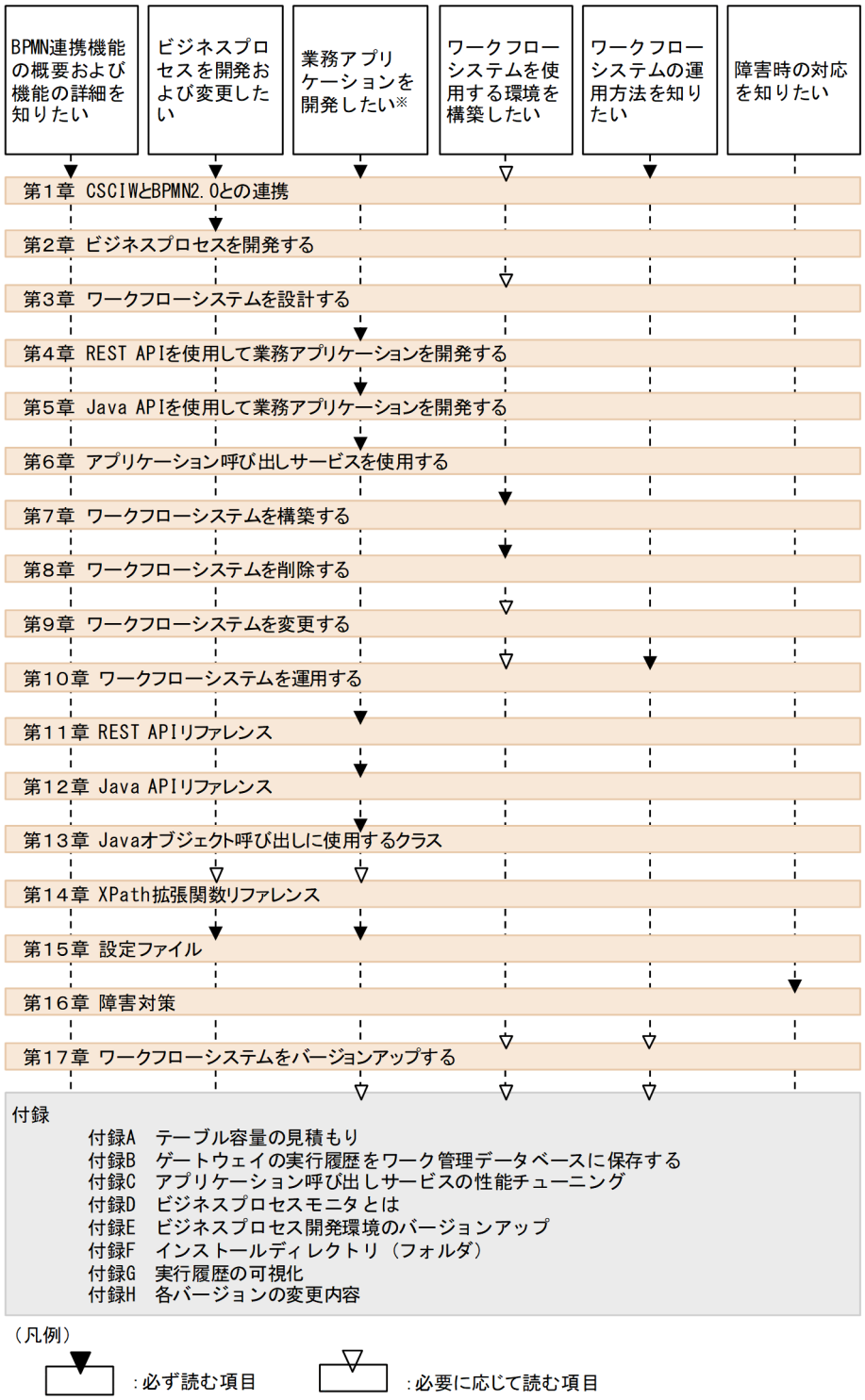
このマニュアルは，CSCIW で BPMN 連携機能を使用して，対話型ワークフローを適用したシステムを構築・運用・開発する方を対象にしています。また，次の知識をお持ちであることを前提にしています。

- CSCIW に関する知識
- Eclipse および Activiti Designer に関する知識や基本操作
- BPMN2.0 に関する知識
- 使用している OS に関する基本的な知識
- Cosminexus および JavaEE に関する知識
- リレーショナルデータベースの操作（SQL 文など）に関する知識

■ 読書手順

このマニュアルは，利用目的に合わせて直接章を選択して読むことができます。利用目的別に次の流れに従ってお読みいただくことをお勧めします。

読書手順



注※

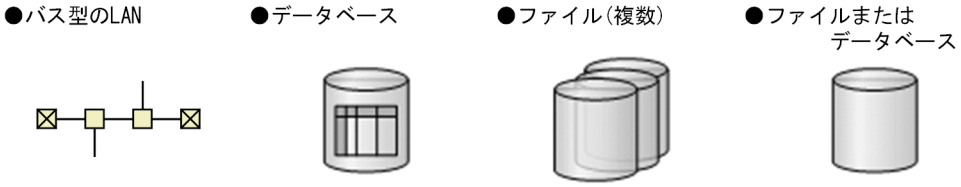
第 4，5，11，12，13 章については，使用する API に関する章だけお読みください。

なお，このマニュアルの「用語解説」では，BPMN 連携機能の用語について説明しています。CSCIW の用語については，マニュアル『uCosminexus Service Coordinator Interactive Workflow システム構築・運用ガイド』の「付録 N　用語解説」を参照してください。

■ このマニュアルで使用する記号

図中で使用する記号

このマニュアルの図中で使用する記号を示します。



このマニュアルで使用する記号

このマニュアルで使用する記号を示します。

記号	意味
＜　＞	可変値を意味します。
斜体	可変値を意味します。 ファイルの記述例など，記号<>で変数を表現しきれない場合に使用します。

コマンドの説明で使用する記号

コマンドの説明で使用する記号を示します。

記号	意味
＜　＞	可変値を意味します。
｜（ストローク）	横に並べられた複数の項目に対し，項目間の区切りを示します。
{　}	この記号で囲まれている複数の項目のうちから 1 つを選択することを意味します。項目が横に並べられ，記号｜で区切られている場合は，そのうちの 1 つを選択します。 例 　{A ｜ B}は，「A と指定する」または「B と指定する」ことを示します。
(　)	この記号で囲まれている複数の項目のうちから 1 つ以上を選択することを意味します。項目が横に並べられ，記号｜で区切られている場合は，そのうちの 1 つ以上を選択します。 例 　(A ｜ B)は，「A と指定する」，「B と指定する」または「A および B と指定する」ことを示します。

記号	意味
[ ]	この記号で囲まれている項目は省略してよいことを意味します。 例 [A]は「何も指定しない」か「A を指定する」ことを示します。

操作の説明で使用する記号

操作の説明で使用する記号を示します。

記号	意味
[ ]	次のどれかを示します。 • ボタン • キーボードのキー • 画面またはダイアログの名称 • 画面またはダイアログに表示される項目
[A] ＋ [B]	＋の前のキーを押したまま，後ろのキーを押すことを示します。
[AAA] － [BBB]	－の前に示したメニューから，－の後ろのメニューを選択することを示します。

■ このマニュアルで使用する注記

このマニュアルで使用する注記を次に示します。

重要

操作を完了させるための重要な情報を示します。

メモ

本文に対して強調したい内容または補足事項を示します。

ヒント

操作する上で効果的な情報，指針，提案を示します。

■ このマニュアルでの表記

このマニュアルで使用する製品名・機能名を示します。

表記	製品名
Chrome	Google Chrome
Cosminexus	uCosminexus Application Server
CSCIW	uCosminexus Service Coordinator Interactive Workflow
Firefox	Mozilla Firefox
HiRDB	HiRDB Server Version 10
	HiRDB SQL Executer
Linux※1※2	Red Hat Enterprise Linux Server 7（64-bit x86_64）
	Red Hat Enterprise Linux Server 8（64-bit x86_64）
	Red Hat Enterprise Linux Server 9（64-bit x86_64）
ORACLE	Oracle(R) Database 19*c*

注※1

仮想化プラットフォームについては，前提となる AP サーバに準拠します。

注※2

Linux の製品名称を特に区別する必要がない場合，UNIX と表記しています。

■ マイクロソフト製品の表記について

このマニュアルでは，マイクロソフト製品の名称を次のように表記しています。

表記		製品名
SQL Server	SQL Server 2005	Microsoft SQL Server 2005
	SQL Server 2008	Microsoft SQL Server 2008
		Microsoft SQL Server 2008 R2
SQL Server の JDBC ドライバ		Microsoft SQL Server 2005 JDBC Driver 1.1
		Microsoft SQL Server 2005 JDBC Driver 1.2
		Microsoft SQL Server JDBC Driver 2.0
		Microsoft SQL Server JDBC Driver 3.0
Windows Server	Windows Server 2016※	Microsoft Windows Server 2016 Standard 日本語版
		Microsoft Windows Server 2016 Datacenter 日本語版
	Windows Server 2019※	Microsoft Windows Server 2019 Standard 日本語版
		Microsoft Windows Server 2019 Datacenter 日本語版

表記		製品名
Windows Server	Windows Server 2022※	Microsoft Windows Server 2022 Standard 日本語版
		Microsoft Windows Server 2022 Datacenter 日本語版
Windows	Windows 10※	Windows 10 Pro 日本語版(64 ビット版)
		Windows 10 Enterprise 日本語版(64 ビット版)
	Windows 11※	Windows 11 Pro 日本語版
		Windows 11 Enterprise 日本語版

Windows Server 2016，Windows Server 2019，Windows Server 2022，Windows 10，および Windows 11 を特に区別する必要がない場合，Windows と表記しています。

注※

仮想化プラットフォームについては，前提となる AP サーバに準拠します。

■ このマニュアルで使用している略語

このマニュアルで使用する英略語を次の表に示します。

英略語	正式名称
BPMN	Business Process Modeling and Notation
DBMS	Database Management System
J2EE	Java 2 Platform, Enterprise Edition，および J2EE
JAR	Java ARchive
JAX-RS	Java API for RESTful Web Service
JDBC	JDBC Java Database Connectivity，および JDBC
JDK	Java Development Kit，および JDK
REST	Representational State Transfer

■ このマニュアルで使用する KB（キロバイト）などの単位表記

1KB（キロバイト），1MB（メガバイト），1GB（ギガバイト），1TB（テラバイト）はそれぞれ 1,024 バイト，$1{,}024^2$ バイト，$1{,}024^3$ バイト，$1{,}024^4$ バイトです。

■ 関連マニュアル

このマニュアルの関連マニュアルを次に示します。必要に応じてお読みください。

CSCIW 関連

- uCosminexus Service Coordinator Interactive Workflow システム構築・運用ガイド（3020-3-M80）
- uCosminexus Service Coordinator Interactive Workflow AP 開発ガイド（3020-3-M81）
- uCosminexus Service Coordinator Interactive Workflow 案件運用操作ガイド（3020-3-M82）
- uCosminexus Service Coordinator Interactive Workflow メッセージ（3020-3-M83）
- uCosminexus Service Coordinator Interactive Workflow コマンド（3020-3-M84）

> **ヒント**
>
> マニュアル『uCosminexus Service Coordinator Interactive Workflow メッセージ』やマニュアル『uCosminexus Service Coordinator Interactive Workflow コマンド』を参照する場合で，参照先に CSCIW-Definer の記述があるときは，マニュアル内の「CSCIW-Definer」の表記を「BPMN エディタ」に置き換えてお読みください。

Cosminexus 関連

- Cosminexus V11 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)（3021-3-J05）
- Cosminexus V11 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（3021-3-J13）
- Cosminexus V11 アプリケーションサーバ 運用管理ポータル操作ガイド（3021-3-J14）
- Cosminexus V11 アプリケーションサーバ リファレンス コマンド編（3021-3-J15）
- Cosminexus V11 アプリケーションサーバ リファレンス 定義編(サーバ定義)（3021-3-J16）
- Cosminexus V11 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)（3021-3-J17）
- Cosminexus V11 アプリケーションサーバ アプリケーション開発ガイド（3021-3-J20）
- Cosminexus V11 アプリケーションサーバ Web サービス開発ガイド（3021-3-J23）

なお，このマニュアルでは，Cosminexus 関連のマニュアルについて，バージョン番号を省略して表記しています。

HiRDB 関連

- HiRDB Version 10 SQL リファレンス（3020-6-561）

なお，このマニュアルでは，HiRDB 関連のマニュアルについて，バージョン番号を省略して表記しています。

■ 適用 OS の違いによる機能相違点の表記

このマニュアルで説明する機能は，適用 OS の種類（UNIX または Windows）によって，異なる場合があります。OS によって機能差がある場合，OS 名を明記しています。また，OS によってバージョン，リビジョンが異なる場合があります。バージョン，リビジョンによって操作方法などが異なる場合はそれぞれの説明に OS 名，バージョン，およびリビジョンを明記しています。

なお，UNIX および Windows の共通の説明部分に使用している「ディレクトリ」という用語は，Windows の場合は，特に断りのないかぎり，「フォルダ」に読み替えてください。また，UNIX のパスの区切り文字として使用している「/」は，Windows の場合には，特に断りのないかぎり，「¥」に読み替えてください。

目次

1 CSCIW と BPMN2.0 との連携

CSCIW では，BPMN2.0 と連携するための機能として，BPMN 連携機能を提供しています。

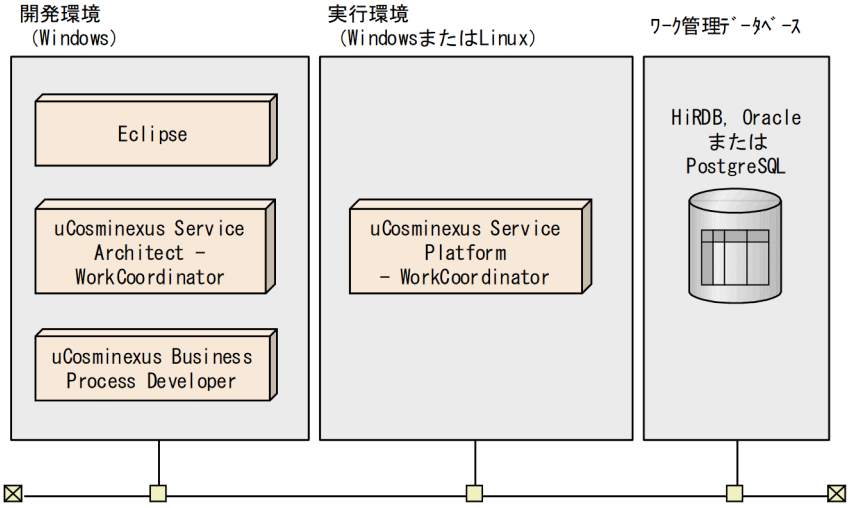
1.1　BPMN 連携機能の概要

BPMN 連携機能とは，CSCIW で BPMN2.0 をサポートするための機能のことです。BPMN 連携機能を使用すると，BPMN2.0 仕様に従ったビジネスプロセスを CSCIW のビジネスプロセスに変換して，CSCIW で使用できます。

1.1.1　ソフトウェア構成

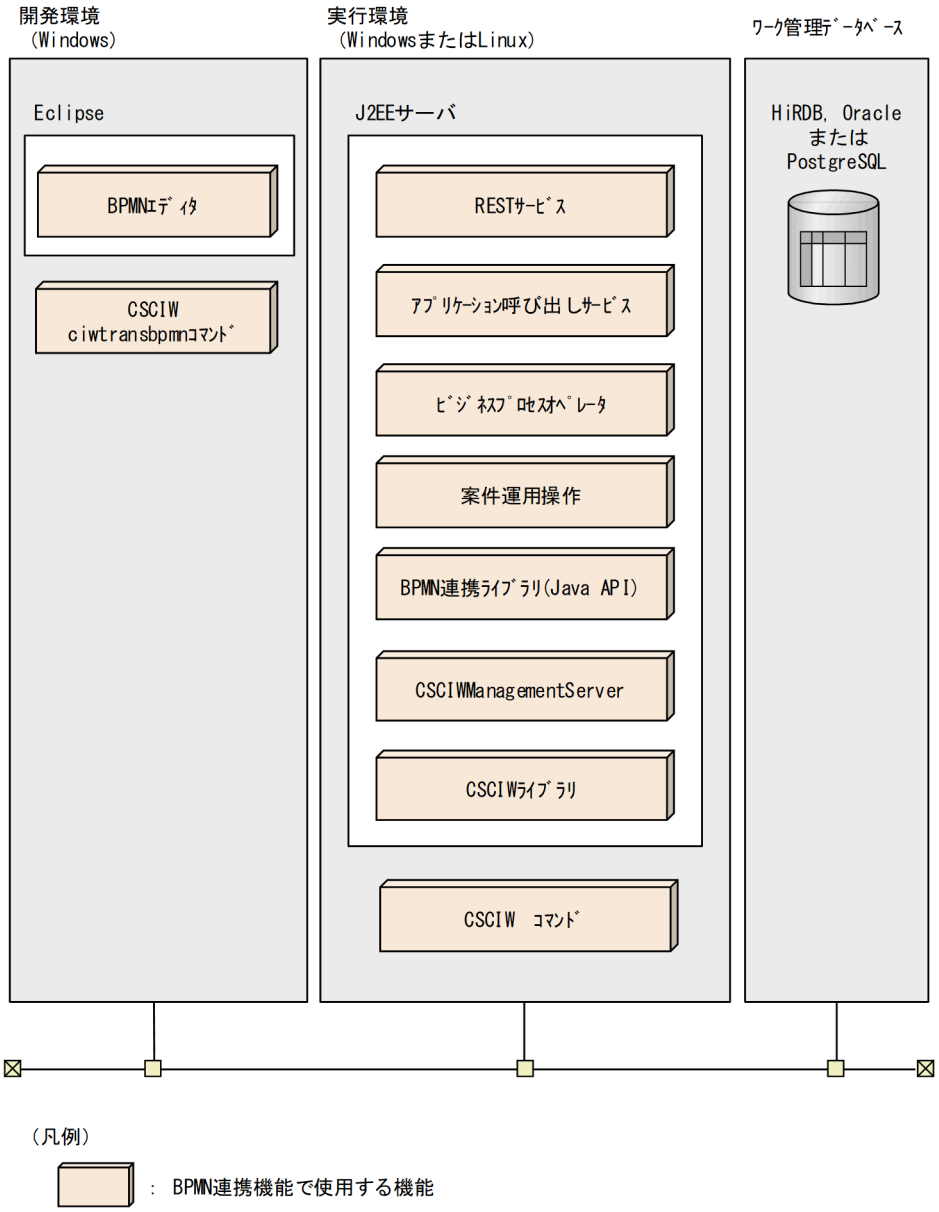
CSCIW では，BPMN 連携機能を実現するためのソフトウェアを提供しています。BPMN 連携機能を使用する場合の各環境のソフトウェア構成を次の図に示します。

図 1-1　ソフトウェア構成



各環境での BPMN 連携機能の機能構成を次の図に示します。

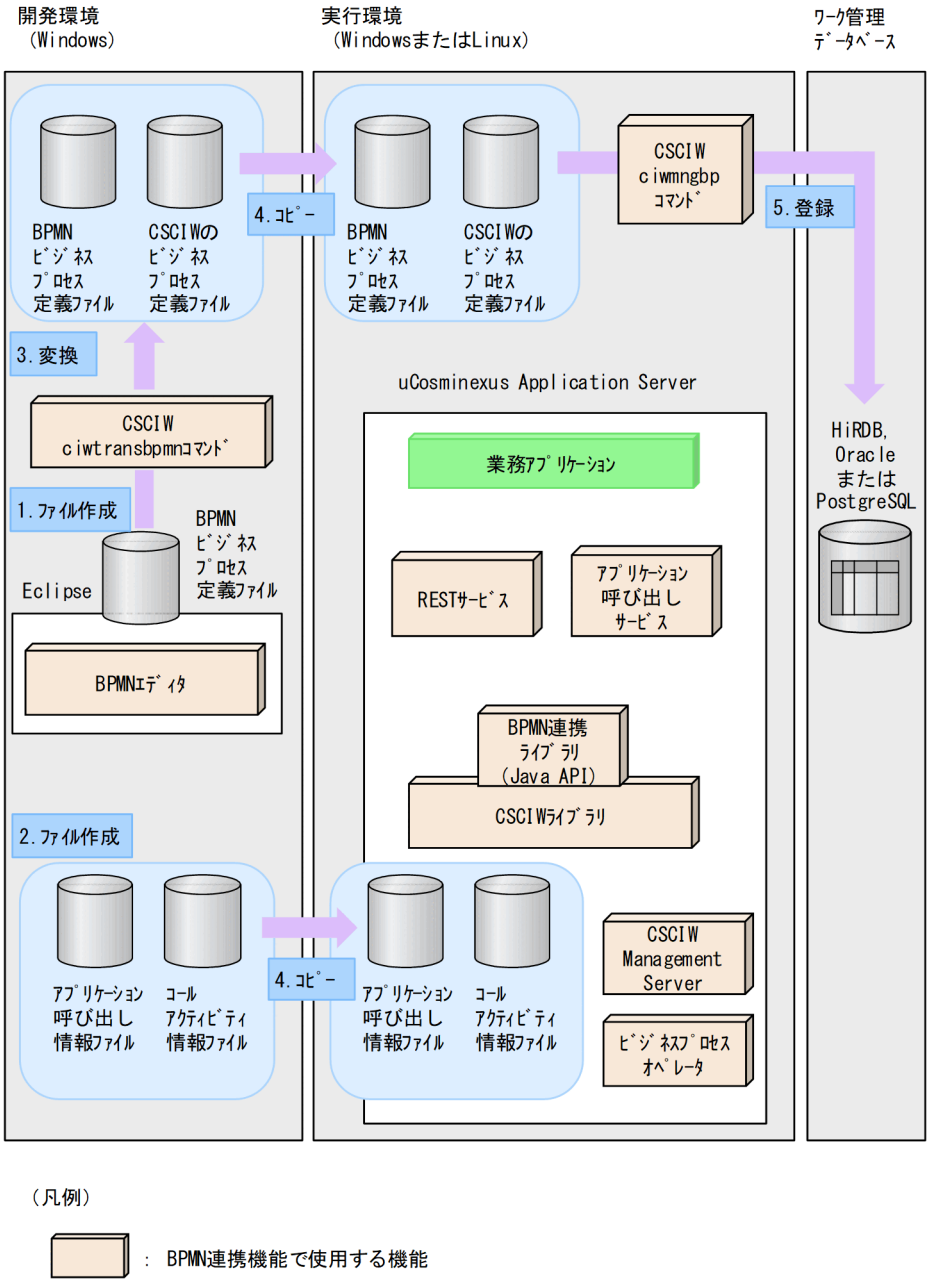
図 1-2　各環境での BPMN 連携機能の機能構成



1.1.2　ビジネスプロセスの開発と実行の流れ

BPMN 連携機能の各機能が連携して処理をする流れ（ファイルの登録・変換に CSCIW コマンドを使用する場合）を次の図で示します。

図 1-3　BPMN 連携機能の各機能の連携イメージ（ファイルの登録・変換に CSCIW コマンドを使用する場合）



[図の説明]

1. BPMN エディタを利用し，XML 形式の BPMN ビジネスプロセス定義ファイルを作成します。
2. 次のファイルを作成します。
   - アプリケーション呼び出し情報ファイル
   - コールアクティビティ情報ファイル
3. ciwtransbpmn コマンドを使用して，BPMN ビジネスプロセス定義ファイルを CSCIW のビジネスプロセス定義ファイルに変換します。
4. 開発環境で作成した次のファイルを実行環境にコピーします。

- アプリケーション呼び出し情報ファイル

  アプリケーション呼び出し情報ファイルの内容に従って，アプリケーション呼び出しサービスで呼び出しが実施されます。

- コールアクティビティ情報ファイル

  コールアクティビティ情報ファイルの内容に従って，BPMN 連携ライブラリでコールアクティビティが処理されます。

- BPMN ビジネスプロセス定義ファイル

  BPMN ビジネスプロセス定義ファイルは，ビジネスプロセスオペレータの利用時に使用されます。

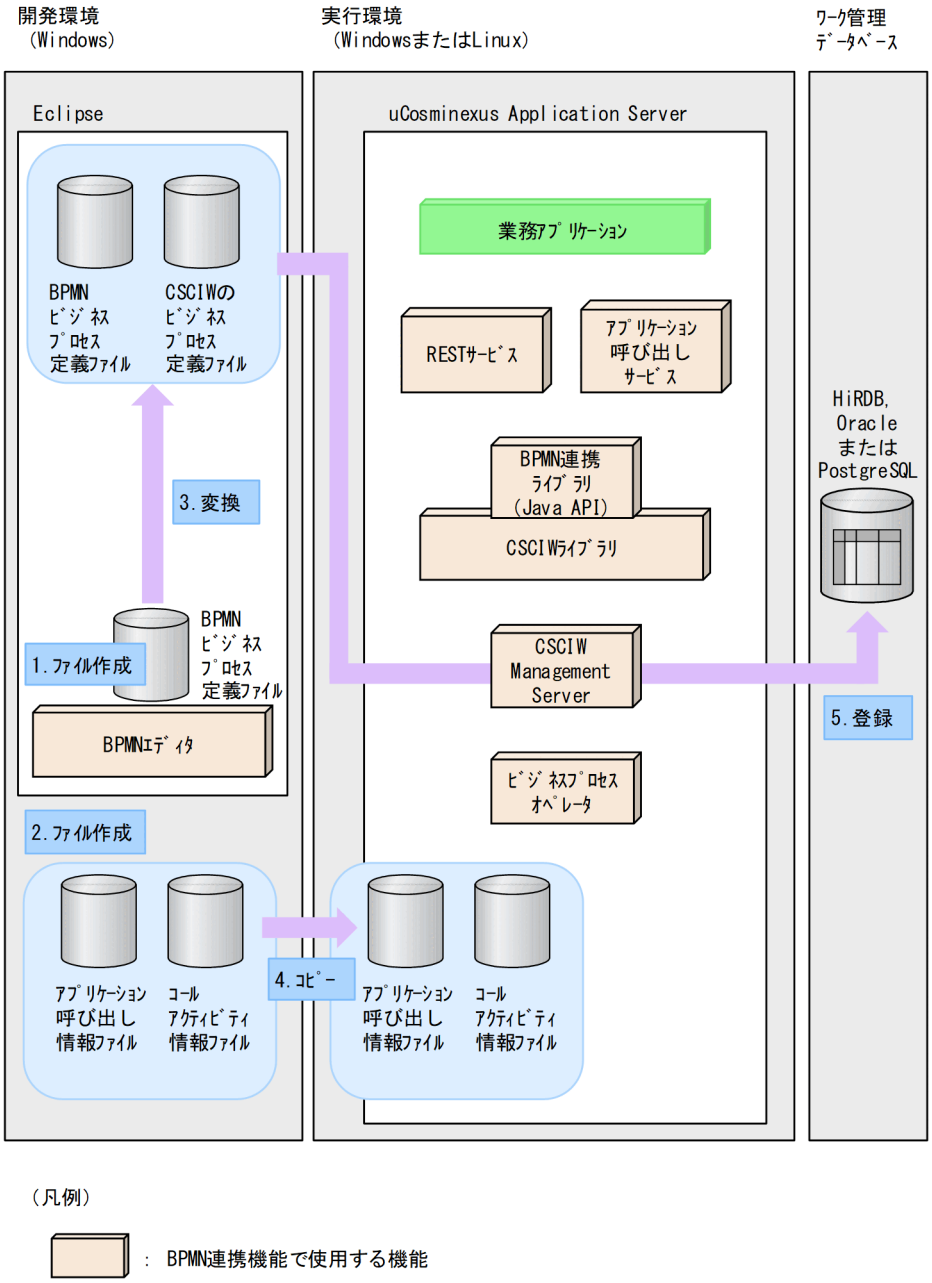
- CSCIW のビジネスプロセス定義ファイル

5. `ciwmngbp` コマンドを使用して，BPMN ビジネスプロセス定義ファイル，および CSCIW のビジネスプロセス定義ファイルをワーク管理データベースに登録します。

> **ヒント**
>
> マニュアル『uCosminexus Service Coordinator Interactive Workflow メッセージ』やマニュアル『uCosminexus Service Coordinator Interactive Workflow コマンド』を参照する場合で，参照先に CSCIW-Definer の記述があるときは，マニュアル内の「CSCIW-Definer」の表記を「BPMN エディタ」に置き換えてお読みください。

BPMN 連携機能の各機能が連携して処理をする流れ（ファイルの登録・変換に BPMN エディタを使用する場合）を次の図で示します。

図 1-4　BPMN 連携機能の各機能の連携イメージ（ファイルの登録・変換に BPMN エディタを使用する場合）



[図の説明]

1. BPMN エディタを利用し，XML 形式の BPMN ビジネスプロセス定義ファイルを作成します。
2. 次のファイルを作成します。
   - アプリケーション呼び出し情報ファイル
   - コールアクティビティ情報ファイル
3. BPMN エディタを使用して，BPMN ビジネスプロセス定義ファイルを CSCIW のビジネスプロセス定義ファイルに変換します。
4. 開発環境で作成した次のファイルを実行環境にコピーします。

- アプリケーション呼び出し情報ファイル

  アプリケーション呼び出し情報ファイルの内容に従って，アプリケーション呼び出しサービスで呼び出しが実施されます。
- コールアクティビティ情報ファイル

  コールアクティビティ情報ファイルの内容に従って，BPMN 連携ライブラリでコールアクティビティが処理されます。

5. CSCIWManagementServer にログインして，BPMN ビジネスプロセス定義ファイル，および CSCIW のビジネスプロセス定義ファイルをワーク管理データベースに登録します。

> **ヒント**
>
> 案件運用操作を使って BPMN ビジネスプロセス定義ファイル，および CSCIW のビジネスプロセス定義ファイルをワーク管理データベースに登録することもできます。詳細については，マニュアル『uCosminexus Service Coordinator Interactive Workflow 案件運用操作ガイド』を参照してください。

1.1.3 システム構成モデル

BPMN 連携機能を使用する場合，次に示す CSCIW のシステム構成モデルを構築できます。

- マルチインスタンス構成
- マルチマシン構成
- マルチプロセス構成
- クラスタ構成

各構成モデルの詳細については，マニュアル『uCosminexus Service Coordinator Interactive Workflow システム構築・運用ガイド』の次の個所を参照してください。

- 「3.2.2　マルチインスタンス構成」
- 「3.2.3　マルチマシン構成」
- 「3.2.4　マルチプロセス構成」
- 「3.2.5　クラスタ構成」

また，各構成モデルを構築する際の注意事項については，マニュアル『uCosminexus Service Coordinator Interactive Workflow システム構築・運用ガイド』の次の個所を参照してください。

- 「4.6.1　マルチインスタンス構成を設定する場合の注意事項」
- 「4.6.2　マルチマシン構成を設定する場合の注意事項」
- 「4.6.3　マルチプロセス構成を設定する場合の注意事項」

- 「4.6.4　クラスタ構成を設定する場合の注意事項」

1.2　BPMN 連携機能で使用できる CSCIW の機能範囲

BPMN 連携機能で使用できる CSCIW の機能範囲を示します。

CSCIW および CSCIW-Definer が提供する機能のうち，BPMN 連携機能を使用した場合に使用できる機能または制限がある機能を表に示します。

表 1-1　BPMN 連携機能で使用できる CSCIW の機能範囲

項番	分類	機能	BPMN 連携機能での使用可否
1	CSCIW-Definer	-	使用できない ビジネスプロセス定義の作成は，BPMN エディタで実行します。
2	案件運用操作	－	使用できる
3	マニュアル『uCosminexus Service Coordinator Interactive Workflow AP 開発ガイド』に記載している API	参照系 API	使用できる
4		更新系 API	使用できない BPMN 連携機能が提供する REST API および Java API を使用して，案件の投入や削除，業務ステップや作業の状態変更などをする必要があります。
5		属性取得 API	使用できる
6	コマンド	ciwchgapwork	使用できる
7		ciwchgdef	使用できない ビジネスプロセス定義の変更はciweditbp コマンドで実行します。
8		ciwchgenv	使用できる
9		ciwcleanup	使用できる
10		ciwdelpi	使用できる
11		ciweditbp	使用できる このコマンドで変更できる内容については，「2.6　登録済みのビジネスプロセス定義を変更する」を参照してください。
12		ciwlistsid	使用できる
13		ciwmngap	使用できる
14		ciwmngapgrp	使用できる
15		ciwmngbp	使用できる
16		ciwmngcr	使用できない 振り分けルールは作成／登録しないため不要です。

項番	分類	機能	BPMN 連携機能での使用可否
17	コマンド	ciwreuseid	使用できる
18		ciwsetenv	使用できる
19		ciwtransbpmn	使用できる

(凡例)

－：該当なし

1.3　BPMN 要素

CSCIW で使用する BPMN 要素の詳細を示します。

メモ

- 変換例などの説明に，要素シグナルに関する情報が記載されている場合がありますが，CSCIW ではシグナルは非サポートです。
- CSCIW のビジネスプロセス定義や案件/業務ステップ/作業の詳細は，マニュアル『uCosminexus Service Coordinator Interactive Workflow システム構築・運用ガイド』の次の説明を参照してください。

  「2.2　ビジネスプロセス定義」

  「2.5.1　案件の基本的な進み方」

  「2.5.2　案件の動的な制御」

1.3.1　BPMN 連携機能で使用できる BPMN 要素

BPMN 連携機能で使用できる BPMN 要素と対応する属性について説明します。

表 1-2　BPMN 連携機能で使用できる BPMN 要素

BPMN 要素		XML 要素	XML 属性
イベント	開始（タイプなし）	startEvent	id, name
	開始（メッセージ）	startEvent	id, name
		messageEventDefinition	messageRef
	イベント・サブプロセス非中断開始（メッセージ）	startEvent	id, name, isInterrupting
		messageEventDefinition	messageRef
	イベント・サブプロセス中断開始（メッセージ）	startEvent	id, name, isInterrupting
		messageEventDefinition	messageRef
	イベント・サブプロセス中断開始（エラー）	startEvent	id, name
		errorEventDefinition	errorRef
	開始（タイマー）	startEvent	id, name
		timerEventDefinition	－
		extensionElements	－
		csciw:periodicDateTime	－
		timeCycle	－

BPMN 要素		XML 要素	XML 属性
イベント	開始（タイマー）	timeDate	－
		timeDuration	－
	イベント・サブプロセス非中断開始（タイマー）	startEvent	id, name, isInterrupting
		timerEventDefinition	－
		extensionElements	－
		csciw:periodicDateTime	－
		csciw:mainProcessData	－
		timeCycle	－
		timeDate	－
		timeDuration	－
	イベント・サブプロセス中断開始（タイマー）	startEvent	id, name, isInterrupting
		timerEventDefinition	－
		extensionElements	－
		csciw:periodicDateTime	－
		csciw:mainProcessData	－
		timeCycle	－
		timeDate	－
		timeDuration	－
	キャッチ（メッセージ）	intermediateCatchEvent	id, name
		messageEventDefinition	messageRef
	キャッチ（リンク）	intermediateCatchEvent	id, name
		linkEventDefinition	[id], name
		source	－
	キャッチ（タイマー）	intermediateCatchEvent	id, name
		timerEventDefinition	－
		extensionElements	－
		csciw:periodicDateTime	－
		csciw:mainProcessData	－
		timeCycle	－
		timeDate	－
		timeDuration	－

BPMN 要素		XML 要素	XML 属性
イベント	スロー（メッセージ）	intermediateThrowEvent	id, name
		messageEventDefinition	messageRef
	スロー（リンク）	intermediateThrowEvent	id, name
		linkEventDefinition	[id], name
		target	―
	境界非中断（メッセージ）	boundaryEvent	id, name, [attachedToRef], cancelActivity
		messageEventDefinition	messageRef
	境界中断（メッセージ）	boundaryEvent	id, name, [attachedToRef], cancelActivity
		messageEventDefinition	messageRef
	境界中断（エラー）	boundaryEvent	id, name, [attachedToRef]
		errorEventDefinition	errorRef
	境界非中断（タイマー）	boundaryEvent	id, name, [attachedToRef], cancelActivity
		timerEventDefinition	―
		extensionElements	―
		csciw:periodicDateTime	―
		csciw:mainProcessData	―
		timeCycle	―
		timeDate	―
		timeDuration	―
	境界中断（タイマー）	boundaryEvent	id, name, [attachedToRef], cancelActivity
		timerEventDefinition	―
		extensionElements	―
		csciw:periodicDateTime	―
		csciw:mainProcessData	―
		timeCycle	―
		timeDate	―
		timeDuration	―
	終了（タイプなし）	endEvent	id, name
	終了（メッセージ）	endEvent	id, name
		messageEventDefinition	messageRef
	終了（エラー）	endEvent	id, name

BPMN 要素		XML 要素	XML 属性
イベント	終了（エラー）	errorEventDefinition	errorRef
	強制終了	endEvent	id, name
		terminateEventDefinition	id
アクティビティ	ユーザタスク	userTask	id, name
		extensionElements	—
		csciw:participantProcessData	—
	ユーザタスク（シーケンシャルマルチインスタンス）	userTask	id, name
		multiInstanceLoopCharacteristics	isSequential
		loopCardinality	—
		completionCondition	—
		extensionElements	—
		csciw:participantProcessData	—
	ユーザタスク（パラレルマルチインスタンス）	userTask	id, name
		multiInstanceLoopCharacteristics	isSequential
		loopCardinality	—
		completionCondition	—
		extensionElements	—
		csciw:participantProcessData	—
	サービスタスク	serviceTask	id, name, operationRef
	サービスタスク（シーケンシャルマルチインスタンス）	serviceTask	id, name, operationRef
		multiInstanceLoopCharacteristics	isSequential
		loopCardinality	—
		completionCondition	—
	サービスタスク（パラレルマルチインスタンス）	serviceTask	id, name, operationRef
		multiInstanceLoopCharacteristics	isSequential
		loopCardinality	—

BPMN 要素		XML 要素	XML 属性
アクティビティ	サービスタスク（パラレルマルチインスタンス）	completionCondition	―
	ビジネスルールタスク	businessRuleTask	id, name, csciw:operationRef
	ビジネスルールタスク（シーケンシャルマルチインスタンス）	businessRuleTask	id, name, csciw:operationRef
		multiInstanceLoopCharacteristics	isSequential
		loopCardinality	―
		completionCondition	―
	ビジネスルールタスク（パラレルマルチインスタンス）	businessRuleTask	id, name, csciw:operationRef
		multiInstanceLoopCharacteristics	isSequential
		loopCardinality	―
		completionCondition	―
	折りたたまれたコールアクティビティ	callActivity	id, name, calledElement
	折りたたまれたコールアクティビティ（シーケンシャルマルチインスタンス）	callActivity	id, name, calledElement
		multiInstanceLoopCharacteristics	isSequential
		loopCardinality	―
		completionCondition	―
	折りたたまれたコールアクティビティ（パラレルマルチインスタンス）	callActivity	id, name, calledElement
		multiInstanceLoopCharacteristics	isSequential
		loopCardinality	―
		completionCondition	―
ゲートウェイ	並列ゲートウェイ	parallelGateway	id, name
	排他ゲートウェイ	exclusiveGateway	id, name, default
	排他イベントゲートウェイ	eventBasedGateway	id, name
接続オブジェクト	シーケンスフロー（条件なし）	sequenceFlow	id, name, [sourceRef], [targetRef]
	シーケンスフロー（条件あり）	sequenceFlow	id, name, [sourceRef], [targetRef]
		conditionExpression	―

BPMN 要素		XML 要素	XML 属性
接続オブジェクト	シーケンスフロー（デフォルト）	sequenceFlow	id, name, [sourceRef], [targetRef]
	メッセージフロー※	messageFlow	name, [sourceRef], [targetRef]
	関連※	association	id, [sourceRef], [targetRef]
	データの関連※	dataInputAssociation	id, csciw:name
		sourceRef	―
		dataOutputAssociation	id, csciw:name
		targetRef	―
データ	データオブジェクト※	dataObject	id, name
	データストア※	dataStoreReference	id, name
コンテナ	プール	participant	id, name, [processRef]
	レーン	laneSet	[id]
		lane	id, name
	プロセス	process	id, name, [isExecutable]
	展開されたサブプロセス	subProcess	id, name, [triggeredByEvent]
	展開されたサブプロセス(シーケンシャルマルチインスタンス)	subProcess	id, name, [triggeredByEvent]
		multiInstanceLoopCharacteristics	isSequential
		loopCardinality	―
		completionCondition	―
	展開されたサブプロセス(パラレルマルチインスタンス)	subProcess	id, name, [triggeredByEvent]
		multiInstanceLoopCharacteristics	isSequential
		loopCardinality	―
		completionCondition	―
	折りたたまれたサブプロセス	subProcess	id, name, [triggeredByEvent]
	折りたたまれたサブプロセス（シーケンシャルマルチインスタンス）	subProcess	id, name, [triggeredByEvent]
		multiInstanceLoopCharacteristics	isSequential
		loopCardinality	―
		completionCondition	―

BPMN 要素		XML 要素	XML 属性
コンテナ	折りたたまれたサブプロセス（パラレルマルチインスタンス）	subProcess	id, name, [triggeredByEvent]
		multiInstanceLoopCharacteristics	isSequential
		loopCardinality	－
		completionCondition	－
	展開されたイベント・サブプロセス	subProcess	id, name, [triggeredByEvent]
	展開されたアドホック・サブプロセス	adHocSubProcess	id, name, cancelRemainingInstances, ordering
		completionCondition	－
	折りたたまれたアドホック・サブプロセス	adHocSubProcess	id, name, cancelRemainingInstances, ordering
		completionCondition	－
その他	テキスト注釈※	textAnnotation	id
		text	－
	グループ※	group	id, name

（凡例）

－：該当なし

[xxx] ：値が自動で設定される属性

注※

BPMN エディタでは定義できますが，実行モデルでは使用しません。

1.3.2 BPMN 連携機能での案件の基本的な進み方

BPMN 連携機能での案件の進み方および BPMN 要素の操作手段について説明します。

BPMN 連携機能での案件の基本的な進み方には，次のようなパターンがあります。

1. 更新系 API による案件の遷移

   開始（タイプなし），ユーザタスク，開始（メッセージ）やキャッチ（メッセージ）などは，業務アプリケーションが REST API や BPMN 連携ライブラリの Java API で提供する更新系 API を発行することで遷移します。

2. アプリケーション呼び出しサービスによる案件の遷移

   サービスタスクやスロー（メッセージ）など，メッセージの送信やサービスの呼び出しなどを行う BPMN 要素に関しては，アプリケーション呼び出しサービスが呼び出しに成功したら，完了させて次に遷移します。

開始（タイマー）やキャッチ（タイマー）など，処理期限に設定されている発火時刻を過ぎたら開始や完了などを行う BPMN 要素に関しては，処理期限に設定されている発火時刻を過ぎたら，アプリケーション呼び出しサービスが案件を開始または完了させて次に遷移します。

3. BPMN 連携ライブラリによる案件の遷移

子案件が終了した際に，子案件の遷移処理の延長で BPMN 連携ライブラリによって遷移します。

図 1-5　BPMN 連携機能での案件の基本的な進み方

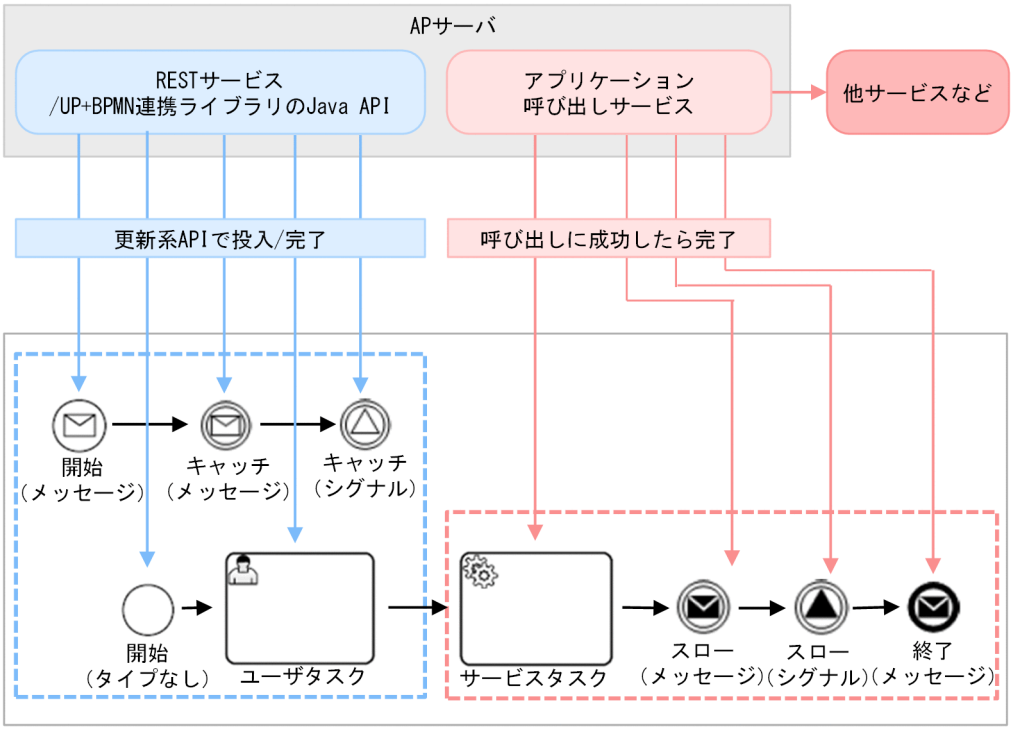


表 1-3　BPMN 要素と操作手段

種別	名称	操作手段
アクティビティ	ユーザタスク※	更新系 API
	サービスタスク※	アプリケーション呼び出しサービス
	ビジネスルールタスク※	アプリケーション呼び出しサービス
	コールアクティビティ※	BPMN 連携ライブラリ
イベント	開始（タイプなし）	更新系 API
	開始（メッセージ）	更新系 API
	終了（タイプなし）	なし（自動で遷移する）
	終了（メッセージ）	アプリケーション呼び出しサービス
	強制終了	なし（自動で遷移する）
	キャッチ（メッセージ）	更新系 API
	キャッチ（リンク）	なし（自動で遷移する）

種別	名称	操作手段
イベント	境界中断（メッセージ）	更新系 API
	イベント・サブプロセス非中断開始（メッセージ）	更新系 API
	スロー（メッセージ）	アプリケーション呼び出しサービス
	スロー（リンク）	なし（自動で遷移する）
	境界非中断（メッセージ）	更新系 API
	イベント・サブプロセス中断開始（メッセージ）	更新系 API
	境界中断（エラー）	更新系 API
	イベント・サブプロセス中断開始（エラー）	更新系 API
	終了（エラー）	アプリケーション呼び出しサービス
	開始（タイマー）	アプリケーション呼び出しサービス
	キャッチ（タイマー）	アプリケーション呼び出しサービス
	境界中断（タイマー）	アプリケーション呼び出しサービス
	境界非中断（タイマー）	アプリケーション呼び出しサービス
	イベント・サブプロセス中断開始（タイマー）	アプリケーション呼び出しサービス
	イベント・サブプロセス非中断開始（タイマー）	アプリケーション呼び出しサービス
ゲートウェイ	排他ゲートウェイ	なし（自動で遷移する）
	並列ゲートウェイ	なし（自動で遷移する）
	排他イベントゲートウェイ	更新系 API
コンテナ	アドホック・サブプロセス	更新系 API

注※

マルチインスタンスも含みます。

遷移先に実行中の業務ステップがすでに存在する場合，遷移先には新たに実行中の業務ステップは生成されません。イベント・サブプロセス非中断開始（メッセージ）や境界非中断（メッセージ）に対して，2回メッセージを送信した場合，2回目のメッセージ送信時にイベント・サブプロセス非中断開始（メッセージ）や境界非中断（メッセージ）の遷移先の業務ステップが実行中のときがあります。このとき，2回目のメッセージ送信時は遷移先に新たに業務ステップは生成されません。

1.3.3　BPMN 要素の CSCIW での扱われ方

(1)　BPMN 要素と CSCIW の変換概要

BPMN 要素と CSCIW の変換について説明します。

BPMN 要素であるタスクやイベントは，CSCIW の業務ステップ定義および作業定義に変換されます。変換された業務ステップ定義および作業定義の定義名は，変換元の BPMN 要素の name 属性値と id 属性値を「_」（半角アンダースコア）でつないだ名称になります。ただし，以降の変換イメージの図では，定義名の id 部分を省略して説明しています。

図 1-6　BPMN 要素と CSCIW の変換概要

<table>
<tr><th colspan="3">BPMN2.0（変換前）</th><th colspan="3">CSCIW（変換後）</th></tr>
<tr><td rowspan="3"> 
STEP1</td><td>name</td><td>STEP1</td><td rowspan="3"> 
STEP1_101
 
STEP1_101</td><td>業務ステップ定義名</td><td>STEP1_101
※1</td></tr>
<tr><td>id</td><td>101</td><td>作業定義名</td><td>STEP1_101
※2</td></tr>
<tr><td>レーン</td><td></td><td>作業者</td><td></td></tr>
<tr><td rowspan="3"> 
msg1受信</td><td>name</td><td>msg1受信</td><td rowspan="3"> 
msg1受信_102
 
msg1受信_102</td><td>業務ステップ定義名</td><td>msg1受信_102
※1</td></tr>
<tr><td>id</td><td>102</td><td>作業定義名</td><td>msg1受信_102</td></tr>
<tr><td>messageRef</td><td>msg1</td><td>作業者</td><td>IWRMSG_msg1</td></tr>
</table>

注※1

サブプロセス（マルチインスタンス）に含まれるタスクやイベントの場合，業務ステップ定義名は「IW<インデクス>_<BPMN 要素の name 属性>_<BPMN 要素の id 属性>」となります。

注※2

タスクがマルチインスタンスである場合，作業定義名は「IWMW_STEP1_101」となります。

基本的に，タスクやイベントから変換された業務ステップ定義には，対応する作業定義が 1 つ生成されます。次の場合，作業定義は複数生成されます。

1. 境界イベントが定義されている場合

次の業務ステップ定義の場合，境界イベントを受信するための作業定義が追加されます。

- 境界イベント（境界中断（メッセージ），境界非中断（メッセージ），または境界中断（エラー））が定義されたアクティビティ内のタスクやイベントから変換された業務ステップ定義
- 境界中断（タイマー）または境界非中断（タイマー）が定義されたタスクから変換された業務ステップ定義

追加する対象は，次のタスクやイベントから変換された業務ステップ定義です。

- ユーザタスク

- サービスタスク
- ビジネスルールタスク
- コールアクティビティ
- キャッチ（メッセージ）
- スロー（メッセージ）
- 終了（メッセージ）
- 排他イベントゲートウェイ
- 終了（エラー）

2. 排他イベントゲートウェイの場合

   排他イベントゲートウェイから変換された業務ステップ定義に，遷移先のすべてのイベントに対応した作業が生成されます。詳細は，「1.3.4　BPMN 要素の CSCIW ビジネスプロセス定義への変換の詳細」の「(28)　排他イベントゲートウェイ」を参照してください。

3. マルチインスタンスの場合

   マルチインスタンスが定義されたアクティビティのタスクから変換された業務ステップ定義には，作業を動的に生成するための作業定義（並列作業）および作業定義（一般作業）が生成されます。生成される対象は，次のタスクから変換された業務ステップ定義です。

   - ユーザタスク（シーケンシャルマルチインスタンス）
   - ユーザタスク（パラレルマルチインスタンス）
   - サービスタスク（シーケンシャルマルチインスタンス）
   - サービスタスク（パラレルマルチインスタンス）
   - ビジネスルールタスク（シーケンシャルマルチインスタンス）
   - ビジネスルールタスク（パラレルマルチインスタンス）
   - コールアクティビティ（シーケンシャルマルチインスタンス）
   - コールアクティビティ（パラレルマルチインスタンス）

(2)　BPMN 要素と CSCIW の作業者

CSCIW の作業の作業者に設定される文字列について説明します。

CSCIW の作業の作業者には，ユーザタスクの場合はレーン名が，イベントの場合は`messageRef` や`errorRef` などから生成された文字列が設定されます。ユーザタスクの作業者の決定方法については，「1.3.4　BPMN 要素の CSCIW ビジネスプロセス定義への変換の詳細」の「(1)　ユーザタスク」を参照してください。

CIWServer クラスの getWorkItemsList メソッドのフィルター条件に状態と作業者を指定することで，指定したユーザの作業（ユーザタスク）の一覧や，指定したイベントを受信待ちしている作業（キャッチなど）の一覧を取得できます。

表 1-4　BPMN 要素と CSCIW の作業の作業者に設定される値

BPMN 要素	作業者
ユーザタスク	<レーン名>※
サービスタスク	IWTOPE_<operationRef>
ビジネスルールタスク	IWTOPE_<operationRef>
コールアクティビティ	IWCALL_GLOBALBP
キャッチ（メッセージ）	IWRMSG_<messageRef>
境界中断（メッセージ）［受信用］	IWRMSG_<messageRef>
スロー（メッセージ）	IWTMSG_<messageRef>
終了（メッセージ）	IWTMSG_<messageRef>
排他イベントゲートウェイ［キャッチ（メッセージ）］	IWRMSG_<messageRef>
境界中断（エラー）［受信用］	IWRERR_<errorRef>
終了（エラー）	IWTERR_<errorRef>
境界非中断（メッセージ）［受信用］	IWRMSG_<messageRef>
キャッチ（タイマー）	IWTTIM_IWTransit
境界中断（タイマー）［受信用］	IWTTIM_IWTransit
境界非中断（タイマー）［受信用］	IWTTIM_IWTransit
イベント・サブプロセス中断開始（タイマー）［受信用］	IWTTIM_IWTransit
イベント・サブプロセス非中断開始（タイマー）［受信用］	IWTTIM_IWTransit

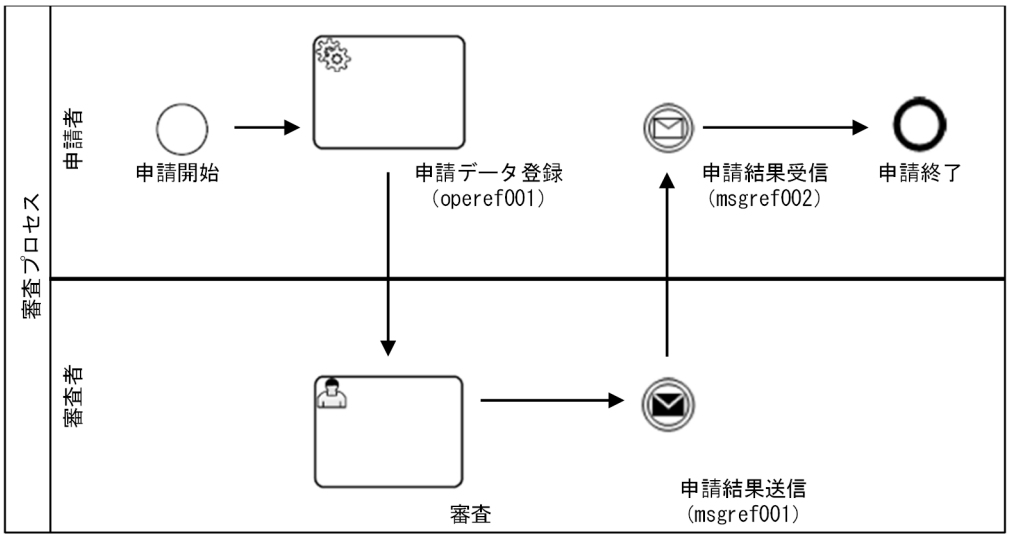
注※

<レーン名>の決定方法については，「1.3.4　BPMN 要素の CSCIW ビジネスプロセス定義への変換の詳細」の「(1)　ユーザタスク」を参照してください。

(3)　BPMN ビジネスプロセス定義の CSCIW での実行例

BPMN ビジネスプロセス定義の実行例で使用する，BPMN ビジネスプロセス定義の例を次の図に示します。

図 1-7　BPMN ビジネスプロセス定義の例



（）内は設定されているoperationRefおよびmessageRef

図に示したビジネスプロセス定義で，申請結果受信が実行中の場合の業務ステップ一覧の例を示します。現在実行中のタスク・イベントに対応する業務ステップの状態は「d（実行中)」になります。実行済みのタスク・イベントに対応する業務ステップの状態は「t（遷移済)」になります。

表 1-5　BPMN のビジネスプロセスと CSCIW の業務ステップ一覧の例

案件 ID	業務ステップ ID	業務ステップ定義名	状態	開始日時※	終了日時※
1001	2001	申請データ登録_101	t（遷移済）	2016/08/01	2016/08/01
1001	2002	審査_102	t（遷移済）	2016/08/01	2016/08/10
1001	2003	申請結果送信_103	t（遷移済）	2016/08/10	2016/08/15
1001	2004	申請結果受信_104	d（実行中）	2016/08/15	-

注※

実際の開始日時および終了日時には，時刻まで含まれます。

作業一覧の例を示します。現在実行中のタスク・イベントに対応する作業の状態は「j（実行開始可能)」になります。実行済みのタスク・イベントに対応する作業の状態は「r（実行済)」になります。

表 1-6　BPMN のビジネスプロセスと CSCIW の作業一覧の例

案件 ID	作業 ID	作業定義名	作業者	状態	作成日時※	終了日時※
1001	2001	申請データ登録_101	IWTOPE_operef001	r（実行済）	2016/08/01	2016/08/01
1001	2002	審査_102	審査者	r（実行済）	2016/08/01	2016/08/10

案件 ID	作業 ID	作業定義名	作業者	状態	作成日時※	終了日時※
1001	2003	申請結果送信_103	IWTMSG_msgref001	r（実行済）	2016/08/10	2016/08/15
1001	2004	申請結果受信_104	IWRMSG_msgref002	j（実行開始可能）	2016/08/15	-

注※

実際の作成日時および終了日時には，時刻まで含まれます。

CIWServer クラスの getWorkItemsList メソッドのフィルター条件に，「ProcessInstanceID=1001 AND Participant='IWRMSG_msgref002' AND StateCode='j'」と指定すると，指定したメッセージで受信待ちしている作業（イベント）を検索できます。

1.3.4　BPMN 要素の CSCIW ビジネスプロセス定義への変換の詳細

BPMN ビジネスプロセス定義ファイルで指定した各 BPMN 要素について，CSCIW ビジネスプロセス定義に変換されたあとのイメージや，CSCIW での動作について説明します。

(1)　ユーザタスク

機能

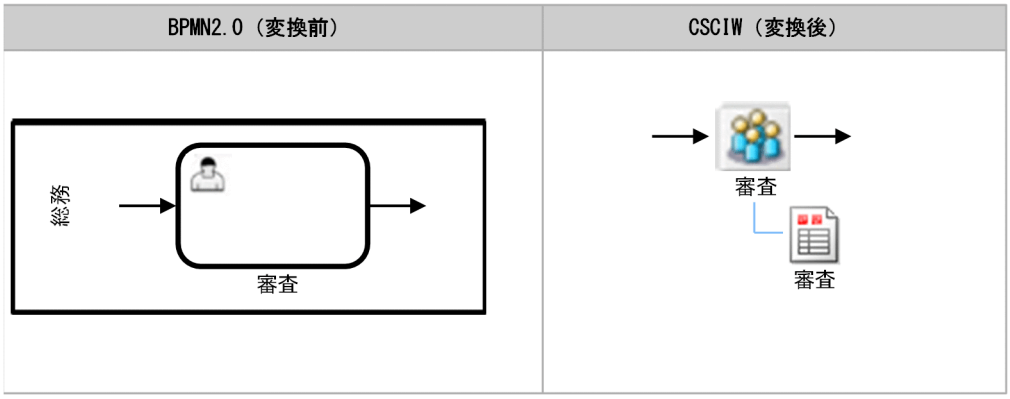
ユーザタスクに遷移すると，個人またはグループがユーザタスクの作業者に設定されます。

変換イメージ

ユーザタスクの BPMN ビジネスプロセス定義から CSCIW ビジネスプロセス定義への変換イメージを示します。

ユーザタスクは業務ステップ定義および作業定義に変換されます。

図 1-8　ユーザタスクの変換イメージ



処理内容

ユーザタスクから変換された業務ステップに遷移すると，ユーザタスクの属するレーン名を作業の作業者に割り当てます※1。レーン名に<SYSTEMID>_WORK_ITEM テーブルのParticipant カラムのバイト数を超える文字列を指定した場合は，文字列のうちParticipant カラムのバイト数を超える部分が切り捨てられます※2。

BPMN エディタでユーザタスクを定義する際にプロセスデータキー名を指定すると，プロセスデータ値を作業の作業者に割り当てます※1。これによって，ユーザタスクの作業者を案件ごとに変更できます。

プロセスデータキー名を指定した場合，ユーザタスクから変換された業務ステップに遷移すると，BPMN エディタで設定したプロセスデータの値を作業の作業者に割り当てます。プロセスデータが存在しない場合は，ユーザタスクの属するレーン名を作業の作業者に割り当てます※1。プロセスデータ値，またはレーン名に<SYSTEMID>_WORK_ITEM テーブルのParticipant カラムのバイト数を超える文字列を指定した場合は，文字列のうちParticipant カラムのバイト数を超える部分が切り捨てられます※2。また，プロセスデータ値に null を指定した場合は，エラーになります。

注※1

レーン名およびプロセスデータ値には「IW」で始まる名称を指定できません。

また，レーン名が未設定の場合は，代わりにプール名が設定されます。レーン名とプール名が未設定の場合は，代わりにビジネスプロセス定義名が設定されます。

注※2

マルチバイト文字を含む文字列の場合，マルチバイト文字の途中でParticipant カラムのバイト数を超えたときは，そのマルチバイト文字も切り捨て対象になります。作業の検索，作業者の変更などの API で，作業者 ID を入力する際は，注意してください。

なお，ユーザタスク（マルチインスタンス），またはサブプロセス（マルチインスタンス）の中にユーザタスクが定義されている場合，マルチインスタンスのインスタンスごとに異なるプロセスデータ値で，作業者を動的変更できます。BPMN エディタでは，{MIIndex}を付与した形式でプロセスデータキー名を指定します。指定例を次に示します。

指定例

```
$SParticipant{MIIndex}
```

(2) ユーザタスク（シーケンシャルマルチインスタンス）

機能

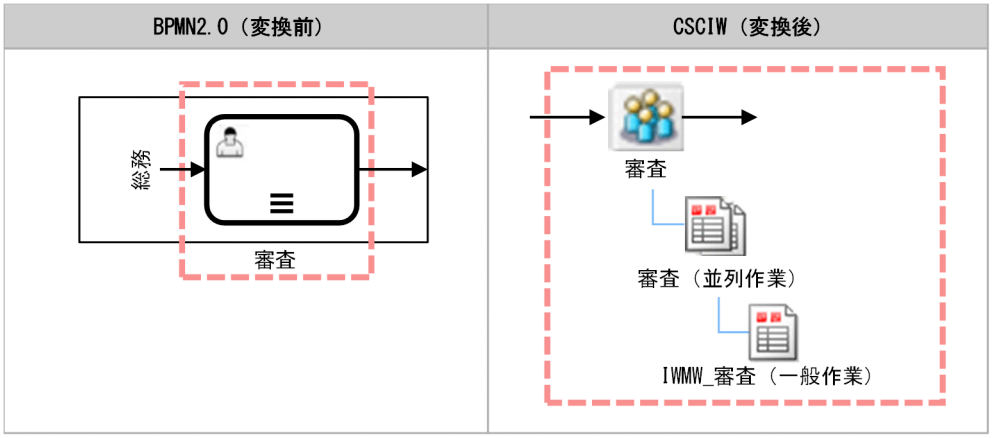
ユーザタスク（シーケンシャルマルチインスタンス）に遷移すると，ユーザは BPMN ビジネスプロセス定義に指定した繰り返し回数（loopCardinality）分の業務処理を逐次処理します。繰り返し回数（loopCardinality）が 0 の場合は何もしないで終了し，次に遷移します。

変換イメージ

シーケンシャルマルチインスタンスがユーザタスクに定義された場合の，BPMN ビジネスプロセス定義から CSCIW ビジネスプロセス定義への変換イメージを示します。

シーケンシャルマルチインスタンスが定義されたユーザタスクは，業務ステップ定義，作業定義（ 審査(並列作業)），および作業定義（IWMW_ 審査（一般作業））に変換されます。

図 1-9　ユーザタスク（シーケンシャルマルチインスタンス）の変換イメージ



処理内容

シーケンシャルマルチインスタンスが定義されたユーザタスクから変換された業務ステップに遷移すると，ユーザタスクから変換された作業（IWMW_審査）を 1 件生成し，ユーザタスクの属するレーン名称またはプロセスデータ値を作業（IWMW_審査）の作業者に割り当てます。

ユーザタスクから変換された作業（IWMW_審査）が完了した場合，マルチインスタンスの完了条件（completionCondition）を評価します。完了条件（completionCondition）が成立した場合，業務ステップは完了します。完了条件（completionCondition）が成立しない場合または完了条件（completionCondition）の設定を省略した場合は，繰り返し回数（loopCardinality）分の作業（IWMW_審査）が完了しているかどうかを評価します。すべての作業（IWMW_審査）が完了した場合，業務ステップは完了します。完了していない作業（IWMW_審査）がある場合，再びユーザタスクから変換された作業（IWMW_審査）を 1 つ生成し，ユーザタスクの属するレーン名称またはプロセスデータ値を作業（IWMW_審査）の作業者に割り当てます。

ユーザタスクから変換された作業（IWMW_審査）が強制終了した場合，ユーザタスクから変換された業務ステップを強制終了します。この場合，ユーザタスクから先には遷移しません。

ユーザタスクから変換された作業（IWMW_審査）の着手と完了を繰り返し遷移して業務ステップが完了すると，遷移先の業務ステップ/制御ノードに遷移します。

(3)　ユーザタスク（パラレルマルチインスタンス）

機能

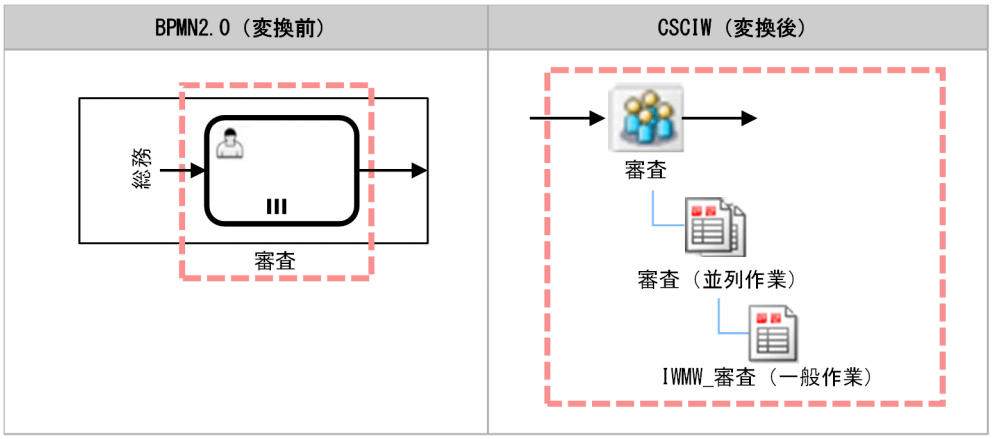
ユーザタスク（パラレルマルチインスタンス）に遷移すると，ユーザは BPMN ビジネスプロセス定義に指定した繰り返し回数（loopCardinality）分の業務処理を並列処理します。繰り返し回数（loopCardinality）が 0 の場合は何もしないで終了し，次に遷移します。

変換イメージ

パラレルマルチインスタンスがユーザタスクに定義された場合の，BPMN ビジネスプロセス定義から CSCIW ビジネスプロセス定義への変換イメージを示します。

パラレルマルチインスタンスが定義されたユーザタスクは，業務ステップ定義，作業定義（ 審査（並列作業)），および作業定義（IWMW_ 審査（一般作業））に変換されます。

図 1-10　ユーザタスク（パラレルマルチインスタンス）の変換イメージ



処理内容

パラレルマルチインスタンスが定義されたユーザタスクから変換された業務ステップに遷移すると，ユーザタスクから変換された作業（IWMW_審査）を繰り返し回数（loopCardinality）分同時に生成し，ユーザタスクの属するレーン名称またはプロセスデータ値を作業（IWMW_審査）の作業者に割り当てます。

ユーザタスクから変換された作業（IWMW_審査）が完了した場合，マルチインスタンスの完了条件（completionCondition）を評価します。完了条件（completionCondition）が成立した場合，業務ステップは完了します。完了条件（completionCondition）が成立しない場合または完了条件（completionCondition）の設定を省略した場合は，繰り返し回数（loopCardinality）分の作業（IWMW_審査）が完了しているかどうかを評価します。すべての作業（IWMW_審査）が完了した場合，業務ステップは完了します。完了していない作業（IWMW_審査）がある場合，業務ステップは実行中のままとなります。

ユーザタスクから変換された作業（IWMW_審査）が強制終了した場合，ユーザタスクから変換された業務ステップを強制終了します。また，すべての実行中の作業（IWMW_審査）も強制終了します。この場合，ユーザタスクから先には遷移しません。

ユーザタスクから変換された作業（IWMW_審査）の着手と完了を繰り返し遷移して業務ステップが完了すると，遷移先の業務ステップ/制御ノードに遷移します。

(4)　サービスタスク

機能

業務処理を，人手の介入なしに自動で実行します。

自動実行する業務処理を次に示します。

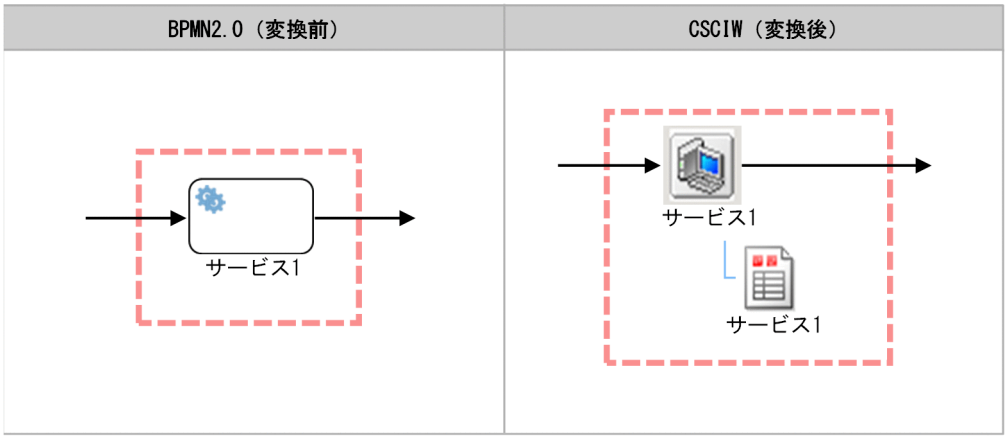
- ユーザが作成した REST アプリケーションの呼び出し
- ユーザが作成した Java オブジェクトの呼び出し

変換イメージ

サービスタスクの変換イメージを示します。

サービスタスクは，業務ステップ定義および作業定義に変換されます。

図 1-11　サービスタスクの変換イメージ



処理内容

サービスタスクの業務実行の動作については，「1.8.1　呼び出し対象の BPMN 要素」を参照してください。

(5)　サービスタスク（シーケンシャルマルチインスタンス）

機能

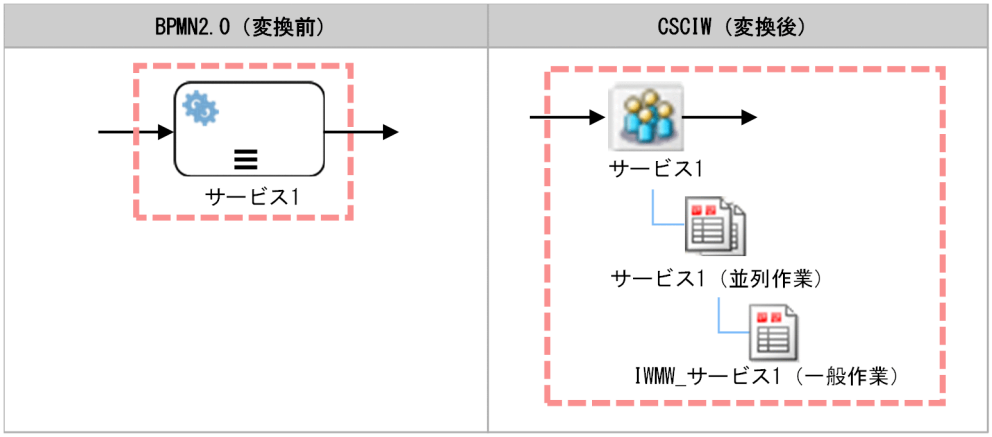
サービスタスク（シーケンシャルマルチインスタンス）に遷移すると，BPMN ビジネスプロセス定義に指定した繰り返し回数（loopCardinality）分の業務処理を自動で逐次処理します。繰り返し回数（loopCardinality）が 0 の場合は何もしないで終了し，次に遷移します。

変換イメージ

シーケンシャルマルチインスタンスがサービスタスクに定義された場合の，BPMN ビジネスプロセス定義から CSCIW ビジネスプロセス定義への変換イメージを示します。

シーケンシャルマルチインスタンスが定義されたサービスタスクは，業務ステップ定義，作業定義（サービス 1（並列作業)），および作業定義（IWMW_サービス 1（一般作業））に変換されます。

図 1-12　サービスタスク（シーケンシャルマルチインスタンス）の変換イメージ



処理内容

シーケンシャルマルチインスタンスが定義されたサービスタスクから変換された業務ステップに遷移すると，1 件の業務処理を自動実行します。

業務処理が完了した場合，マルチインスタンスの完了条件（completionCondition）を評価します。完了条件（completionCondition）が成立した場合，業務ステップは完了します。完了条件（completionCondition）が成立しない場合または完了条件（completionCondition）の設定を省略した場合は，繰り返し回数（loopCardinality）分の業務処理が完了しているかどうかを評価します。すべての業務処理が完了した場合，業務ステップは完了します。完了していない業務処理がある場合，再び業務処理を 1 件自動実行します。

業務処理が強制終了した場合，サービスタスクから変換された業務ステップを強制終了します。この場合，サービスタスクから先には遷移しません。

業務処理の自動実行を繰り返し遷移して業務ステップが完了すると，遷移先の業務ステップ/制御ノードに遷移します。

(6)　サービスタスク（パラレルマルチインスタンス）

機能

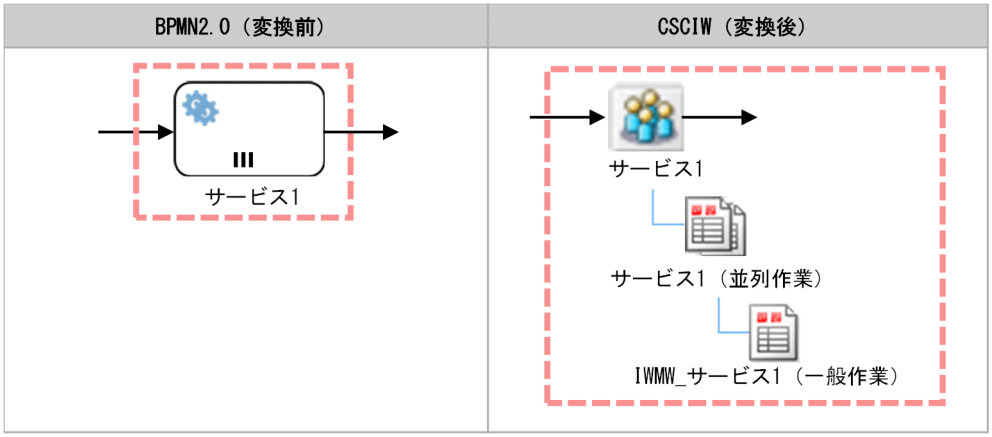
サービスタスク（パラレルマルチインスタンス）に遷移すると，BPMN ビジネスプロセス定義に指定した繰り返し回数（loopCardinality）分の業務処理を自動で並列処理します。繰り返し回数（loopCardinality）が 0 の場合は何もしないで終了し，次に遷移します。

変換イメージ

パラレルマルチインスタンスがサービスタスクに定義された場合の，BPMN ビジネスプロセス定義から CSCIW ビジネスプロセス定義への変換イメージを示します。

パラレルマルチインスタンスが定義されたサービスタスクは，業務ステップ定義，作業定義（サービス 1（並列作業）），および作業定義（IWMW_サービス 1（一般作業））に変換されます。

図 1-13　サービスタスク（パラレルマルチインスタンス）の変換イメージ



処理内容

パラレルマルチインスタンスが定義されたサービスタスクから変換された業務ステップに遷移すると，繰り返し回数（loopCardinality）分の業務処理を並列に自動実行します。

業務処理が完了した場合，マルチインスタンスの完了条件（completionCondition）を評価します。完了条件（completionCondition）が成立した場合，業務ステップは完了します。完了条件（completionCondition）が成立しない場合または完了条件（completionCondition）の設定を省略した場合は，繰り返し回数（loopCardinality）分の業務処理が完了しているかどうかを評価します。すべての業務処理が完了した場合，業務ステップは完了します。完了していない業務処理がある場合，業務ステップは実行中のままとなります。

業務処理が強制終了した場合，サービスタスクから変換された業務ステップを強制終了します。また，すべての業務処理も強制終了します。この場合，サービスタスクから先には遷移しません。

業務処理の自動実行を繰り返し遷移して業務ステップが完了すると，遷移先の業務ステップ/制御ノードに遷移します。

(7)　ビジネスルールタスク

機能

ユーザが作成したビジネスルールエンジンを呼び出す業務処理を，自動で実行します。

自動実行する業務処理を次に示します。

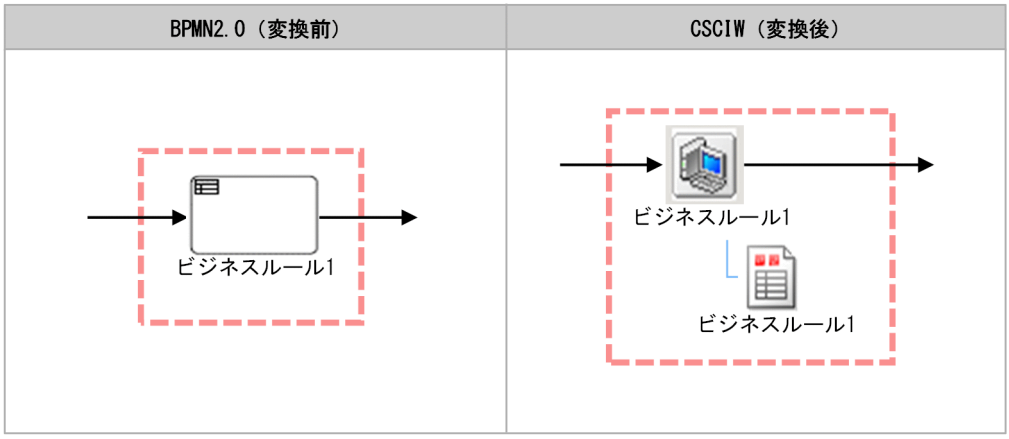
- ユーザが作成した REST アプリケーションの呼び出し
- ユーザが作成した Java オブジェクトの呼び出し

変換イメージ

ビジネスルールタスクの変換イメージを示します。

ビジネスルールタスクは，業務ステップ定義および作業定義に変換されます。

図 1-14　ビジネスルールタスクの変換イメージ



処理内容

ビジネスルールタスクの業務実行の動作については，「1.8.1　呼び出し対象の BPMN 要素」を参照してください。

(8)　ビジネスルールタスク（シーケンシャルマルチインスタンス）

機能

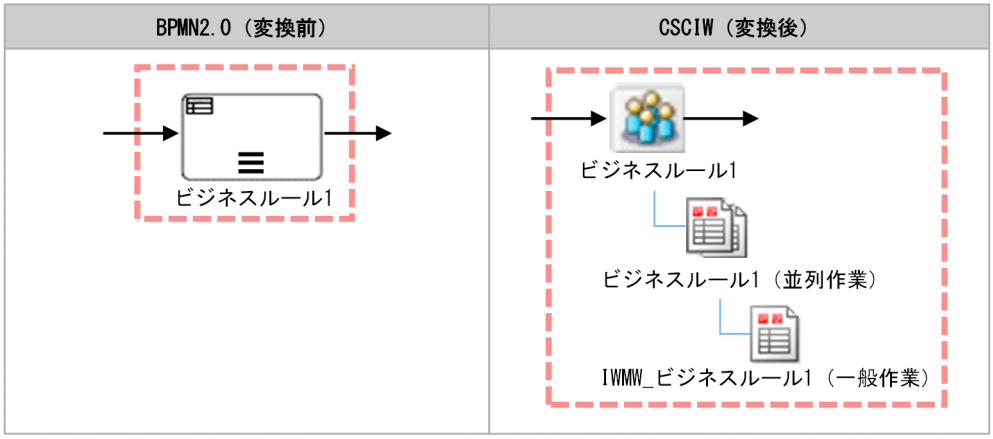
ビジネスルールタスク（シーケンシャルマルチインスタンス）に遷移すると，BPMN ビジネスプロセス定義に指定した繰り返し回数（loopCardinality）分，ユーザが作成したビジネスルールエンジンを呼び出す業務処理を，自動で逐次処理します。繰り返し回数（loopCardinality）が 0 の場合は何もしないで終了し，次に遷移します。

変換イメージ

シーケンシャルマルチインスタンスがビジネスルールタスクに定義された場合の，BPMN ビジネスプロセス定義から CSCIW ビジネスプロセス定義への変換イメージを示します。

シーケンシャルマルチインスタンスが定義されたビジネスルールタスクは，業務ステップ定義，作業定義（ビジネスルール 1（並列作業）），および作業定義（IWMW_ビジネスルール 1（一般作業））に変換されます。

図 1-15　ビジネスルールタスク（シーケンシャルマルチインスタンス）の変換イメージ



処理内容

シーケンシャルマルチインスタンスが定義されたビジネスルールタスクから変換された業務ステップに遷移すると，1 件の業務処理を自動実行します。

業務処理が完了した場合，マルチインスタンスの完了条件（completionCondition）を評価します。完了条件（completionCondition）が成立した場合，業務ステップは完了します。完了条件（completionCondition）が成立しない場合または完了条件（completionCondition）の設定を省略した場合は，繰り返し回数（loopCardinality）分の業務処理が完了しているかどうかを評価します。すべての業務処理が完了した場合，業務ステップは完了します。完了していない業務処理がある場合，再び業務処理を 1 件自動実行します。

業務処理が強制終了した場合，ビジネスルールタスクから変換された業務ステップを強制終了します。この場合，ビジネスルールタスクから先には遷移しません。

業務処理の自動実行を繰り返し遷移して業務ステップが完了すると，遷移先の業務ステップ/制御ノードに遷移します。

(9)　ビジネスルールタスク（パラレルマルチインスタンス）

機能

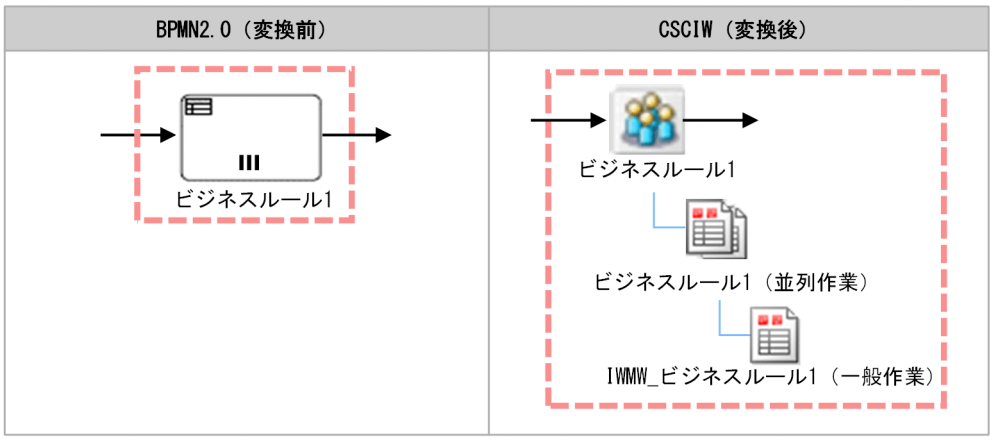
ビジネスルールタスク（パラレルマルチインスタンス）に遷移すると，BPMN ビジネスプロセス定義に指定した繰り返し回数（loopCardinality）分，ユーザが作成したビジネスルールエンジンを呼び出す業務処理を，自動で並列処理します。繰り返し回数（loopCardinality）が 0 の場合は何もしないで終了し，次に遷移します。

変換イメージ

パラレルマルチインスタンスがビジネスルールタスクに定義された場合の，BPMN ビジネスプロセス定義から CSCIW ビジネスプロセス定義への変換イメージを示します。

パラレルマルチインスタンスが定義されたビジネスルールタスクは，業務ステップ定義，作業定義（ビジネスルール 1（並列作業)），および作業定義（IWMW_ビジネスルール 1（一般作業)）に変換されます。

図 1-16　ビジネスルールタスク（パラレルマルチインスタンス）の変換イメージ



処理内容

パラレルマルチインスタンスが定義されたビジネスルールタスクから変換された業務ステップに遷移すると，繰り返し回数（loopCardinality）分の業務処理を並列に自動実行します。

業務処理が完了した場合，マルチインスタンスの完了条件（completionCondition）を評価します。完了条件（completionCondition）が成立した場合，業務ステップは完了します。完了条件（completionCondition）が成立しない場合または完了条件（completionCondition）の設定を省略した場合は，繰り返し回数（loopCardinality）分の業務処理が完了しているかどうかを評価します。すべての業務処理が完了した場合，業務ステップは完了します。完了していない業務処理がある場合，業務ステップは実行中のままとなります。

業務処理が強制終了した場合，ビジネスルールタスクから変換された業務ステップを強制終了します。また，すべての業務処理も強制終了します。この場合，ビジネスルールタスクから先には遷移しません。

業務処理の自動実行を繰り返し遷移して業務ステップが完了すると，遷移先の業務ステップ/制御ノードに遷移します。

(10)　折りたたまれたサブプロセス

機能

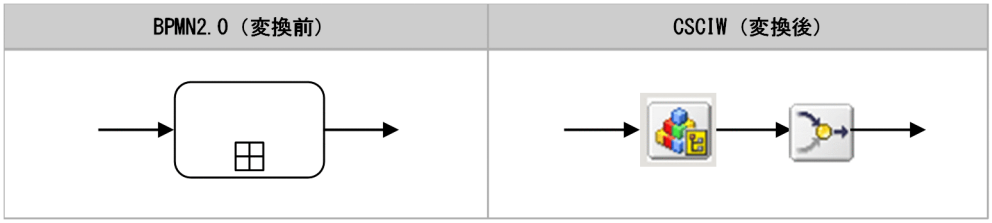
折りたたまれたサブプロセスは，親ビジネスプロセス内のアクティビティになります。

また，別ビジネスプロセスにはならないで，親ビジネスプロセスの遷移元からの処理の延長で動作します。このため，折りたたまれたサブプロセス内のフロー定義は，親ビジネスプロセス定義のフロー定義の延長と考えて問題ありません。

変換イメージ

BPMN ビジネスプロセス定義から CSCIW ビジネスプロセス定義への変換イメージを示します。折りたたまれたサブプロセスは階層定義および先着ノードに変換されます。

図 1-17　折りたたまれたサブプロセスの変換イメージ



処理内容

折りたたまれたサブプロセスから変換された階層定義に遷移すると，階層定義内のソースノードの次の業務ステップ / 制御ノードに遷移します。

(11)　展開されたサブプロセス

機能

展開されたサブプロセスは，親ビジネスプロセス内のアクティビティになります。

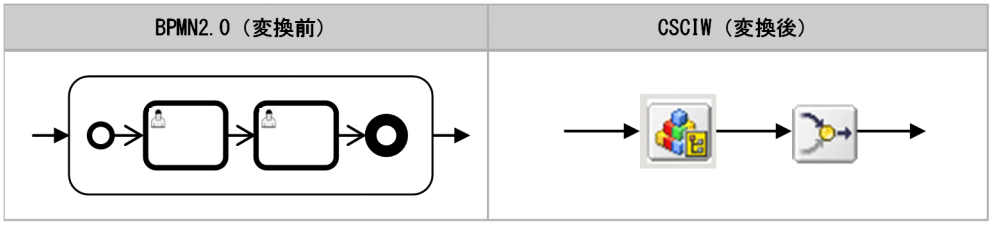
また，別ビジネスプロセスとはならないで，親ビジネスプロセスの遷移元からの処理の延長で動作します。

このため，展開されたサブプロセス内のフロー定義は，親ビジネスプロセス定義のフロー定義の延長と考えて問題ありません。

変換イメージ

BPMN ビジネスプロセス定義から CSCIW ビジネスプロセス定義への変換イメージを示します。展開されたサブプロセスは階層定義および先着ノードに変換されます。

図 1-18　展開されたサブプロセスの変換イメージ



処理内容

展開されたサブプロセスから変換された階層定義に遷移すると，階層定義内のソースノードの次の業務ステップ / 制御ノードに遷移します。

(12)　コールアクティビティ

機能

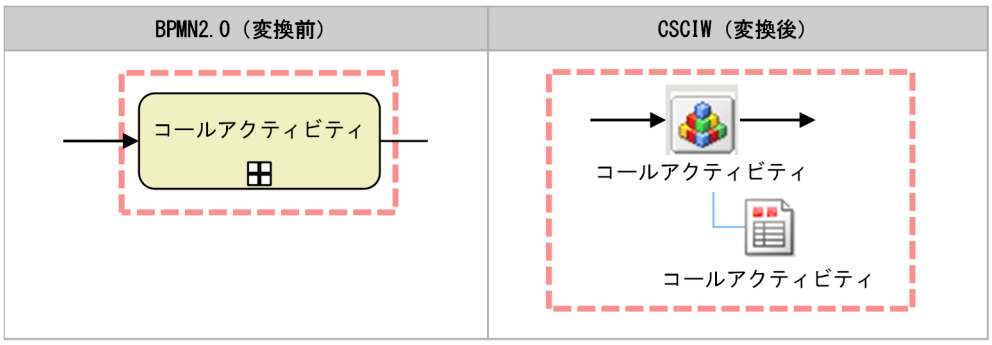
ほかのビジネスプロセス定義に子案件を投入します。子案件が完了すると呼び元のコールアクティビティも完了します。

コールアクティビティには，開始（タイプなし）が定義してあるビジネスプロセス定義を指定する必要があります。

変換イメージ

コールアクティビティの BPMN ビジネスプロセス定義から CSCIW ビジネスプロセス定義への変換イメージを示します。コールアクティビティは業務ステップ定義および作業定義に変換されます。

図 1-19　コールアクティビティの変換イメージ



処理内容

コールアクティビティから変換された業務ステップに遷移すると，calledElement 属性値で指定されたコールアクティビティ情報ファイルに従って子案件を投入します。子案件の投入時にコールアクティビティ情報ファイルに従って親案件のプロセスデータを子案件に登録します。

子案件が終了した場合の動作

- 子案件が完了した場合

  親案件のコールアクティビティから変換された業務ステップを完了して，次の業務ステップ/制御ノードに遷移します。業務ステップ完了時に，コールアクティビティ情報ファイルに従って子案件のプロセスデータを親案件に登録します。

- 子案件が強制終了した場合

  親案件のコールアクティビティから変換された業務ステップを強制終了します。この場合，コールアクティビティから先には遷移しません。

親案件またはコールアクティビティが終了した場合の動作

- 親案件が強制終了した場合

  すべての実行中の子案件も強制終了します。

- 親案件のコールアクティビティから変換された業務ステップまたは作業が，完了または強制終了した場合

  対象のコールアクティビティから投入された子案件も強制終了します。

案件の削除処理

- API やコマンドでは子案件を直接削除できません。
- ルート案件を削除すると，子案件もまとめて削除されます。

(13)　コールアクティビティ（シーケンシャルマルチインスタンス）

機能

コールアクティビティ（シーケンシャルマルチインスタンス）に遷移すると，BPMN ビジネスプロセス定義に指定した繰り返し回数（loopCardinality）分の子案件を逐次投入します。繰り返し回数（loopCardinality）が 0 の場合は何もしないで終了し，次に遷移します。

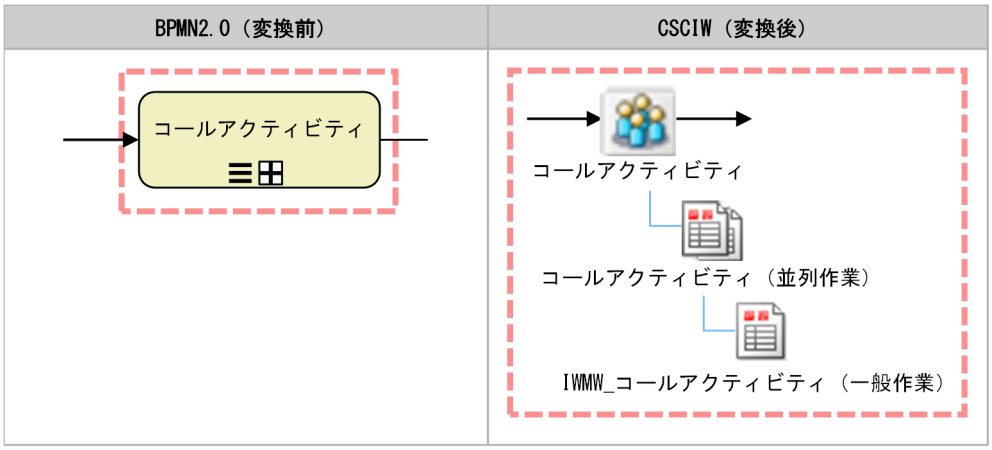
コールアクティビティ（シーケンシャルマルチインスタンス）には開始（タイプなし）が定義してあるビジネスプロセス定義を指定する必要があります。

変換イメージ

シーケンシャルマルチインスタンスがコールアクティビティに定義された場合の，BPMN ビジネスプロセス定義から CSCIW ビジネスプロセス定義への変換イメージを示します。

シーケンシャルマルチインスタンスが定義されたコールアクティビティは，業務ステップ定義，作業定義（コールアクティビティ（並列作業）），および作業定義（IWMW_コールアクティビティ（一般作業））に変換されます。

図 1-20　コールアクティビティ（シーケンシャルマルチインスタンス）の変換イメージ



処理内容

シーケンシャルマルチインスタンスが定義されたコールアクティビティから変換された業務ステップに遷移すると，1 件の子案件を投入します。子案件の投入時に，コールアクティビティ情報ファイルに従って，親案件のプロセスデータを子案件に登録します。

子案件が終了した場合の動作

- 子案件が完了した場合

  コールアクティビティ情報ファイルに従って子案件のプロセスデータを親案件に登録し，マルチインスタンスの完了条件（completionCondition）を評価します。

  完了条件（completionCondition）が 成立した場合，業務ステップは完了します。完了条件（completionCondition）が成立しない場合または完了条件（completionCondition）の設定を省略した場合は，繰り返し回数（loopCardinality）分の子案件が完了しているかどうかを評価します。すべての子案件が完了した場合，業務ステップは完了します。完了していない子案件がある場合，再び子案件を 1 件投入します。子案件の投入時に，コールアクティビティ情報ファイルに従って親案件のプロセスデータを子案件に登録します。

- 子案件が強制終了した場合

  親案件のコールアクティビティから変換された業務ステップを強制終了します。この場合，コールアクティビティから先には遷移しません。

親案件またはコールアクティビティが終了した場合の動作

- 親案件が強制終了した場合

  すべての実行中の子案件も強制終了します。

- 親案件のコールアクティビティから変換された業務ステップまたは作業が，完了または強制終了した場合

  対象のコールアクティビティから投入された子案件も強制終了します。

子案件の投入，完了を繰り返し遷移して業務ステップが完了すると，次の業務ステップ/制御ノードに遷移します。

(14) コールアクティビティ（パラレルマルチインスタンス）

機能

コールアクティビティ（パラレルマルチインスタンス）に遷移すると，BPMN ビジネスプロセス定義に指定した繰り返し回数（loopCardinality）分の子案件を並列に投入します。繰り返し回数（loopCardinality）が 0 の場合は何もしないで終了し，次に遷移します。

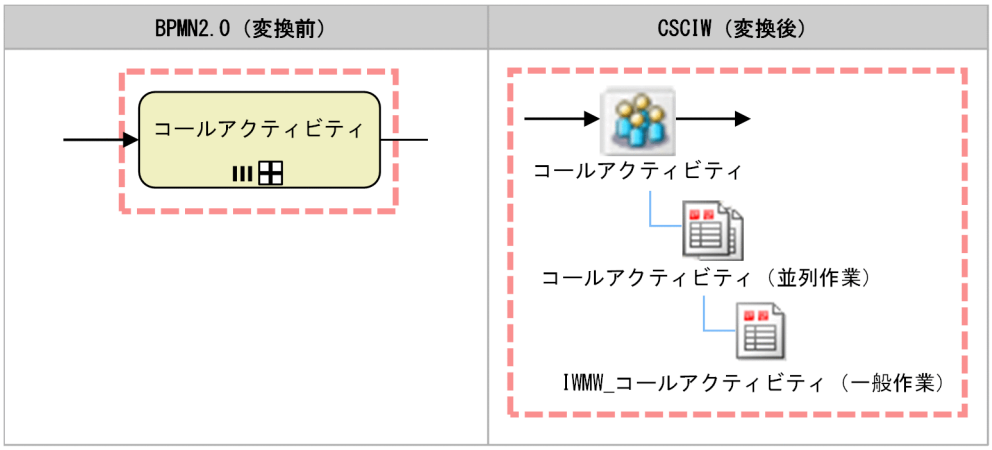
コールアクティビティ（パラレルマルチインスタンス）には開始（タイプなし）が定義してあるビジネスプロセス定義を指定する必要があります。

変換イメージ

パラレルマルチインスタンスがコールアクティビティに定義された場合の，BPMN ビジネスプロセス定義から CSCIW ビジネスプロセス定義への変換イメージを示します。

パラレルマルチインスタンスが定義されたコールアクティビティは，業務ステップ定義，作業定義（コールアクティビティ（並列作業)），および作業定義（IWMW_コールアクティビティ（一般作業)）に変換されます。

図 1-21　コールアクティビティ（パラレルマルチインスタンス）の変換イメージ



処理内容

パラレルマルチインスタンスが定義されたコールアクティビティから変換された業務ステップに遷移すると，繰り返し回数（loopCardinality）分の子案件を同時に投入します。子案件の投入時に，コールアクティビティ情報ファイルに従って，親案件のプロセスデータを子案件に登録します。

子案件が終了した場合の動作

- 子案件が完了した場合

  コールアクティビティ情報ファイルに従って子案件のプロセスデータを親案件に登録し，マルチインスタンスの完了条件（completionCondition）を評価します。

  完了条件（completionCondition）が 成立した場合，業務ステップは完了します。完了条件（completionCondition）が成立しない場合または完了条件（completionCondition）の設定を省略した場合は，繰り返し回数（loopCardinality）分の子案件が完了しているかどうかを評価します。すべての子案件が完了した場合，業務ステップは完了します。完了していない子案件がある場合，業務ステップは実行中のままとなります。

- 子案件が強制終了した場合

  親案件のコールアクティビティから変換された業務ステップを強制終了します。また，すべての実行中の子案件も強制終了します。この場合，コールアクティビティから先には遷移しません。

親案件またはコールアクティビティが終了した場合の動作

- 親案件が強制終了した場合

  すべての実行中の子案件も強制終了します。

- 親案件のコールアクティビティから変換された業務ステップまたは作業が，完了または強制終了した場合

  対象のコールアクティビティから投入された子案件も強制終了します。

子案件の投入，完了を繰り返し遷移して業務ステップが完了すると，次の業務ステップ/制御ノードに遷移します。

(15)　イベント・サブプロセス

機能

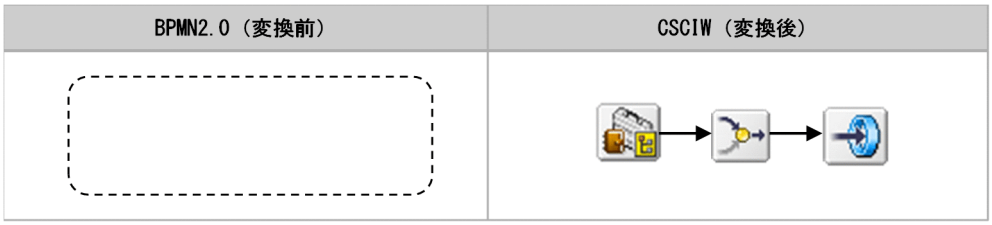
親ビジネスプロセスとイベント・サブプロセスは，並行実行されます。

なお，親ビジネスプロセスとイベント・サブプロセスを合わせて，1 つのビジネスプロセス定義とします。変換後の CSCIW ビジネスプロセス定義も，親ビジネスプロセスとイベント・サブプロセスは同じビジネスプロセス定義ファイル内に定義します。

変換イメージ

BPMN ビジネスプロセス定義から CSCIW ビジネスプロセス定義への変換イメージの例を示します。イベント・サブプロセスは，階層定義，先着ノードおよびシンクノードに変換されます。イベント・サブプロセス内の各 BPMN 要素はそれぞれ変換されて，階層定義内に定義されます。

図 1-22　イベント・サブプロセスの変換イメージ



処理内容

イベント・サブプロセス内に定義された内容に従って遷移します。

(16)　開始（タイプなし）

機能

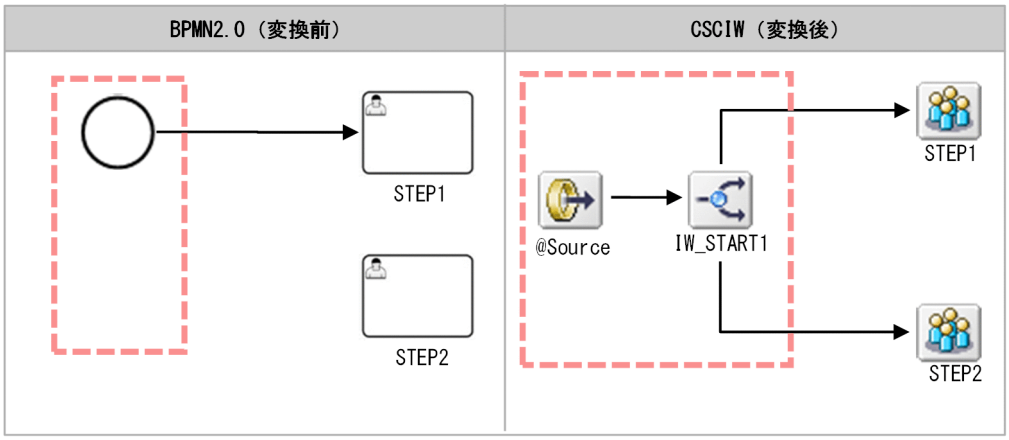
指定されたビジネスプロセスを，開始（タイプなし）イベントから開始します。

開始（タイプなし）は，省略もできます。

変換イメージ

開始（タイプなし）イベントの BPMN ビジネスプロセス定義から CSCIW ビジネスプロセス定義への変換イメージを示します。開始（タイプなし）イベントは，ソースノードおよび分業ノードに変換されます。開始（タイプなし）が省略されている場合も，ソースノードおよび分業ノードに変換されます。

図 1-23　開始（タイプなし）の変換イメージ



処理内容

案件投入（イベントなし）API が呼び出されると案件が生成され，開始します。複数の開始（タイプなし）イベントがある場合は，すべての遷移先の業務ステップに遷移します。

開始（タイプなし）（省略されている場合も含む）が定義されていないビジネスプロセスに対して案件投入（メッセージ）API が指定された場合は，例外が発生します。

(17)　開始（メッセージ）

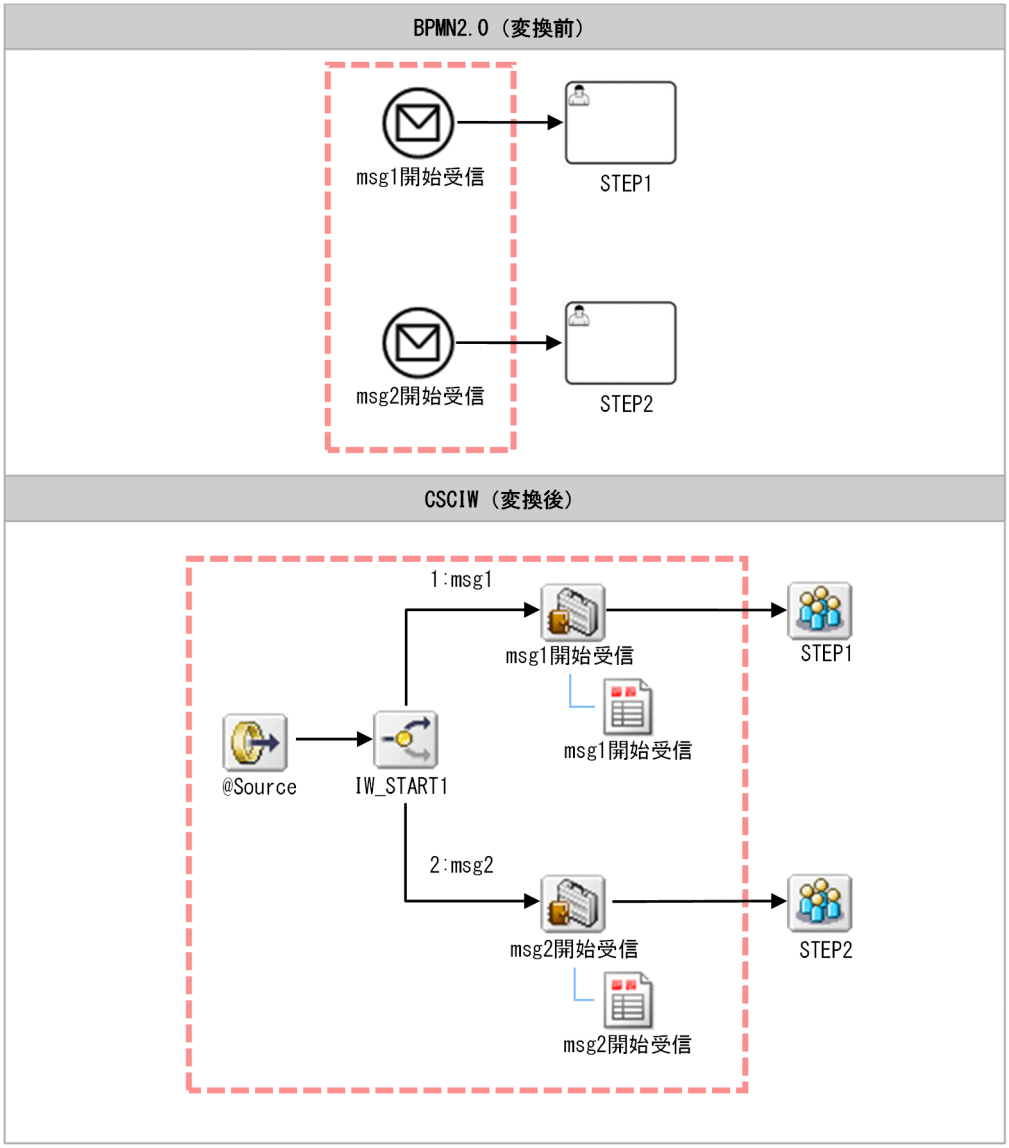
機能

指定されたビジネスプロセスを，指定したメッセージの開始（メッセージ）イベントから開始します。

変換イメージ

開始（メッセージ）イベントの BPMN ビジネスプロセス定義から CSCIW ビジネスプロセス定義への変換イメージを示します。開始（メッセージ）イベントは，ソースノード，分岐ノード，分岐条件，業務ステップ定義，および作業定義に変換されます。

図 1-24　開始（メッセージ）の変換イメージ



処理内容

案件投入（メッセージ）API が呼び出されると案件を生成し，開始します。指定されたmessageRef に従って分岐し，開始（メッセージ）から変換された業務ステップに遷移します。開始（メッセージ）から変換された業務ステップは自動的に完了し，次の業務ステップ / 制御ノードに遷移します。案件投入（イベントなし）API が呼び出されると分岐条件が成立しないため，例外が発生します。

(18)　終了（タイプなし）

機能

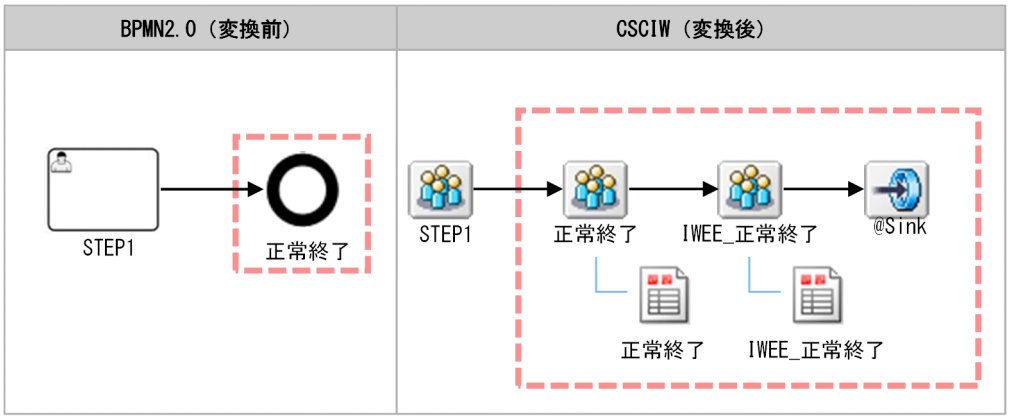
終了（タイプなし）イベントに遷移すると，ほかに実行中のアクティビティがない場合は，案件が完了します。ほかに実行中のアクティビティがある場合は，案件は完了しません。

変換イメージ

終了（タイプなし）イベントの BPMN ビジネスプロセス定義から CSCIW ビジネスプロセス定義への変換イメージを示します。

終了（タイプなし）イベントは，業務ステップ定義，作業定義，およびシンクノードに変換されます。

図 1-25　終了（タイプなし）の変換イメージ



処理内容

終了（タイプなし）から変換されたシンクノードに遷移すると，ほかに実行中の業務ステップがない場合は，案件が完了します。ほかに実行中の業務ステップがある場合は，案件が完了しません。

サブプロセスの場合はシンクノードに遷移すると，同一のサブプロセス内にほかに実行中の業務ステップがない場合は，サブプロセスの先に遷移します。同一のサブプロセス内に実行中の業務ステップがある場合は，サブプロセスの先には遷移しません。

子案件の場合は，案件の完了時に呼び元のコールアクティビティから変換された作業を完了します。

(19)　終了（メッセージ）

機能

終了（メッセージ）イベントに遷移すると，メッセージをスローし，ほかに実行中のアクティビティがない場合は，案件が完了します。ほかに実行中のアクティビティがある場合，案件は完了しません。

終了（メッセージ）イベントのメッセージスロー先を示します。

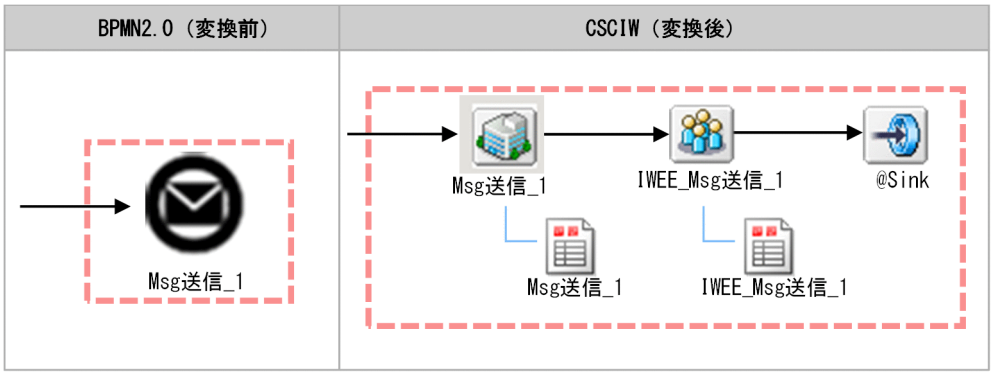
- 開始（メッセージ）
- キャッチ（メッセージ）
- 境界中断（メッセージ）
- イベント・サブプロセス非中断開始（メッセージ）
- REST アプリケーション

変換イメージ

終了（メッセージ）イベントの BPMN ビジネスプロセス定義から CSCIW ビジネスプロセス定義への変換イメージを示します。

終了（メッセージ）イベントは，業務ステップ定義，作業定義，およびシンクノードに変換されます。

図 1-26　終了（メッセージ）の変換イメージ



処理内容

終了（メッセージ）の動作については，「1.8.1　呼び出し対象の BPMN 要素」を参照してください。

メッセージの送信に成功した場合，シンクノードに遷移します。シンクノードに遷移した場合の動作については，「1.3.4　BPMN 要素の CSCIW ビジネスプロセス定義への変換の詳細」の「(18)　終了（タイプなし）」を参照してください。

(20)　強制終了

機能

共通設定ファイルのTerminateCompatMode パラメタにtrue を指定している場合（03-10 までと同じ）

強制終了イベントに遷移すると，ほかに実行中のアクティビティがある場合でも案件を強制終了します。

共通設定ファイルのTerminateCompatMode パラメタにfalse を指定している場合

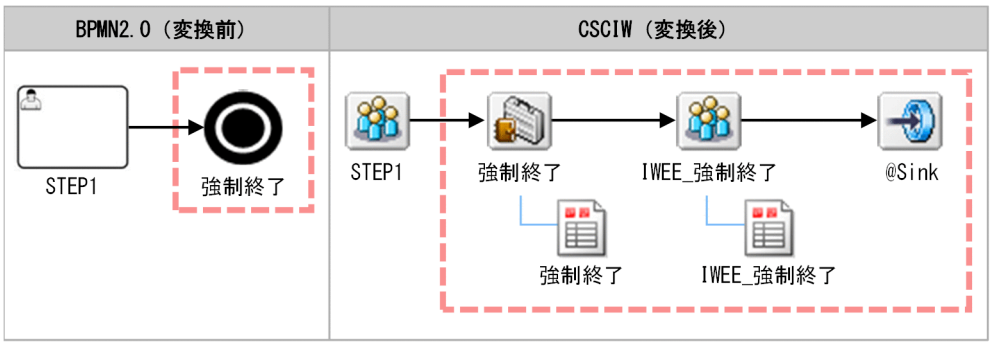
強制終了イベントに遷移すると，トップレベルの場合は，ほかに実行中のアクティビティがあっても案件は強制終了します。サブプロセスやイベント・サブプロセス内の場合は，そのサブプロセスやイベント・サブプロセス内の実行中のアクティビティだけを強制終了します。

変換イメージ

強制終了イベントの BPMN ビジネスプロセス定義から CSCIW ビジネスプロセス定義への変換イメージを示します。

強制終了イベントは業務ステップ定義，作業定義，およびシンクノードに変換されます。

図 1-27　強制終了の変換イメージ



処理内容

共通設定ファイルのTerminateCompatMode パラメタの指定によって，強制終了イベントに遷移する時の案件および業務ステップの状態遷移が異なります。パラメタの指定ごとの状態遷移を次の表で示します。

表 1-7　強制終了イベントへ遷移時の案件および業務ステップの状態遷移

TerminateCompatMode パラメタの指定	強制終了イベントを定義するスコープ	遷移後の状態			
		案件	ほかの実行中の業務ステップ		強制終了イベントの業務ステップ
			スコープ内	スコープ外	
true	スコープに関係なく強制終了イベントを定義	強制終了	強制終了		強制終了
false	トップレベル	強制終了	強制終了		強制終了
	サブプロセス	実行中	強制終了	実行中	強制終了
	イベント・サブプロセス	実行中	強制終了	実行中	強制終了

共通設定ファイルのTerminateCompatMode パラメタにtrue を指定している場合（03-10 までと同じ）

強制終了イベントから変換された業務ステップに遷移すると，ほかの「実行中」状態の業務ステップの有無に関係なく案件を強制終了します。

子案件の場合は，呼び元のコールアクティビティから変換された作業および業務ステップを強制終了します。

表 1-8　強制終了イベントへ遷移時の状態遷移

遷移前	遷移後				
ほかの業務ステップ	案件	ほかの業務ステップ		強制終了イベントの業務ステップ	強制終了イベントの業務ステップ内の作業
		未終了	終了		
未終了なし	強制終了	－	変更なし	強制終了	強制終了
未終了あり	強制終了	強制終了	変更なし	強制終了	強制終了

（凡例）

－：該当なし

共通設定ファイルのTerminateCompatMode パラメタにfalse を指定している場合

トップレベルの強制終了イベントから変換された業務ステップに遷移すると，ほかの「実行中」状態の業務ステップの有無に関係なく案件を強制終了します。

サブプロセス，イベント・サブプロセス内の強制終了イベントから変換された業務ステップに遷移すると，サブプロセス，イベント・サブプロセス内の「実行中」状態の業務ステップを強制終了して，強制終了イベントから変換された業務ステップも強制終了します。サブプロセス，イベント・サブプロセスの先には遷移しません。対象となるサブプロセス，イベント・サブプロセス以外に「実行中」状態の業務ステップがないと，案件は「実行中」状態のままですが「実行中」状態の業務ステップがなくなり，強制終了しかできなくなります。

（凡例）

－：該当なし

(21) キャッチ（メッセージ）

機能

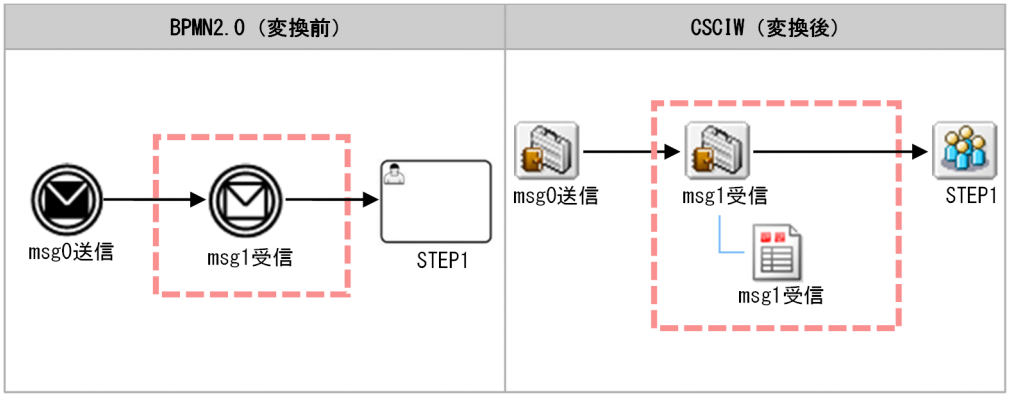
キャッチ（メッセージ）イベントに遷移すると，設定されたメッセージが送信されるまで受信待ち状態になります。メッセージを受信すると，次のアクティビティに遷移します。

変換イメージ

キャッチ（メッセージ）イベントの BPMN ビジネスプロセス定義から CSCIW ビジネスプロセス定義への変換イメージを示します。

キャッチ（メッセージ）イベントは，業務ステップ定義および作業定義に変換されます。

図 1-28 キャッチ（メッセージ）の変換イメージ



処理内容

キャッチ（メッセージ）から変換された業務ステップに遷移すると，設定されたmessageRef の受信待ちになります。

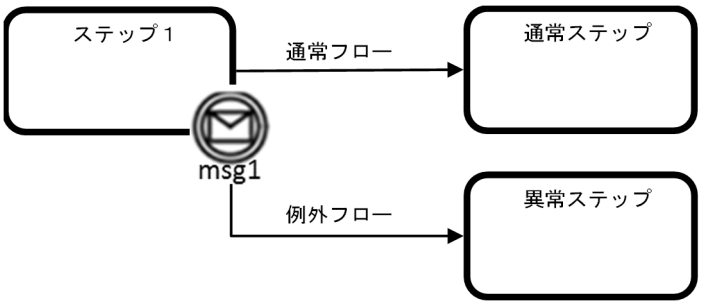
メッセージイベント送信 API によってメッセージを受信すると，業務ステップを完了して，次の業務ステップ / 制御ノードに遷移します。

(22) 境界中断（メッセージ）

機能

アクティビティの境界上にキャッチ（メッセージ）が定義される場合，その境界のアクティビティが実行中の間だけ受信待ち状態になります。アクティビティが完了する前に，メッセージが発生した場合は，アクティビティを停止して例外フローに遷移します。アクティビティが完了した場合，通常フローに遷移します。

図 1-29 境界中断（メッセージ）イベント



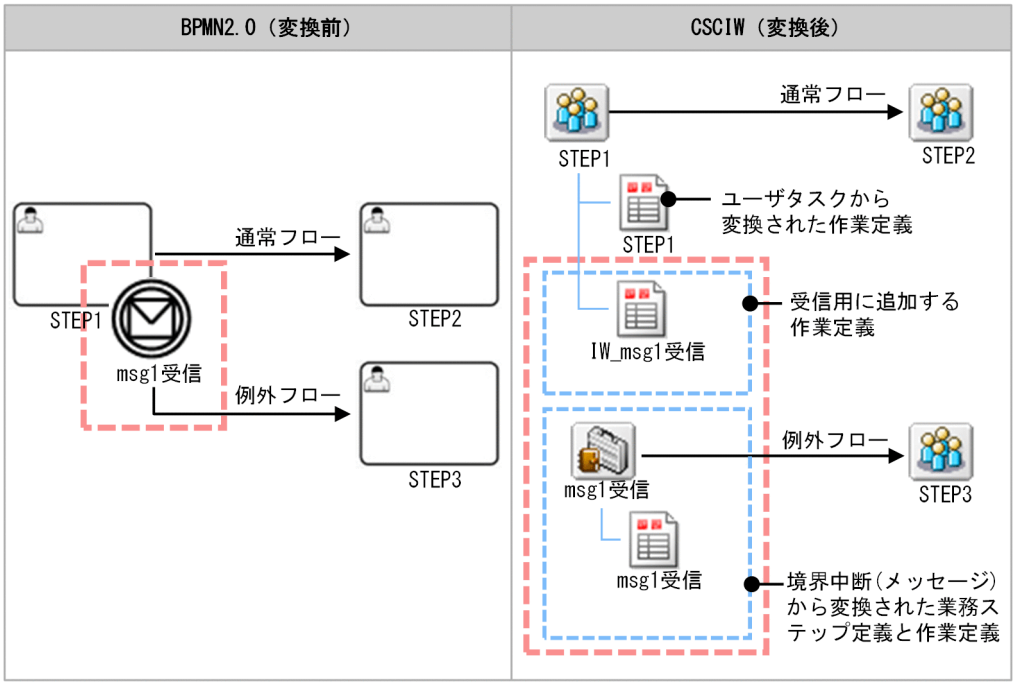
変換イメージ 1（ユーザタスク / サービスタスク / ビジネスルールタスク / コールアクティビティ）

境界中断（メッセージ）イベントがユーザタスク / サービスタスク / ビジネスルールタスク / コールアクティビティに定義された場合の BPMN ビジネスプロセス定義から CSCIW ビジネスプロセス定義への変換イメージを示します。

境界中断（メッセージ）イベント（msg1 受信）は，業務ステップ定義（msg1 受信）および作業定義（msg1 受信）に変換されます。

境界中断（メッセージ）が定義された対象から変換された業務ステップ定義（STEP1）には，境界中断（メッセージ）の受信用の作業定義（IW_msg1 受信）が定義されます。

図 1-30　境界中断（メッセージ）の変換イメージ 1



処理内容 1（ユーザタスク / サービスタスク / ビジネスルールタスク / コールアクティビティ）

境界中断（メッセージ）が定義されたアクティビティから変換された業務ステップに遷移すると，境界中断（メッセージ）に設定されたmessageRef の受信待ちになります。

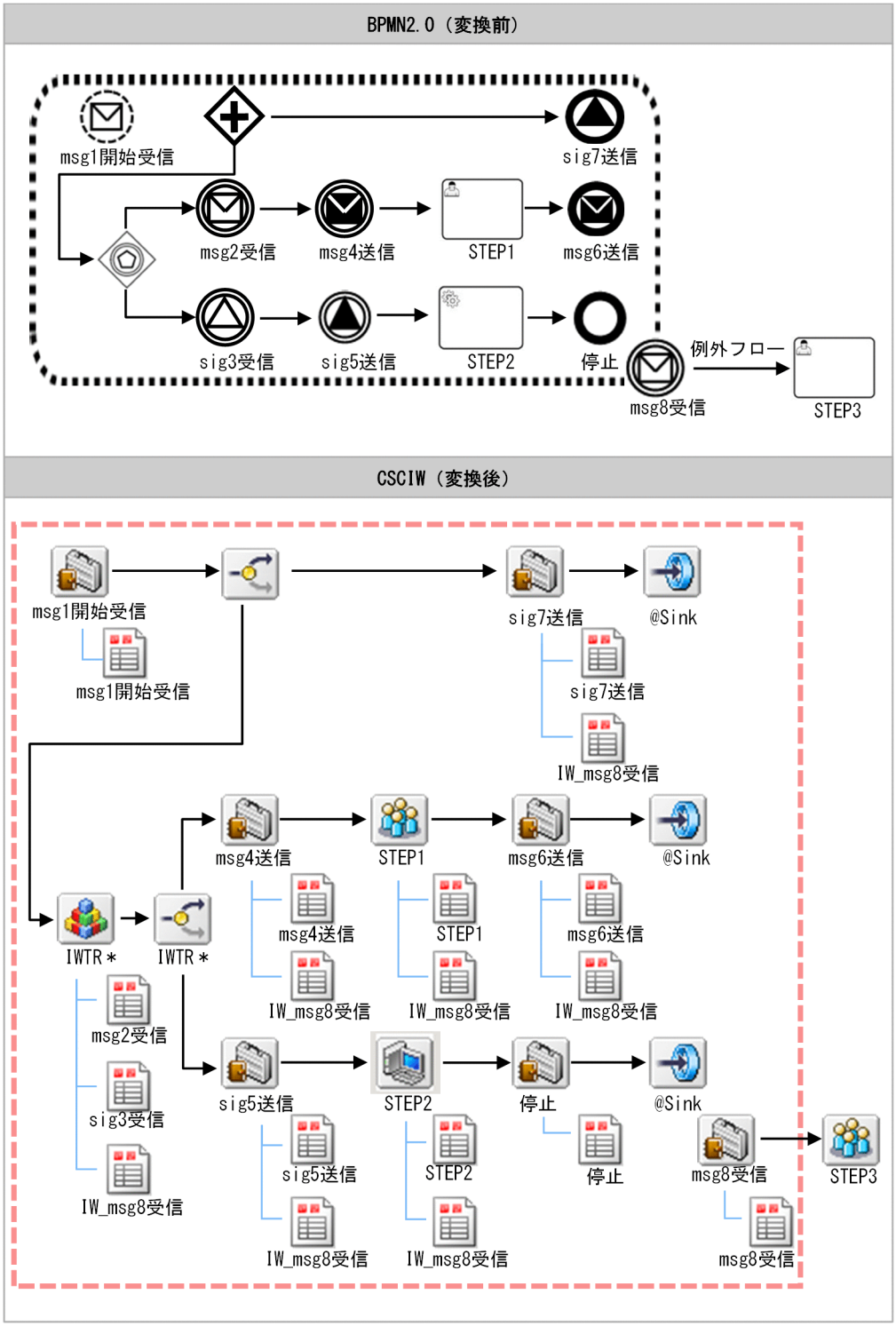
メッセージイベント送信 API によってメッセージを受信すると，業務ステップを強制終了して，境界中断（メッセージ）から変換された業務ステップを生成します。境界中断（メッセージ）から変換された業務ステップは，自動的に完了し，例外フロー遷移先の業務ステップ / 制御ノードに遷移します。

ユーザタスクから変換された作業を完了しても業務ステップ内には受信用の作業が存在しているため，業務ステップは完了しません。作業完了の API の中で業務ステップが完了していなかった場合は，業務ステップを完了します。業務ステップが完了すると通常フローの遷移先の業務ステップ / 制御ノードに遷移します。

変換イメージ 2（サブプロセス / イベント・サブプロセス）

境界中断（メッセージ）がサブプロセス / イベント・サブプロセスに定義された場合の BPMN ビジネスプロセス定義から CSCIW ビジネスプロセス定義への変換イメージを示します。

図 1-31　境界中断（メッセージ）の変換イメージ 2



境界中断（メッセージ）「msg8 受信」が定義されたイベント・サブプロセス内で実行中になる業務ステップ定義内には受信用の作業定義「IW_msg8 受信」が定義されます。受信用の作業定義が定義される対象が複数になる以外は，「変換イメージ 1（ユーザタスク / サービスタスク / ビジネスルールタスク / コールアクティビティ）」と同じになります。

処理内容 2（サブプロセス / イベント・サブプロセス）

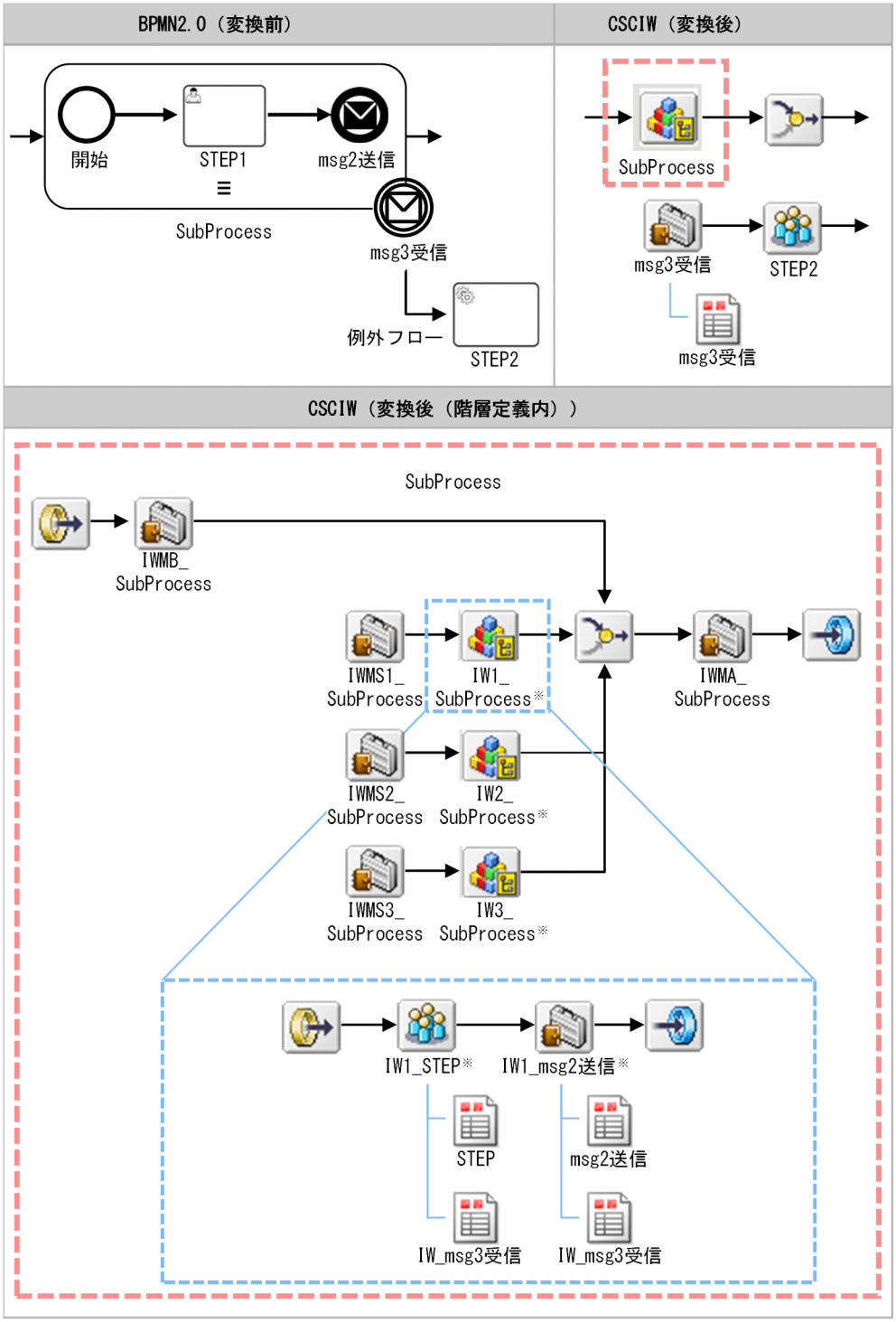
境界中断（メッセージ）が定義されたサブプロセス内ではタスクやイベントから変換された業務ステップに遷移すると，境界中断（メッセージ）に設定された`messageRef` の受信待ちになります。

メッセージイベント送信 API によってメッセージを受信すると，サブプロセス内で受信待ちしていたすべての業務ステップを強制終了して，境界中断（メッセージ）から変換された業務ステップを生成します。境界中断（メッセージ）から変換された業務ステップは自動的に完了し，例外フロー遷移先の業務ステップ / 制御ノードに遷移します。

変換イメージ 3（サブプロセス（マルチインスタンス））

境界中断（メッセージ）がサブプロセス（マルチインスタンス）に定義された場合の BPMN ビジネスプロセス定義から CSCIW ビジネスプロセス定義への変換イメージを示します。

図 1-32　境界中断（メッセージ）の変換イメージ 3



注※

階層定義名，および階層定義内の業務ステップ定義名は，「IW<インデクス>_<BPMN 要素の name 属性>_<BPMN 要素の id 属性>」となります。

境界中断（メッセージ）「msg3 受信」が定義されたサブプロセス内で実行中になる業務ステップ定義内には，受信用の作業定義「IW_msg3 受信」が定義されます。サブプロセス（マルチインスタンス）から変

換されたすべての階層（IW<インデクス>_SubProcess）内の業務ステップ定義内に定義される以外は，「変換イメージ 2（サブプロセス / イベント・サブプロセス)」と同じになります。

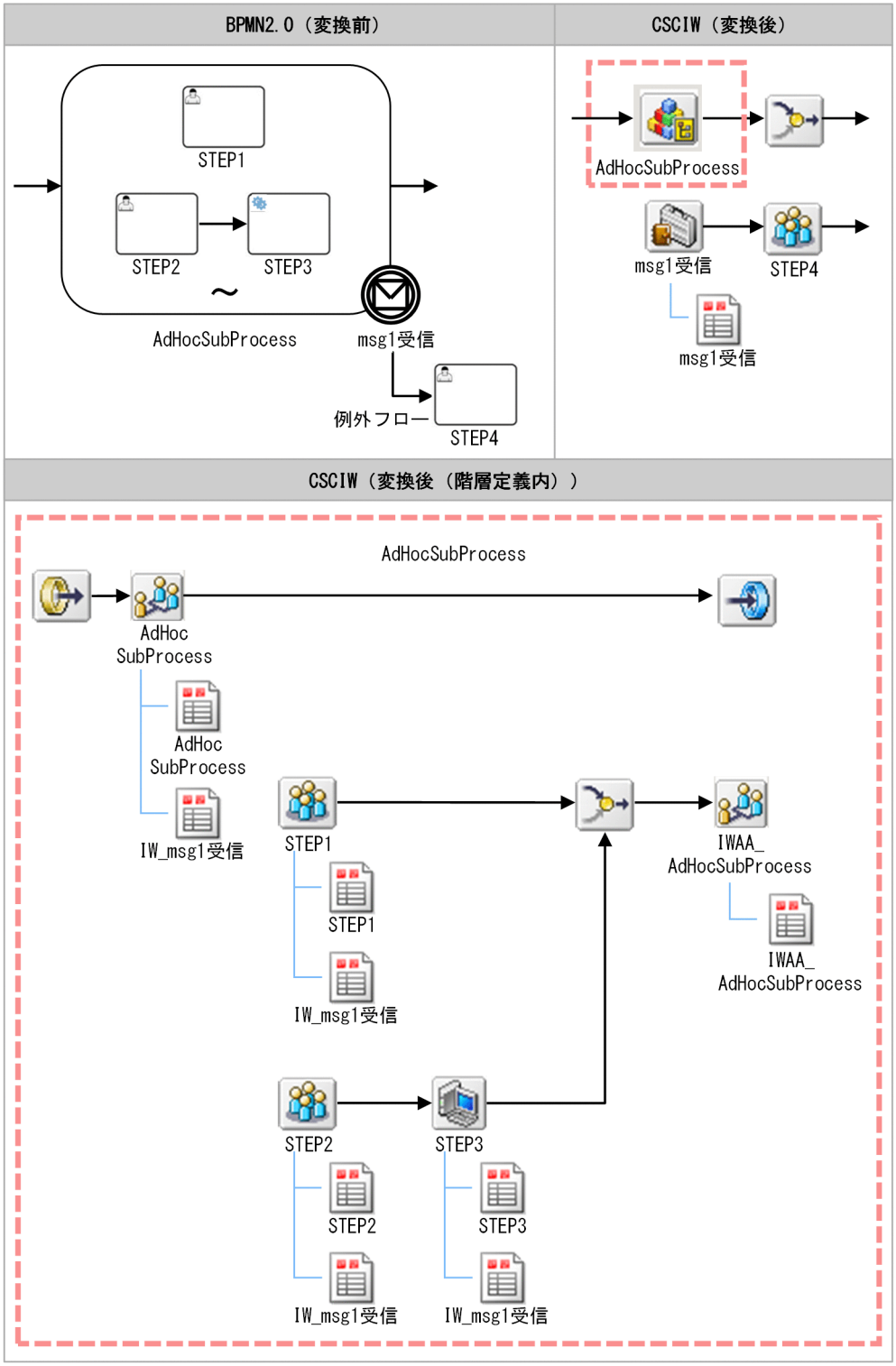
処理内容 3（サブプロセス（マルチインスタンス））

境界中断（メッセージ）が定義されたサブプロセス（マルチインスタンス）の動作は，「処理内容 2（サブプロセス / イベント・サブプロセス)」と同じになります。

変換イメージ 4（アドホック・サブプロセス）

境界中断（メッセージ）がアドホック・サブプロセスに定義された場合の BPMN ビジネスプロセス定義から CSCIW ビジネスプロセス定義への変換イメージを示します。

図 1-33　境界中断（メッセージ）の変換イメージ 4



境界中断（メッセージ）「msg1 受信」が定義されたアドホック・サブプロセス内で実行中になる業務ステップ定義内には，受信用の作業定義「IW_msg1 受信」が定義されます。境界中断（メッセージ）が定義されたアドホック・サブプロセスの変換は，「変換イメージ 2（サブプロセス / イベント・サブプロセス）」と同じになります。

処理内容 4（アドホック・サブプロセス）

境界中断（メッセージ）が定義されたアドホック・サブプロセスの動作は，「処理内容 2（サブプロセス / イベント・サブプロセス）」と同じになります。

(23)　イベント・サブプロセス非中断開始（メッセージ）

機能

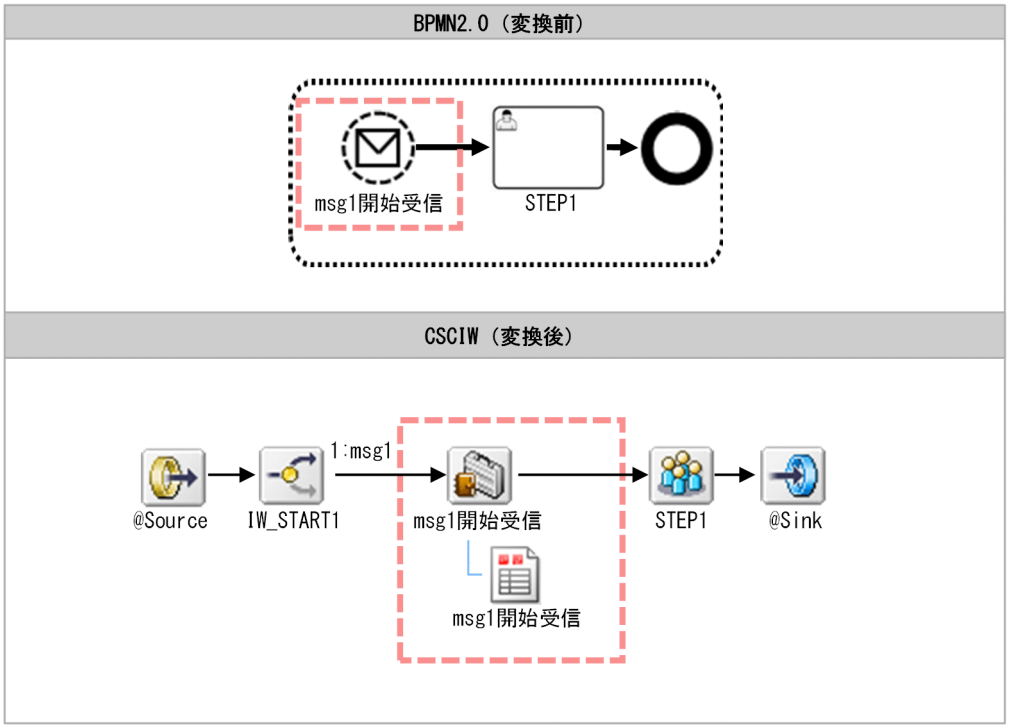
イベント・サブプロセス非中断開始（メッセージ）は，ビジネスプロセスが実行中の間だけイベントの受信待ちになります。サブプロセスに定義されている場合は，そのサブプロセス（下位のサブプロセスも含む）が実行中の間だけイベントの受信待ちになります。メッセージを受信するとイベント・サブプロセスが開始されます。

変換イメージ

イベント・サブプロセス非中断開始（メッセージ）の BPMN ビジネスプロセス定義から CSCIW ビジネスプロセス定義への変換イメージを示します。

イベント・サブプロセス非中断開始（メッセージ）はイベント・サブプロセスから変換された階層定義の中でソースノード，分岐ノード，分岐条件，業務ステップ定義，および作業定義に変換されます。作業定義の説明にはmessageRef から生成された文字列が設定されます。

図 1-34　イベント・サブプロセス非中断開始（メッセージ）の変換イメージ



処理内容

メッセージイベント送信 API によってメッセージを受信すると，指定されたmessageRef のイベント・サブプロセス非中断開始（メッセージ）から変換された業務ステップが生成されます。生成された業務ステップは自動的に完了し，次の業務ステップ / 制御ノードに遷移します。

(24)　スロー（メッセージ）

機能

メッセージをスローします。

スロー（メッセージ）のスロー先を示します。

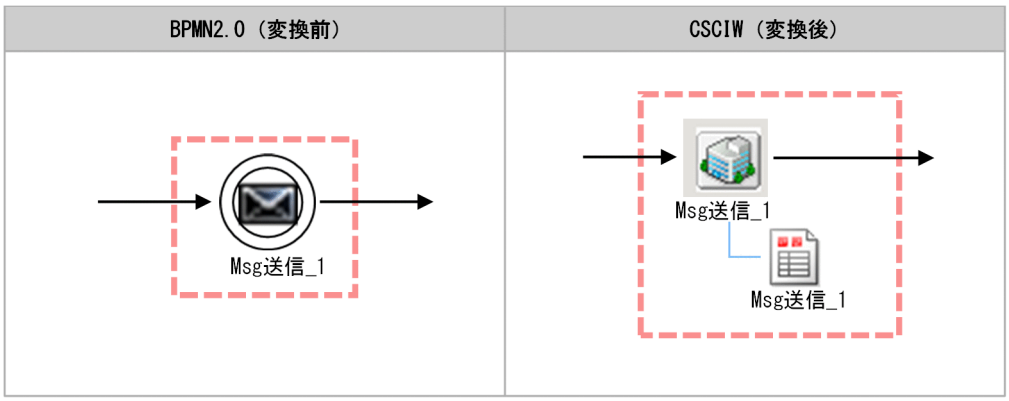
- 開始（メッセージ）
- キャッチ（メッセージ）
- 境界中断（メッセージ）
- イベント・サブプロセス非中断開始（メッセージ）
- REST アプリケーション

変換イメージ

スロー（メッセージ）イベントの BPMN ビジネスプロセス定義から CSCIW ビジネスプロセス定義への変換イメージを示します。

スロー（メッセージ）イベントは，業務ステップ定義および作業定義に変換されます。

図 1-35　スロー（メッセージ）の変換イメージ



処理内容

スロー（メッセージ）イベントの動作については，「1.8.1　呼び出し対象の BPMN 要素」を参照してください。

(25)　スロー（リンク）／キャッチ（リンク）

機能

スロー（リンク）からキャッチ（リンク）に遷移します。

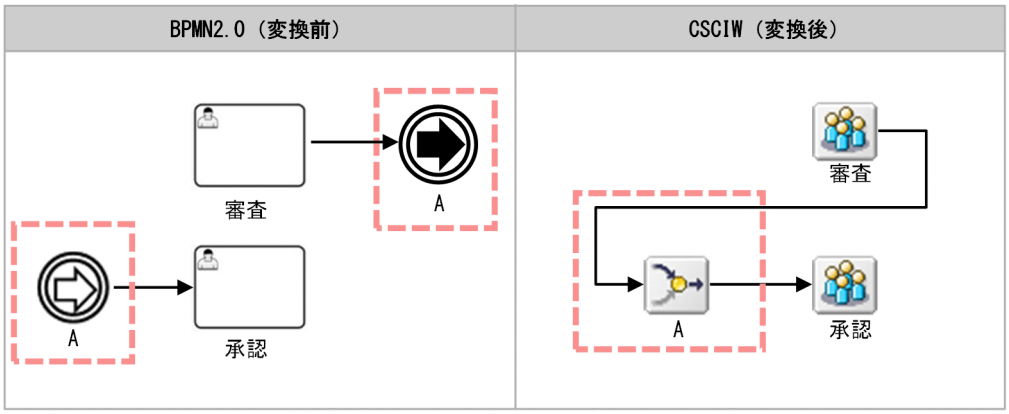
ビジネスプロセス定義が大き過ぎて 1 画面（ページ）に収まらない場合に，1 画面（ページ）の最後にスロー（リンク）を定義し，2 画面（ページ）の先頭に対応するキャッチ（リンク）を定義するときに使用します。

変換イメージ

スロー（リンク）／キャッチ（リンク）の BPMN ビジネスプロセス定義から CSCIW ビジネスプロセス定義への変換イメージを示します。

スロー（リンク）およびキャッチ（リンク）は先着ノード（後続停止なし）とフロー定義に変換されます。

図 1-36　スロー（リンク）／キャッチ（リンク）の変換イメージ



処理内容

フロー定義に従ってスロー（リンク）の遷移元から，キャッチ（リンク）の遷移先に遷移します。

(26)　排他ゲートウェイ

機能

分岐する場合と合流する場合の 2 パターンがあります。

分岐の場合，1 つの入力フローに対し複数の出力フローを定義し，任意の出力フローが 1 つだけ有効になります。デフォルトシーケンスフロー以外の出力フローには分岐条件として XPath 式を設定します。分岐条件の詳細は，「1.6.4　プロセスデータの利用方法」を参照してください。

合流の場合，複数の入力フローに対し 1 つの出力フローを定義し，任意の入力フローが合流したあとに，次のアクティビティに遷移します。

変換イメージ

排他ゲートウェイの BPMN ビジネスプロセス定義から CSCIW ビジネスプロセス定義への変換イメージを示します。

排他ゲートウェイ・分岐は分岐ノードおよび分岐条件に，排他ゲートウェイ・合流は先着ノード（後続停止なし）に変換されます。

図 1-37　排他ゲートウェイ・分岐の変換イメージ

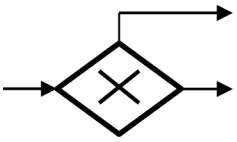
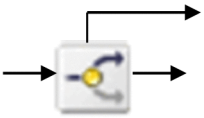
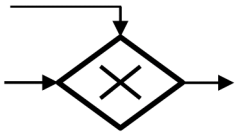
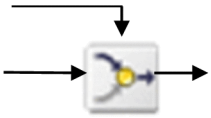
BPMN2.0（変換前）	CSCIW（変換後）
	

図 1-38　排他ゲートウェイ・合流の変換イメージ

BPMN2.0（変換前）	CSCIW（変換後）
	

処理内容

排他ゲートウェイ・分岐

分岐ノードに遷移すると，分岐条件が順番に評価され最初に成立した分岐先に遷移します。

デフォルトシーケンスフローを定義しなかった場合，分岐条件が 1 つも成立しないとエラーになります。

排他ゲートウェイ・合流

先着ノード（後続停止なし）に遷移すると，次の業務ステップ / 制御ノードに遷移します。

(27)　並列ゲートウェイ

機能

分岐する場合と合流する場合の 2 パターンがあります。

分岐の場合，1 つの入力フローに対し複数の出力フローを定義し，すべての出力フローへ並行に遷移します。

合流の場合，複数の入力フローに対し 1 つの出力フローを定義し，すべての入力フローが合流したあとに，次のアクティビティへ遷移します。

変換イメージ

並列ゲートウェイの BPMN ビジネスプロセス定義から CSCIW ビジネスプロセス定義への変換イメージを示します。

並列ゲートウェイ・分岐は分業ノードに，並列ゲートウェイ・合流は待合ノードに変換されます。

図 1-39　並列ゲートウェイ・分岐の変換イメージ

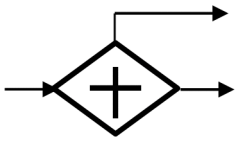
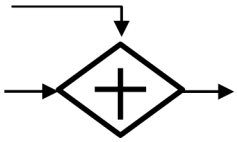
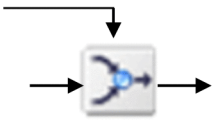
BPMN2.0（変換前）	CSCIW（変換後）
	

図 1-40　並列ゲートウェイ・合流の変換イメージ

BPMN2.0（変換前）	CSCIW（変換後）
	

処理内容

分業ノードに遷移した場合，遷移先として定義されたすべての業務ステップ / 制御ノードに遷移します。

待合ノードはすべての遷移元から遷移した場合に，次の業務ステップ / 制御ノードに遷移します。

注意事項

並列ゲートウェイを使用したビジネスプロセス定義で差し戻し／引き戻しをする場合，注意事項があります。差し戻し／引き戻しの注意事項は，マニュアル『uCosminexus Service Coordinator Interactive Workflow システム構築・運用ガイド』の「2.5.2　案件の動的な制御」の「(1) 引き戻しと差し戻し」を参照してください。

(28) 排他イベントゲートウェイ

機能

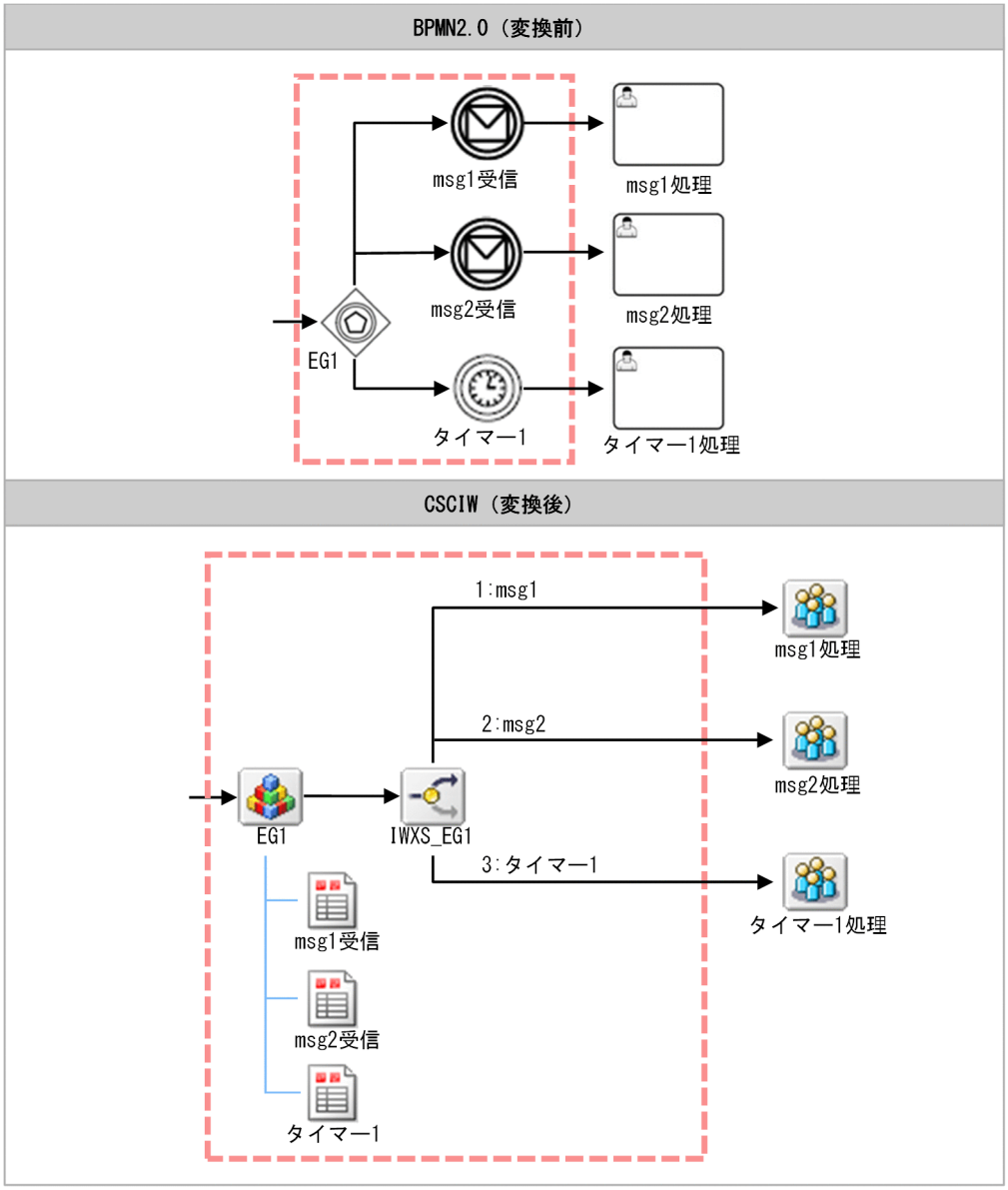
排他イベントゲートウェイに遷移すると，続く複数のイベントが受信待ち状態になります。受信待ちの複数のイベントの中から最初に受信したイベントの遷移先に遷移します。

変換イメージ

排他イベントゲートウェイの BPMN ビジネスプロセス定義から CSCIW ビジネスプロセス定義への変換イメージを示します。

排他イベントゲートウェイは，業務ステップ，作業，分岐ノード，および分岐条件に変換されます。業務ステップには受信用のイベントから変換された作業が定義されます。分岐ノードからは各受信用のイベントの遷移先にフロー定義が定義され，それぞれに分岐条件が設定されます。

図 1-41　排他イベントゲートウェイの変換イメージ



処理内容

排他イベントゲートウェイから変換された業務ステップに遷移すると，各メッセージイベントのmessageRefの受信待ち，およびタイマーイベントの発火時刻の待ち状態になります。

メッセージイベント送信 API によってメッセージを受信すると，指定されたmessageRef に従って分岐し，次の業務ステップ / 制御ノードに遷移します。タイマーの発火時刻を過ぎると，アプリケーション呼び出しサービスによって，発火時刻を過ぎたタイマーの遷移先の業務ステップ / 制御ノードに遷移します。

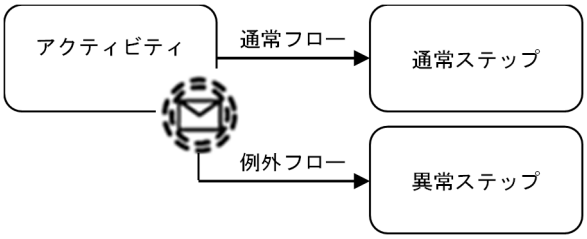
(29)　境界非中断（メッセージ）

機能

アクティビティの境界上にキャッチ（メッセージ）が定義される場合，その境界のアクティビティが実行中の間だけ受信待ち状態になります。アクティビティが完了する前にメッセージが発生した場合は，例外フローに遷移します。アクティビティが完了すると通常フローに遷移します。

境界中断（メッセージ）とは，受信した際にアクティビティを停止するかどうかが異なります。

図 1-42　境界非中断（メッセージ）イベント



変換イメージ

境界中断（メッセージ）と同じです。「1.3.4　BPMN 要素の CSCIW ビジネスプロセス定義への変換の詳細」の「(22)　境界中断（メッセージ）」を参照してください。

処理内容

境界非中断（メッセージ）が定義されたユーザタスク，サービスタスク，ビジネスルールタスク，コールアクティビティや，境界非中断（メッセージ）が定義されたサブプロセス内のタスクやイベントから変換された業務ステップに遷移すると，境界非中断（メッセージ）に設定されたmessageRef の受信待ちになります。

メッセージイベント送信 API によってメッセージを受信すると，境界非中断（メッセージ）から変換された業務ステップを生成します。境界非中断（メッセージ）から変換された業務ステップは自動的に完了し，例外フロー遷移先の業務ステップ / 制御ノードに遷移します。

(30)　イベント・サブプロセス中断開始（メッセージ）

機能

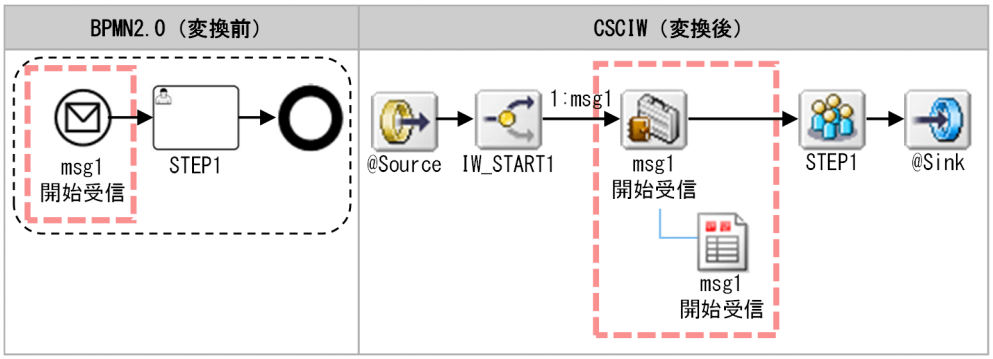
イベント・サブプロセス中断開始（メッセージ）は，ビジネスプロセスが実行中の間だけイベントの受信待ちになります。サブプロセスに定義されている場合は，そのサブプロセス（下位のサブプロセスも含む）が実行中の間だけイベントの受信待ちになります。トップレベルに定義されている場合は，メッセージを受信すると同一案件内で実行中のタスク / イベントを強制終了してから，イベント・サブプロセスが開始されます。サブプロセスに定義されている場合は，サブプロセス内の実行中のタスク / イベントを強制終了してから，イベント・サブプロセスが開始されます。

変換イメージ

イベント・サブプロセス中断開始（メッセージ）の BPMN ビジネスプロセス定義から CSCIW ビジネスプロセス定義への変換イメージを示します。

イベント・サブプロセス中断開始（メッセージ）はイベント・サブプロセスから変換された階層定義の中でソースノード，分岐ノード，分岐条件，業務ステップ定義，および作業定義に変換されます。作業定義の説明にはmessageRef から生成された文字列が設定されます。

図 1-43　イベント・サブプロセス中断開始（メッセージ）の変換イメージ



処理内容

メッセージイベント送信 API によってメッセージを受信すると，指定されたmessageRef のイベント・サブプロセス中断開始（メッセージ）から変換された業務ステップが生成されます。生成された業務ステップは自動的に完了し，次の業務ステップ / 制御ノードに遷移します。

トップレベルに定義されている場合は，同一案件内の実行中の業務ステップを強制終了します。サブプロセスに定義されている場合は，サブプロセス内の実行中の業務ステップを強制終了します。

(31)　終了（エラー）

機能

終了（エラー）イベントに遷移すると，エラーをスローします。ほかに実行中のアクティビティがない場合，案件が完了します。ほかに実行中のアクティビティがある場合，案件は完了しません。

終了（エラー）イベントのエラースロー先を次に示します。

- 境界中断（エラー）
- イベント・サブプロセス中断開始（エラー）

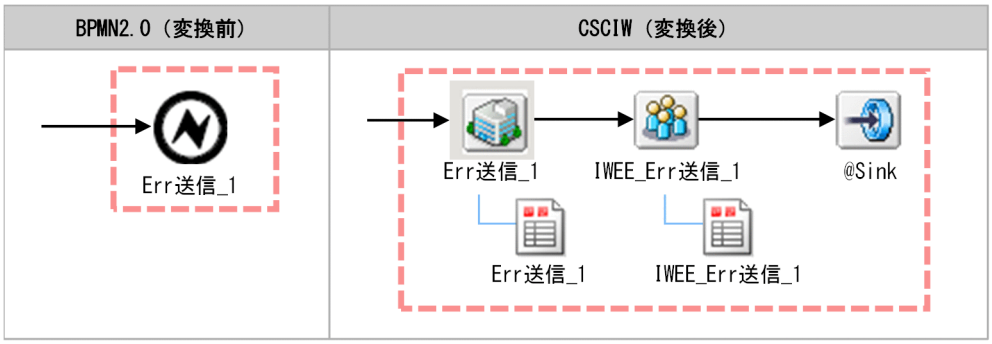
子案件の場合，キャッチできる境界中断（エラー）やイベント・サブプロセス中断開始（エラー）が対象案件に存在しないときは，親案件に対してエラーをスローします。親案件にも存在しないときは，親案件の親案件に対してエラーをスローします。これをルート案件まで繰り返します。

変換イメージ

終了（エラー）イベントの BPMN ビジネスプロセス定義から CSCIW ビジネスプロセス定義への変換イメージを示します。

終了（エラー）イベントは，業務ステップ定義，作業定義，およびシンクノードに変換されます。

図 1-44　終了（エラー）の変換イメージ



処理内容

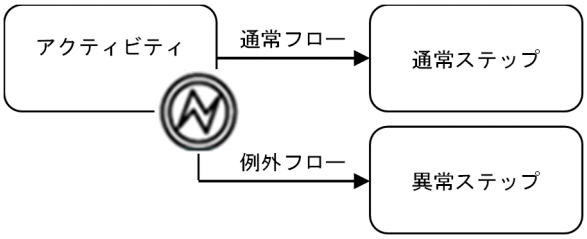
終了（エラー）の動作については，「1.8.1　呼び出し対象の BPMN 要素」を参照してください。

(32)　境界中断（エラー）

機能

アクティビティの境界上に境界中断（エラー）が定義される場合，その境界のアクティビティが実行中の間だけ受信待ち状態になります。アクティビティが完了する前に，エラーイベントが発生した場合は，アクティビティを停止して例外フローに遷移します。アクティビティが完了した場合，通常フローに遷移します。

図 1-45　境界中断（エラー）イベント



変換イメージ

境界中断（メッセージ）と同じです。「1.3.4　BPMN 要素の CSCIW ビジネスプロセス定義への変換の詳細」の「(22)　境界中断（メッセージ）」を参照してください。

処理内容

境界中断（エラー）が定義されたユーザタスク，サービスタスク，ビジネスルールタスク，コールアクティビティや，境界中断（エラー）が定義されたサブプロセス内のタスクやイベントから変換された業務ステップに遷移すると，境界中断（エラー）に設定されたerrorRef の受信待ちになります。

エラーを受信すると，受信待ちをしていたすべての業務ステップを強制終了し，境界中断（エラー）から変換された業務ステップを生成します。境界中断（エラー）から変換された業務ステップは，自動的に完了し，例外フロー遷移先の業務ステップ / 制御ノードに遷移します。

(33)　イベント・サブプロセス中断開始（エラー）

機能

イベント・サブプロセス中断開始（エラー）は，ビジネスプロセスが実行中の間だけイベントの受信待ちになります。サブプロセスに定義されている場合は，そのサブプロセス（下位のサブプロセスも含む）が実行中の間だけイベントの受信待ちになります。

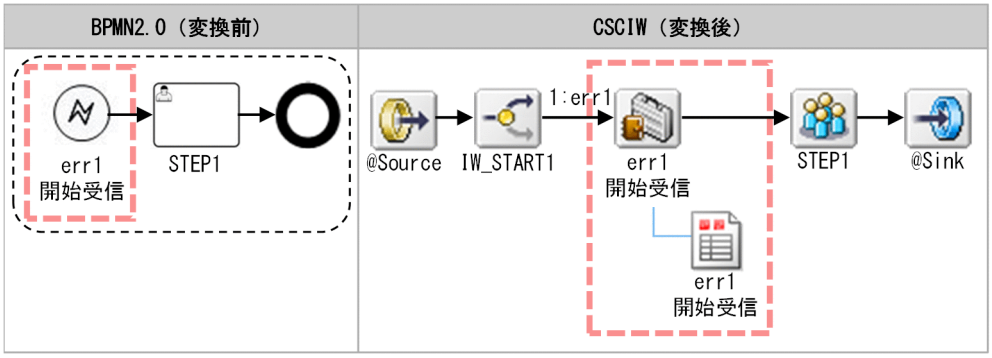
トップレベルに定義されている場合は，エラーを受信すると同一案件内で実行中のタスク / イベントを強制終了してから，イベント・サブプロセスが開始されます。サブプロセスに定義されている場合は，サブプロセス内の実行中のタスク / イベントを強制終了してから，イベント・サブプロセスが開始されます。

変換イメージ

イベント・サブプロセス中断開始（エラー）の BPMN ビジネスプロセス定義から CSCIW ビジネスプロセス定義への変換イメージを示します。

イベント・サブプロセス中断開始（エラー）はイベント・サブプロセスから変換された階層定義の中でソースノード，分岐ノード，分岐条件，業務ステップ定義，および作業定義に変換されます。作業定義の説明にはerrorRef から生成された文字列が設定されます。

図 1-46　イベント・サブプロセス中断開始（エラー）の変換イメージ



処理内容

エラーを受信すると，指定されたerrorRef のイベント・サブプロセス中断開始（エラー）から変換された業務ステップが生成されます。生成された業務ステップは自動的に完了し，次の業務ステップ / 制御ノードに遷移します。

トップレベルに定義されている場合は，同一案件内の実行中の業務ステップを強制終了します。サブプロセスに定義されている場合は，サブプロセス内の実行中の業務ステップを強制終了します。

(34)　開始（タイマー）

機能

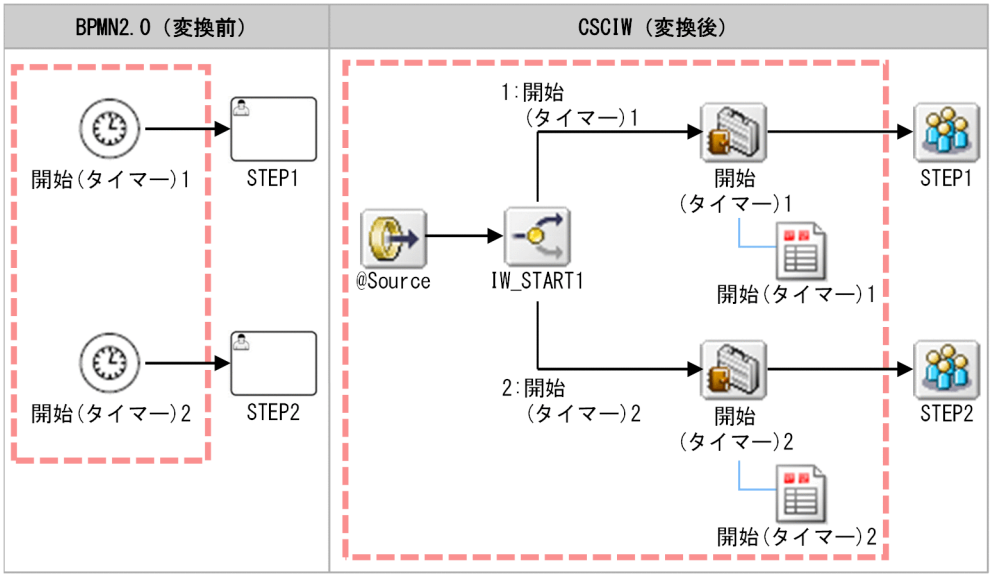
設定されたタイマールールを評価して求めた発火時刻を過ぎたときに，案件を開始します。

変換イメージ

開始（タイマー）イベントの BPMN ビジネスプロセス定義から CSCIW ビジネスプロセス定義への変換イメージを示します。

開始（タイマー）イベントは，ソースノード，分岐ノード，分岐条件，業務ステップ定義，および作業定義に変換されます。

図 1-47　開始（タイマー）の変換イメージ



処理内容

開始（タイマー）の処理内容については，「1.8.4　開始（タイマー）の場合の処理の流れ」を参照してください。

(35)　キャッチ（タイマー）

機能

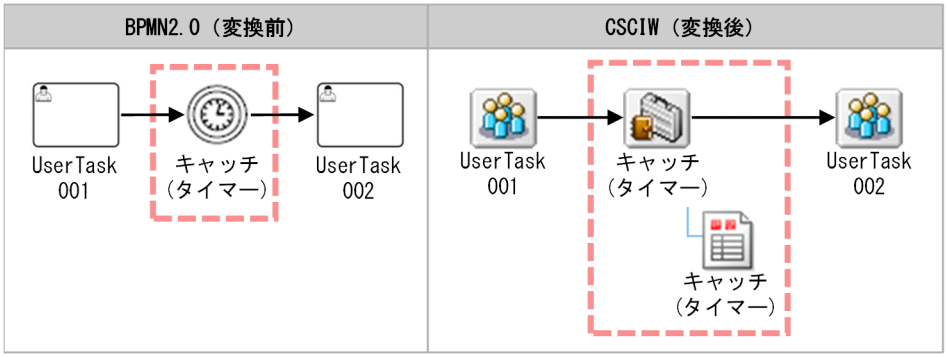
キャッチ（タイマー）イベントに遷移すると，設定されたタイマールールを評価して求めた発火時刻になるまで待ち状態になります。発火時刻を過ぎると，次のアクティビティに遷移します。

変換イメージ

キャッチ（タイマー）イベントの BPMN ビジネスプロセス定義から CSCIW ビジネスプロセス定義への変換イメージを示します。

キャッチ（メッセージ）イベントは，業務ステップ定義および作業定義に変換されます。

図 1-48　キャッチ（タイマー）の変換イメージ



処理内容

キャッチ（タイマー）から変換された業務ステップに遷移すると，設定されたタイマールールを評価して求めた発火時刻になるまで待ち状態になります。

作業は，次のとおり設定されます。

- 処理期限：設定されたタイマールールを評価して求めた発火時刻
- 優先度（実行回数として使用）：「0」
- 作業者：「IWTTIM_IWTransit」

発火時刻を過ぎると，アプリケーション呼び出しサービスが業務ステップを完了して，次の業務ステップ / 制御ノードに遷移します。

タイマーイベントによる案件の遷移については，「1.8.5　開始（タイマー）以外のタイマーイベントの場合の処理の流れ」を参照してください。

(36) 境界中断（タイマー）

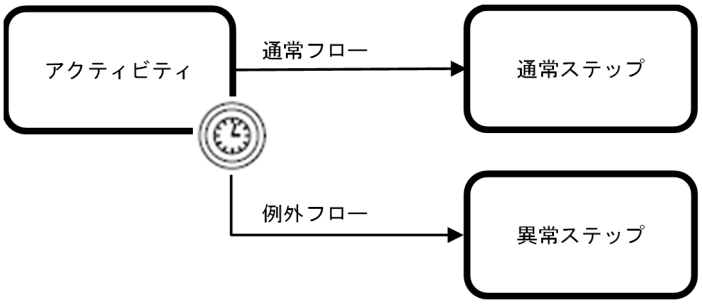
機能

境界中断（タイマー）イベントが定義されたアクティビティに遷移すると，設定されたタイマールールを評価して求めた発火時刻になるまで待ち状態になります。

アクティビティが完了する前に発火時刻を過ぎると，アクティビティを強制終了して例外フローに遷移します。

発火時刻を過ぎる前にアクティビティが完了すると，通常フローに遷移します。

図 1-49　境界中断（タイマー）イベント



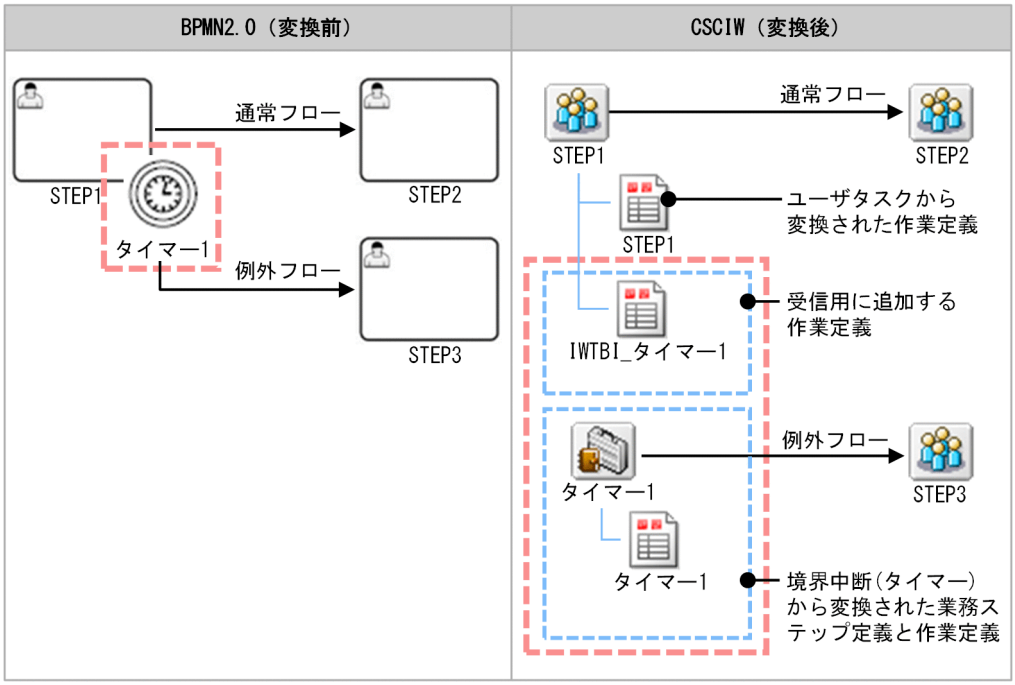
変換イメージ 1（ユーザタスク / サービスタスク / ビジネスルールタスク / コールアクティビティ）

境界中断（タイマー）イベントがユーザタスク / サービスタスク / ビジネスルールタスク / コールアクティビティに定義された場合の，BPMN ビジネスプロセス定義から CSCIW ビジネスプロセス定義への変換イメージを示します。

境界中断（タイマー）イベントは，業務ステップ定義（タイマー 1）と作業定義（タイマー 1）に変換されます。

境界中断（タイマー）が定義された対象から変換された業務ステップ定義（STEP1）には，境界中断（タイマー）の受信用の作業定義（IWTBI_タイマー 1）が定義されます。

図 1-50　境界中断（タイマー）の変換イメージ 1



処理内容 1（ユーザタスク / サービスタスク / ビジネスルールタスク / コールアクティビティ）

境界中断（タイマー）が定義されたタスクから変換された業務ステップに遷移すると，受信用の作業が生成されて，「実行開始可能」状態に遷移します。

受信用の作業は，次のとおり設定されます。

- 処理期限：設定されたタイマールールを評価して求めた発火時刻
- 優先度（実行回数として使用）：「0」
- 作業者：「IWTTIM_IWTransit」

発火時刻を過ぎると，アプリケーション呼び出しサービスから，境界中断（タイマー）の受信用の作業に対して遷移要求が投げられます。遷移要求を受けると，タスクから変換された業務ステップを強制終了し，境界中断（タイマー）から変換された業務ステップを生成して開始します。

境界中断（タイマー）から変換された業務ステップは自動的に完了し，例外フロー遷移先の業務ステップ / 制御ノードに遷移します。

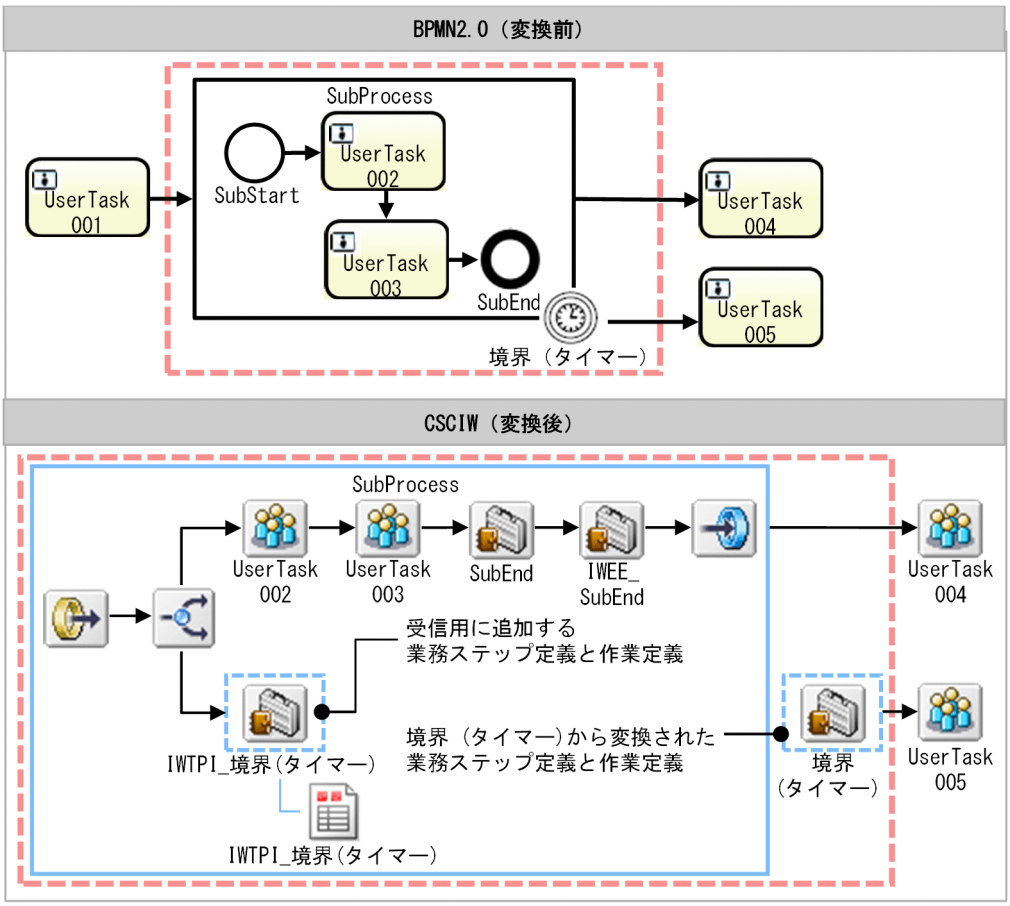
変換イメージ 2（サブプロセス）

境界中断（タイマー）イベントがサブプロセスに定義された場合の，BPMN ビジネスプロセス定義から CSCIW ビジネスプロセス定義への変換イメージを示します。

境界中断（タイマー）イベントは，業務ステップ定義（境界（タイマー））および作業定義（境界（タイマー））に変換されます。

境界中断（タイマー）が定義された対象から変換された階層定義（SubProcess）の階層定義内には，分業ノードと境界中断（タイマー）の受信用の業務ステップおよび作業定義（IWTPI_境界（タイマー））が定義されます。

図 1-51　境界中断（タイマー）の変換イメージ 2



処理内容 2（サブプロセス）

境界中断（タイマー）が定義されたサブプロセスから変換された階層に遷移すると，分業して境界中断（タイマー）の受信用の業務ステップが実行中になります。そして，受信用の作業が生成されて，「実行開始可能」状態に遷移します。

受信用の作業は，次のように設定されます。

- 処理期限：設定されたタイマールールを評価して求めた発火時刻
- 優先度（実行回数として使用）：「0」
- 作業者：「IWTTIM_IWTransit」

発火時刻を過ぎると，アプリケーション呼び出しサービスから，境界中断（タイマー）の受信用の作業に対して遷移要求が投げられます。遷移要求を受けると，境界中断（タイマー）の場合は，同一階層内で実行中の業務ステップおよび受信用の業務ステップを強制終了します。

境界中断（タイマー）から変換された業務ステップを生成して開始します。境界中断（タイマー）から変換された業務ステップは自動的に完了し，例外フロー遷移先の業務ステップ / 制御ノードに遷移します。

正常フローに遷移するケース

対象のサブプロセスのシンクノードに遷移すると，同一階層内で実行中の業務ステップが次に示すものだけの場合は，タイマーの業務ステップを強制終了します。また，終了判定用の業務ステップを完了し，階層の遷移先に遷移します。

- 境界中断（タイマー）
- 境界非中断（タイマー）
- イベント・サブプロセス中断開始（タイマー）
- イベント・サブプロセス非中断開始（タイマー）

変換イメージ３（サブプロセス（マルチインスタンス））

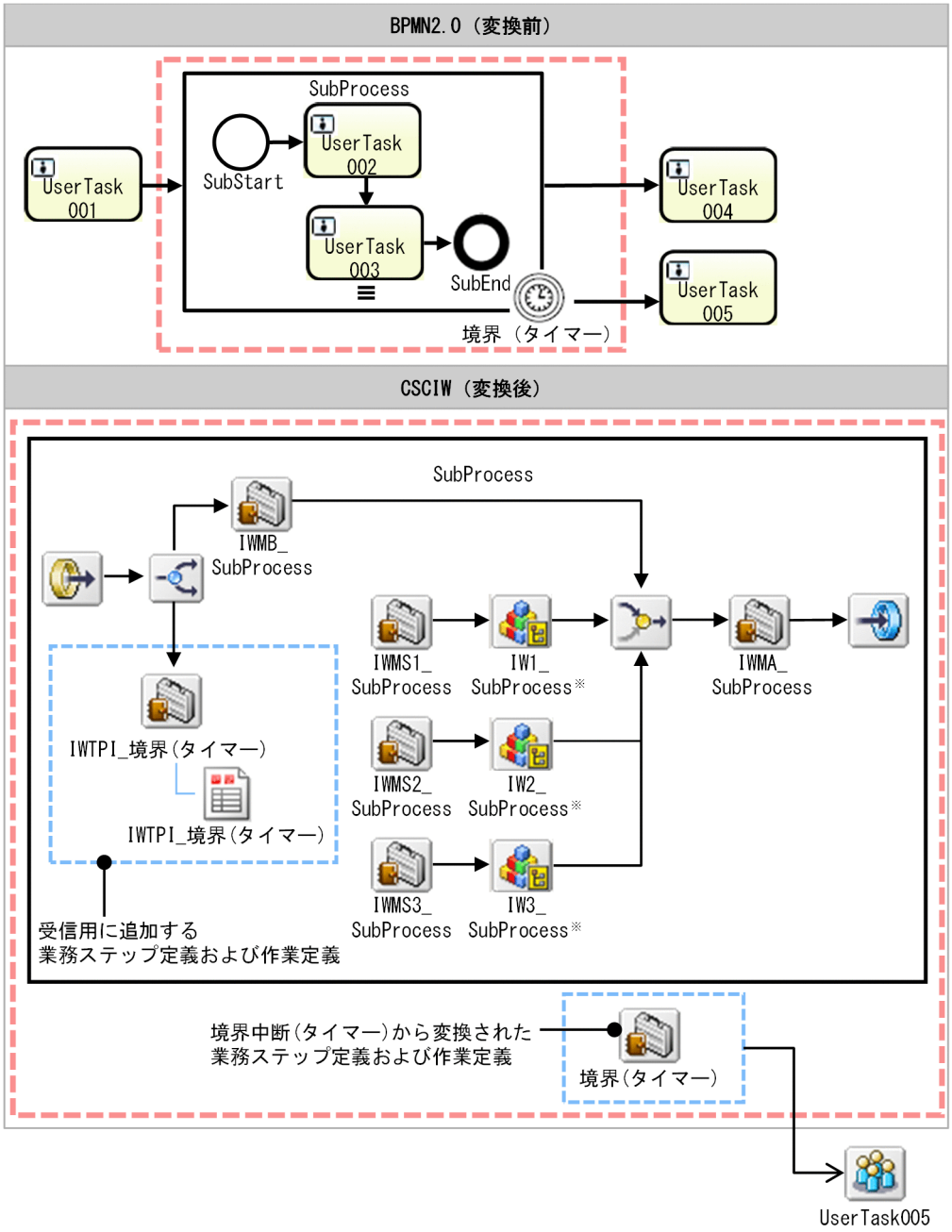
境界中断（タイマー）イベントがサブプロセス（マルチインスタンス）に定義された場合の，BPMN ビジネスプロセス定義から CSCIW ビジネスプロセス定義への変換イメージを示します。

境界中断（タイマー）イベントは，業務ステップ定義（境界（タイマー））および作業定義（境界（タイマー））に変換されます。

境界中断（タイマー）が定義された対象から変換された階層定義（SubProcess）内には，分業ノード，境界中断（タイマー）の受信用の業務ステップ，および作業定義（IWTPI_境界（タイマー））が定義されます。

境界中断（タイマー）イベント以外の変換については，「折りたたまれたサブプロセス（シーケンシャルマルチインスタンス / パラレルマルチインスタンス)」，または「展開されたサブプロセス（シーケンシャルマルチインスタンス / パラレルマルチインスタンス)」と同じになります。「1.3.4　BPMN 要素の CSCIW ビジネスプロセス定義への変換の詳細」の「(40)　折りたたまれたサブプロセス（シーケンシャルマルチインスタンス)」，または「(42)　展開されたサブプロセス（シーケンシャルマルチインスタンス)」を参照してください。

図 1-52　境界中断（タイマー）の変換イメージ 3



注※

階層定義名，および階層定義内の業務ステップ定義名は，「IW<インデクス>_<BPMN 要素の name 属性>_<BPMN 要素の id 属性>」となります。

処理内容 3（サブプロセス（マルチインスタンス））

境界中断（タイマー）が定義されたサブプロセス（マルチインスタンス）の動作については，「処理内容 2（サブプロセス）」と同じになります。

変換イメージ４（アドホック・サブプロセス）

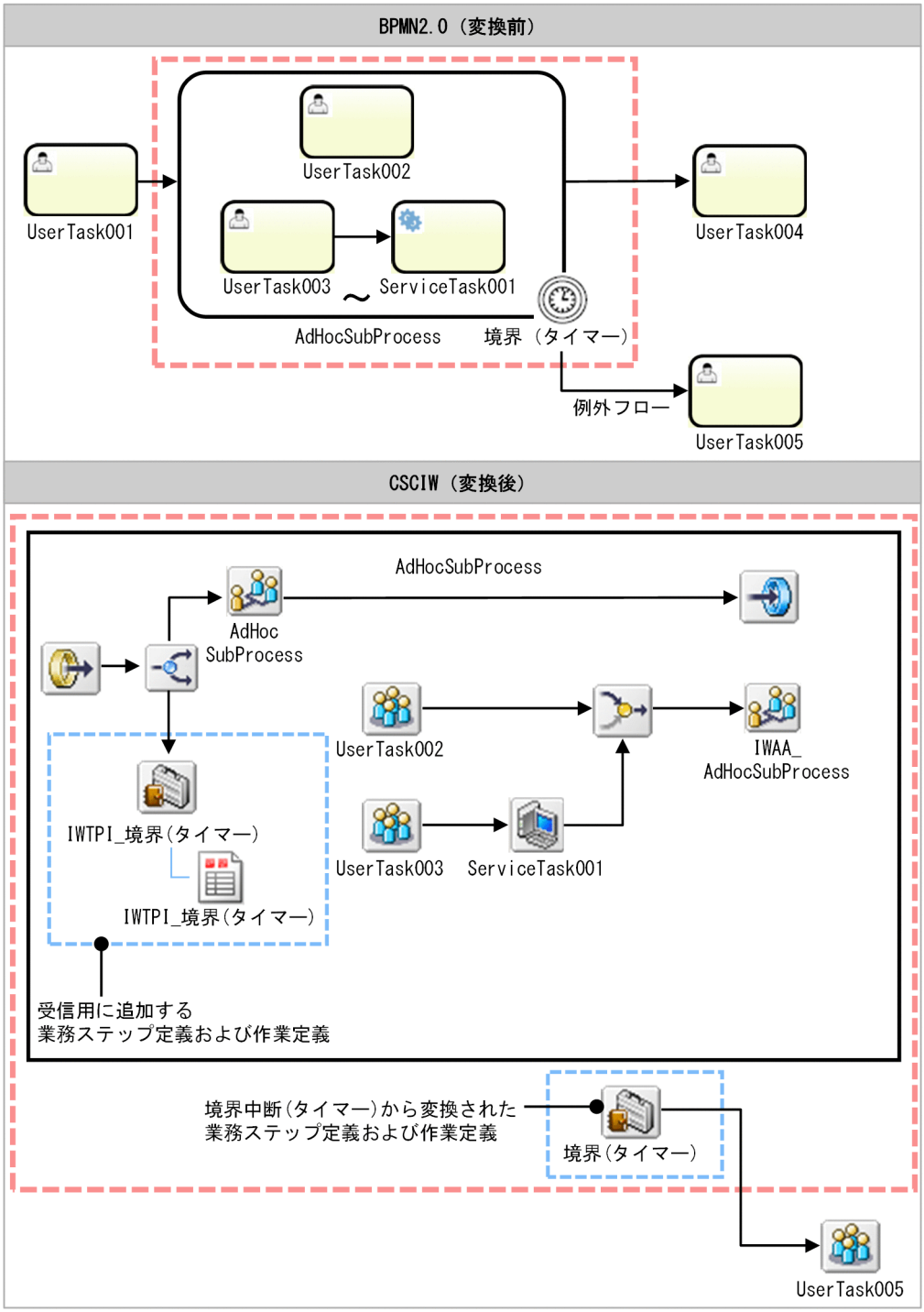
境界中断（タイマー）イベントがアドホック・サブプロセスに定義された場合の，BPMN ビジネスプロセス定義から CSCIW ビジネスプロセス定義への変換イメージを示します。

境界中断（タイマー）イベントは，業務ステップ定義（境界（タイマー））および作業定義（境界（タイマー））に変換されます。

境界中断（タイマー）が定義された対象から変換された階層定義（AdHocSubProcess）内には，分業ノード，境界中断（タイマー）の受信用の業務ステップ，および作業定義（IWTPI_境界（タイマー））が定義されます。

境界中断（タイマー）イベント以外の変換については，「展開されたアドホック・サブプロセス」，または「折りたたまれたアドホック・サブプロセス」と同じになります。「1.3.4　BPMN 要素の CSCIW ビジネスプロセス定義への変換の詳細」の「(44)　展開されたアドホック・サブプロセス」，または「(45)　折りたたまれたアドホック・サブプロセス」を参照してください。

図 1-53　境界中断（タイマー）の変換イメージ 4



処理内容 4（アドホック・サブプロセス）

境界中断（タイマー）が定義されたアドホック・サブプロセスの動作については，「処理内容 2（サブプロセス）」と同じになります。

(37) 境界非中断（タイマー）

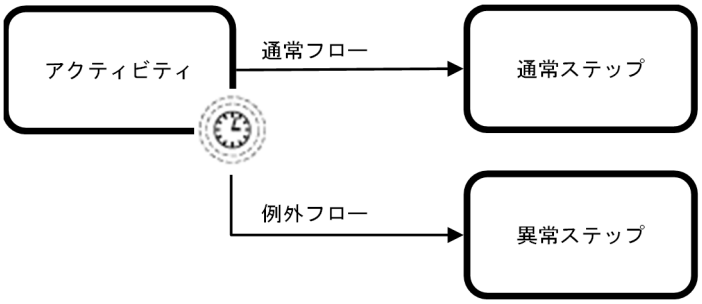
機能

境界非中断（タイマー）イベントが定義されたアクティビティに遷移すると，設定されたタイマールールを評価して求めた発火時刻になるまで待ち状態になります。

アクティビティが完了する前に発火時刻を過ぎると，アクティビティは実行中のまま例外フローに遷移します。繰り返しが設定されている場合は，次の発火時刻まで待ち状態になります。

アクティビティが完了すると，通常フローに遷移します。

図 1-54　境界非中断（タイマー）イベント



変換イメージ

変換については，業務ステップ / 作業定義名以外は「境界中断（タイマー)」と同じになります。境界非中断（タイマー）の業務ステップ / 作業定義名は次に示すとおりです。

- 対象がタスクの場合：
  IWTBN_境界非中断（タイマー）のラベル名
- 対象がサブプロセスの場合：
  IWTPN_境界非中断（タイマー）のラベル名

処理内容

受信待ち状態になるところまでは，「境界中断（タイマー)」と同じになります。

発火時刻を過ぎると，アプリケーション呼び出しサービスから境界非中断（タイマー）の受信用の作業に対して遷移要求が投げられます。遷移要求を受けると，境界非中断（タイマー）から変換された業務ステップを生成して開始します。

境界非中断（タイマー）から変換された業務ステップは，自動的に完了し，例外フロー遷移先の業務ステップ / 制御ノードに遷移します。

処理の実行回数が上限まで達していない場合は，タイマールールを再評価して，次の発火時刻を求めて処理期限に設定します。実行回数が上限に達した場合は，受信用の作業 / 業務ステップを強制終了します。

(38) イベント・サブプロセス中断開始（タイマー）

機能

イベント・サブプロセス中断開始（タイマー）は，イベント・サブプロセス中断開始（タイマー）が定義されたビジネスプロセスまたはサブプロセスが実行中になった時点から，設定されたタイマールールを評価して求めた発火時刻になるまで待ち状態になります。

ビジネスプロセスまたはサブプロセスが完了する前に発火時刻を過ぎると，そのプロセス内で実行中のタスクやイベントを強制終了してから，イベント・サブプロセスが開始されます。

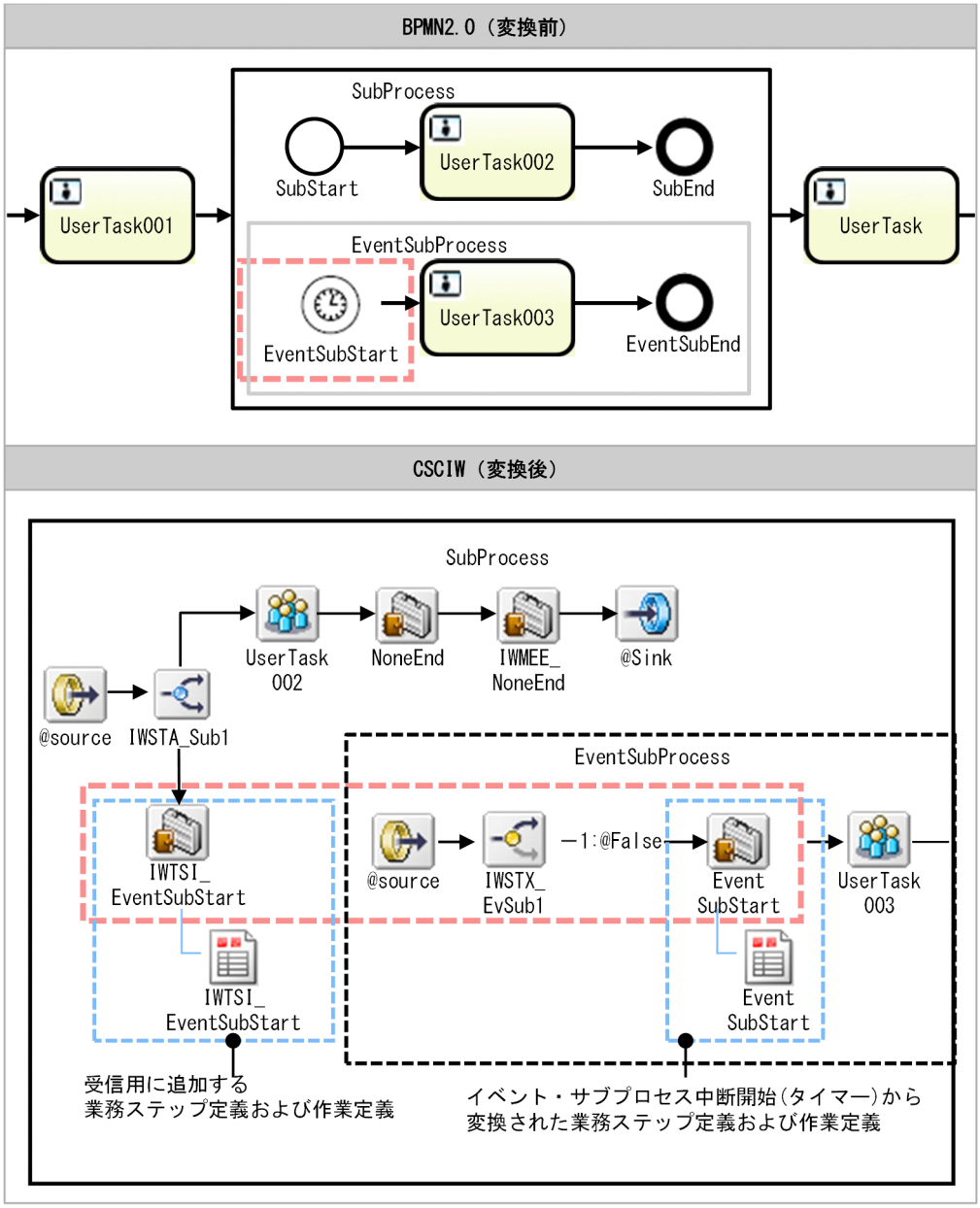
変換イメージ

イベント・サブプロセス中断開始（タイマー）イベントの BPMN ビジネスプロセス定義から CSCIW ビジネスプロセス定義への変換イメージを示します。

イベント・サブプロセス中断開始（タイマー）イベントは，業務ステップ定義（EventSubStart）と作業定義（EventSubStart）に変換されます。

イベント・サブプロセス中断開始（タイマー）が定義されたビジネスプロセスまたはサブプロセスから変換されたトップレベル / 階層定義（SubProcess）には，分業ノード，イベント・サブプロセス中断開始（タイマー）の受信用の業務ステップ，および作業定義（IWTSI_EventSubStart）が定義されます。

図 1-55　イベント・サブプロセス中断開始（タイマー）の変換イメージ



処理内容

イベント・サブプロセス中断開始（タイマー）が定義されたサブプロセスから変換された階層に遷移すると，分業してイベント・サブプロセス中断開始（タイマー）の受信用の業務ステップが実行中になります。また，受信用の作業が生成されて「実行開始可能」状態に遷移します。

受信用の作業は，次のように設定されます。

- 処理期限：設定されたタイマールールを評価して求めた発火時刻
- 優先度（実行回数として使用）：「0」
- 作業者：「IWTTIM_IWTransit」

発火時刻を過ぎると，アプリケーション呼び出しサービスから，イベント・サブプロセス中断開始（タイマー）の受信用の作業に対して遷移要求が投げられます。遷移要求を受けると，同一階層内で実行中の業務ステップ，およびイベント・サブプロセス中断開始（タイマー）受信用の業務ステップを強制終了します。

イベント・サブプロセス中断開始（タイマー）から変換された業務ステップを生成して開始すると，自動的に完了し，遷移先の業務ステップ / 制御ノードに遷移します。

トップレベルに定義されている場合は，同一案件内で実行中の業務ステップを強制終了します。サブプロセスに定義されている場合は，サブプロセス内で実行中の業務ステップを強制終了します。

(39) イベント・サブプロセス非中断開始（タイマー）

機能

イベント・サブプロセス非中断開始（タイマー）は，イベント・サブプロセス非中断開始（タイマー）が定義されたビジネスプロセスまたはサブプロセスが実行中になった時点から，設定されたタイマールールを評価して求めた発火時刻になるまで待ち状態になります。

ビジネスプロセスまたはサブプロセスが完了する前に発火時刻を過ぎると，イベント・サブプロセスが開始されます。繰り返しが設定されている場合は，次の発火時刻まで待ち状態になります。

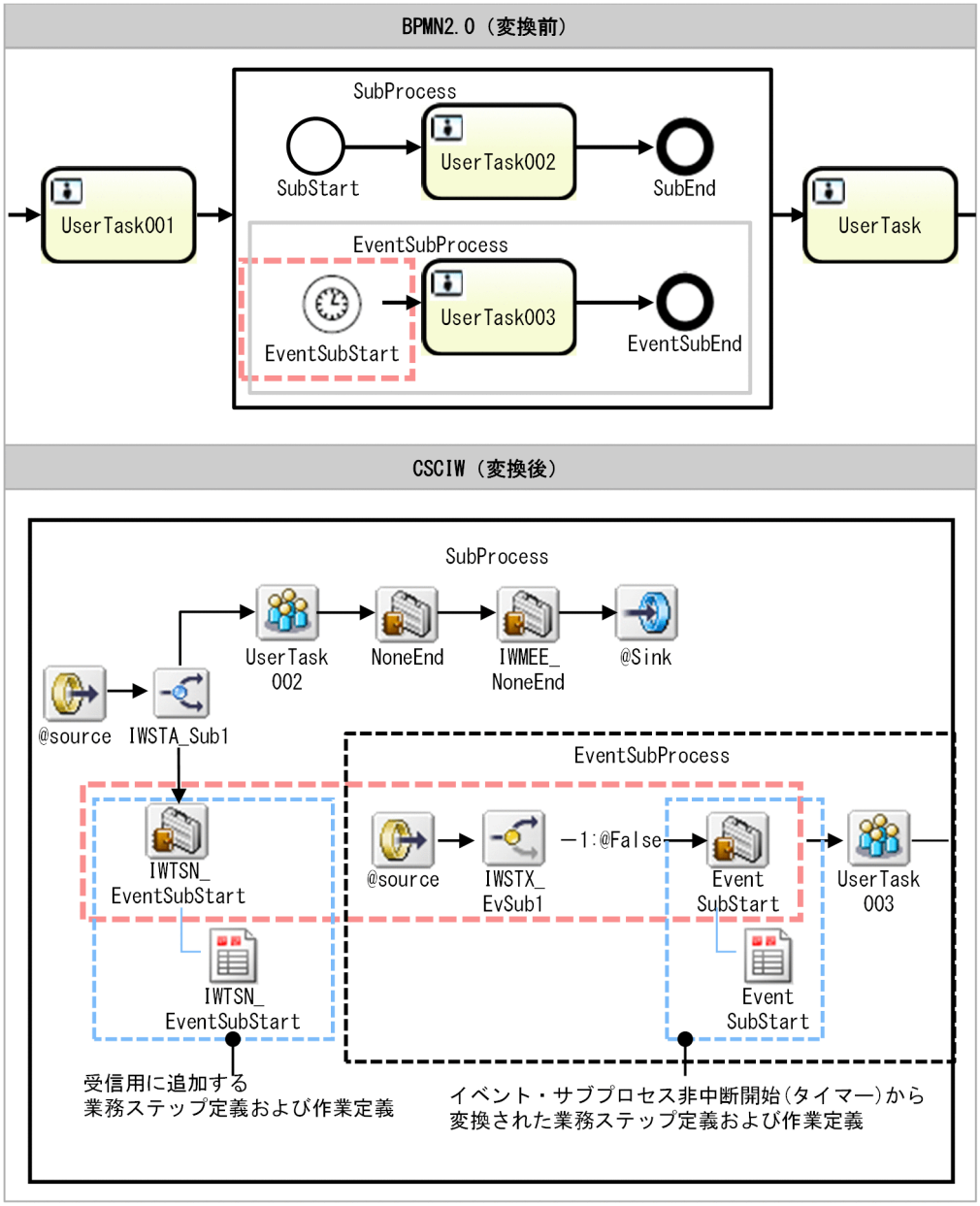
変換イメージ

イベント・サブプロセス非中断開始（タイマー）イベントの BPMN ビジネスプロセス定義から CSCIW ビジネスプロセス定義への変換イメージを示します。

イベント・サブプロセス非中断開始（タイマー）イベントは，業務ステップ定義（EventSubStart）と作業定義（EventSubStart）に変換されます。

イベント・サブプロセス非中断開始（タイマー）が定義されたビジネスプロセスまたはサブプロセスから変換されたトップレベル / 階層定義（SubProcess）には，分業ノード，イベント・サブプロセス中断開始（タイマー）の受信用の業務ステップおよび作業定義（IWTSN_EventSubStart）が定義されます。

図 1-56　イベント・サブプロセス非中断開始（タイマー）の変換イメージ



処理内容

イベント・サブプロセス非中断開始（タイマー）が定義されたサブプロセスから変換された階層に遷移すると，分業してイベント・サブプロセス非中断開始（タイマー）の受信用の業務ステップが実行中になります。また，受信用の作業が生成されて「実行開始可能」状態に遷移します。

受信用の作業は，次のように設定されます。

- 処理期限：設定されたタイマールールを評価して求めた発火時刻
- 実行回数として使用する優先度：「0」
- 作業者：「IWTTIM_IWTransit」

発火時刻を過ぎると，アプリケーション呼び出しサービスから，イベント・サブプロセス非中断開始（タイマー）の受信用の作業に対して遷移要求が投げられます。遷移要求を受けると，イベント・サブプロセス非中断開始（タイマー）から変換された業務ステップを生成して開始します。そして，自動的に完了し，遷移先の業務ステップ / 制御ノードに遷移します。

実行回数が上限まで達していない場合は，受信用の作業は，次のように設定されます。

- 処理期限：タイマールールを再評価して求めた，次の発火時刻
- 優先度（実行回数として使用）：1 を加算

実行回数が上限に達した場合は，受信用の作業および業務ステップを強制終了します。

トップレベルに定義されている場合は，同一案件内で実行中の業務ステップを強制終了します。サブプロセスに定義されている場合は，サブプロセス内で実行中の業務ステップを強制終了します。

(40) 折りたたまれたサブプロセス（シーケンシャルマルチインスタンス）

機能

折りたたまれたサブプロセス（シーケンシャルマルチインスタンス）は，親ビジネスプロセス内のアクティビティになります。

折りたたまれたサブプロセス（シーケンシャルマルチインスタンス）に遷移すると，BPMN ビジネスプロセス定義に指定した繰り返し回数（loopCardinality）分のサブプロセスが 1 件ずつ生成され，逐次に遷移します。繰り返し回数（loopCardinality）が 0 の場合は，何もせずに終了し，次に遷移します。

変換イメージ

BPMN ビジネスプロセス定義から CSCIW ビジネスプロセス定義への変換イメージを示します。

折りたたまれたサブプロセスは，階層定義（SubProcess），および先着ノードに変換されます。

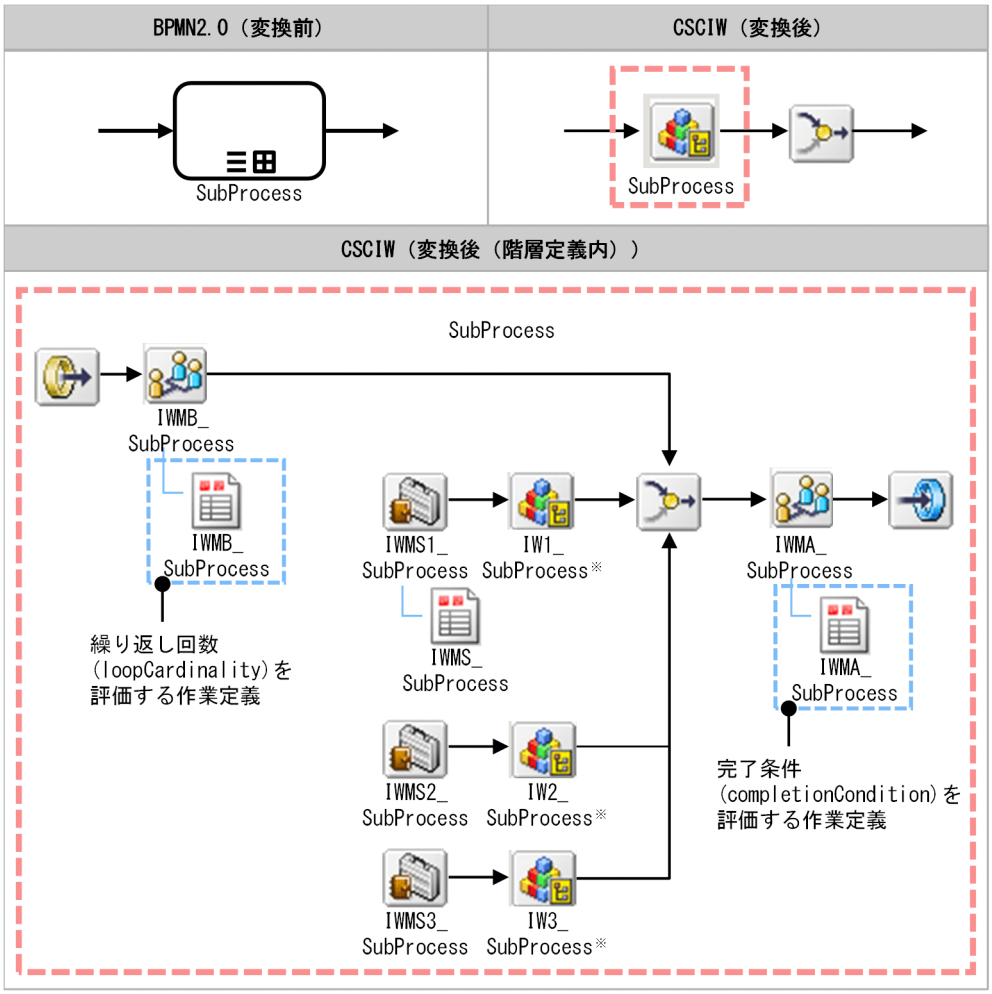
階層定義内（SubProcess）には，次の業務ステップ定義，作業定義，および階層定義が定義されます。

- 繰り返し回数評価用の業務ステップ定義（IWMB_SubProcess），および作業定義（IWMB_SubProcess）
- 繰り返し回数（loopCardinality）分の，インスタンスの動的生成用の業務ステップ定義（IWMS<インデクス>_SubProcess），および作業定義（IWMS_SubProcess）
- 折りたたまれたサブプロセス内に定義された，BPMN 要素の定義用の階層定義（IW<インデクス>_SubProcess）
- 完了条件（completionCondition）評価用の業務ステップ定義（IWMA_SubProcess），および作業定義（IWMA_SubProcess）

業務ステップ定義（IWMS<インデクス>_SubProcess），作業定義（IWMS_SubProcess），および階層定義（IW<インデクス>_SubProcess）が定義される個数については，`ciwtransbpmn` コマンドの`-mimax`

オプションに指定した，サブプロセス（マルチインスタンス）使用時の生成個数，または繰り返し回数（loopCardinality）の設定値に依存します。ciwtransbpmn コマンドの詳細については，マニュアル『uCosminexus Service Coordinator Interactive Workflow 』コマンドを参照してください。なお，この例は，ciwtransbpmn コマンドの-mimax オプションに 3 を指定した場合の変換イメージです。

図 1-57　折りたたまれたサブプロセス（シーケンシャルマルチインスタンス）の変換イメージ



注※

階層定義名，および階層定義内の業務ステップ定義名は，「IW<インデクス>_<BPMN 要素の name 属性>_<BPMN 要素の id 属性>」となります。

処理内容

折りたたまれたサブプロセス（シーケンシャルマルチインスタンス）から変換された階層定義（SubProcess）に遷移すると，業務ステップ（IWMB_SubProcess）に遷移します。

業務ステップ（IWMB_SubProcess）に遷移すると，業務ステップ（IWMB_SubProcess）を完了し，業務ステップ（IWMS1_SubProcess）を 1 件生成します。業務ステップ（IWMS1_SubProcess）は自動的に完了し，階層定義（IW1_SubProcess）に遷移します。

階層定義（IW1_SubProcess）に遷移すると，サブプロセス（シーケンシャルマルチインスタンス）内に定義された BPMN 要素から変換された業務ステップ / 制御ノードに遷移します。

階層定義（IW<インデクス>_SubProcess）が完了した場合の動作

階層定義（IW<インデクス>_SubProcess）が完了し，業務ステップ（IWMA_SubProcess）に遷移すると，マルチインスタンスの完了条件（completionCondition）を評価します。完了条件（completionCondition）が成立した場合，業務ステップ（IWMA_SubProcess）は完了します。完了条件（completionCondition）が成立しない場合または完了条件（completionCondition）の設定を省略した場合は，繰り返し回数（loopCardinality）分のサブプロセスが完了しているかを評価します。

すべてのサブプロセスが完了した場合，業務ステップ（IWMA_SubProcess）は完了します。1 つでも完了していないサブプロセスがある場合，業務ステップ（IWMA_SubProcess）は強制終了され，業務ステップ（IWMS<次のインデクス>_SubProcess）を 1 件生成します。業務ステップ（IWMS<次のインデクス>_SubProcess）は自動的に完了し，階層定義（IW<次のインデクス>_SubProcess）に遷移します。

階層定義（IW<インデクス>_SubProcess）内の業務ステップ / 作業を強制終了した場合の動作

階層定義（IW<インデクス>_SubProcess）内に実行中の業務ステップがなくなった場合，次の業務ステップ / 制御ノードには遷移しなくなります。

階層定義（IW<インデクス>_SubProcess）への遷移と完了を繰り返し，業務ステップ（IWMA_SubProcess）が完了すると，次の業務ステップ / 制御ノードに遷移します。

(41) 折りたたまれたサブプロセス（パラレルマルチインスタンス）

機能

折りたたまれたサブプロセス（パラレルマルチインスタンス）は，親ビジネスプロセス内のアクティビティになります。

折りたたまれたサブプロセス（パラレルマルチインスタンス）に遷移すると，BPMN ビジネスプロセス定義に指定した繰り返し回数（loopCardinality）分の折りたたまれたサブプロセスが同時に生成され，並列に遷移します。繰り返し回数（loopCardinality）が 0 の場合は，何もせずに終了し，次に遷移します。

変換イメージ

BPMN ビジネスプロセス定義から CSCIW ビジネスプロセス定義への変換イメージを示します。

折りたたまれたサブプロセスは，階層定義および先着ノードに変換されます。

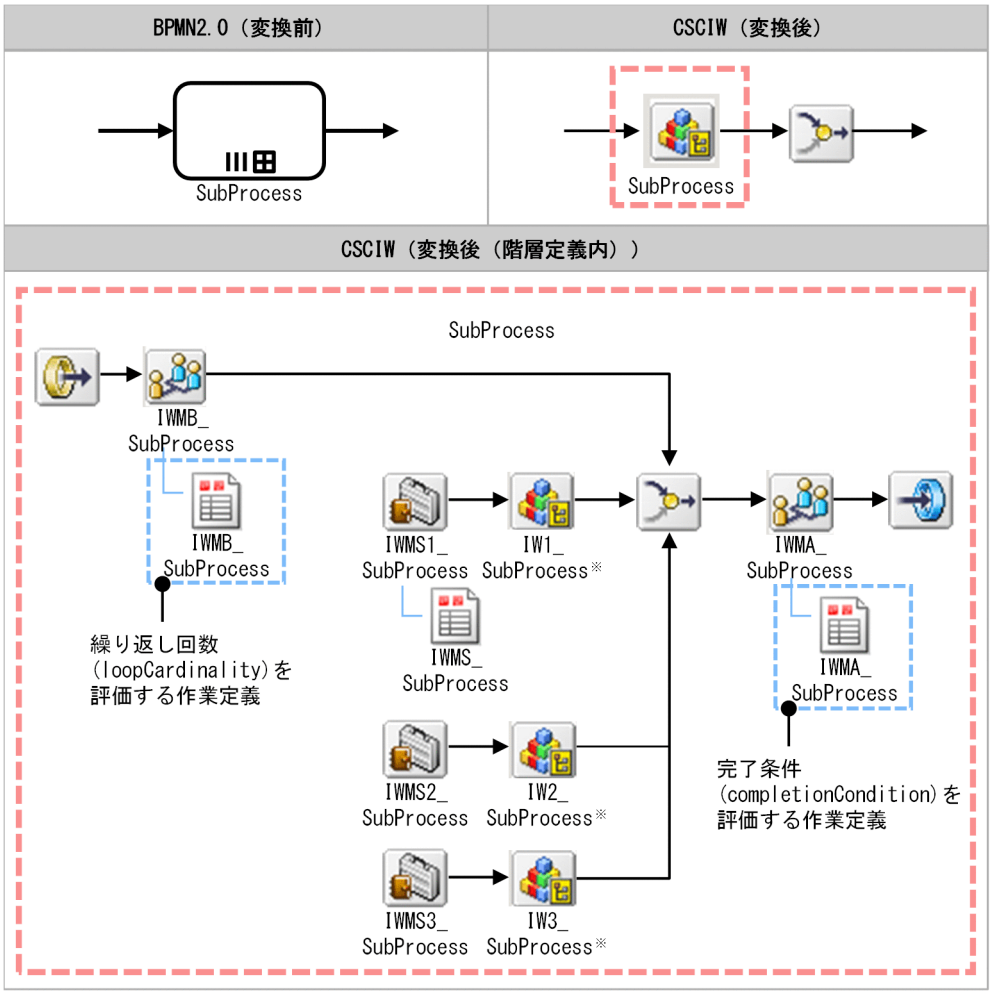
階層定義（SubProcess）内には，次の業務ステップ定義，作業定義，および階層定義が定義されます。

- 繰り返し回数（loopCardinality）評価用の業務ステップ定義，および作業定義（IWMB_SubProcess）
- 繰り返し回数（loopCardinality）分の，インスタンスの動的生成用の業務ステップ定義（IWMS<インデクス>_SubProcess），および作業定義（IWMS_SubProcess）

- 折りたたまれたサブプロセス内に定義された BPMN 要素の定義用の階層定義（IW<インデクス>_SubProcess）
- 完了条件（completionCondition）評価用の業務ステップ定義（IWMA_SubProcess），および作業定義（IWMA_SubProcess）

業務ステップ定義（IWMS<インデクス>_SubProcess），作業定義（IWMS_SubProcess），および階層定義（IW<インデクス>_SubProcess）が定義される個数については，ciwtransbpmn コマンドの-mimax オプションに指定した，サブプロセス（マルチインスタンス）使用時の生成個数，または繰り返し回数（loopCardinality）の設定値に依存します。ciwtransbpmn コマンドの詳細については，マニュアル『uCosminexus Service Coordinator Interactive Workflow コマンド』を参照してください。なお，この例は，ciwtransbpmn コマンドの-mimax オプションに 3 を指定した場合の変換イメージです。

図 1-58　折りたたまれたサブプロセス（パラレルマルチインスタンス）の変換イメージ



注※

階層定義名，および階層定義内の業務ステップ定義名は，「IW<インデクス>_<BPMN 要素の name 属性>_<BPMN 要素の id 属性>」となります。

処理内容

折りたたまれたサブプロセス（パラレルマルチインスタンス）から変換された階層定義（SubProcess）に遷移すると，業務ステップ（IWMB_SubProcess）に遷移します。

業務ステップ（IWMB_SubProcess）に遷移すると，業務ステップ（IWMB_SubProcess）を完了し，繰り返し回数（loopCardinality）分の業務ステップ（IWMS<インデクス>_SubProcess）を同時に生成します。各業務ステップ（IWMS<インデクス>_SubProcess）は自動的に完了し，階層定義（IW<インデクス>_SubProcess）に遷移します。

階層定義（IW<インデクス>_SubProcess）内に遷移すると，サブプロセス（パラレルマルチインスタンス）内に定義された BPMN 要素から変換された業務ステップ / 制御ノードに遷移します。

階層定義（IW<インデクス>_SubProcess）が完了した場合の動作

階層定義（IW<インデクス>_SubProcess）が完了し，業務ステップ（IWMA_SubProcess）に遷移すると，マルチインスタンスの完了条件（completionCondition）を評価します。完了条件（completionCondition）が成立した場合，業務ステップ（IWMA_SubProcess）は完了します。階層定義（SubProcess）内に実行中の業務ステップが存在する場合，すべての業務ステップを強制終了します。完了条件（completionCondition）が成立しない場合または完了条件（completionCondition）の設定を省略した場合は，繰り返し回数（loopCardinality）分のサブプロセスが完了しているかを評価します。

すべてのサブプロセスが完了している場合，業務ステップ（IWMA_SubProcess）は完了します。1 つでも完了していないサブプロセスがある場合，サブプロセスは実行中のままとなり，業務ステップ（IWMA_SubProcess）は強制終了されます。

階層定義（IW<インデクス>_SubProcess）内の業務ステップ / 作業を強制終了した場合の動作

階層定義（IW<インデクス>_SubProcess）内に実行中の業務ステップがなくなった場合，次に遷移しなくなったサブプロセスは完了済みのサブプロセスとは評価されません。そのため，完了条件（completionCondition）が成立する以外，サブプロセス（パラレルマルチインスタンス）の次の業務ステップ / 制御ノードには遷移しません。

階層定義（IW<インデクス>_SubProcess）への遷移と完了を繰り返し，業務ステップ（IWMA_SubProcess）が完了すると，次の業務ステップ / 制御ノードに遷移します。

(42)　展開されたサブプロセス（シーケンシャルマルチインスタンス）

機能

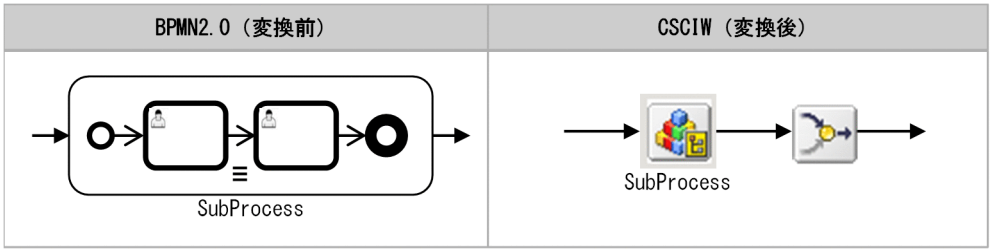
展開されたサブプロセス（シーケンシャルマルチインスタンス）は，親ビジネスプロセス内のアクティビティになります。

展開されたサブプロセス（シーケンシャルマルチインスタンス）に遷移すると，BPMN ビジネスプロセス定義に指定した繰り返し回数（loopCardinality）分のサブプロセスが 1 件ずつ逐次に生成され，遷移します。繰り返し回数（loopCardinality）が 0 の場合は，何もせずに終了し，次に遷移します。

変換イメージ

BPMN ビジネスプロセス定義から CSCIW ビジネスプロセス定義への変換イメージを示します。変換後の変換イメージは，「折りたたまれたサブプロセス（シーケンシャルマルチインスタンス）」と同じです。「1.3.4　BPMN 要素の CSCIW ビジネスプロセス定義への変換の詳細」の「(40)　折りたたまれたサブプロセス（シーケンシャルマルチインスタンス）」を参照してください。

図 1-59　展開されたサブプロセス（シーケンシャルマルチインスタンス）の変換イメージ



処理内容

「折りたたまれたサブプロセス（シーケンシャルマルチインスタンス）」と同じです。「1.3.4　BPMN 要素の CSCIW ビジネスプロセス定義への変換の詳細」の「(40)　折りたたまれたサブプロセス（シーケンシャルマルチインスタンス）」を参照してください。

(43)　展開されたサブプロセス（パラレルマルチインスタンス）

機能

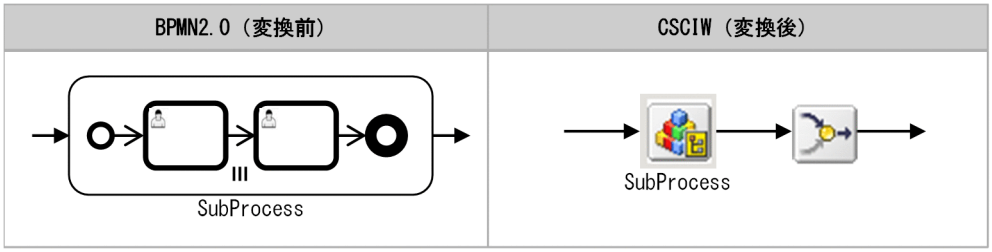
展開されたサブプロセス（パラレルマルチインスタンス）は，親ビジネスプロセス内のアクティビティになります。

展開されたサブプロセス（パラレルマルチインスタンス）に遷移すると，BPMN ビジネスプロセス定義に指定した繰り返し回数（loopCardinality）分のサブプロセスが同時に生成され，並列に遷移します。繰り返し回数（loopCardinality）が 0 の場合は，何もせずに終了し，次に遷移します。

変換イメージ

BPMN ビジネスプロセス定義から CSCIW ビジネスプロセス定義への変換イメージを示します。変換後の変換イメージは，「折りたたまれたサブプロセス（パラレルマルチインスタンス）」と同じです。「1.3.4　BPMN 要素の CSCIW ビジネスプロセス定義への変換の詳細」の「(41)　折りたたまれたサブプロセス（パラレルマルチインスタンス）」を参照してください。

図 1-60　展開されたサブプロセス（パラレルマルチインスタンス）の変換イメージ



処理内容

「折りたたまれたサブプロセス（パラレルマルチインスタンス）」と同じです。「1.3.4　BPMN 要素の CSCIW ビジネスプロセス定義への変換の詳細」の「(41)　折りたたまれたサブプロセス（パラレルマルチインスタンス）」を参照してください。

(44)　展開されたアドホック・サブプロセス

機能

展開されたアドホック・サブプロセスは，親ビジネスプロセス内のアクティビティになります。

展開されたアドホック・サブプロセスに遷移すると，アドホック・サブプロセス内に定義されたフローノード（業務ステップ）を任意のタイミングで生成できます。

アドホック・サブプロセスに遷移したあとの処理の起点は，ユーザの操作になります。ユーザが業務プログラムを介した BPMN 連携ライブラリまたは REST API で，アドホック・サブプロセス内のフローノード（業務ステップ）のフロー制御を行います。

変換イメージ

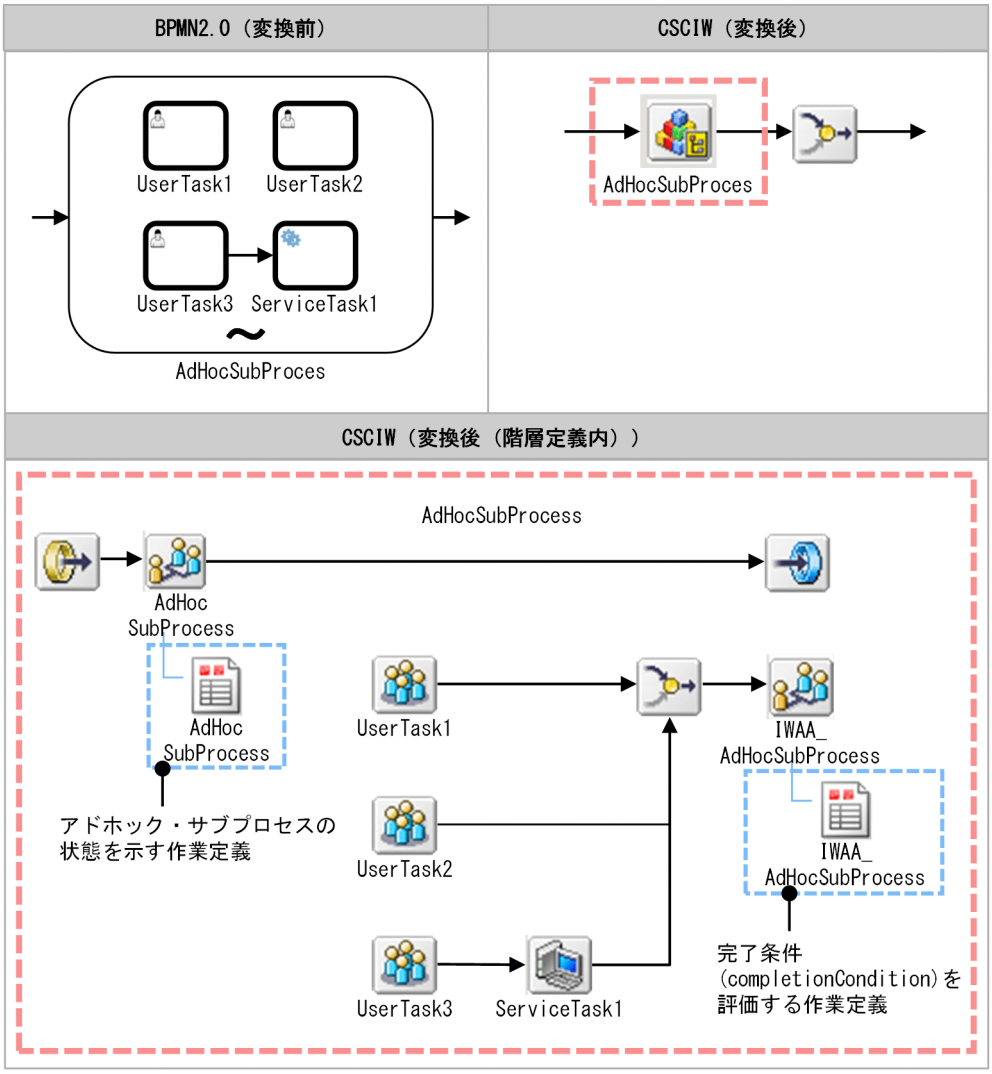
BPMN ビジネスプロセス定義から CSCIW ビジネスプロセス定義への変換イメージを示します。

展開されたアドホック・サブプロセスは，階層定義および先着ノードに変換されます。

階層定義（AdHocSubProcess）内には，次の業務ステップ定義および作業定義が定義されます。

- アドホック・サブプロセスの状態表示用の業務ステップ定義（AdHocSubProcess），および作業定義（AdHocSubProcess）
- アドホック・サブプロセス内に定義された各 BPMN 要素から変換された業務ステップ定義，および作業定義
- 完了条件（completionCondition）評価用の業務ステップ定義（IWAA_AdHocSubProcess），および作業定義（IWAA_AdHocSubProcess）

図 1-61　展開されたアドホック・サブプロセスの変換イメージ



処理内容

展開されたアドホック・サブプロセスから変換された階層定義（AdHocSubProcess）に遷移すると，業務ステップ（AdHocSubProcess）に遷移します。

業務ステップ（AdHocSubProcess）に遷移すると，業務ステップは「実行中」状態に遷移し，アドホック・サブプロセスは「生成可」（作業（AdHocSubProcess）は「実行開始可能」）状態に遷移します。アドホック・サブプロセスが「生成可」（作業（AdHocSubProcess）は「実行開始可能」）状態に遷移すると，ユーザは業務プログラムを介した BPMN 連携ライブラリまたは REST API で，アドホック・サブプロセス内のフロー先端のフローノード（業務ステップ）のインスタンスを生成できます。

アドホック・サブプロセス内のフロー終端のフローノード（UserTask1）が完了すると，業務ステップ（IWAA_AdHocSubProcess）に遷移し，アドホック・サブプロセスの完了条件（completionCondition）を評価します。

- 完了条件（completionCondition）が成立し，かつインスタンスのキャンセル属性（cancelRemainingInstances）が`true`の場合

業務ステップ（IWAA_AdHocSubProcess）を強制終了します。アドホック・サブプロセスは「完了」（作業（AdHocSubProcess）は「実行済」）状態に遷移します。階層定義（AdHocSubProcess）内に実行中の業務ステップが存在するときは，すべての業務ステップを強制終了します。

- 完了条件（completionCondition）が成立し，かつインスタンスのキャンセル属性（cancelRemainingInstances）が`false`の場合
  - 階層定義（AdHocSubProcess）内に実行中の業務ステップが存在するとき

    業務ステップ（IWAA_AdHocSubProcess）を強制終了します。アドホック・サブプロセスは「生成不可」（作業（AdHocSubProcess）は「作業者実行」）状態に遷移します。アドホック・サブプロセスは，アドホック・サブプロセス内の実行中のすべての業務ステップが完了し，フロー終端に到達したあとに完了します。

    ユーザは，アドホック・サブプロセス内のフローノード（業務ステップ）のインスタンスを生成することはできません。
  - 階層定義（AdHocSubProcess）内に実行中の業務ステップが存在しないとき

    業務ステップ（IWAA_AdHocSubProcess）を強制終了します。アドホック・サブプロセスは「完了」（作業（AdHocSubProcess）は「実行済」）状態に遷移します。
- 完了条件（completionCondition）が成立しない場合または完了条件（completionCondition）の設定を省略した場合

  業務ステップ（IWAA_AdHocSubProcess）を強制終了します。アドホック・サブプロセスは「生成可」（作業（AdHocSubProcess）は「実行開始可能」）状態のままになります。

  ユーザはアドホック・サブプロセス内のフローノード（業務ステップ）のインスタンスを再び生成できます。

アドホック・サブプロセス内のフローノード（業務ステップ）のインスタンスの生成と完了を繰り返し，アドホック・サブプロセスが「完了」（作業（AdHocSubProcess）が「実行済」）状態に遷移すると，次の業務ステップ / 制御ノードに遷移します。

(45)　折りたたまれたアドホック・サブプロセス

機能

折りたたまれたアドホック・サブプロセスは，親ビジネスプロセス内のアクティビティになります。

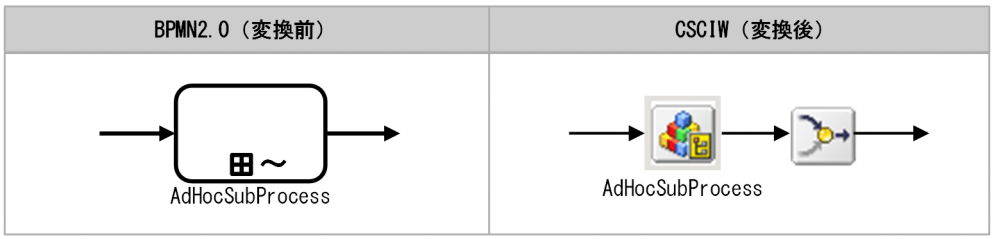
折りたたまれたアドホック・サブプロセスに遷移すると，アドホック・サブプロセス内に定義されたフローノード（業務ステップ）を任意のタイミングで生成できます。アドホック・サブプロセスに遷移したあとの処理の起点は，ユーザの操作になります。ユーザが業務プログラムを介した BPMN 連携ライブラリまたは REST API で，アドホック・サブプロセス内のフローノード（業務ステップ）のフロー制御を行います。

変換イメージ

BPMN ビジネスプロセス定義から CSCIW ビジネスプロセス定義への変換イメージを示します。変換後の変換イメージは，「展開されたアドホック・サブプロセス」と同じです。「1.3.4　BPMN 要素の CSCIW

ビジネスプロセス定義への変換の詳細」の「(44)　展開されたアドホック・サブプロセス」を参照してください。

図 1-62　折りたたまれたアドホック・サブプロセスの変換イメージ



処理内容

「展開されたアドホック・サブプロセス」と同じです。「1.3.4　BPMN 要素の CSCIW ビジネスプロセス定義への変換の詳細」の「(44)　展開されたアドホック・サブプロセス」を参照してください。

1.4　タイマーイベントとは

ここでは，タイマーイベントについて説明します。また，発火時刻とタイマールールの概要についても説明します。

タイマーを指定できる BPMN 要素を次に示します。

- 開始（タイマー）イベント
- キャッチ（タイマー）イベント
- イベント・サブプロセス中断開始（タイマー）イベント
- イベント・サブプロセス非中断開始（タイマー）イベント
- 境界中断（タイマー）イベント
- 境界非中断（タイマー）イベント

タイマーイベントを定義したときの動作を次に示します。

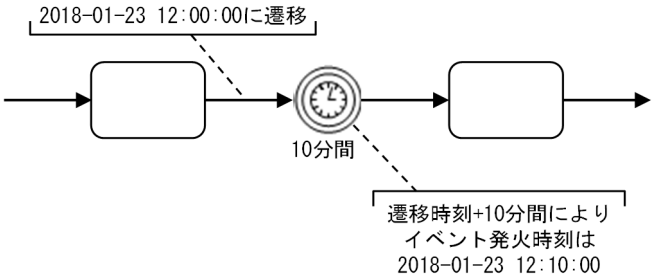
表 1-9　タイマーイベントを定義したときの動作

定義するタイマーイベント	動作
開始イベント	• 指定したパターンの日時での案件投入 （例）毎月 10 日の 12:00 に案件投入する。 • 指定した間隔での案件投入 （例）1 日おきに案件投入する。
キャッチイベント	• 指定した間隔だけ待機 （例）10 分間待機する。
イベント・サブプロセス開始イベント，および境界イベント	• 指定したパターンの日時が経過したあと後のタイムアウト （例）ある日の 17:15 を過ぎたタイミングで，タイムアウトする。 • 指定した間隔を超過したあとのタイムアウト （例）10 分間を超過したあとで，タイムアウトする。

「案件投入」,「待機完了」,「タイムアウト」の動作が発生する時刻を，「タイマーイベントのイベント発火時刻」と定義します。

次に，キャッチ（タイマー）イベントの例を示します。この例では，イベント発火時刻は「2018-01-23 12:10:00」となります。

図 1-63　タイマーイベントの例



1.4.1　タイマーイベントのイベント発火時刻

タイマーイベントのイベント発火について説明します。

BPMN 要素にタイマーイベントを指定した場合の，タイマーイベントの発火までの流れを次に示します。

1. イベント発火時刻の決定タイミングで，次のイベント発火時刻を決定する※1
2. 現在日時がイベント発火時刻を経過したら※2，タイマーイベントを指定した BPMN 要素から遷移する
3. タイマーイベントの定義で最大実行回数（Number of execution）が 2 回以上の場合，次のイベント発火時刻を設定する※1

注※1

イベントの発火時刻は，タイマーイベントのタイマールールによって異なる方式で決定されます。詳細については，「1.4.2　タイマーイベントのタイマールール」を参照してください。1.のイベント発火時刻の決定タイミングは，タイマーイベントを指定した BPMN 要素によって異なります。詳細については，以降の説明を参照してください。

注※2

イベント発火時刻を経過したかどうかは，アプリケーション呼び出しサービスが定期的にチェックします。

(1)　開始（タイマー）イベントのイベント発火時刻

開始（タイマー）イベントのイベント発火時刻の決定タイミングについて説明します。

対象のビジネスプロセス定義に対して，次の条件を最初にすべて満たしたタイミングで，イベント発火時刻を決定します。

- ビジネスプロセス定義が登録済みである
- アプリケーション呼び出しサービスの検索で，対象のビジネスプロセス定義がヒットする

タイマールールで定義された最大実行回数が 2 回以上の場合，アプリケーション呼び出しサービスによる案件投入が成功したあとに，次のイベント発火時刻を決定します。

(2) キャッチ（タイマー）イベントのイベント発火時刻

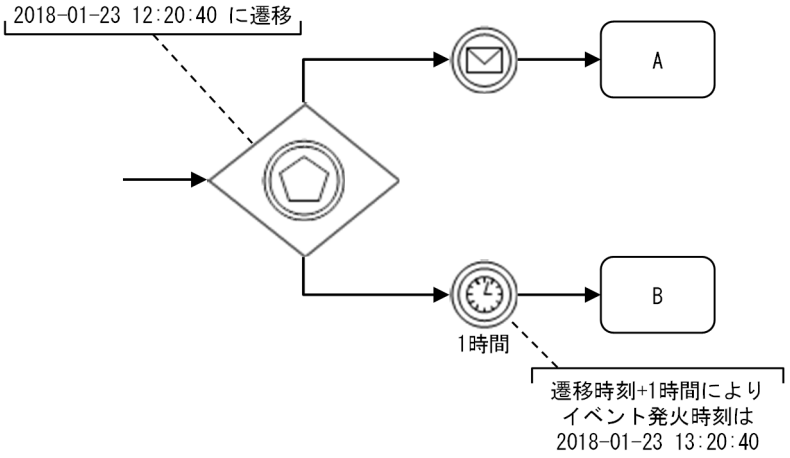
キャッチ（タイマー）イベントのイベント発火時刻の決定タイミングについて説明します。

キャッチ（タイマー）イベントに遷移した時点で，イベント発火時刻を決定します。ただし，排他イベントゲートウェイの遷移先にキャッチ（タイマー）イベントがある場合は，排他イベントゲートウェイに遷移した時点でイベント発火時刻を決定します。

キャッチ（タイマー）イベントのイベント発火時刻の例を次に示します。

この例では，排他イベントゲートウェイに遷移してから 1 時間経過する前にメッセージを受信した場合は，タスク A に遷移します。メッセージを受信しなかった場合は，1 時間後にタスク B に遷移します。

図 1-64　キャッチ（タイマー）イベントのイベント発火時刻の例



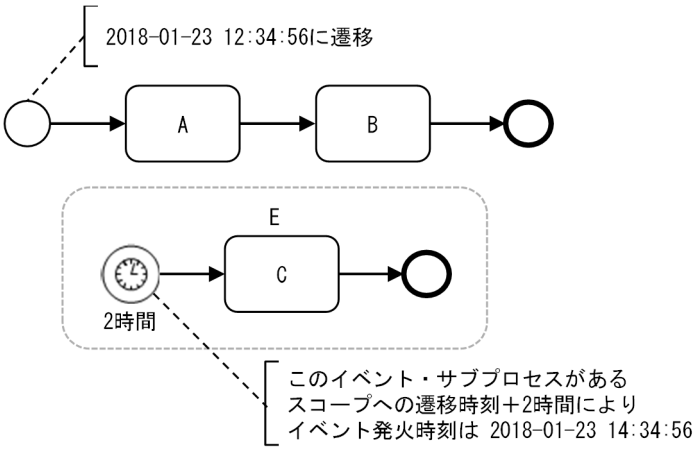
(3) イベント・サブプロセス中断開始（タイマー）イベント／イベント・サブプロセス非中断開始（タイマー）イベントのイベント発火時刻

イベント・サブプロセス中断開始（タイマー）イベント／イベント・サブプロセス非中断開始（タイマー）イベントのイベント発火時刻の決定タイミングについて説明します。

イベント・サブプロセスの対象スコープの開始イベントに遷移した時点で，イベント発火時刻を決定します。

イベント・サブプロセス中断開始（タイマー）イベントのイベント発火時刻の例を次に示します。

図 1-65　イベント・サブプロセス中断開始（タイマー）イベントのイベント発火時刻の例



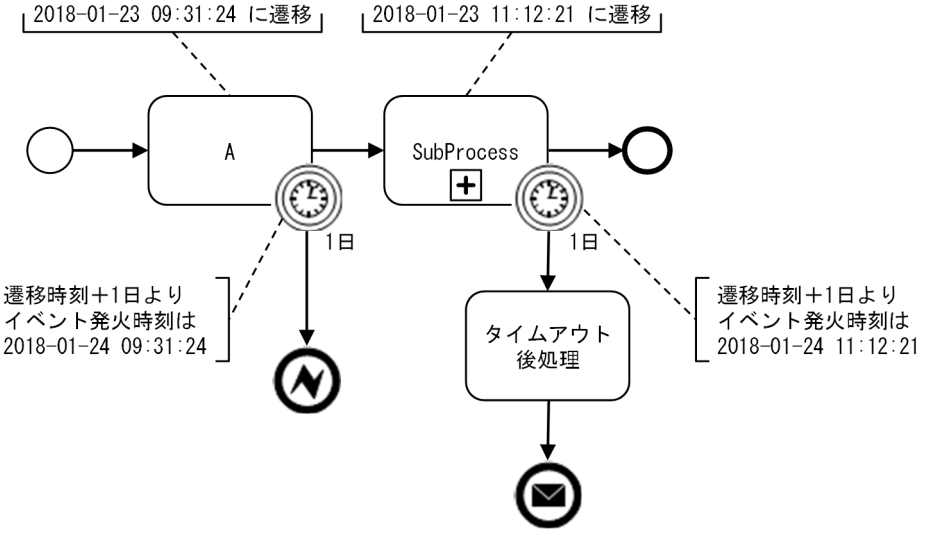
(4)　境界中断（タイマー）イベント／境界非中断（タイマー）イベントのイベント発火時刻

境界中断（タイマー）イベント／境界非中断（タイマー）イベントのイベント発火時刻の決定タイミングについて説明します。

境界イベントがアタッチされた BPMN 要素に遷移した時点で，イベント発火時刻を決定します。

境界中断（タイマー）イベントのイベント発火時刻の例を次に示します。

図 1-66　境界中断（タイマー）イベントのイベント発火時刻の例



1.4.2　タイマーイベントのタイマールール

タイマーイベントのタイマールールの種類について，説明します。

タイマールールとは，タイマーイベントの発火時刻と，イベントの最大実行回数の設定方式です。

タイマーイベントの動作はタイマールールで決まります。タイマールールには次に示す 3 種類があります。

- 固定日時方式（Fixed date and time）
- 間隔方式（Duration）
- 定期日時方式（Periodic date and time）

タイマールールの概要を次の表に示します。

表 1-10　タイマールールの概要

タイマールールの種類	発火時刻の指定方法の概要	最大実行回数※
固定日時方式	ある特定の日時（年月日，および時分）を指定します。 （例） 2018 年 1 月 23 日午前 4 時 56 分 タイマーイベントの発火タイミングは，指定された日時を経過したあととなります。	指定できません。実行回数は 1 になります。
間隔方式	時間間隔を指定します。 （例） 20 分間 タイマーイベントの発火タイミングは，指定された間隔を超過したあととなります。	1 以上を指定します。
定期日時方式	日時のパターンを指定します。 （例） 任意月の 1 日午前 12 時 00 分 最大実行回数を無限とすると，繰り返し（「毎月」や「毎日」など）を指定できます。 （例） 毎月 1 日の午前 12 時 00 分	1 以上を指定します。

注※

次の BPMN 要素の最大実行回数（Number of execution）は，BPMN エディタでの指定値に関わらず，「1」として扱われます。

- キャッチ（タイマー）イベント
- イベント・サブプロセス中断開始（タイマー）イベント
- 境界中断（タイマー）イベント

タイマーイベントを作成する方法については，「(4)　タイマーイベントの作成方法」を参照してください。

(1)　固定日時方式（Fixed date and time）

タイマールールの固定日時方式（Fixed date and time）について説明します。

概要

あらかじめ決められた日時にイベントを発火します。年月日，および時分を指定する必要があります。主に，「タイマーイベントのタイマールール動的変更」と組み合わせて使用します。「タイマーイベントのタイマールール動的変更」の詳細については，「1.4.3　タイマーイベントのタイマールール動的変更」を参照してください。

イベント発火時刻の決め方

イベント発火時刻は，タイマールールで指定された日時となります。イベント発火時刻の決定タイミングに依存しません。

(2)　間隔方式（Duration）

タイマールールの間隔方式（Duration）について説明します。

概要

イベント発火時刻の決定タイミングからの時間間隔を指定します。指定できる単位は，年月日と時分の量となります。

（例）

90 分間

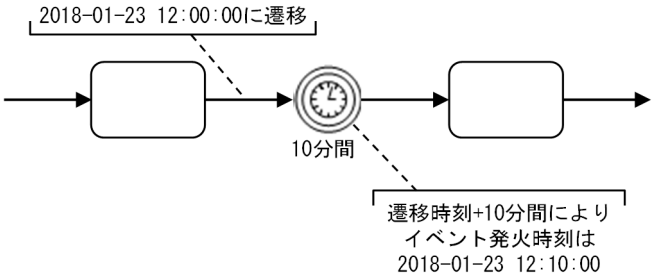
36 時間

イベント発火時刻の決め方

イベント発火時刻は，イベント発火時刻の決定タイミングに依存します。イベント発火時刻の決定タイミングの日時に，指定された時間間隔を加算したものが，イベント発火時刻となります。

間隔方式（Duration）の例を次に示します。

図 1-67　間隔方式（Duration）の例



(3)　定期日時方式（Periodic date and time）

タイマールールの定期日時方式（Periodic date and time）について説明します。

概要

設定した日時パターンに該当する場合に，イベントを発火します。月日と時の値に，任意（Every）を指定できます。年は常に任意の値となります。

（例）

- 「月：任意，日：任意，時：12，分：00」の場合
  任意の日の 12 時 00 分を表します。
- 「月：任意，日：01，時：09，分：00，実行回数：無限」の場合
  毎月 1 日の 9 時 00 分を表します。

イベント発火時刻の決め方

イベント発火時刻は，イベント発火時刻の決定タイミングに依存します。

イベント発火時刻の決定タイミングの日時を経過したときに，設定した日時パターンに該当する日時の中で，直近のものがイベント発火時刻となります。なお，イベント発火時刻の秒以降の単位は，すべて 0 とします。

（例）

次の条件の場合の次回イベント発火時刻を求めます。

- 日時パターン
  月：任意，日：01，時：09，分：00，実行回数：無限
- イベント発火時刻の決定タイミング
  2018 年 1 月 1 日の 12 時 00 分

日時パターンに該当する値は次のとおりです。

```
2018-01-01 09:00
2018-02-01 09:00
2018-03-01 09:00
2018-04-01 09:00
2018-05-01 09:00
       :
```

イベント発火時刻の決定タイミングの日時を超過する直近の時刻がイベント発火時刻なので，この場合の次回イベント発火時刻は，「2018-02-01 09:00:00」となります。

また，日時パターンの「日」を「29」，「30」，「31」のどれかの固定値とした場合，その値が存在しない月に対しては「月末日」に変換されます。

（例）

次の条件の場合の次回イベント発火時刻を求めます。

- 日時パターン
  月：任意，日：31，時：09，分：00

- イベント発火時刻の決定タイミング
  2018 年 2 月 1 日の 12 時 00 分

2 月の月末日がイベント発火時刻なので，この場合の次回イベント発火時刻は「2018-02-28 09:00:00」となります。

1.4.3　タイマーイベントのタイマールール動的変更

タイマーイベントのタイマールール動的変更の概要について，説明します。

BPMN エディタで設定したタイマールールの内容の一部を，案件ごとに変更できます。これによって，外部のサービスで決定するタイムアウト期限や待機時間を，案件ごとに設定できます。

動的変更するには，プロセスデータを使用します。使用するプロセスデータのキー名は，BPMN エディタで設定します。ただし，プロセスデータを利用するため，開始イベントによる案件投入のタイマールールは変更できません。また，最大実行回数も変更できません。

プロセスデータの値は，イベント発火時刻の決定タイミングで参照します。イベント発火時刻の決定タイミングで指定されたプロセスデータが存在しない場合，BPMN エディタで設定したタイマールールで動作します。設定されるプロセスデータ値は，タイマールールの種類によって異なります。タイマールールの種類によって設定されるプロセスデータについて，以降で説明します。

(1)　固定日時方式（Fixed date and time）の場合

固定日時方式（Fixed date and time）のタイマールールに変更する場合のプロセスデータ値について説明します。

形式

次の形式（ISO 8601 の拡張形式）で，プロセスデータに日付と時刻を指定します。

```
yyyy-MM-ddThh:mm:ss
```

yyyy：年を 4 桁で指定します。

MM：月を 2 桁（01～12）で指定します。

dd：日を 2 桁（01～31）で指定します。

hh：時を 2 桁（00～23）で指定します。

mm：分を 2 桁（00～59）で指定します。

ss：秒を 2 桁（00～59）で指定します。

ポイント

年のフィールドは必ず 4 桁で指定してください。また，月，日，時，分，秒のフィールドは必ず 2 桁で指定してください。

正しい指定例

```
2018-01-02T03:04:05
```

誤った指定例

```
2018-1-2T3:4:5
```

(2) 間隔方式（Duration）の場合

間隔方式（Duration）のタイマールールに変更する場合のプロセスデータ値について説明します。

形式

次の形式（ISO 8601 の間隔を指定する形式の一つ）で，プロセスデータに日付と時刻を指定します。

```
PnyYnmMndDTnhHnmiM
```

n_y：年を整数で指定します。

n_m：月を整数で指定します。

n_d：日を整数で指定します。

n_h：時を整数で指定します。

n_{mi}：分を整数で指定します。

ポイント

量が 0 であることを表す場合

整数値とフィールドの指定文字（Y，M，D，H，M）を省略できます。

1 時間 30 分の表記法を次に示します。年，月，日を示すフィールドの指定文字（Y，M，D）を省略できます。

```
PT1H30M
```

時，および分の量が 0 であることを表す場合

T 以降をすべて省略できます。

1 か月間の表記法を次に示します。年，日を示すフィールドの指定文字（Y，D），および T 以降を省略できます。

```
P1M
```

ただし，すべてのフィールドを省略することはできません。

(3) 定期日時方式（Periodic date and time）の場合

定期日時方式（Periodic date and time）のタイマールールに変更する場合のプロセスデータ値について説明します。

形式

次の形式で，プロセスデータに日付と時刻を指定します。

```
*-M-dTh:m
```

M：月を整数（1～12）で指定します。または「*」を指定します。

d：日を整数（1～31）で指定します。または「*」を指定します。

h：時を整数（0～23）で指定します。または「*」を指定します。

m：分を整数（00～59）で指定します。

ポイント

すべての値を取得したいフィールド（月，日，時）には「*」を指定します。「年」のフィールドは，常にすべての値を取得します。

ある月の 15 日午前 9 時 00 分の表記法

```
*-*-15T9:00
```

指定された文字列から求められる日時の「秒」の値は 0 として扱われます。

(4) サブプロセス（マルチインスタンス）でのタイマールール動的変更

サブプロセス（マルチインスタンス）の場合に，タイマールールを動的に変更するときの設定方法について説明します。

サブプロセス（マルチインスタンス）の中にタイマーイベントが定義されている場合，マルチインスタンスのインスタンスごとに異なるプロセスデータ値で，タイマールールを動的変更できます。

BPMN エディタでは，{MIIndex}を付与した形式でプロセスデータキー名を指定します。指定例を次に示します。

指定例

```
$STimerDateTime{MIIndex}
```

実行環境では，リスト型プロセスデータを登録する必要があります。

1.4.4　タイマーイベントのイベント発火時刻の変更

タイマーイベントのイベント発火時刻の変更について，説明します。

次の条件をすべて満たす場合は，CSCIW が提供する機能（JavaAPI，または REST API）によって，すでに設定されたタイマーイベントのイベント発火時刻を変更できます。

- タイマーイベントを指定した BPMN 要素が次のどれかである
  - キャッチ（タイマー）イベント
  - イベント・サブプロセス中断開始（タイマー）イベント
  - イベント・サブプロセス非中断開始（タイマー）イベント
  - 境界中断（タイマー）イベント
  - 境界非中断（タイマー）イベント
- イベント発火時刻の決定タイミング経過後であり，イベント発火時刻が規定の作業のインスタンスに設定されている
- 規定の作業のインスタンスが「未終了」である

イベント発火時刻が設定された規定の作業定義を，次の表に示します。

表 1-11　イベント発火時刻を設定する作業定義

BPMN 要素	作業定義名
中間イベント（キャッチ）	＜ BPMN 要素名＞_＜ BPMN 要素 ID ＞
開始イベント（中断）	IWTSI_＜ BPMN 要素名＞_＜ BPMN 要素 ID ＞
開始イベント（非中断）	IWTSN_＜ BPMN 要素名＞_＜ BPMN 要素 ID ＞
境界イベント（中断） （アタッチ先がサブプロセスのとき）	IWTPI_＜ BPMN 要素名＞_＜ BPMN 要素 ID ＞
境界イベント（中断） （アタッチ先がサブプロセス以外のとき）	IWTBI_＜ BPMN 要素名＞_＜ BPMN 要素 ID ＞
境界イベント（非中断） （アタッチ先がサブプロセスのとき）	IWTPN_＜ BPMN 要素名＞_＜ BPMN 要素 ID ＞
境界イベント（非中断）	IWTBN_＜ BPMN 要素名＞_＜ BPMN 要素 ID ＞

BPMN 要素	作業定義名
（アタッチ先がサブプロセス以外のとき）	IWTBN_< BPMN 要素名>_< BPMN 要素 ID >

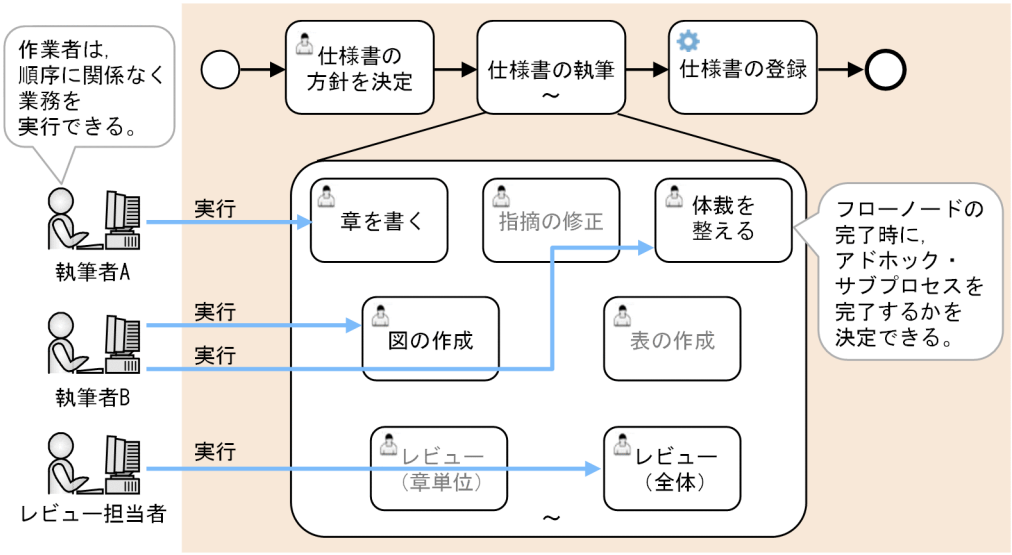
変更のための JavaAPI は，CIWBPMNLib#setDeadlineForTimer（Connection,CIWServer,Integer,Integer,Date）です。

1.5　アドホック・サブプロセスとは

アドホック・サブプロセスについて説明します。

アドホック・サブプロセスは，BPMN2.0 で規定される非定型プロセスを表記するための BPMN 要素です。アドホック・サブプロセスでは，通常のサブプロセスとは異なり，BPMN 要素の実行順序を規定しません。アドホック・サブプロセスは，次に示す使用例のように「仕様書を執筆する」という非定型のプロセスを BPMN で表記する場合に使用します。

図 1-68　アドホック・サブプロセスの使用例



[説明]

アドホック・サブプロセスには，実行候補のフローノードを定義しておきます。

アドホック・サブプロセスに遷移すると，仕様書の執筆者は，アドホック・サブプロセスに定義したフローノードのどれかを実行します。執筆者は，仕様書のページ数や難易度によって，フローノードの実行順序を変更できます。また，アドホック・サブプロセスが完了するタイミングを自由に決定できます。

1.5.1　アドホック・サブプロセスの定義内容

BPMN エディタで，次に示すアドホック・サブプロセスの属性を定義できます。

表 1-12　アドホック・サブプロセスの定義内容

属性	説明
completionCondition	アドホック・サブプロセスの完了条件の評価式です。 評価式が`true` に評価されると，アドホック・サブプロセスは完了します。評価式の指定を省略した場合，完了条件は評価されません。この場合，API 呼び出しによって，アドホック・サブプロセスを完了させることができます。

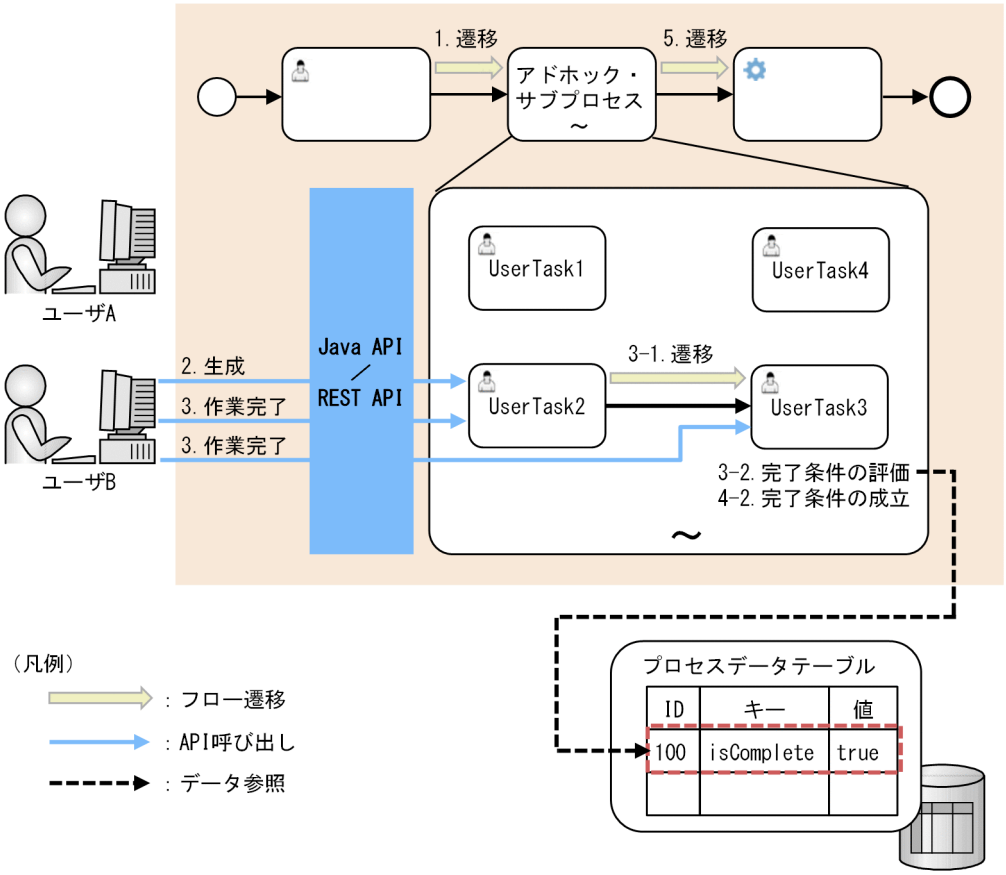
属性	説明
ordering	アドホック・サブプロセス内のフローノードの実行方式です。実行方式の選択値によって，フローノードの生成条件が変わります。 • Parallel を選択した場合 API 呼び出しによって，アドホック・サブプロセス内のフローノードを複数同時に生成します。 • Sequential を選択した場合 API 呼び出しによって，アドホック・サブプロセス内のフローノードを 1 件ずつ逐次に生成します。アドホック・サブプロセス内に「実行中」状態のフローノードが存在する場合は，フローノードを生成できません。
cancelRemainingInstances	完了条件（completionCondition）が成立した場合に，「実行中」状態のフローノードを強制終了するかどうかの設定値です。 • true を選択した場合 すべての「実行中」状態のフローノードを強制終了したあとに，アドホック・サブプロセスを完了します。 • false を選択した場合 「実行中」状態のフローノードはそのまま実行されます。すべてのフローノードが完了したあとに，アドホック・サブプロセスを完了します。

1.5.2 アドホック・サブプロセスの基本的な処理の流れ

アドホック・サブプロセスに遷移したあとの処理の起点は，ユーザ（業務プログラム）の操作となります。

アドホック・サブプロセスの基本的な処理の流れを，次の図に示します。

図 1-69　アドホック・サブプロセスの基本的な処理の流れ



[説明]

1. アドホック・サブプロセスに遷移すると，アドホック・サブプロセスは「生成可」状態になります。アドホック・サブプロセスが「生成可」状態の場合，ユーザ（業務プログラム）は，API 呼び出しによって，アドホック・サブプロセス内のフロー先端※1 のフローノード（業務ステップ）を生成できます。
2. アドホック・サブプロセス内のフローノード（業務ステップ）を生成すると，対応する作業が「実行可能」状態になります。ユーザ（業務プログラム）は，作業を処理します。
3. 作業が完了すると，定義内容によって，次のどちらかの処理を行います。

   3-1. 作業がフロー終端※2 ではない場合，次のフローノード（業務ステップ）が生成されます。2.に戻ります。

   3-2. 作業がフロー終端※2 の場合，アドホック・サブプロセスの完了条件（completionCondition）が評価されます。4.へ進みます。
4. 完了条件（completionCondition）の評価結果によって，次のどちらかの処理を行います。この図では 4-2.の処理に進みます。

   4-1. 完了条件（completionCondition）を満たさない場合または完了条件（completionCondition）の設定を省略した場合は，アドホック・サブプロセスは「生成可」状態のままになります。2.に戻ります。

4-2. 完了条件（completionCondition）を満たす場合，アドホック・サブプロセスは「完了」状態になります。5.へ進みます。※3

5. アドホック・サブプロセスが「完了」状態になると，次の業務ステップ/制御ノードへ遷移します。

注※1

フロー先端とは，入力シーケンスフローが付与されていないアクティビティのことです。この例では，UserTask1，UserTask2，および UserTask4 がフロー先端です。

注※2

フロー終端とは，出力シーケンスフローが付与されていないアクティビティのことです。この例では，UserTask1，UserTask3，および UserTask4 がフロー終端です。

注※3

完了条件（completionCondition）を満たす場合の動作は，インスタンスキャンセル属性（cancelRemainingInstances）の設定値によって変更できます。詳細は，「1.5.3　完了条件（completionCondition）の評価時の処理の流れ」を参照してください。

1.5.3　完了条件（completionCondition）の評価時の処理の流れ

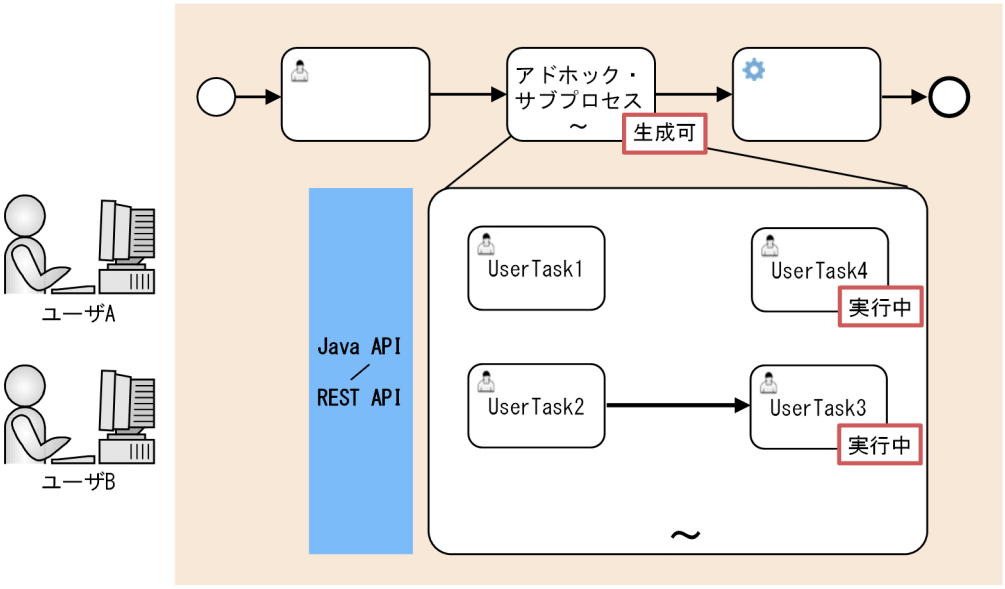
完了条件（completionCondition）を満たす場合，アドホック・サブプロセスのインスタンスキャンセル属性（cancelRemainingInstances）の設定値によって，次に示す処理が実行されます。

(1)　インスタンスキャンセル属性（cancelRemainingInstances）が true の場合

インスタンスキャンセル属性がtrue の場合，すべての「実行中」状態のフローノード（業務ステップ）が強制終了されたあとに，アドホック・サブプロセスは「完了」状態になります。

インスタンスキャンセル属性がtrue の場合の処理の流れを，次の図に示します。

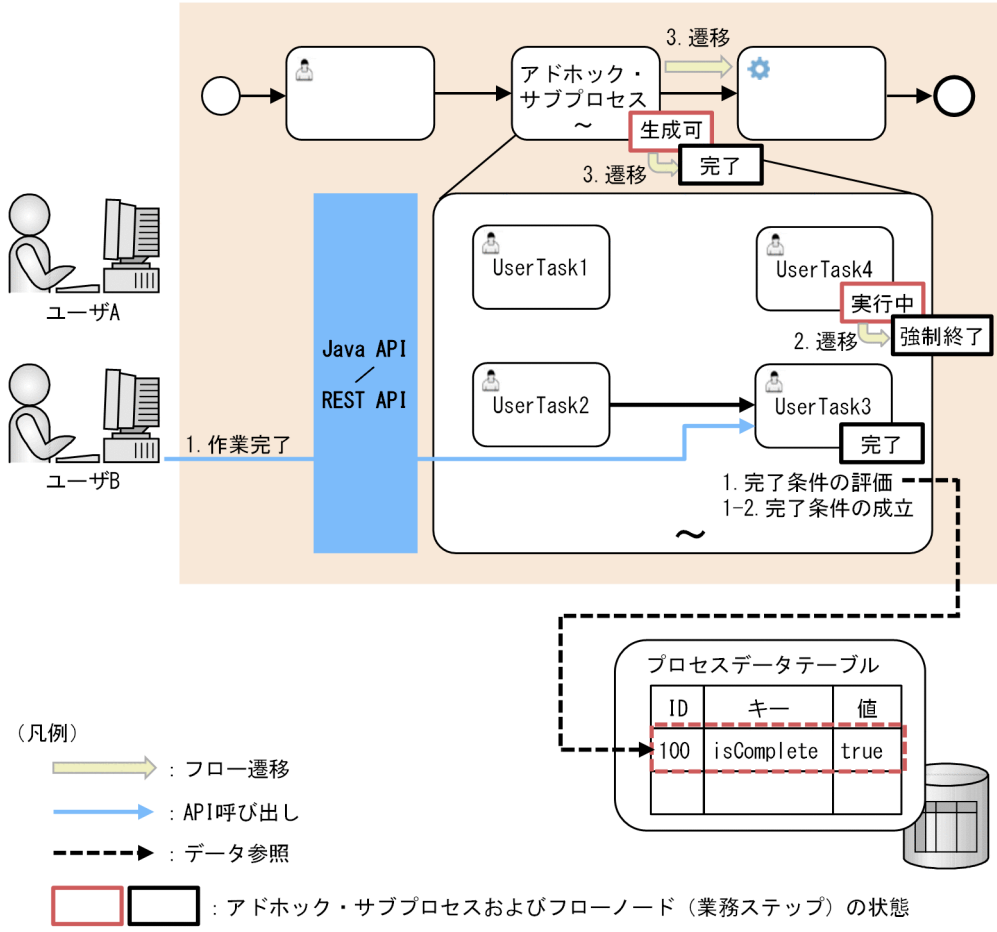
図 1-70　完了条件（completionCondition）の評価前の状態



（凡例）

：アドホック・サブプロセスおよびフローノード（業務ステップ）の状態

図 1-71　完了条件（completionCondition）の評価時の処理の流れ



（凡例）

：フロー遷移

：API呼び出し

：データ参照

：アドホック・サブプロセスおよびフローノード（業務ステップ）の状態

［説明］

1. 完了条件（completionCondition）の評価結果によって，次のどちらかの処理を行います。この図では 1-2.の処理に進みます。

   1-1. 完了条件（completionCondition）を満たさない場合，アドホック・サブプロセスは「生成可」状態のままになります。

   1-2. 完了条件（completionCondition）を満たす場合，2.へ進みます。

2. アドホック・サブプロセス内に存在するすべての「実行中」状態のフローノード（業務ステップ）は，強制終了されます。

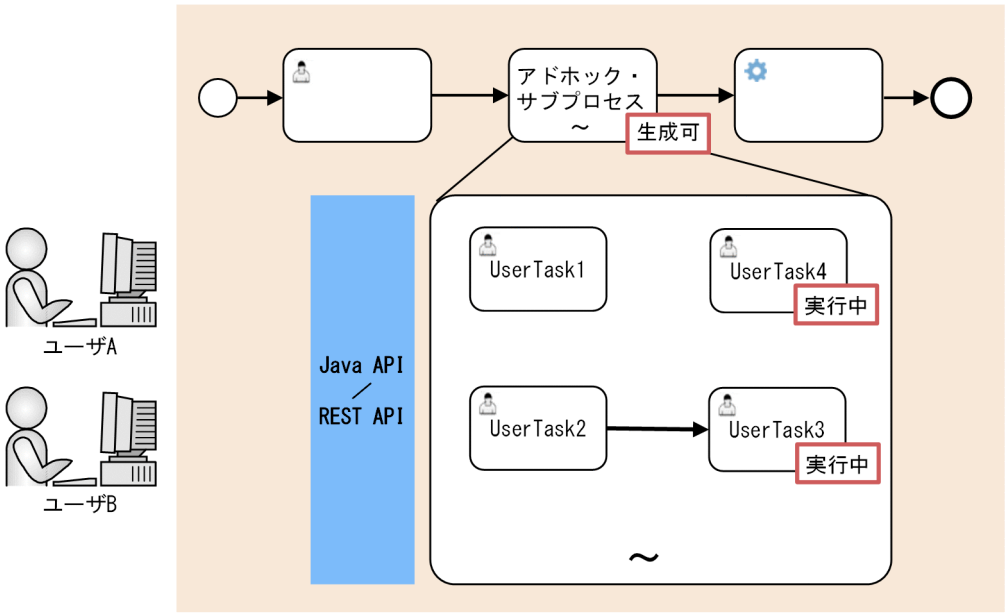
3. アドホック・サブプロセスは「完了」状態になり，次の業務ステップ/制御ノードへ遷移します。

(2) インスタンスキャンセル属性（cancelRemainingInstances）が false の場合

インスタンスキャンセル属性がfalse の場合，「実行中」状態のフローノード（業務ステップ）はそのまま実行されます。すべてのフローノード（業務ステップ）が完了したあとに，アドホック・サブプロセスは「完了」状態になります。

インスタンスキャンセル属性がfalse の場合の処理の流れを，次の図に示します。

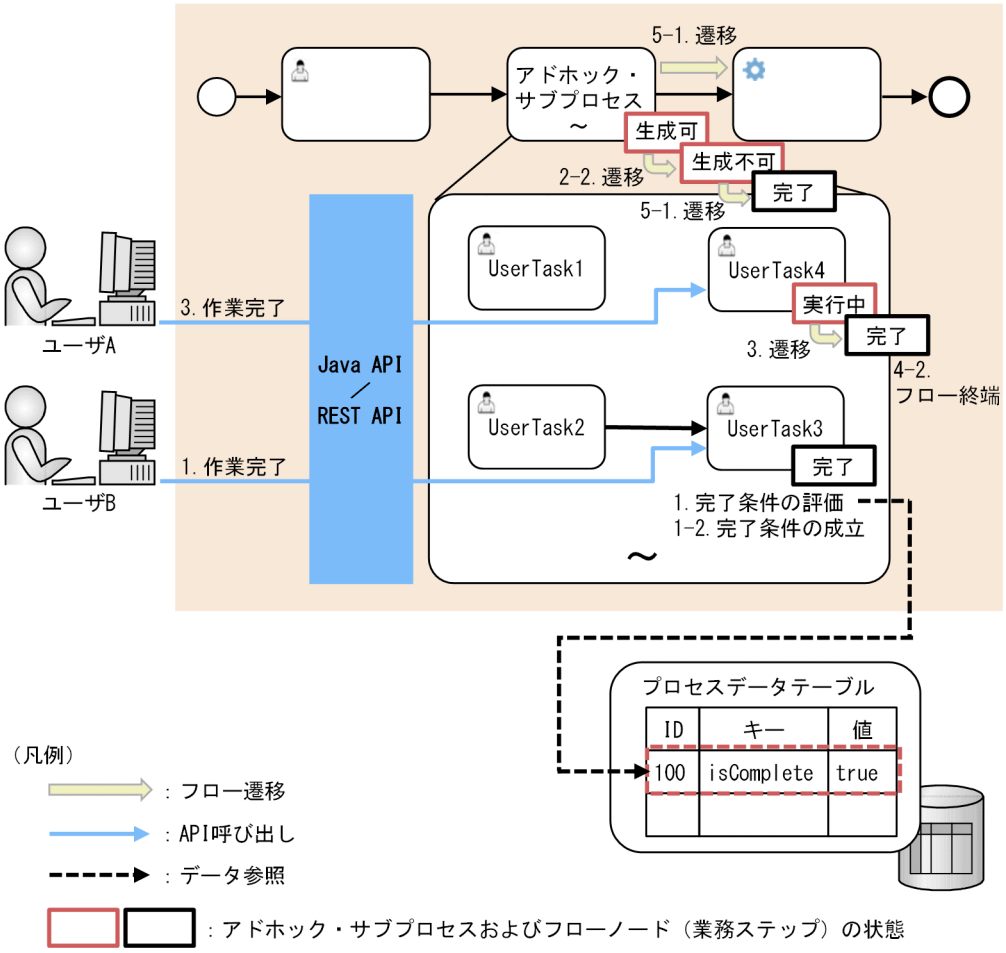
図 1-72 完了条件（completionCondition）の評価前の状態



（凡例）

：アドホック・サブプロセスおよびフローノード（業務ステップ）の状態

図 1-73　完了条件（completionCondition）の評価時の処理の流れ



[説明]

1. 作業が完了すると，完了条件（completionCondition）の評価結果によって，次のどちらかの処理を行います。この図では 1-2.の処理に進みます。

   1-1. 完了条件（completionCondition）を満たさない場合，アドホック・サブプロセスは「生成可」状態のままになります。

   1-2. 完了条件（completionCondition）を満たす場合，2.へ進みます。

2. アドホック・サブプロセス内に存在するフローノード（業務ステップ）の状態によって，次のどちらかの処理を行います。この図では 2-2.の処理に進みます。

   2-1. 「実行中」状態のフローノード（業務ステップ）が存在しない場合，アドホック・サブプロセスは「完了」状態になります。

   2-2. 「実行中」状態のフローノード（業務ステップ）が存在する場合，アドホック・サブプロセスは「生成不可」状態になります。3.へ進みます。

3. アドホック・サブプロセスの状態が「生成不可」状態になると，「実行中」状態のフローノード（業務ステップ）は，そのまま実行されます。ユーザ（業務プログラム）は，フローノード（業務ステップ）に対応する作業を処理します。

4. 作業が完了すると，定義内容によって，次のどちらかの処理を行います。この図では 4-2.の処理に進みます。

4-1. フロー終端ではない場合，次のフローノード（業務ステップ）が生成されます。3.に戻ります。

4-2. フロー終端の場合，5.へ進みます。

5. アドホック・サブプロセス内に存在するフローノード（業務ステップ）の状態によって，次のどちらかの処理を行います。この図では 5-1.の処理に進みます。

5-1. 「実行中」状態のフローノード（業務ステップ）が存在しない場合，アドホック・サブプロセスは「完了」状態になります。

5-2. 「実行中」状態のフローノード（業務ステップ）が存在する場合，3.に戻ります。

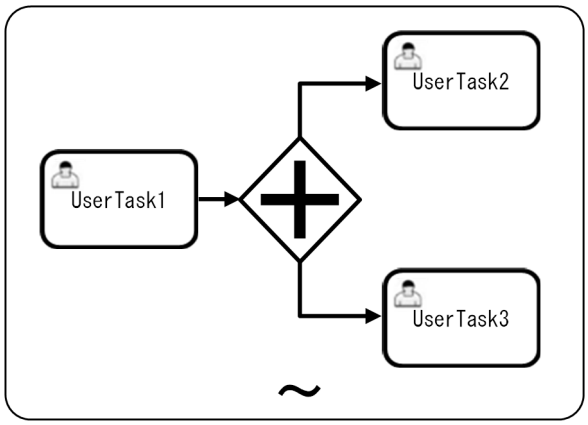
1.5.4 完了条件（completionCondition）の評価のタイミング

アドホック・サブプロセスの完了条件（completionCondition）は，アドホック・サブプロセス内に定義されたフロー終端のフローノードが完了した時点で，評価されます。

並列ゲートウェイが定義されている場合の評価のタイミング

アドホック・サブプロセス内に並列ゲートウェイが定義されている場合の例を，次に示します。

図 1-74 並列ゲートウェイが定義されている場合の例



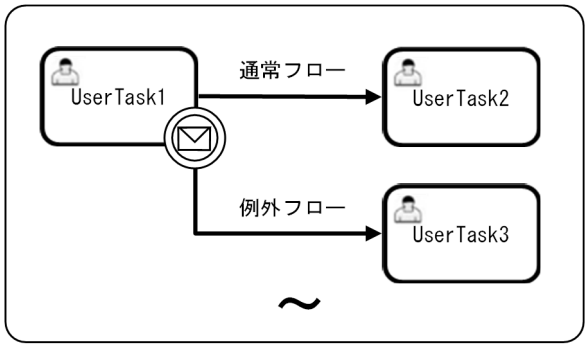
この例では，次のタイミングで完了条件（completionCondition）が評価されます。

- UserTask2 が完了した時点
- UserTask3 が完了した時点

境界イベントの例外フローが定義されている場合の評価のタイミング

アドホック・サブプロセス内に境界イベントの例外フローが定義されている場合の例を，次に示します。

図 1-75　境界イベントの例外フローが定義されている場合の例



この例では，次のタイミングで完了条件（completionCondition）が評価されます。

- UserTask2 が完了した時点
- UserTask3 が完了した時点

1.5.5　アドホック・サブプロセス内に定義できる BPMN 要素

CSCIW でサポートする，アドホック・サブプロセス内に定義できる BPMN 要素を，次の表に示します。

表 1-13　アドホック・サブプロセス内に定義できる BPMN 要素

BPMN 要素	
イベント	キャッチ（メッセージ）
	キャッチ（リンク）
	キャッチ（タイマー）
	スロー（メッセージ）
	スロー（リンク）
	境界非中断（メッセージ）
	境界中断（メッセージ）
	境界中断（エラー）
	境界非中断（タイマー）
	境界中断（タイマー）
アクティビティ	ユーザタスク
	ユーザタスク（シーケンシャルマルチインスタンス）
	ユーザタスク（パラレルマルチインスタンス）
	サービスタスク
	サービスタスク（シーケンシャルマルチインスタンス）

BPMN 要素	
アクティビティ	サービスタスク（パラレルマルチインスタンス）
	ビジネスルールタスク
	ビジネスルールタスク（シーケンシャルマルチインスタンス）
	ビジネスルールタスク（パラレルマルチインスタンス）
	コールアクティビティ
	コールアクティビティ（シーケンシャルマルチインスタンス）
	コールアクティビティ（パラレルマルチインスタンス）
ゲートウェイ	並列ゲートウェイ
	排他ゲートウェイ
	排他イベントゲートウェイ
接続オブジェクト	シーケンスフロー
	関連
	データの関連
その他	テキスト注釈
	グループ

アドホック・サブプロセスには，1 つ以上のアクティビティを定義する必要があります。

アドホック・サブプロセス内のフロー先端に定義できるのは，アクティビティだけです。フロー終端には，アクティビティ，並列ゲートウェイ，または排他ゲートウェイのどれかを定義できます。

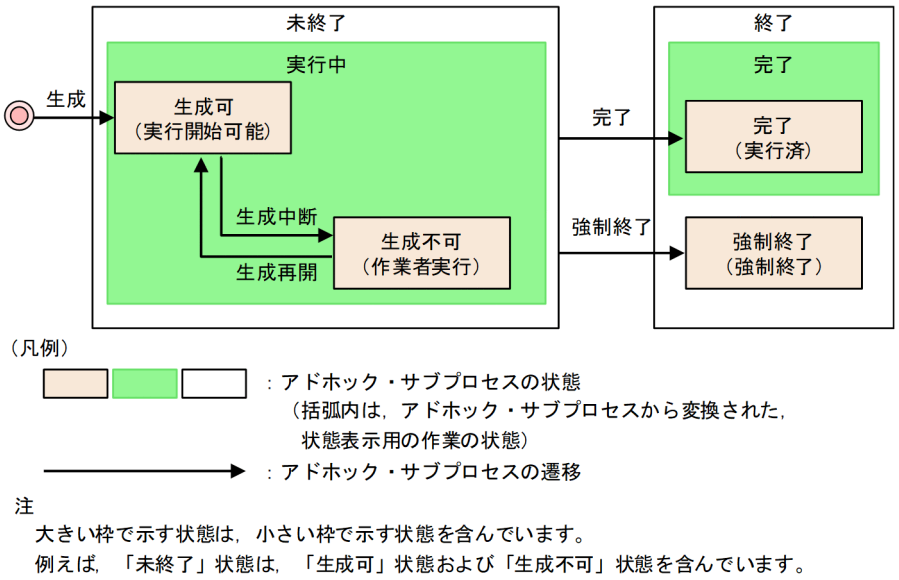
1.5.6　アドホック・サブプロセスの状態遷移モデル

アドホック・サブプロセスの状態遷移について説明します。

(1)　アドホック・サブプロセスの状態遷移

アドホック・サブプロセスの状態遷移を，次の図に示します。

図 1-76　アドホック・サブプロセスの状態遷移



注
大きい枠で示す状態は，小さい枠で示す状態を含んでいます。
例えば，「未終了」状態は，「生成可」状態および「生成不可」状態を含んでいます。

(2)　アドホック・サブプロセスの状態

アドホック・サブプロセスの状態を，次の表に示します。

表 1-14　アドホック・サブプロセスの状態

状態	説明
生成可	アドホック・サブプロセス内のフローノード（業務ステップ）を生成できる状態です。 アドホック・サブプロセスが「生成可」状態のときだけ，API 呼び出しによるアドホック・サブプロセス内のフローノード（業務ステップ）を生成できます。また，フロー終端のフローノード（業務ステップ）が完了した時点で，完了条件（completionCondition）が評価されます。
生成不可	アドホック・サブプロセス内のフローノード（業務ステップ）を生成できない状態です。 アドホック・サブプロセスが「生成不可」状態になると，API 呼び出しによるアドホック・サブプロセス内のフローノード（業務ステップ）を生成できなくなります。また，完了条件（completionCondition）は評価されなくなります。すでに実行されているフローノード（業務ステップ）はそのまま実行されます。
完了	アドホック・サブプロセスが完了し，次にフロー遷移した状態です。 「完了」状態のアドホック・サブプロセスに含まれている，すべてのフローノード（業務ステップ）は「完了」状態または「強制終了」状態です。
強制終了	アドホック・サブプロセスが強制的に終了された状態です。 境界イベントによって，アドホック・サブプロセスが中断要求された場合，アドホック・サブプロセスは「強制終了」状態になります。

(3)　アドホック・サブプロセスの遷移

アドホック・サブプロセスの遷移を，次の表に示します。

表 1-15　アドホック・サブプロセスの状態一覧

遷移種別	要因 / 動作	説明
生成	要因	ワークフローシステムの制御（アドホック・サブプロセスへのフロー遷移）
	動作	アドホック・サブプロセスを「生成可」状態で生成します。
生成中断	要因	• API 呼び出しによる遷移要求 • 完了条件（completionCondition）を満たす場合に，インスタンスのキャンセル属性（cancelRemainingInstances）が`false`，かつアドホック・サブプロセス内に実行中のフローノード（業務ステップ）が存在するとき
	動作	アドホック・サブプロセスを「生成不可」状態へ遷移させます。
生成再開	要因	API 呼び出しによる遷移要求
	動作	アドホック・サブプロセスを「生成可」状態へ遷移させます。
完了	要因	• API 呼び出しによる遷移要求 •「生成不可」状態の場合に，フロー終端のフローノード（業務ステップ）が完了時にアドホック・サブプロセス内に「実行中」状態のフローノード（業務ステップ）が存在しないとき • 完了条件（completionCondition）を満たす場合に，次のどちらかのとき 　• インスタンスのキャンセル属性（cancelRemainingInstances）が`true` 　• インスタンスのキャンセル属性（cancelRemainingInstances）が`false`で，かつアドホック・サブプロセス内に実行中のフローノード（業務ステップ）が存在しない
	動作	アドホック・サブプロセスを「完了」状態へ遷移させます。また，そのアドホック・サブプロセス内のすべての「実行中」状態のフローノード（業務ステップ）を，「強制終了」状態へ遷移させます。
強制終了	要因	境界イベントによる中断要求
	動作	アドホック・サブプロセスを「強制終了」状態へ遷移させます。また，そのアドホック・サブプロセス内のすべての「実行中」状態のフローノード（業務ステップ）を，「強制終了」状態へ遷移させます。

1.5.7　アドホック・サブプロセスの状態の確認方法

アドホック・サブプロセスの状態は，アドホック・サブプロセスから変換された，状態表示用の作業の状態から確認できます。アドホック・サブプロセスの状態とアドホック・サブプロセスから変換された，状態表示用の作業の状態の対応を，次の表に示します。

表 1-16　アドホック・サブプロセスの状態と作業の状態の対応

アドホック・サブプロセスの状態	状態表示用の作業の状態 状態表示用の作業の状態の列挙型定数（コード値）
生成可	実行開始可能 READY(j)

アドホック・サブプロセスの状態	状態表示用の作業の状態 状態表示用の作業の状態の列挙型定数（コード値）
生成不可	作業者実行 PERFORMING(f)
完了	実行済 EXECUTED(r)
強制終了	強制終了 TERMINATED(u)

アドホック・サブプロセスの状態表示用の作業の作業定義名は，「＜アドホック・サブプロセスの BPMN 要素名＞_＜アドホック・サブプロセスの BPMN 要素 ID ＞」です。この作業は，次に示す Java API または REST API によって取得できます。

Java API

- `CIWServer#getWorkItemsList(String, String, int, int, Set)`
- `CIWBPMNLib#getFlowNodeInstancesListByPDName(Connection, CIWServer, String, String, String, Set, Integer, Set)`
- `CIWBPMNLib#getFlowNodeInstancesListByPIID(Connection, CIWServer, Integer, String, String, Set, Integer, Set)`

REST API

- 作業の一覧取得
- フローノードの一覧取得

1.6　プロセスデータとは

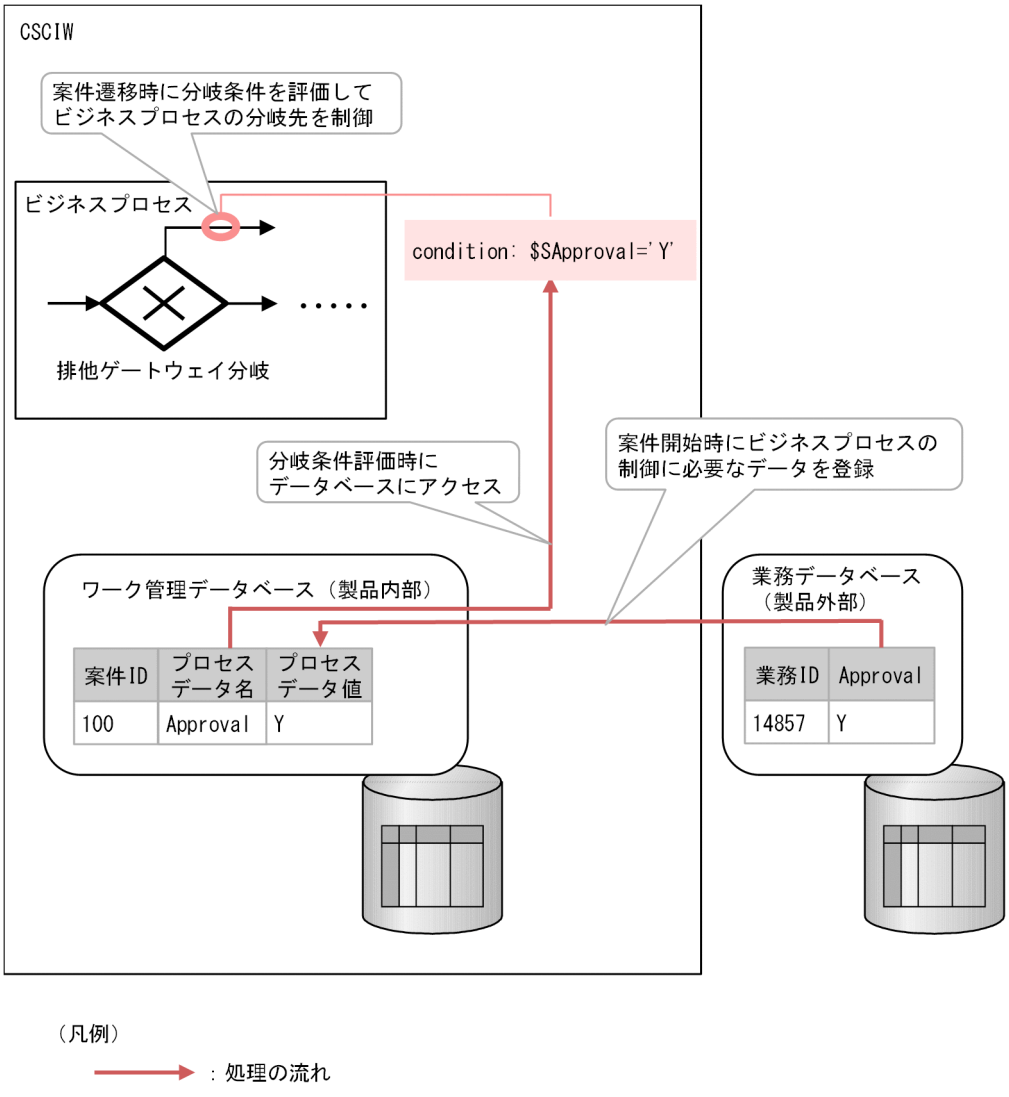
プロセスデータの概要や利用場面について説明します。

プロセスデータは，各案件に付随するデータ項目です。

プロセスデータの利用用途の一つとして，ビジネスプロセスの排他ゲートウェイの分岐条件があります。ビジネスプロセスの設計時に，分岐条件にプロセスデータを指定することで，ビジネスプロセスの分岐先の遷移を動的に制御できます。

プロセスデータは，製品内部のワーク管理データベースに保存されます。分岐条件で使うデータを製品内部に登録することで，製品外部のデータベースにアクセスしなくてもビジネスプロセスを制御できます。

図 1-77　ビジネスプロセスの分岐先の動的制御



（凡例）

：処理の流れ

1.6.1　プロセスデータの利用用途

プロセスデータの利用用途を示します。

表 1-17　プロセスデータの利用用途

利用用途	利用手段
ビジネスプロセスの動的制御	次に示す設定項目にプロセスデータを記述します。 • BPMN ビジネスプロセス定義のシーケンスフローに設定する分岐条件（condition） • BPMN ビジネスプロセス定義のマルチインスタンスに設定する繰り返し回数（loopCardinality）および完了条件（completionCondition） • BPMN ビジネスプロセス定義のタイマーイベントに設定するタイマールール • BPMN ビジネスプロセス定義のアドホック・サブプロセスに設定する完了条件（completionCondition） • BPMN ビジネスプロセス定義のユーザタスクに設定する作業者
案件に紐付くプロセスデータを登録	次に示す BPMN 連携ライブラリの Java API および REST API を利用します。 • プロセスデータを登録する API • 案件の更新系 API 次に示す設定ファイルにプロセスデータを記述します。 • アプリケーション呼び出し情報ファイル • コールアクティビティ情報ファイル
案件を起点としてプロセスデータを取得	次に示す BPMN 連携ライブラリの Java API および REST API を利用します。 • 案件 ID からプロセスデータを取得する API 次に示す設定ファイルにプロセスデータを記述します。 • アプリケーション呼び出し情報ファイル • コールアクティビティ情報ファイル
プロセスデータを起点として案件を取得	次に示す BPMN 連携ライブラリの Java API および REST API を利用します。 • プロセスデータから案件 ID を取得する API
BPMN 要素のマルチインスタンスインデクスを取得	次に示す BPMN 連携ライブラリの Java API および REST API を利用します。 • リスト型プロセスデータのインデクスを取得する API

1.6.2　プロセスデータの基本構成

プロセスデータには，単一型プロセスデータとリスト型プロセスデータの 2 種類があります。次の図に示すように，単一型プロセスデータは，1 つのプロセスデータキー名に対して，1 つの値を持ちます。リスト型プロセスデータは，1 つのプロセスデータキー名に対して，複数の値を持ちます。

プロセスデータは，各案件に付随するデータ項目です。そのため，同じプロセスデータキー名でも，案件ごとに別のプロセスデータとして扱われ，それぞれ別の値を持ちます。

特定のプロセスデータ値にアクセスするには，案件を特定する方法（例：案件 ID）とプロセスデータキー名が必要です。

図 1-78　単一型プロセスデータの例

案件ID	プロセスデータキー名	プロセスデータ値
100	$SCustomerName	AAA
100	$NCustomerID	20387484
101	$SCustomerName	BBB
101	$NCustomerID	10049607

図 1-79　リスト型プロセスデータの例

案件ID	プロセスデータキー名	プロセスデータ値
100	$SProductNameList{}	{"Apple","Orange","Nuts"}
2000	$SCustomerNameList{}	{"AAA","BBB"}
2000	$NCustomerIDList{}	{20387484,20049607}

リスト型プロセスデータを使用すると，同じ目的で使用する複数のプロセスデータを一括で登録したり検索したりできます。これによって，1 つのプロセスデータキー名で複数の値をまとめて管理できます。

また，マルチインスタンスでは，リスト型プロセスデータ内の値の数に応じたインスタンスを生成したり，リスト型プロセスデータ内の各値を受け渡したりできます。詳細については，「1.6.4　プロセスデータの利用方法」のマルチインスタンスでのインスタンス生成時および作業実行時の説明を参照してください。

1.6.3　プロセスデータテーブルの内容

プロセスデータで扱えるプロセスデータ値は，文字列型，数値型およびリスト型の 3 種類です。各プロセスデータの型（リスト型の場合はリスト内の要素の型）に対応するプロセスデータテーブルを定義します。

プロセスデータテーブルの概要を表に示します。

表 1-18　プロセスデータテーブルの概要

名前	列名	説明
案件 ID	ProcessInstanceID	プロセスデータが所属する案件の ID
プロセスデータ名	ProcessDataName	プロセスデータキー名から先頭の"$"と種別を表す文字を除いた名称
プロセスデータ値	ProcessDataValue	プロセスデータの値

プロセスデータの値によって，内部では異なるテーブルが使用されます。ユーザがプロセスデータを利用する場合は，プロセスデータキー名を指定します。そのため，テーブルの違いを意識する必要はありません。

プロセスデータキー名の命名規則，java 変数の型，およびデータベースの型の対応関係を表に示します。

表 1-19　プロセスデータキー名の命名規則とプロセスデータ型の対応関係

プロセスデータキー名の命名規則※	java 変数の型	プロセスデータデータベースの型
• "$N"から始まる	Integer	整数型

プロセスデータキー名の命名規則※	java 変数の型	プロセスデータデータベースの型
• "$N"から始まり，末尾が"{<リスト内識別子>}"で終わる	Integer	整数型
"$N"から始まり，末尾が"{}"で終わる	java.util.List<Integer>	
• "$S"から始まる • "$S"から始まり，末尾が"{<リスト内識別子>}"で終わる	String	可変長文字列型
"$S"から始まり，末尾が"{}"で終わる	java.util.List<String>	

注※

この規則で示す文字列以外に使用できる文字列は，英数字だけです。ただし，"IW"から始まる文字列は使用できません。また，英字は大文字と小文字が区別されます。

なお，プロセスデータテーブルおよびインデクス定義の詳細については，「付録 A.1　テーブル定義」を参照してください。

1.6.4　プロセスデータの利用方法

ここではプロセスデータの指定方法および記述形式と，単一型プロセスデータの場合とリスト型プロセスデータの場合のアクセス方法を説明します。

プロセスデータの指定方法および記述形式

案件を特定する情報とプロセスデータキー名を次のどれかに指定すると，プロセスデータ値にアクセスできます。

- BPMN ビジネスプロセス定義のシーケンスフローに設定する分岐条件（condition）
- BPMN ビジネスプロセス定義のマルチインスタンスに設定する繰り返し回数（loopCardinality）および完了条件（completionCondition）
- BPMN ビジネスプロセス定義のタイマーイベントに設定するタイマールール
- BPMN ビジネスプロセス定義のアドホック・サブプロセスに設定する完了条件（completionCondition）
- BPMN ビジネスプロセス定義のユーザタスクに設定する作業者
- アプリケーション呼び出し情報ファイル
- コールアクティビティ情報ファイル
- REST API のリクエストボディ
- BPMN 連携ライブラリの Java API の引数

案件を特定する情報の指定方法は，機能によって異なります。また，機能によっては，対象の案件が自明な場合は，案件を特定する情報を省略できるときがあります。

単一型プロセスデータへの参照アクセス

単一型プロセスデータにアクセスするためのプロセスデータキー名に使用できる文字列を次に示します。

- プロセスデータ値が文字列型の場合：

  $S<ProcessDataName>

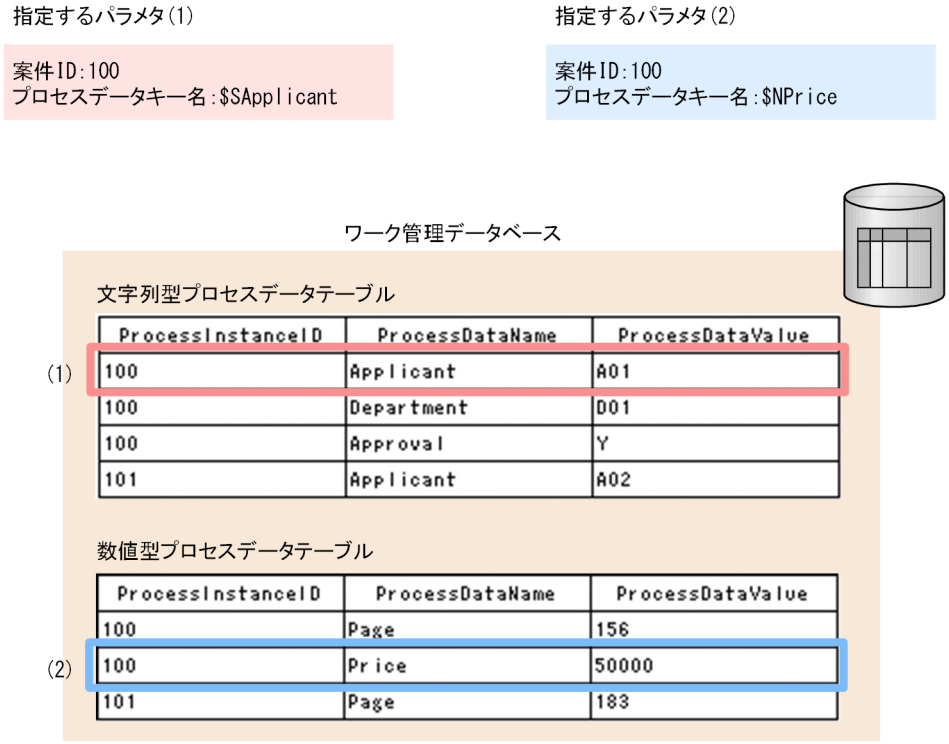
- プロセスデータ値が数値型の場合：

  $N<ProcessDataName>

<ProcessDataName>には，英大文字，英小文字，または数字だけから構成される文字列を指定してください。ただし，"IW"から始まる文字列は指定しないでください。

案件 ID が"100"の案件のプロセスデータにアクセスする例を次の図に示します。

図 1-80　単一型プロセスデータへの参照アクセスの例



[説明]

■パラメタ(1)を指定してアクセスした場合

案件 ID に"100"，プロセスデータキー名に"$SApplicant"を指定することで，文字列型プロセスデータテーブルの行（ProcessInstanceID="100"，ProcessDataName="Applicant"）が特定されます。対応するプロセスデータ値（ProcessDataValue）は"A01"です。

■パラメタ(2)を指定してアクセスした場合

案件 ID に"100"，プロセスデータキー名に"$NPrice"を指定することで，数値型プロセスデータテーブルの行（ProcessInstanceID="100"，ProcessDataName="Price"）が特定されます。対応するプロセスデータ値（ProcessDataValue）は"50000"です。

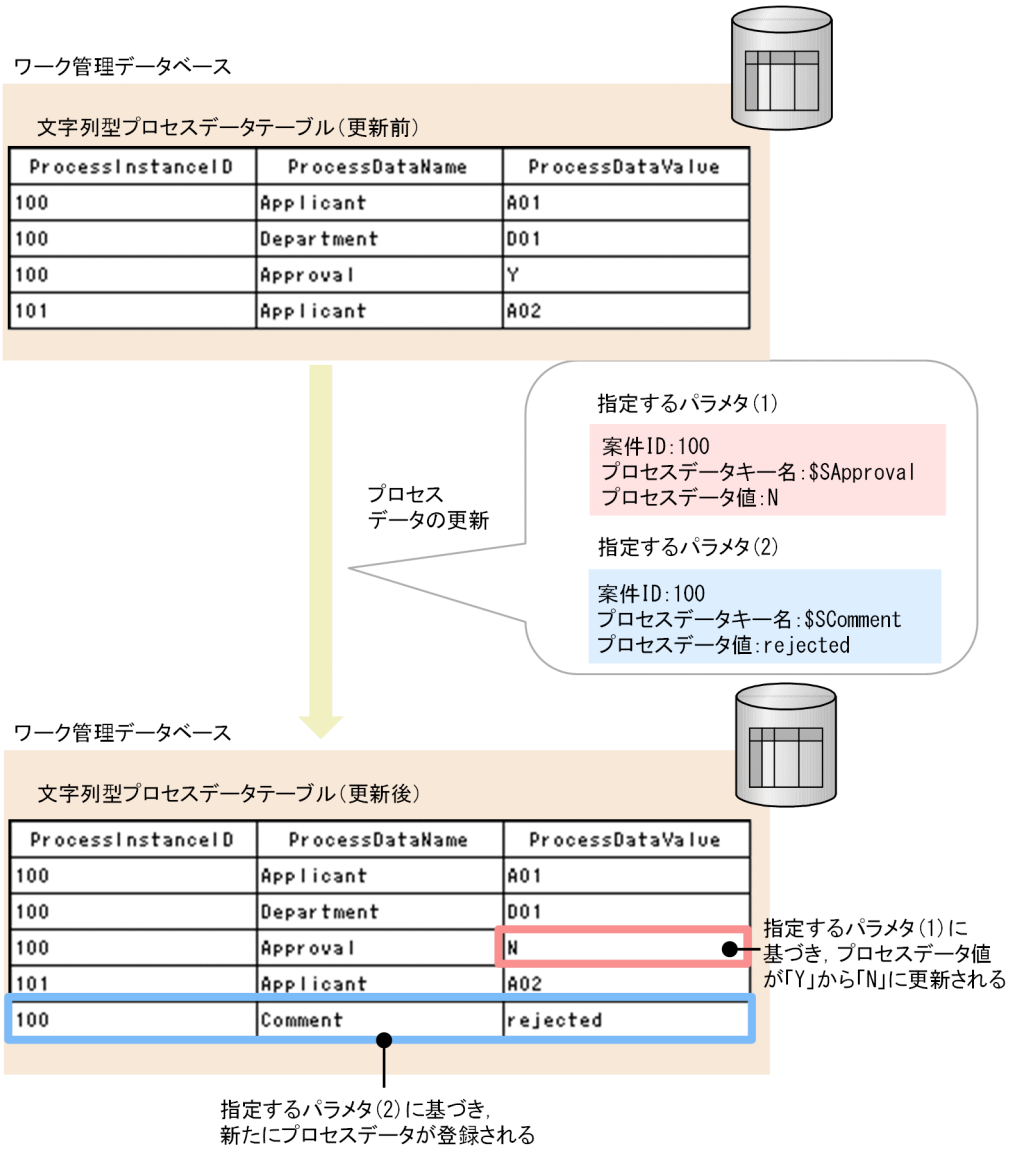
単一型プロセスデータ値の更新処理の例

プロセスデータを登録する際に，案件 ID およびプロセスデータキー名が同一のプロセスデータがすでに存在していた場合は，値を更新します（既存のプロセスデータ値を上書きします）。

単一型プロセスデータにアクセスするためのプロセスデータキー名に使用できる文字列については「単一型プロセスデータへの参照アクセス」を参照してください。

案件 ID が"100"の案件のリスト型プロセスデータを更新する例を次の図に示します。

図 1-81　単一型プロセスデータ値の更新処理の例



[説明]

■パラメタ(1)を指定してアクセスした場合

案件 ID（ProcessInstanceID）が"100"で，文字列型プロセスデータ名（ProcessDataValue）が"Approval"のプロセスデータが存在するため，プロセスデータ値"Y"を"N"に更新します。

■パラメタ(2)を指定してアクセスした場合

案件 ID が"100"で，文字列型プロセスデータ名が"Comment"のプロセスデータが存在しないため，プロセスデータ値が"rejected"のプロセスデータを登録します。

リスト型プロセスデータの全要素への参照アクセス

リスト型プロセスデータにアクセスするためのプロセスデータキー名に使用できる文字列を次に示します。

- プロセスデータ値が文字列型の場合：

  $S<ProcessDataName>{}

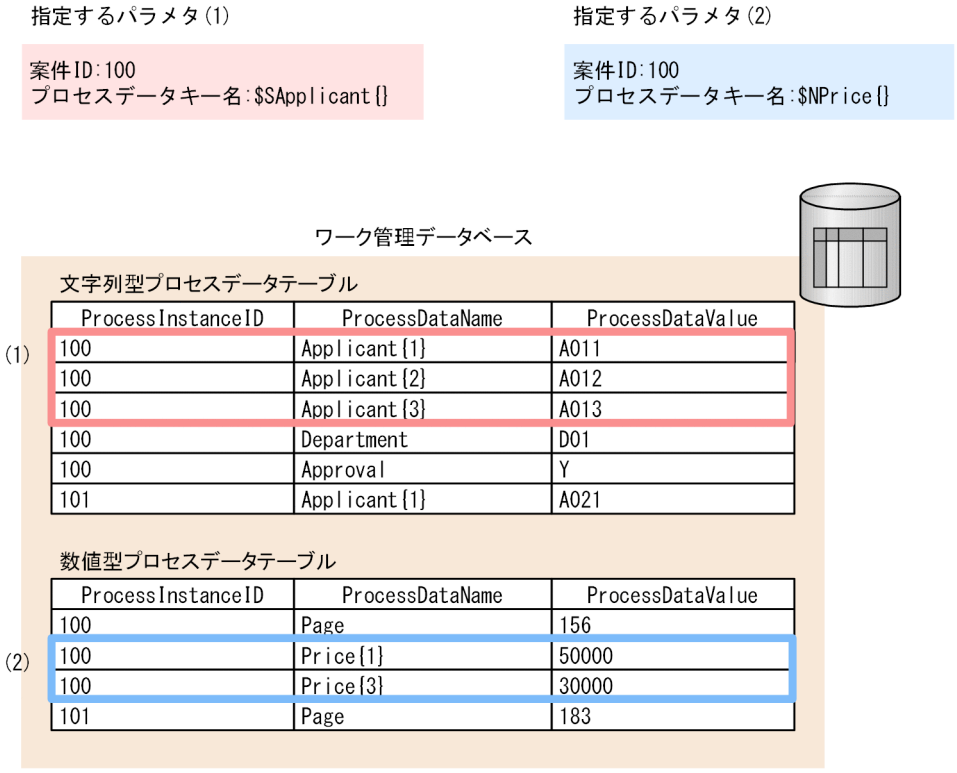
- プロセスデータ値が数値型の場合：

  $N<ProcessDataName>{}

<ProcessDataName>には，英大文字，英小文字，または数字だけから構成される文字列を指定してください。ただし，"IW"から始まる文字列は指定しないでください。

案件 ID が"100"の案件のプロセスデータにアクセスする例を次の図に示します。

図 1-82　リスト型プロセスデータの全要素への参照アクセスの例



[説明]

■パラメタ(1)を指定してアクセスした場合

案件 ID に"100"，プロセスデータキー名に"$SApplicant{}"を指定することで，文字列型プロセスデータテーブルの 3 行が特定されます。対応するプロセスデータ値（ProcessDataValue）は，"A011"，"A012"および"A013"です。

なお，ワーク管理データベース内に保存されたリスト型プロセスデータは，ProcessDataName の末尾に"{<番号>}"が付与されています。リスト型プロセスデータの全要素に参照アクセスする際は，番号（リスト内識別子）を意識する必要はありません。

■パラメタ(2)を指定してアクセスした場合

案件 ID に"100"，プロセスデータキー名に"$NPrice{}"を指定することで，数値型プロセスデータテーブルの 2 行が特定されます。対応するプロセスデータ値（ProcessDataValue）は，"50000"および"30000"です。

なお，プロセスデータ名のリスト内識別子が連番ではない場合，存在するプロセスデータ値だけを取得します。

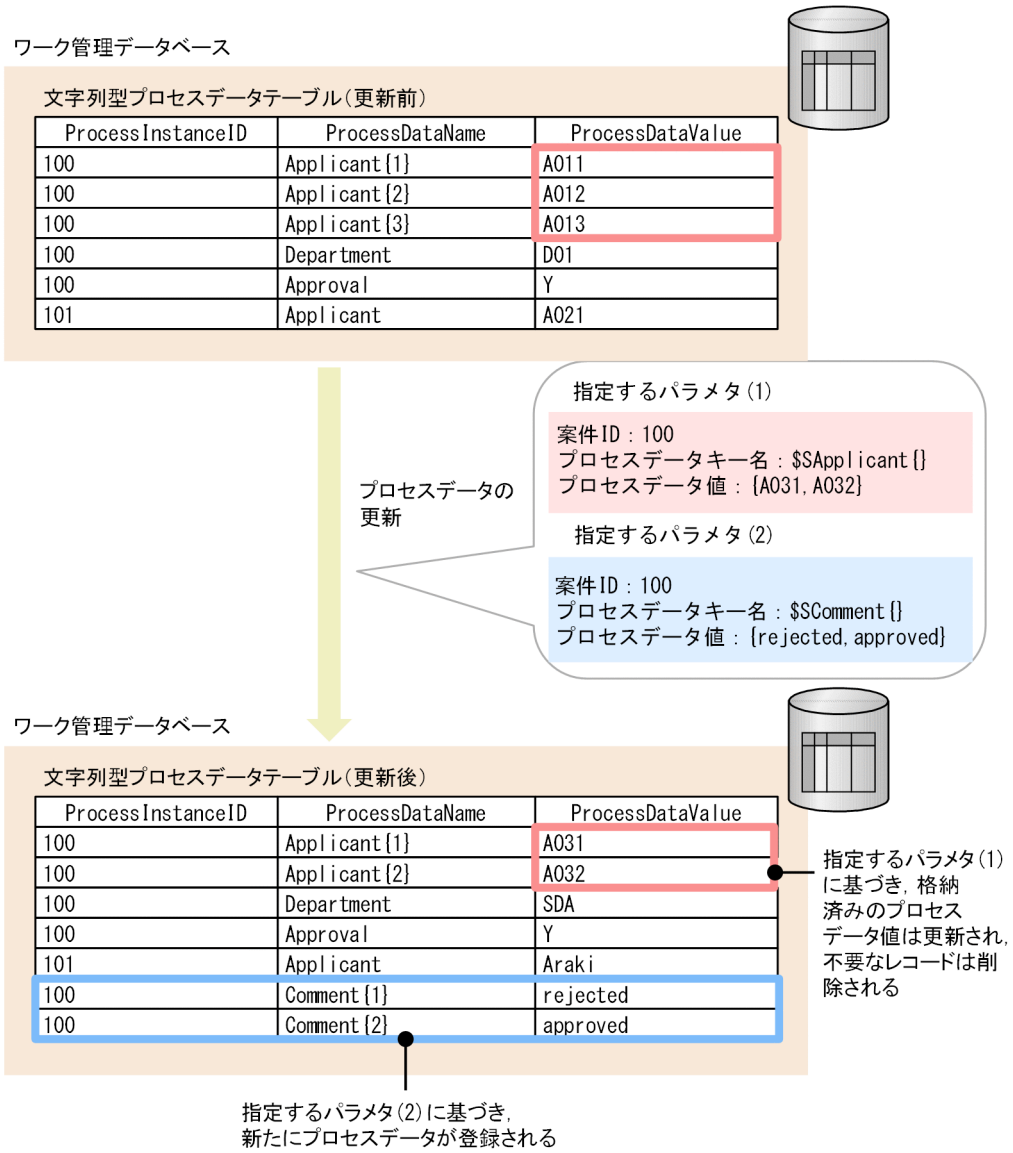
リスト型プロセスデータ値の全要素の更新処理の例

プロセスデータを登録する際に，案件 ID およびプロセスデータキー名が同一のプロセスデータがすでに存在していた場合は，値を更新します（既存のプロセスデータ値を上書きします）。

リスト型プロセスデータにアクセスするためのプロセスデータキー名に使用できる文字列については「リスト型プロセスデータの全要素への参照アクセス」を参照してください。

案件 ID が"100"の案件のリスト型プロセスデータを更新する例を次の図に示します。

図 1-83　リスト型プロセスデータ値の全要素の更新処理の例



[説明]

■パラメタ(1)を指定してアクセスした場合

案件 ID に"100"，プロセスデータキー名に"$SApplicant{}"を指定することで，文字列型プロセスデータテーブルの 3 行が特定されます。更新するリスト型プロセスデータの要素数は 2 のため，すでに存在する行（ProcessDataName が"Applicant{1}"および"Applicant{2}"）は更新され，不要になった行（ProcessDataName が"Applicant{3}"）は削除されます。

■パラメタ(2)を指定してアクセスした場合

案件 ID が"100"，プロセスデータキー名が"$SComment{}"に対応するリスト型プロセスデータが存在しないため，プロセスデータ値が"rejected"，および"approved"のプロセスデータが登録されます。

なお，ProcessDataName の"{}"内には，追加対象リストの先頭のデータから順番に，各プロセスデータ値を識別するための番号（リスト内識別子）が自動的に付与されます。そのため，この例の ProcessDataName は"Comment{1}"，および"Comment{2}"となります。

リスト型プロセスデータの 1 要素への参照アクセス

リスト型プロセスデータの 1 要素を特定する文字列をプロセスデータキー名に付与することで，単一型プロセスデータへのアクセスと同様の方法で，リスト型プロセスデータの 1 要素へアクセスできます。

リスト型プロセスデータの 1 要素にアクセスするためのプロセスデータキー名に使用できる文字列を次に示します。

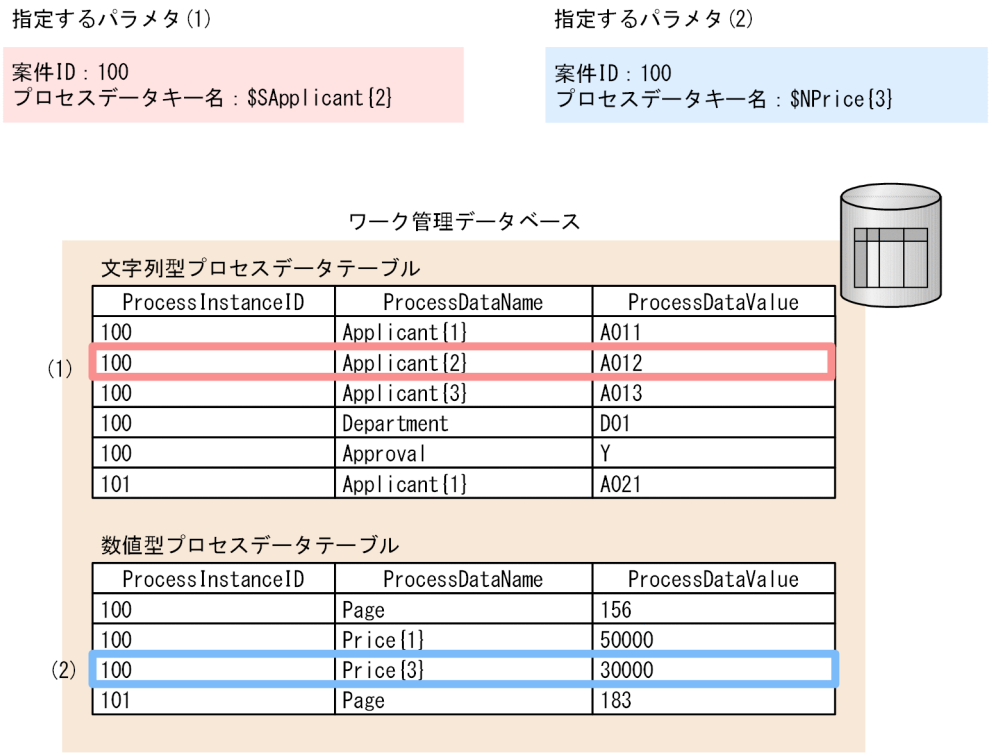
- プロセスデータ値が文字列型の場合：
  $S<ProcessDataName>{<リスト内識別子>}
- プロセスデータ値が数値型の場合：
  $N<ProcessDataName>{<リスト内識別子>}

<ProcessDataName>には，英大文字，英小文字，または数字だけから構成される文字列を指定してください。ただし，"IW"から始まる文字列は指定しないでください。

リスト内識別子の範囲は 1 から 2,147,483,647 の整数です。

案件 ID が"100"の案件のリスト型プロセスデータの 1 要素にアクセスする例を次の図に示します。

図 1-84　リスト型プロセスデータの 1 要素への参照アクセスの例



[説明]

■パラメタ(1)を指定してアクセスした場合

案件 ID に"100"，プロセスデータキー名に"$SApplicant{2}"を指定することで，文字列型プロセスデータテーブルの行（ProcessInstanceID="100"，ProcessDataName="Applicant{2}"）が特定されます。対応するプロセスデータ値（ProcessDataValue）は"A012"です。

■パラメタ(2)を指定してアクセスした場合

案件 ID に"100"，プロセスデータキー名に"$NPrice{3}"を指定することで，数値型プロセスデータテーブルの行（ProcessInstanceID="100"，ProcessDataName="Price{3}"）が特定されます。対応するプロセスデータ値（ProcessDataValue）は"30000"です。

リスト型プロセスデータの 1 要素の更新処理の例

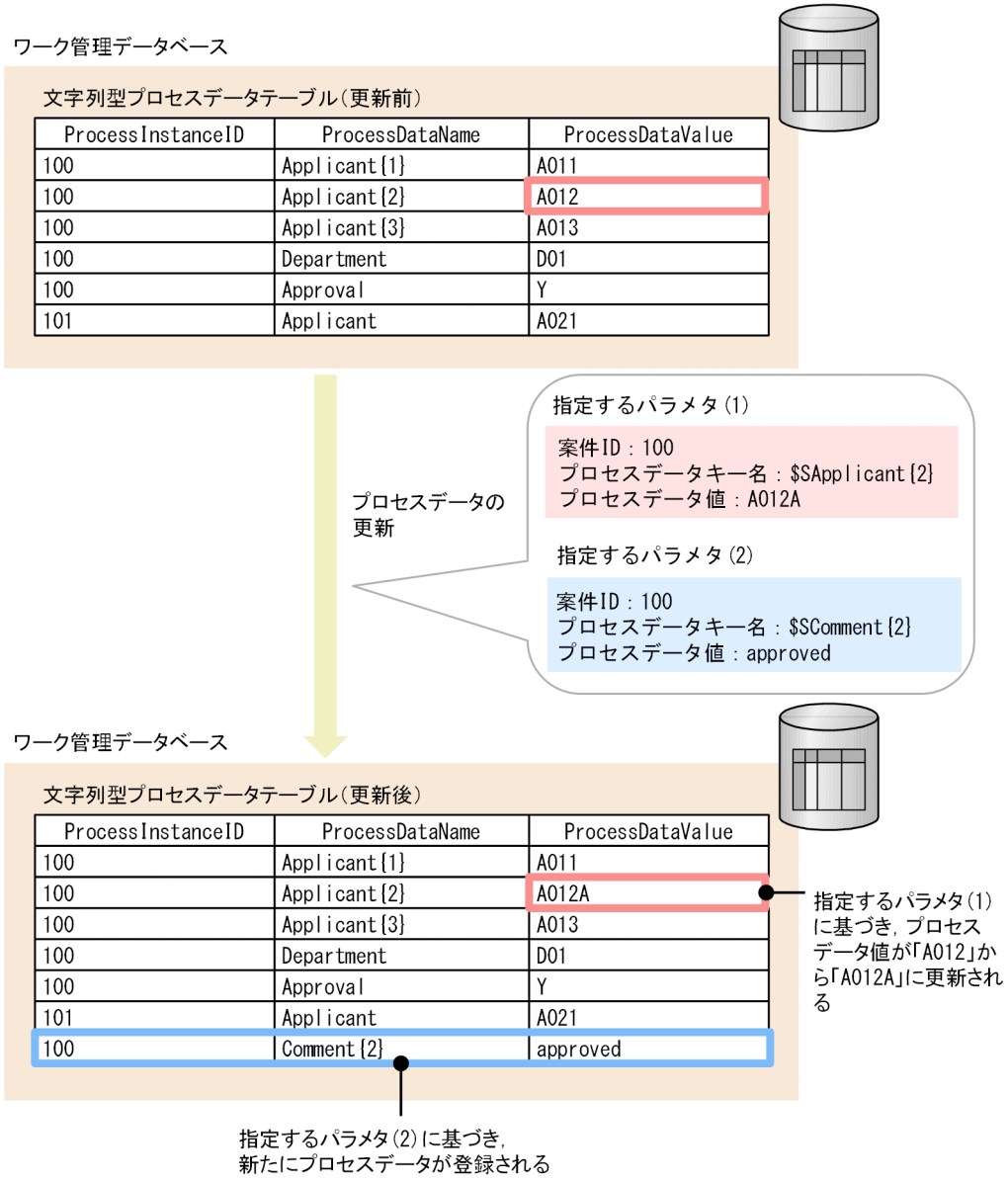
リスト型プロセスデータの 1 要素を特定する文字列をプロセスデータキー名に付与することで，単一型プロセスデータへのアクセスと同様の方法で，リスト型プロセスデータの 1 要素にアクセスできます。

リスト型プロセスデータの 1 要素にアクセスするためのプロセスデータキー名に使用できる文字列については「リスト型プロセスデータの 1 要素への参照アクセス」を参照してください。

リスト内識別子の範囲は 1 から 2,147,483,647 の整数です。

案件 ID が"100"の案件のリスト型プロセスデータの 1 要素を更新する例を次の図に示します。

図 1-85　リスト型プロセスデータの 1 要素の更新処理の例



[説明]

■パラメタ(1)を指定してアクセスした場合

案件 ID（ProcessInstanceID）が"100"で，文字列型プロセスデータ名（ProcessDataName）が"Applicant{2}"のプロセスデータが存在するため，プロセスデータ値"A012"を"A012A"に更新します。

■パラメタ(2)を指定してアクセスした場合

案件 ID が"100"で，文字列型プロセスデータ名が"Comment{2}"のプロセスデータが存在しないため，プロセスデータ値が"approved"のプロセスデータを登録します。

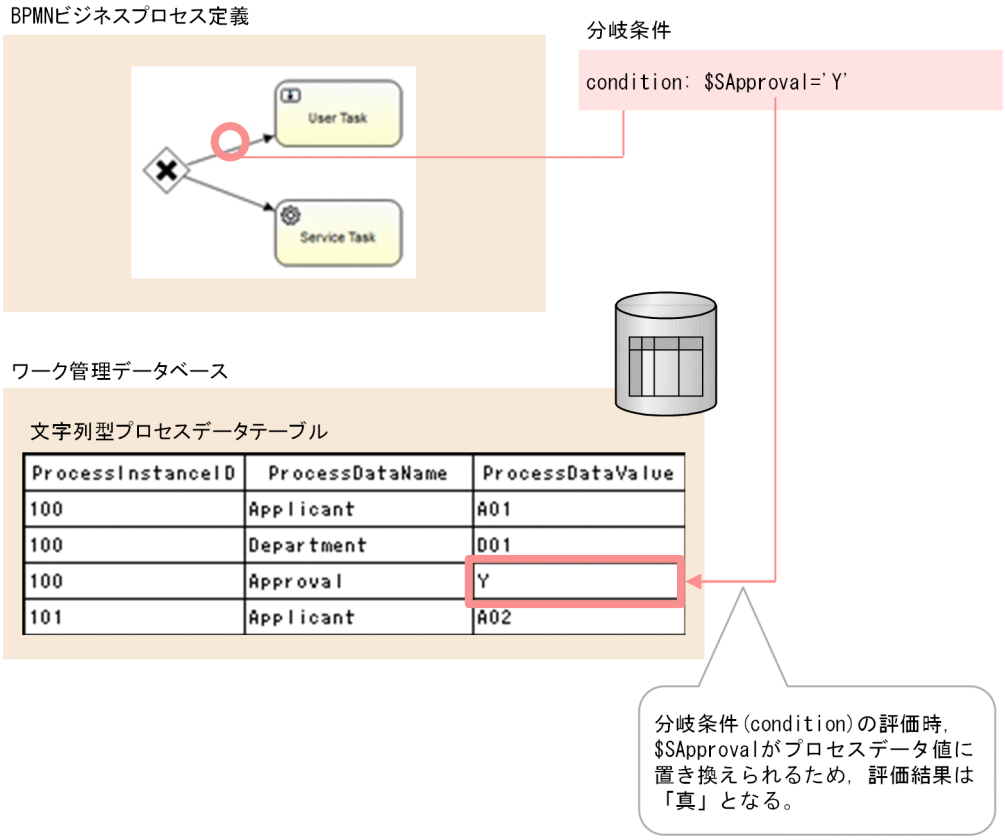
BPMN ビジネスプロセス定義の分岐条件でのプロセスデータの使用例

[説明]

1. BPMN ビジネスプロセス定義で，シーケンスフローに分岐条件の評価式（condition）を定義します。

評価式には，XPath 式を記述してください。次の図では，プロセスデータキー名が"$SApproval"の値を取得しています。XPath の評価時にプロセスデータ値に置き換えられるため，XPath 式"'Y'='Y'"として評価され，評価結果は「真」となります。

図 1-86　BPMN ビジネスプロセス定義の分岐条件でのプロセスデータの使用例



重要

変数と定数だけを使用していて，boolean 値を返す比較演算子を使用している評価式だけ記述できます。

分岐条件に記述できる XPath 式に関する規則は，W3C で規定されている XML Path Language (XPath) Version 1.0 に準拠します。また，XPath の仕様で規定されている XPath 関数に加えて，XPath 拡張関数を記述できます。XPath 拡張関数の仕様については，「14.　XPath 拡張関数リファレンス 」を参照してください。

サブプロセス（マルチインスタンス）内に定義された分岐条件では，リスト型プロセスデータの 1 要素を参照および更新するように記述できます。サブプロセス（マルチインスタンス）内に定義された分岐条件でのプロセスデータの使用方法については，「マルチインスタンスでの作業実行時のプロセスデータ使用例（サブプロセスの場合）」を参照してください。

主な規則は次のとおりです。

- ロケーションパスは使用できません。
- CSCIW の組み込み変数は使用できません。

例えば，"$SApproval=CSCIWPICreato"や"$SApproval=@PICreator"のように記述しても，CSCIW で保持する値には置換されません。

- 条件式は，<SYSTEMID>_CONDITION_JAVA_DEF テーブルのExParameter カラムで定義したバイト数（デフォルト 2,000 バイト）以内で指定してください。
- 命名規則に従わないプロセスデータキー名は使用できません。なお，リスト型プロセスデータの全要素を示すプロセスデータキー名は変数として使用できません。

ヒント

記述できる Xpath 式の例

- concat($SWCO,”/”,$SCSCIW,”/”,$SESB) = ’WCO/CSCIW/ESB’
  $SWCO='WCO'，$SCSCIW='CSCIW'，$ESB='ESB'の場合，結果は「真」になります。
- substring-before($SYMD, ”/”) = ’2016’
  $SYMD='2016/12/31'の場合，結果は「真」になります。
- $NTANKA * $NKAZU{3} > 2000000
  $NTANKA=1000000 かつ$NKAZU{3}=3 の場合，「真」になります。
- $NCustomerIDList{MIIndex}=1000000
  分岐条件のマルチインスタンスインデクス=3，かつ$NCustomerIDList{3}=1000000 の場合，結果は「真」になります。

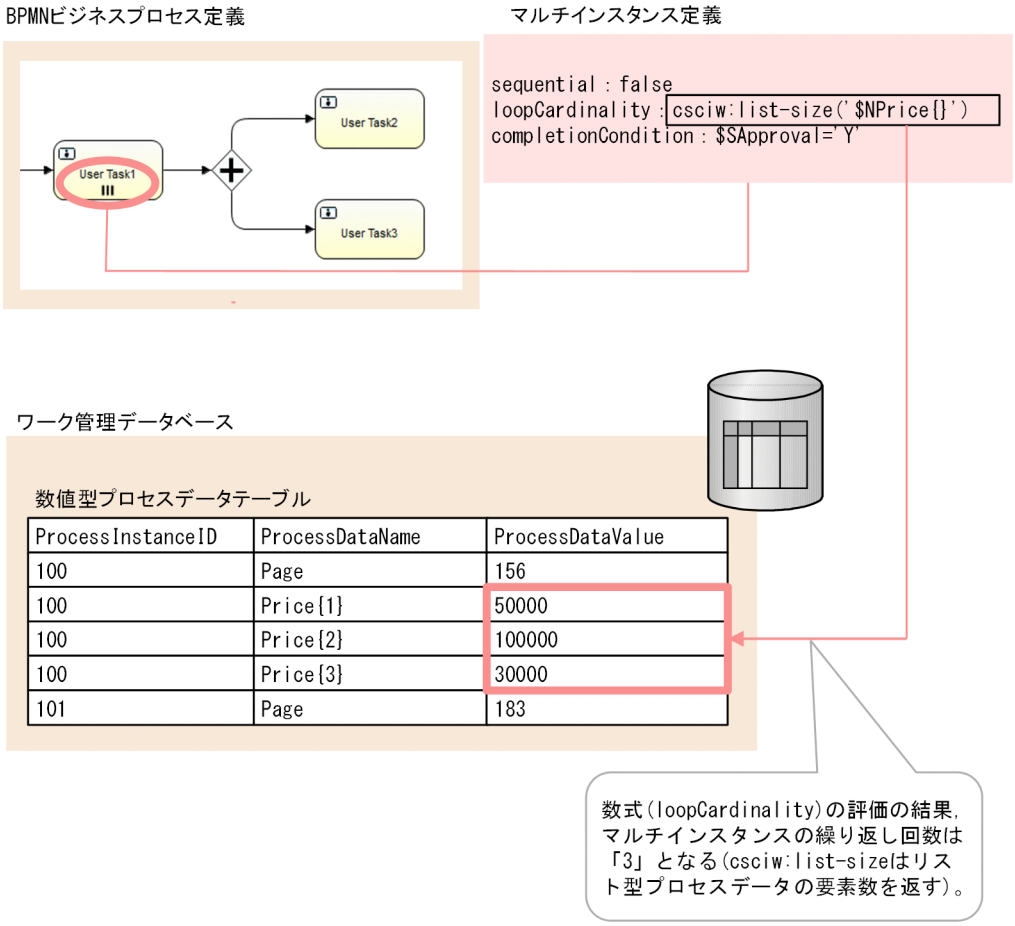
記述できない Xpath 式の例

- $SAcount$SApproval=’hitachiYes’
  プロセスデータキー名を連結する記述はできません。
- ’123’$SApproval=’123Y’および123$SApproval=’123Y’
  文字列と連結したプロセスデータキー名は使用できません。
- $SApproval{}=’123Y’
  リスト型プロセスデータの全要素を示すプロセスデータキー名は記述できません。この場合，リスト型と文字列型の比較になるため，結果は「偽」になります。

マルチインスタンス定義でのプロセスデータ使用例

マルチインスタンス定義でのプロセスデータの使用例を次の図で説明します。

図 1-87　マルチインスタンス定義でのプロセスデータ使用例



［説明］

BPMN ビジネスプロセス定義のマルチインスタンス定義では，図の上方で示すように，実行種別（sequential），繰り返し回数を示す数式（loopCardinality）および完了条件の評価式（completionCondition）の 3 つの属性を定義します。この中で，繰り返し回数を示す数式（loopCardinality）および完了条件の評価式（completionCondition）にプロセスデータキー名を使用できます。

数式（loopCardinality）と評価式（completionCondition）には XPath 式を指定します。この例では，プロセスデータキー名（$NPrice{}）に対応するプロセスデータの要素数をデータベースから取得するため，数式（loopCardinality）の評価結果は'3'となります。

重要

数式（loopCardinality）に XPath 拡張関数のcsciw:list-size を定義する場合，リスト内識別子に欠番があるリスト型プロセスデータは引数として指定できません。

数式（loopCardinality）と評価式（completionCondition）には，XPath 式のすべての文法を記述できるわけではありません（例えば，ロケーションパスは記述できません）。数式（loopCardinality）には，変数と定数だけを使用していて，数値を返す演算子による数式だけを記述できます。

評価式（completionCondition）には，変数と定数だけを使用していて，boolean 値を返す比較演算子を使用している評価式だけを記述できます。記述の仕様は分岐条件に記述する評価式と同じです。詳細については「BPMN ビジネスプロセス定義の分岐条件でのプロセスデータの使用例」を参照してください。

ヒント

記述できる数式（loopCardinality）の例と，注意事項を示します。

- `$NTANKA * 2`

  `$NTANKA=1000000` の場合，結果は「2000000」になります。

- `csciw:list-size('$SCOUNT{}')`

  `$SCOUNT{}`が示すリスト型プロセスデータ内の要素の数が「8」の場合，結果は「8」になります。

注意

評価結果が boolean 値の場合，例外が発生します。

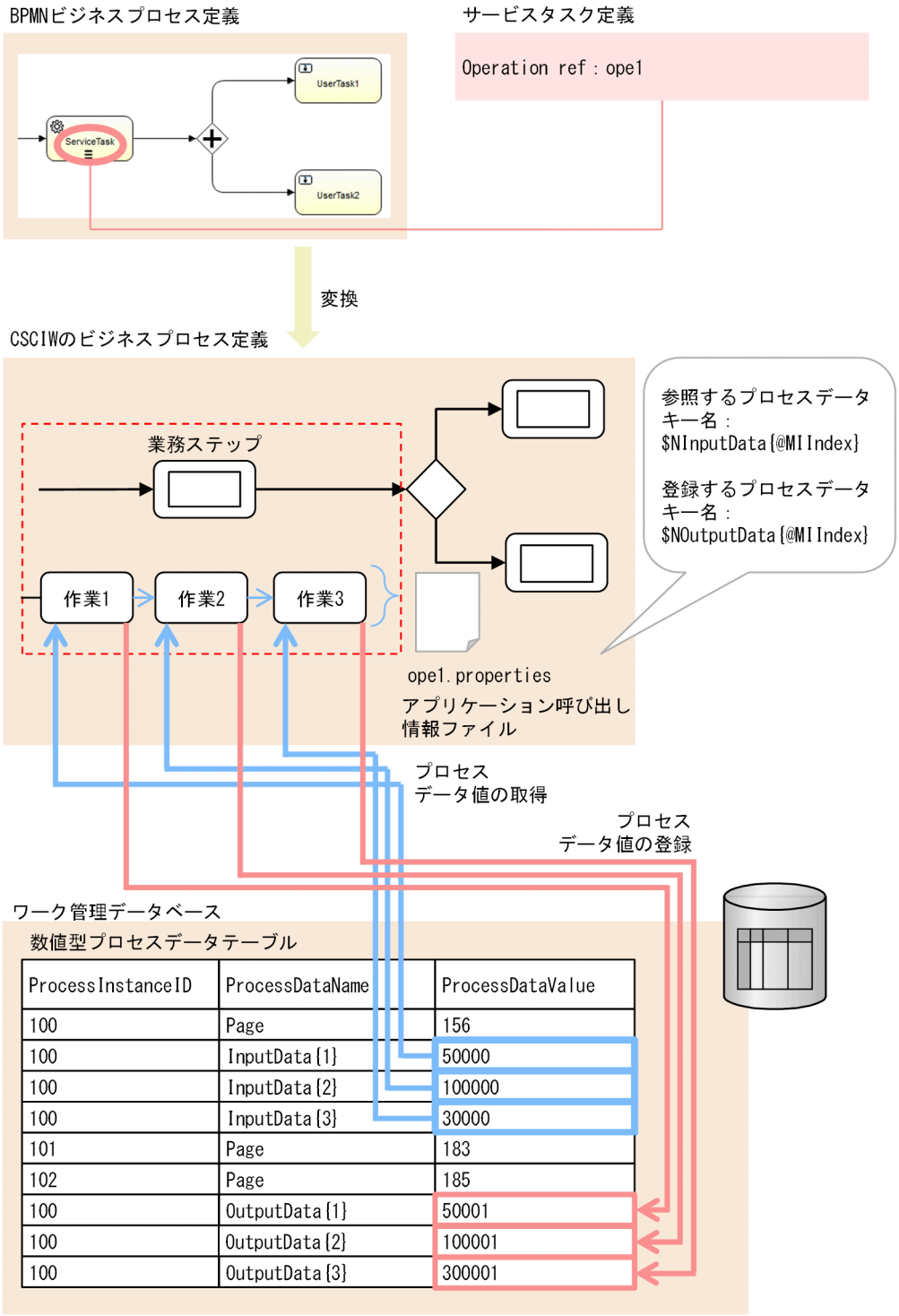
評価結果が int 値の範囲外の場合，int 値に丸めた値となります。なお，小数点以下を含む値の場合，小数点以下を切り捨てます。

マルチインスタンスでの作業実行時のプロセスデータ使用例（サービスタスク / ビジネスルールタスク / コールアクティビティの場合）

マルチインスタンスが定義されたサービスタスク，ビジネスルールタスク，またはコールアクティビティから生成された作業では，あらかじめ登録されたリスト型プロセスデータの 1 要素を取得し，新たにリスト型プロセスデータに 1 要素を登録できます。マルチインスタンスが定義されたサービスタスク，ビジネスルールタスク，またはコールアクティビティでのリスト型プロセスデータの各要素の受け渡しについては，アプリケーション呼び出し情報ファイルまたはコールアクティビティ情報ファイルに定義します。

シーケンシャルマルチインスタンスが定義されたサービスタスク，ビジネスルールタスク，またはコールアクティビティでの作業実行時の，プロセスデータの使用例を説明します。

図 1-88　シーケンシャルマルチインスタンスでの作業実行時のプロセスデータの使用例



ProcessInstanceID	ProcessDataName	ProcessDataValue
100	Page	156
100	InputData{1}	50000
100	InputData{2}	100000
100	InputData{3}	30000
101	Page	183
102	Page	185
100	OutputData{1}	50001
100	OutputData{2}	100001
100	OutputData{3}	300001

[説明]

マルチインスタンスが定義されたサービスタスクによって，CSCIW の作業が逐次に生成され，それぞれの作業にマルチインスタンスインデクスが付与されます。また，シーケンシャルマルチインスタンスの場合，生成した作業の順にマルチインスタンスインデクスが 1 から割り当てられます。リスト型プロセスデータの 1 要素を参照および更新するようにアプリケーション呼び出し情報ファイルに記述することで，作業ごとに異なるプロセスデータ要素にアクセスします。この例では，作業 1（マルチインスタンスインデクス=1）はInputData{1}を取得してOutputData{1}を登録します。作業 2（マルチインスタ

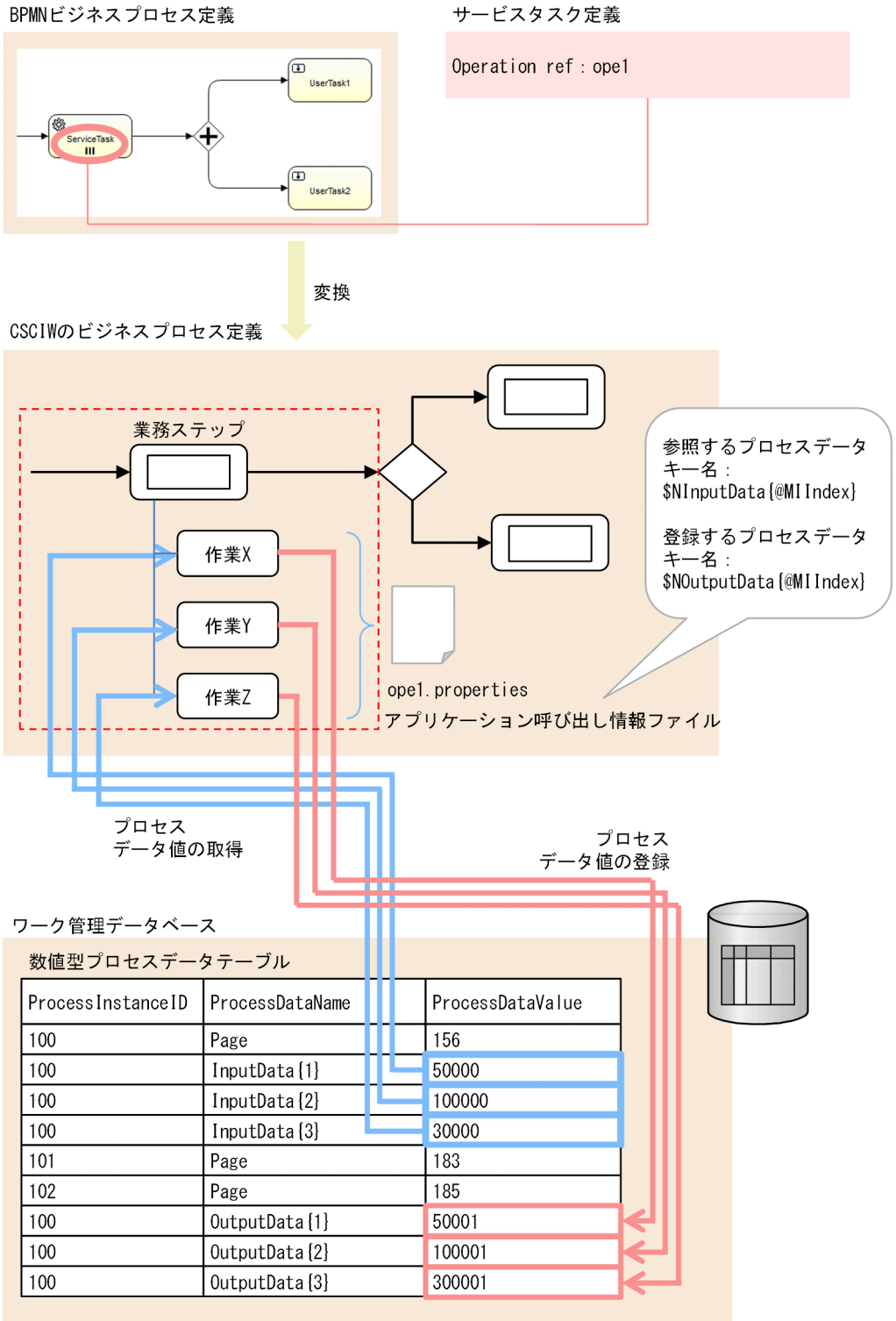
ンスインデクス=2）はInputData{2}を取得してOutputData{2}を登録します。作業３（マルチインスタンスインデクス=3）はInputData{3}を取得してOutputData{3}を登録します。{}内の番号をリスト内識別子と呼びます。

このように，それぞれの作業は，割り当てられたマルチインスタンスインデクスに対応したリスト型プロセスデータのリスト内識別子を参照します。それぞれの作業が参照するリスト型プロセスデータのリスト内識別子と，作業が登録するリスト型プロセスデータのリスト内識別子は同一になります。

なお，アプリケーション呼び出し情報ファイルへのプロセスデータキー名の記述方法については，「15.3.14　REST アプリケーションの呼び出しの場合にファイルに指定する内容」を参照してください。

次に，パラレルマルチインスタンスが定義されたサービスタスク，ビジネスルールタスク，またはコールアクティビティでの作業実行時の，プロセスデータの使用例を説明します。

図 1-89　パラレルマルチインスタンスでの作業実行時のプロセスデータの使用例



ProcessInstanceID	ProcessDataName	ProcessDataValue
100	Page	156
100	InputData{1}	50000
100	InputData{2}	100000
100	InputData{3}	30000
101	Page	183
102	Page	185
100	OutputData{1}	50001
100	OutputData{2}	100001
100	OutputData{3}	300001

[説明]

マルチインスタンスが定義されたサービスタスクによって，CSCIW の作業が並列に生成され，それぞれの作業にマルチインスタンスインデクスが付与されます。また，パラレルマルチインスタンスの場合，生成した作業の順にマルチインスタンスインデクスが 1 から割り当てられます。ただし，作業が同時に生成されるため，各作業にどのリスト内識別子が割り当てられるかは不定です。リスト型プロセスデータの 1 要素を参照および更新するようにアプリケーション呼び出し情報ファイルに記述すること

で，作業ごとに異なるプロセスデータ要素にアクセスします。この例では，作業 X（マルチインスタンスインデクス=1）はInputData{1}を取得してOutputData{1}を登録します。作業 Y（マルチインスタンスインデクス=2）はInputData{2}を取得してOutputData{2}を登録します。作業 Z（マルチインスタンスインデクス=3）はInputData{3}を取得してOutputData{3}を登録します。{}内の番号をリスト内識別子と呼びます。

このように，それぞれの作業は，割り当てられたマルチインスタンスインデクスに対応したリスト型プロセスデータのリスト内識別子を参照します。それぞれの作業が参照するリスト型プロセスデータのリスト内識別子と，作業が登録するリスト型プロセスデータのリスト内識別子は同一になります。

ヒント

プロセスデータテーブルの詳細については，「付録 A.1　テーブル定義」の「(3) <SYSTEMID>_PROCESS_DATA_S の内容」および「(4) <SYSTEMID>_PROCESS_DATA_N の内容」を参照してください。

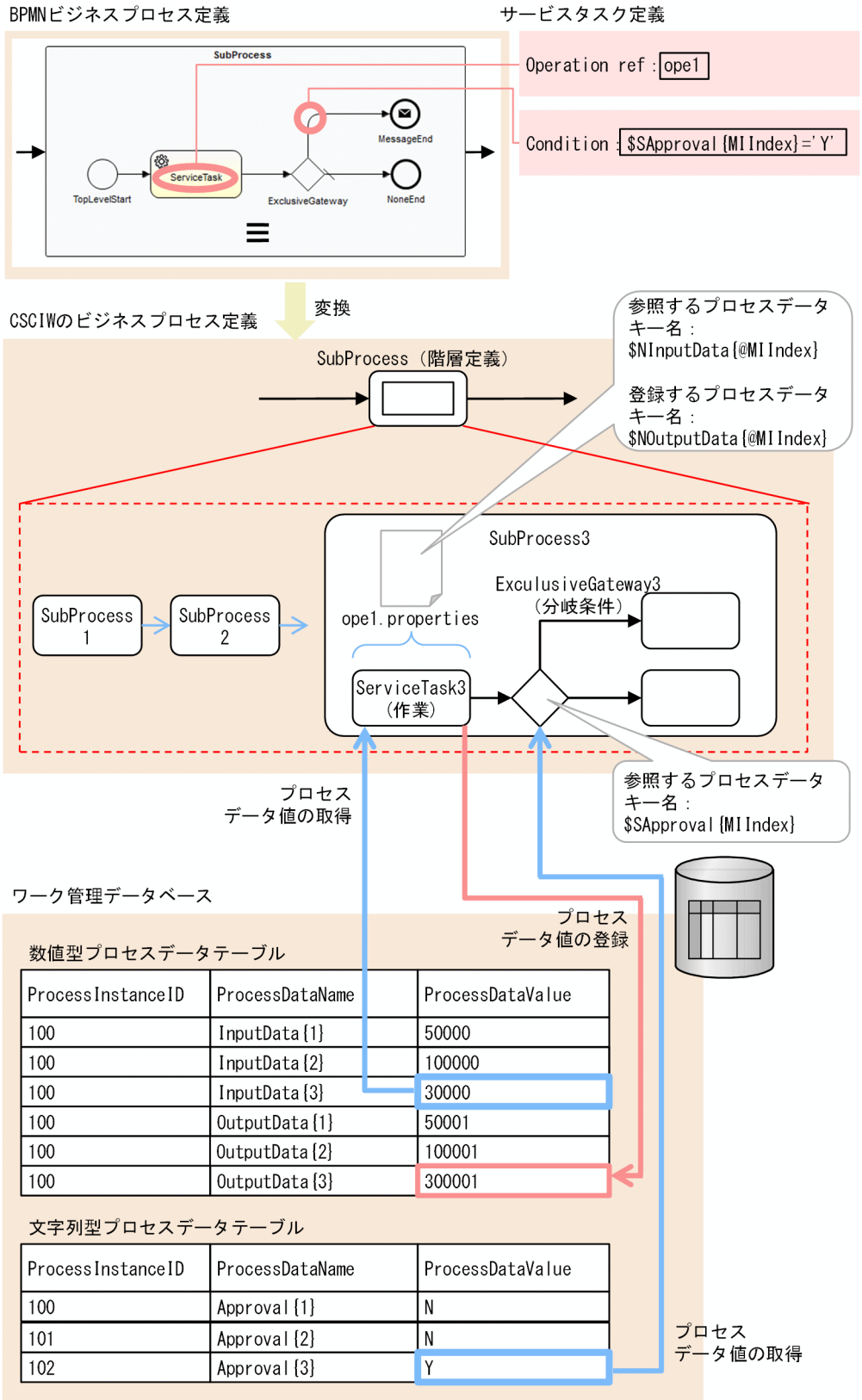
マルチインスタンスでの作業実行時のプロセスデータ使用例（サブプロセスの場合）

サブプロセス（マルチインスタンス）内の定義によって，プロセスデータを使って次の操作ができます。

サブプロセス内の定義	操作
サービスタスク	サービスタスク，ビジネスルールタスク，またはコールアクティビティにマルチインスタンスが定義された場合と同様に，リスト型プロセスデータの 1 要素を参照または更新できます。
ビジネスルールタスク	
コールアクティビティ	
排他ゲートウェイ	リスト型プロセスデータの 1 要素を参照し，分岐条件を評価できます。
スロー（メッセージ）	リスト型プロセスデータの 1 要素を参照または更新できます。
終了（メッセージ）	
終了（エラー）	
キャッチ（タイマー）	リスト型プロセスデータの 1 要素を参照し，タイムアウト期限や待機時間を設定できます。
境界中断（タイマー）	
境界非中断（タイマー）	
イベント・サブプロセス中断開始（タイマー）	
イベント・サブプロセス非中断開始（タイマー）	
ユーザタスク	リスト型プロセスデータの 1 要素を参照し，ユーザタスクの作業者を割り当てることができます。
アドホック・サブプロセス	リスト型プロセスデータの 1 要素を参照し，完了条件を評価できます。

シーケンシャルマルチインスタンスが定義されたサブプロセス実行時の，プロセスデータの使用例を説明します。

図 1-90　シーケンシャルマルチインスタンスでの作業実行時のプロセスデータの使用例（サブプロセスの場合）



［説明］

シーケンシャルマルチインスタンスが定義されたサブプロセスによって，サブプロセスが 1 件ずつ生成され，逐次に遷移します。それぞれのサブプロセス内に定義された CSCIW の作業と分岐条件には，マルチインスタンスインデクスが付与されます。

シーケンシャルマルチインスタンスの場合，マルチインスタンスインデクスは生成したサブプロセスの順に 1 から割り当てられます。例えば，1 回目に遷移したサブプロセス内に定義された CSCIW の作業および分岐条件のマルチインスタンスインデクスは，1 になります。

BPMN 要素ごとの動作を次の表で説明します。

BPMN 要素	動作
サービスタスク	アプリケーション呼び出し情報ファイルまたはコールアクティビティ情報ファイルに，リスト型プロセスデータの 1 要素を参照や更新をするように記述すると，それぞれの作業は，割り当てられたマルチインスタンスインデクスに対応したリスト型プロセスデータのリスト内識別子を参照します。
ビジネスルールタスク	
コールアクティビティ	
排他ゲートウェイ	排他ゲートウェイの出力シーケンスフローの評価式にリスト型プロセスデータの 1 要素を参照するように記述すると，それぞれの評価式では，割り当てられたマルチインスタンスインデクスに対応したリスト型プロセスデータのリスト内識別子を参照します。
スロー（メッセージ）	アプリケーション呼び出し情報ファイルにリスト型プロセスデータの 1 要素を参照または更新するように記述すると，それぞれの作業は，割り当てられたマルチインスタンスインデクスに対応したリスト型プロセスデータのリスト内識別子を参照します。
終了（メッセージ）	
終了（エラー）	
キャッチ（タイマー）	BPMN エディタで，タイマールールにリスト型プロセスデータの 1 要素を参照するように設定すると，それぞれの作業は，割り当てられたマルチインスタンスインデクスに対応したリスト型プロセスデータのリスト内識別子を参照します。
境界中断（タイマー）	
境界非中断（タイマー）	
イベント・サブプロセス中断開始（タイマー）	
イベント・サブプロセス非中断開始（タイマー）	
ユーザタスク	BPMN エディタで，ユーザタスクの作業者にリスト型プロセスデータの 1 要素を参照するように設定すると，それぞれの作業は，割り当てられたマルチインスタンスインデクスに対応したリスト型プロセスデータのリスト内識別子を参照します。
アドホック・サブプロセス	アドホック・サブプロセスの完了条件にリスト型プロセスデータの 1 要素を参照するように記述すると，それぞれのアドホック・サブプロセスの完了条件では，割り当てられたマルチインスタンスインデクスに対応したリスト型プロセスデータのリスト内識別子を参照します。

次に，パラレルマルチインスタンスが定義されたサブプロセスでの作業実行時の，プロセスデータの使用例を説明します。

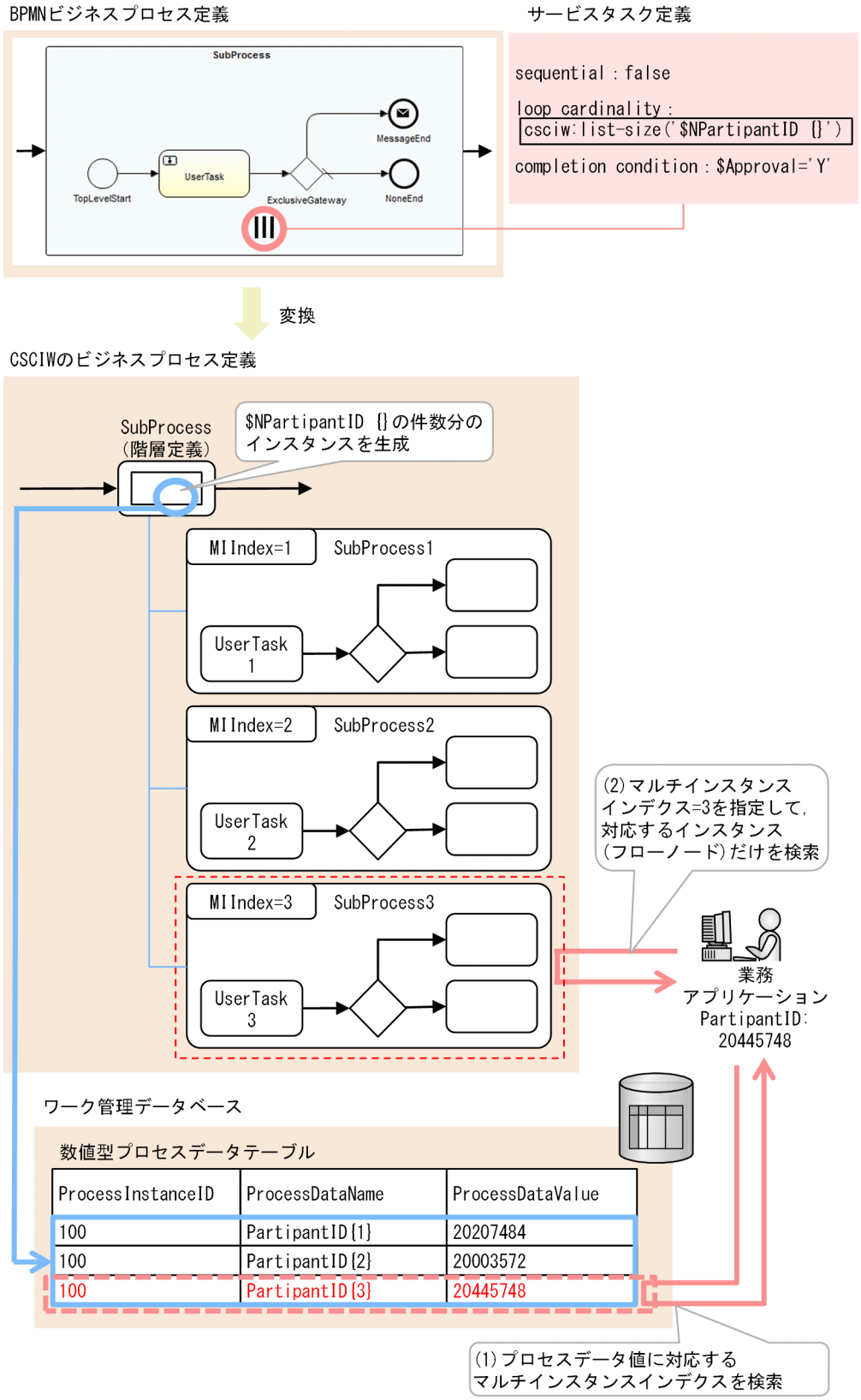
［説明］

パラレルマルチインスタンスが定義されたサブプロセスでは，サブプロセスが同時に生成され，並列に遷移します。また，パラレルマルチインスタンスの場合，マルチインスタンスインデクスが 1 から割り当てられます。ただし，サブプロセスが同時に生成されるため，各サブプロセスにどのマルチインスタンスインデクスが割り当てられるかは不定です。それ以外については，シーケンシャルマルチインスタンスの場合と同じです。

マルチインスタンスの場合の検索時のプロセスデータ使用例

マルチインスタンスの場合の検索時の，プロセスデータの使用例を説明します。

図 1-91　マルチインスタンスの場合の検索時のプロセスデータ使用例



[説明]

マルチインスタンスが定義されたサブプロセスによって，リスト型プロセスデータの要素数分のインスタンスが生成されます。マルチインスタンスで生成された各インスタンスのフローノード ID（BPMN 要素の id 属性値），およびフローノード名（BPMN 要素の name 属性値）は同一になります。

BPMN 連携ライブラリによって，指定したリスト型プロセスデータのプロセスデータ値に対応するマルチインスタンスインデクスを取得し，取得したマルチインスタンスインデクスに対応したインスタンス（フローノード）が検索できます。

1.7 REST サービスとは

REST サービスについて説明します。

REST サービスは，REST API を使用してリソースを操作する機能を提供する J2EE アプリケーションです。

業務アプリケーションは，リソースの取得やクエリの実行などをしたいリソースがある場合，REST サービスが提供する REST API を使用してリソースにアクセスできます。CSCIW の REST サービスはサーブレットアプリケーションで実装されます。REST サービスは，HTTP リクエストを受け取り，リソースを操作してレスポンスを返します。

操作対象のリソースは，URL 形式で指定します。REST API で REST サービスにアクセスすると，結果は XML 形式で返却されます。REST サービスはステートレスで実装されているため，サーバ側には状態を持ちません。

1.8　アプリケーション呼び出しサービスとは

アプリケーション呼び出しサービスについて説明します。

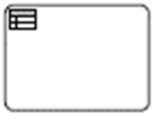
アプリケーション呼び出しサービスは，BPMN ビジネスプロセス定義のサービスタスクやビジネスルールタスクに定義された業務アプリケーションを呼び出す J2EE アプリケーションです。呼び出す業務アプリケーションは，REST に基づいて作成されたアプリケーション（REST アプリケーション）または CSCIW が提供するインタフェースを実装した Java オブジェクトです。また，アプリケーション呼び出しサービスでは，REST アプリケーションや Java オブジェクトの呼び出し以外に，次の処理も行います。

- BPMN のメッセージの送信
- BPMN のエラーのスロー
- BPMN のタイマーイベントの発火

1.8.1　呼び出し対象の BPMN 要素

アプリケーション呼び出しサービスで呼び出し対象となる BPMN 要素と，各 BPMN 要素に対応する呼び出しの種類を示します。

表 1-20　呼び出し対象の BPMN 要素と呼び出しの種類

項番	BPMN 要素		呼び出しの種類
1	サービスタスク		REST アプリケーションの呼び出し Java オブジェクトの呼び出し
2	ビジネスルールタスク		REST アプリケーションの呼び出し Java オブジェクトの呼び出し
3	スロー（メッセージ）		• REST アプリケーションの呼び出し • メッセージによる案件の投入 • メッセージによる自案件の呼び出し • メッセージによる他案件の呼び出し
4	終了（メッセージ）		
5	終了（エラー）		エラーによる自案件の呼び出し

また，アプリケーション呼び出しサービスが呼び出しの対象とする BPMN 要素を定義する際，システム ID 内で一意の文字列として定義した ref 識別子を設定します。アプリケーション呼び出しサービスではこの ref 識別子ごとに呼び出しを設定，処理，および管理します。

アプリケーション呼び出しサービスが案件遷移または案件投入を行う，タイマーイベントの BPMN 要素とそれぞれの動作を次の表に示します。

表 1-21　サポートするタイマーイベントの BPMN 要素

項番	BPMN 要素		動作
1	キャッチ（タイマー）		タイマーによる案件遷移
2	境界中断（タイマー）		
3	境界非中断（タイマー）		
4	イベント・サブプロセス中断開始（タイマー）		
5	イベント・サブプロセス非中断開始（タイマー）		
6	開始（タイマー）		タイマーによる案件投入

タイマーイベントは ref 識別子を持ちません。

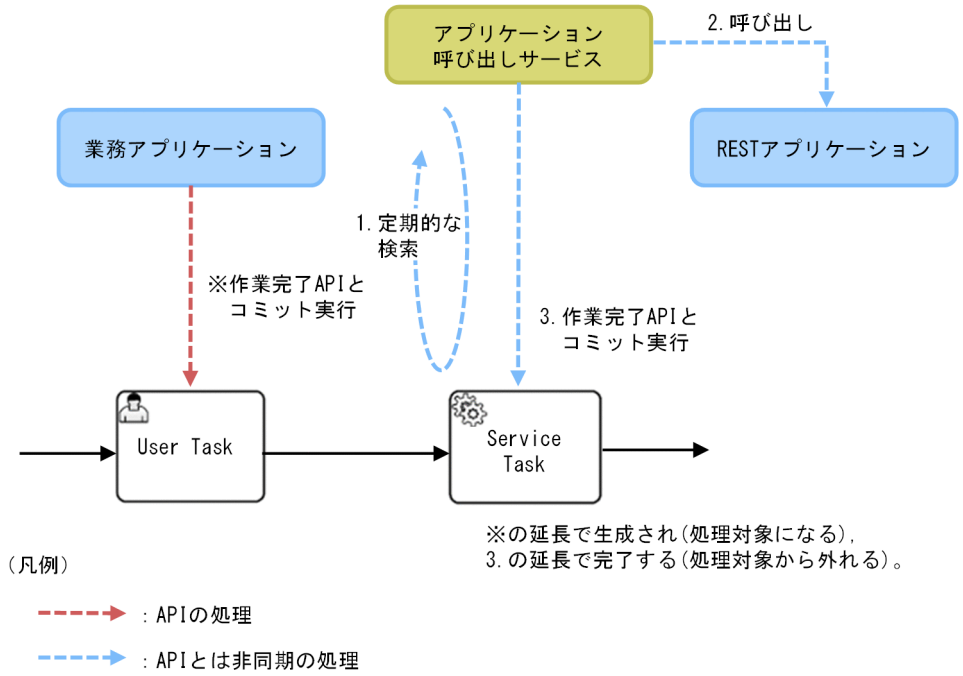
関連項目

- 1.3.1　BPMN 連携機能で使用できる BPMN 要素

1.8.2　アプリケーション呼び出しサービスの動作の概要

アプリケーション呼び出しサービスの動作の概要を，次の図で示します。

図 1-92　アプリケーション呼び出しサービスの動作の概要



[説明]

1. アプリケーション呼び出しサービスは，処理対象の BPMN 要素に対応する作業「Service Task」が生成されているかどうかを，定期的に検索します。作業「Service Task」は，アプリケーション呼び出しサービスと非同期に動作する業務アプリケーションが，手前の作業「User Task」に対して作業完了 API とトランザクションのコミット（図中の※）を実行した延長で生成されます。
2. アプリケーション呼び出しサービスは，処理対象の作業「Service Task」が生成されていることを検知すると，REST アプリケーションを呼び出します。
3. REST アプリケーションの呼び出しが成功すると，アプリケーション呼び出しサービスは，処理対象の作業「Service Task」に対して作業完了 API とトランザクションのコミットを実行します。これによって作業「Service Task」が処理対象から外れます。

1.8.3　サービスタスク・ビジネスルールタスク・メッセージイベント・エラーイベントの場合の処理の流れ

アプリケーション呼び出しサービスの処理対象の BPMN 要素がサービスタスク，ビジネスルールタスク，メッセージイベント，またはエラーイベントの場合の，アプリケーション呼び出しが実施されるまでの処理の流れを説明します。

ここでは，アプリケーション呼び出しサービスの処理対象の BPMN 要素が次のどれかである場合の，処理の流れを説明します。

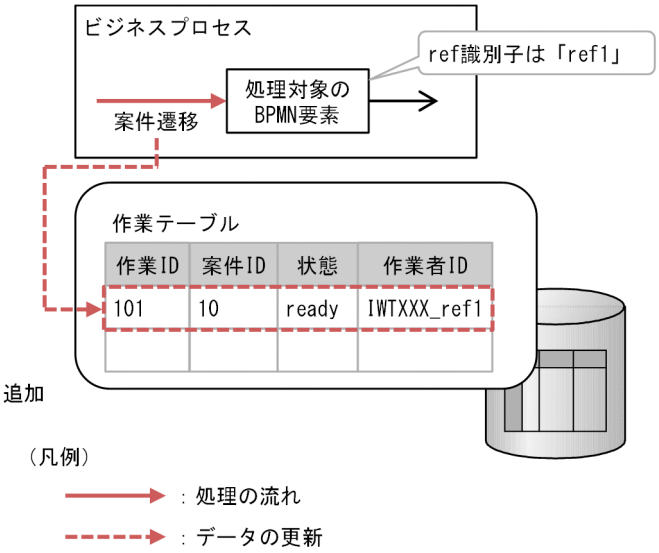
- サービスタスク
- ビジネスルールタスク

- スロー（メッセージ）
- 終了（メッセージ）
- 終了（エラー）

BPMN 連携ライブラリの処理の流れ

アプリケーション呼び出しサービスの処理対象の BPMN 要素に対応する作業が，作業テーブルに追加されるまでの BPMN 連携ライブラリの処理の流れを，次の図で示します。

図 1-93　BPMN 連携ライブラリの処理の流れ



［説明］

業務アプリケーションまたはアプリケーション呼び出しサービスが，アプリケーション呼び出しサービスの処理対象の BPMN 要素の 1 つ前の BPMN 要素に対して遷移要求すると，BPMN 連携ライブラリは案件を遷移させます（図中の「案件遷移」）。

その際，BPMN 連携ライブラリは，アプリケーション呼び出しサービスの処理対象の BPMN 要素に対応する作業（図では作業 ID=101 の作業）を作業テーブルに追加します。また，作業の作業者 ID に「IWTXXX_ref1」※を設定します。

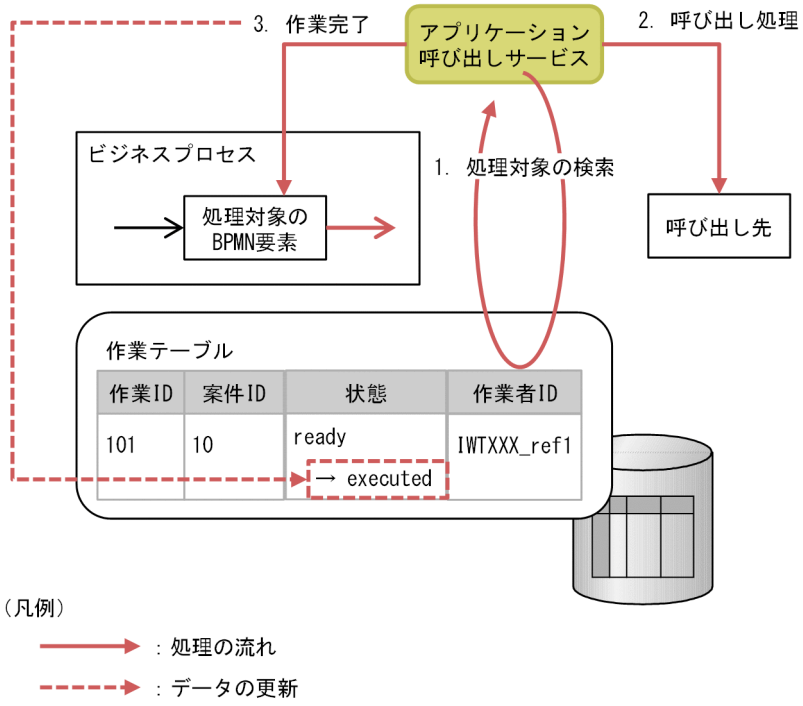
注※

ref 識別子が「ref1」の場合の例です。また，「XXX」には BPMN 要素ごとに異なる値が入ります。

アプリケーション呼び出しサービスの処理の流れ

アプリケーション呼び出しサービスの処理の流れを，次の図で示します。

図 1-94　アプリケーション呼び出しサービスの処理の流れ（サービスタスク・ビジネスルールタスク・メッセージイベント・エラーイベントの場合）



[説明]

アプリケーション呼び出しサービスは，定期的に次の 1.～3.の処理を行います。

1. 処理対象の ref 識別子を決め，その ref 識別子を持つ作業を作業テーブルから検索します。図の例では，次の条件で検索しています。
   - ref 識別子：`ref1`
   - 状態：「実行開始可能」
   - 作業者 ID：`IWTXXX_ref1`（「XXX」は BPMN 要素ごとに異なる文字列）
2. 作業（図の例では作業 ID=101 の作業）が見つかった場合，ref 識別子に対応するアプリケーション呼び出し情報ファイルに従い，BPMN 要素ごとの呼び出し処理を行います。

   BPMN 要素ごとの呼び出し処理の詳細については，「(1)　BPMN 要素ごとの呼び出し処理」の説明を参照してください。
3. BPMN 要素ごとの呼び出し処理が成功すると，アプリケーション呼び出しサービスは，作業の状態を「実行開始可能」から「実行済」に変更します。

(1)　BPMN 要素ごとの呼び出し処理

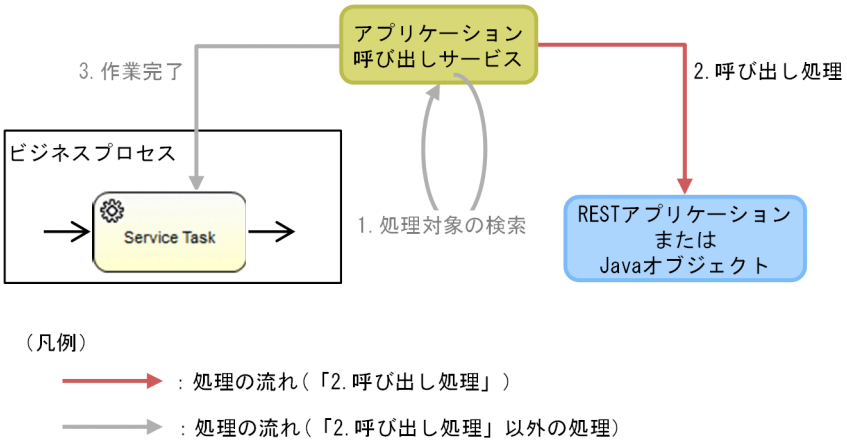
BPMN 要素（サービスタスク，ビジネスルールタスク，メッセージイベント，またはエラーイベント）ごとのアプリケーション呼び出しサービスの呼び出し処理の流れについて説明します。

「図 1-94　アプリケーション呼び出しサービスの処理の流れ（サービスタスク・ビジネスルールタスク・メッセージイベント・エラーイベントの場合）」の「2. 呼び出し処理」で，アプリケーション呼び出しサービスは BPMN 要素ごとに次の処理をします。

サービスタスクまたはビジネスルールタスクの場合

サービスタスクまたはビジネスルールタスクの場合，「図 1-94　アプリケーション呼び出しサービスの処理の流れ（サービスタスク・ビジネスルールタスク・メッセージイベント・エラーイベントの場合）」の「2. 呼び出し処理」で，アプリケーション呼び出しサービスは REST アプリケーション，または Java オブジェクトを呼び出します。

図 1-95　REST アプリケーションまたは Java オブジェクト呼び出し



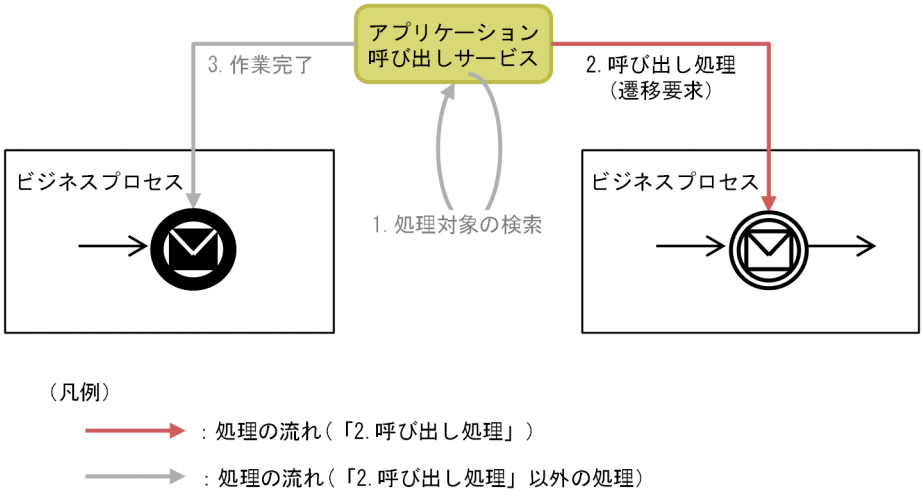
メッセージイベントの場合

メッセージイベントの場合，「図 1-94　アプリケーション呼び出しサービスの処理の流れ（サービスタスク・ビジネスルールタスク・メッセージイベント・エラーイベントの場合）」の「2. 呼び出し処理」で，アプリケーション呼び出しサービスはメッセージを送信します。

CSCIW では，メッセージの送信は，メッセージ送信先のビジネスプロセス定義に対する案件投入，またはメッセージ送信先の作業に対する遷移要求として行われます。

メッセージの送信元が終了（メッセージ）イベントで，送信先がキャッチ（メッセージ）イベントの場合の，メッセージ送信の処理を次の図で示します。

図 1-96　メッセージ送信



メッセージの送信元および送信先になる BPMN 要素を次に示します。

メッセージの送信元※1 になる BPMN 要素

スロー（メッセージ）

終了（メッセージ）

メッセージの送信先※2 になる BPMN 要素

開始（メッセージ）

イベント・サブプロセス中断開始（メッセージ）

イベント・サブプロセス非中断開始（メッセージ）

キャッチ（メッセージ）

境界中断（メッセージ）

境界非中断（メッセージ）

注※1

メッセージの送信元は，「図 1-94　アプリケーション呼び出しサービスの処理の流れ（サービスタスク・ビジネスルールタスク・メッセージイベント・エラーイベントの場合）」の「処理対象の BPMN 要素」に対応しています。

注※2

メッセージの送信先は，「図 1-94　アプリケーション呼び出しサービスの処理の流れ（サービスタスク・ビジネスルールタスク・メッセージイベント・エラーイベントの場合）」の「呼び出し先」に対応しています。

各 BPMN 要素の動作については，「1.3.4　BPMN 要素の CSCIW ビジネスプロセス定義への変換の詳細」の，各 BPMN 要素の説明を参照してください。

なお，メッセージイベントの場合，REST アプリケーションを呼び出すこともできます。

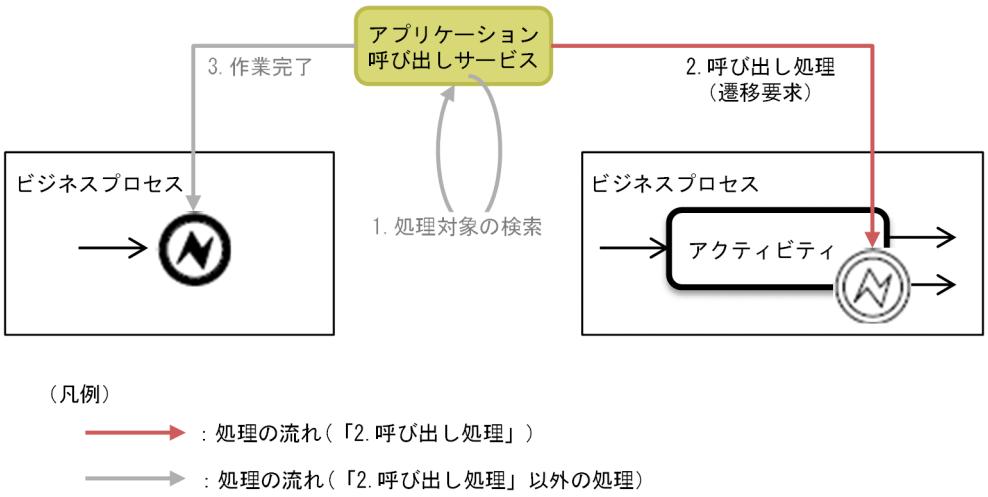
エラーイベントの場合

エラーイベントの場合，「図 1-94　アプリケーション呼び出しサービスの処理の流れ（サービスタスク・ビジネスルールタスク・メッセージイベント・エラーイベントの場合）」の「2. 呼び出し処理」で，アプリケーション呼び出しサービスはエラーをスローします。

CSCIW では，エラーのスローは，エラーのスロー先の作業に対する遷移要求として実行されます。

エラーのスロー元が終了（エラー），エラーのスロー先が境界中断（エラー）の場合の，エラースローの処理を次の図で示します。

図 1-97　エラースロー



エラーのスロー元またはスロー先になる BPMN 要素を次に示します。

エラーのスロー元※1 になる BPMN 要素

終了（エラー）

エラーのスロー先※2 になる BPMN 要素

イベント・サブプロセス中断開始（エラー）

境界中断（エラー）

注※1

エラーのスロー元は，「図 1-94　アプリケーション呼び出しサービスの処理の流れ（サービスタスク・ビジネスルールタスク・メッセージイベント・エラーイベントの場合）」の「処理対象の BPMN 要素」に対応しています。

注※2

エラーのスロー先は，「図 1-94　アプリケーション呼び出しサービスの処理の流れ（サービスタスク・ビジネスルールタスク・メッセージイベント・エラーイベントの場合）」の「呼び出し先」に対応しています。

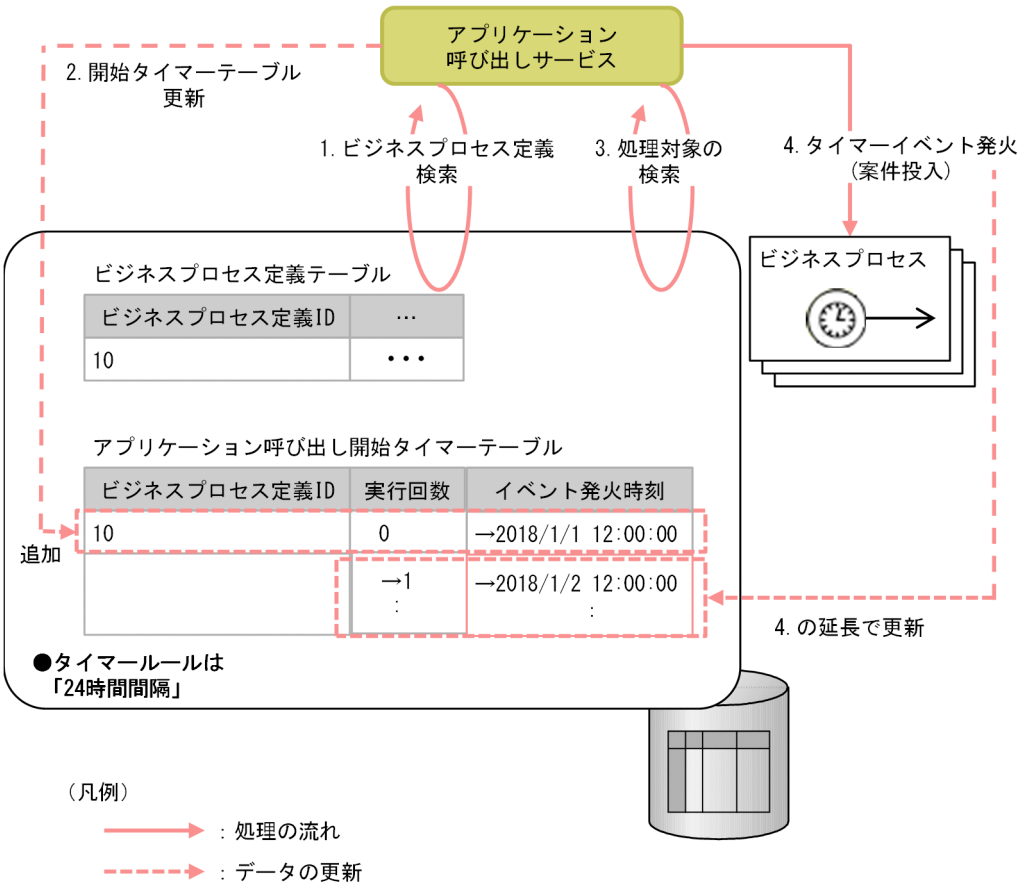
各 BPMN 要素の動作については，「1.3.4　BPMN 要素の CSCIW ビジネスプロセス定義への変換の詳細」の，各 BPMN 要素の説明を参照してください。

1.8.4　開始（タイマー）の場合の処理の流れ

アプリケーション呼び出しサービスが開始（タイマー）を処理する流れを説明します。

アプリケーション呼び出しサービスが開始（タイマー）を処理する流れを，次の図で示します。

図 1-98　開始（タイマー）の場合のアプリケーション呼び出しサービスの処理の流れ



［説明］

アプリケーション呼び出しサービスが定期的に次の 1.～4.の処理を行います。

1. ビジネスプロセス定義テーブルを検索し，開始（タイマー）を含むビジネスプロセス定義が存在するかどうかをチェックします。
2. 1.でビジネスプロセス定義が存在していた場合，そのビジネスプロセス定義がアプリケーション呼び出し開始タイマーテーブルに存在しなければ追加します。また，追加したビジネスプロセス定義のタイマールールを評価して決定したイベント発火時刻とタイマーの実行回数「0」をアプリケーション呼び出し開始タイマーテーブルに設定します。
3. アプリケーション呼び出し開始タイマーテーブルから，ビジネスプロセス定義を次の条件で検索します。
   ・状態が「活性」
   ・イベント発火時刻≦現在時刻
4. 3.でビジネスプロセス定義が見つかった場合（図の例では，ビジネスプロセス定義 ID ＝ 10），開始（タイマー）の発火を行います。この発火は，CSCIW ではビジネスプロセス定義に対する案件投入として行われます。その際，次の処理も行われます。
   1. タイマーの実行回数を 1 増加させて，アプリケーション呼び出し開始タイマーテーブルに設定する

2. タイマーの実行回数が最大実行回数（Number of execution）に達していない場合，タイマールールを評価して決定した次回のイベント発火時刻をアプリケーション呼び出し開始タイマーテーブルに設定する

1.8.5　開始（タイマー）以外のタイマーイベントの場合の処理の流れ

ここでは，アプリケーション呼び出しサービスが開始（タイマー）以外のタイマーイベントを処理する流れを説明します。

ここで説明する，開始（タイマー）以外のタイマーイベントとは，次のタイマーイベントのことです。

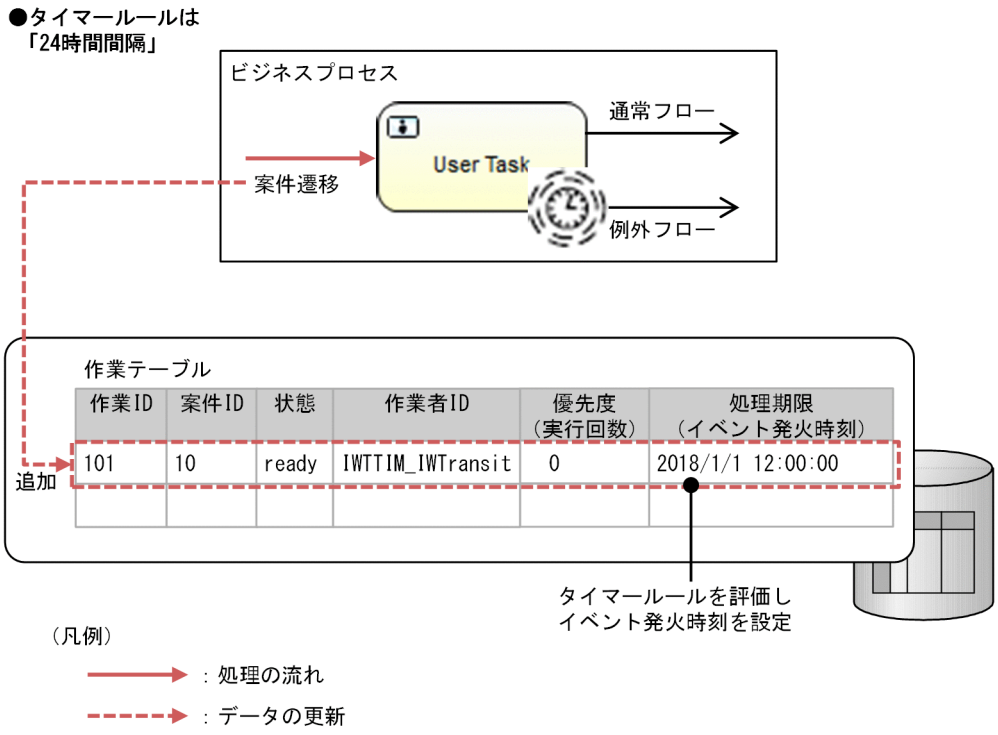
- キャッチ（タイマー）
- 境界中断（タイマー）
- 境界非中断（タイマー）
- イベント・サブプロセス中断開始（タイマー）
- イベント・サブプロセス非中断開始（タイマー）

ここでは，境界非中断（タイマー）を例として処理の流れを説明します。

BPMN 連携ライブラリの処理の流れ

境界非中断（タイマー）に対応する作業が作業テーブルに追加されるまでの BPMN 連携ライブラリの処理の流れを，次の図に示します。

図 1-99　開始（タイマー）以外のタイマーイベントの場合の BPMN 連携ライブラリの処理の流れ



[説明]

業務アプリケーションまたはアプリケーション呼び出しサービスが，境界非中断（タイマー）の 1 つ前の BPMN 要素に対して遷移要求を行うと，BPMN 連携ライブラリは案件を遷移させます。

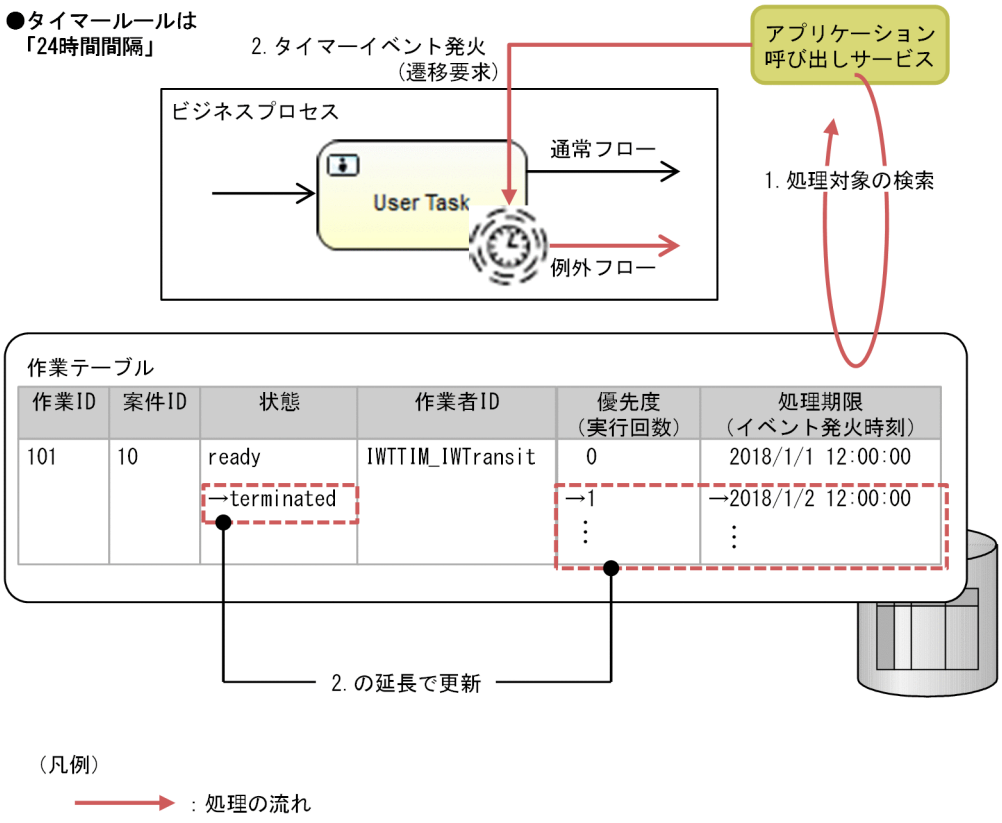
その際，BPMN 連携ライブラリは，境界非中断（タイマー）に対応する作業（図の例では作業 ID=101 の作業）を作業テーブルに追加します。また，次の設定をします。

- 作業の処理期限：タイマールールを評価して決定したイベント発火時刻
- 作業の優先度：タイマーの実行回数「0」

アプリケーション呼び出しサービスの処理の流れ

アプリケーション呼び出しサービスの処理の流れを，次の図に示します。

図 1-100　開始（タイマー）以外のタイマーイベントの場合のアプリケーション呼び出しサービスの処理の流れ



[説明]

アプリケーション呼び出しサービスが定期的に次の 1.～2.の処理を行います。

1. 境界非中断（タイマー）に対応する作業を検索します。図の例では，次の条件で検索しています。
   - 状態：実行開始可能
   - 作業者 ID：IWTTIM_IWTransit
   - イベント発火時刻≦現在時刻

2. 作業が見つかった場合（図の例では作業 ID=101 の作業），境界非中断（タイマー）を発火させます。この発火は，CSCIW では境界非中断（タイマー）に対応する作業に対する遷移要求として行われます。その際，次の処理も行います。

   1. タイマーの実行回数を 1 増加させ，作業の優先度に設定する。

   2. タイマーの実行回数が最大実行回数（Number of execution）に達していた場合，作業の状態を「実行開始可能」から「終了」に変更する。タイマーの実行回数が最大実行回数に達していない場合，タイマールールを評価して決定した次回のイベント発火時刻を作業の処理期限に設定する。

1.8.6 実行間隔とポーリング間隔

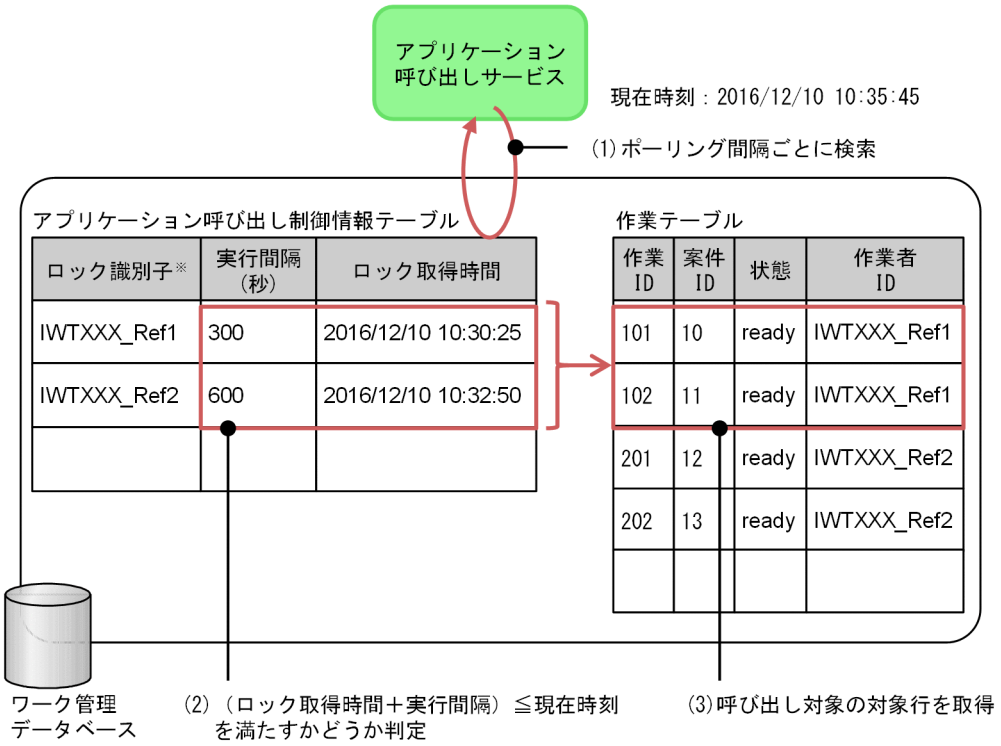
アプリケーション呼び出しサービスが処理を実行する契機は，実行間隔とポーリング間隔によって決まります。

実行間隔は，ref 識別子ごとに設定する間隔で，呼び出す必要がある作業があるかどうかを検索する間隔です。

ポーリング間隔は，アプリケーション呼び出しサービスごとに設定する間隔で，呼び出す必要がある ref 識別子があるかどうかを検索する間隔です。

次の図で，実行間隔とポーリング間隔について説明します。

図 1-101 実行間隔とポーリング間隔



注※

ref識別子に対応する作業者IDの値を指します。

［説明］

アプリケーション呼び出しサービスは，呼び出す必要がある ref 識別子を見つけるために，ポーリング間隔ごとにアプリケーション呼び出し制御情報テーブルを検索します（図中の(1)）。アプリケーション呼び出し制御情報テーブルには，前回呼び出し処理を行った時間（ロック取得時間）が格納されています。図の例では，Ref1 は 2016/12/10 10:30:25 に，Ref2 は 2016/12/10 10:32:50 に前回の呼び出し処理を行っています。

さらに，アプリケーション呼び出しサービスは，Ref1 およびRef2 が実行間隔を過ぎているかどうかを判定します（図中の(2)）。図の例では，Ref1 の実行間隔が 300 秒，Ref2 の実行間隔が 600 秒に設定されています。ロック取得時間に対して，設定されている実行間隔を現在時刻が過ぎているRef1 が呼び出し処理の対象となります（図中の(3)）。

アプリケーション呼び出しサービスは，呼び出し処理の対象となったRef1 を作業者 ID に持ち，状態が「実行開始可能（ready）」の作業に対して，呼び出し処理を行います。

実行間隔は，ref 識別子単位で設定できます。アプリケーション呼び出し制御情報の「実行間隔（ExecuteInterval）」に設定してください。アプリケーション呼び出し制御情報の詳細については，マニュアル『uCosminexus Service Coordinator Interactive Workflow コマンド』の「ciwmngap（アプリケーション呼び出し制御情報の管理）」を参照してください。

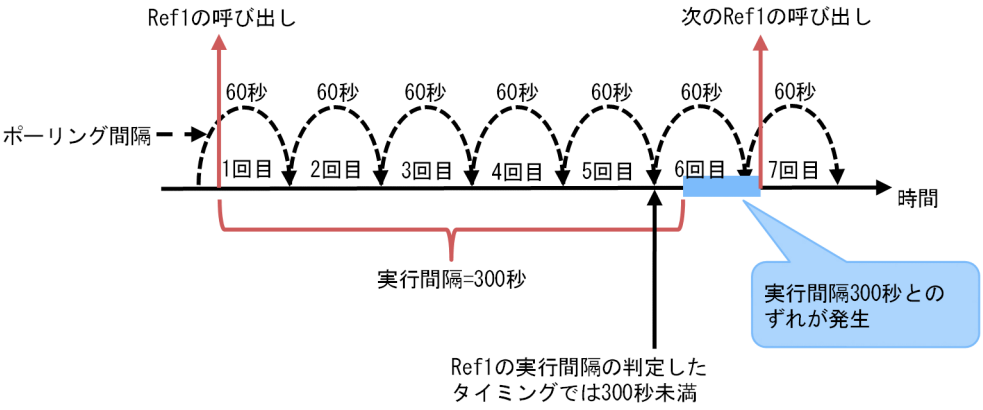
ポーリング間隔は，システム ID 単位で設定できます。共通設定ファイルの「AppCallServicePollingInterval」に設定してください。AppCallServicePollingInterval の詳細については，「15.2.5　共通設定ファイルに指定する内容」の「(4)　AppCallServicePollingInterval」を参照してください。

ヒント

アプリケーション呼び出しサービスが実際に作業を呼び出す間隔は，実行間隔の設定値から最大でポーリング間隔分のずれが発生します。ポーリング間隔を小さくするほど，実行間隔の設定値と実際に作業を呼び出す間隔とのずれは小さくなりますが，その分アプリケーション呼び出し制御情報テーブルの検索実行回数が多くなります。実行間隔とポーリング間隔のずれを次の図に示します。

図の例では，ポーリング間隔 60 秒ごとに，アプリケーション呼び出しサービスがアプリケーション呼び出し制御情報テーブルを検索しています。まず，1 回目のポーリング間隔の途中の時刻にRef1 を呼び出します。そのあと，ポーリングを繰り返し，6 回目のポーリング間隔の最初にRef1 の実行間隔経過を判定します。しかし，判定のタイミングでは，1 回目のポーリング間隔でのRef1 の呼び出しから 300 秒が経過していません。よって，次のRef1 の呼び出し処理は 7 回目のポーリング間隔の間に行われるため，実行間隔の設定値である 300 秒とのずれが発生します。

図 1-102　実行間隔とポーリング間隔のずれ



1.8.7　呼び出しの流量制御

アプリケーション呼び出しサービスでは，呼び出す作業数を制御して，負荷を抑えることができます。呼び出しは，最大作業件数および WorkManager の最大スレッド数の設定によって制御できます。

最大作業件数

最大作業件数は，1 回の実行間隔内で発生した作業のうち，呼び出し処理を行う作業件数の最大値です。業務処理のピーク時に呼び出し対象の作業数が増大しても，呼び出し処理を実行する作業数を制限して，負荷増加を抑えることができます。呼び出し処理の対象から外れた作業が発生した場合，該当する作業は，次回の実行間隔の経過後に呼び出し処理の対象となります。

最大作業件数は，ref 識別子単位で設定できます。アプリケーション呼び出し制御情報の「最大作業件数(`WorkItemMax`)」に設定してください。詳細については，マニュアル『uCosminexus Service Coordinator Interactive Workflow コマンド』の「ciwmngap（アプリケーション呼び出し制御情報の管理)」を参照してください。

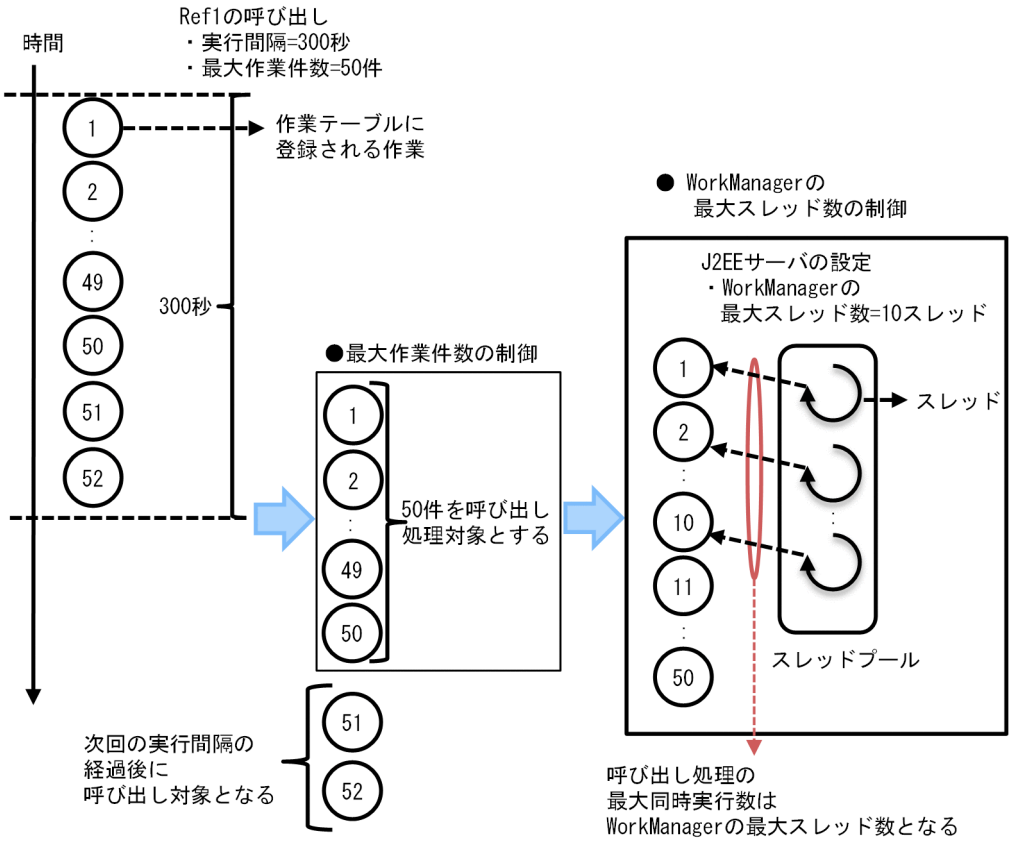
WorkManager の最大スレッド数

WorkManager の最大スレッド数は，呼び出し対象となった作業に対して，並列で同時に処理を実行するためのスレッド数の最大値です。呼び出し対象の作業数が増大しても，呼び出し処理を実行するスレッド数を制限して，負荷増加を抑えることができます。スレッドプール内の最大スレッド数で指定した数のスレッドによって，作業の呼び出し処理が行われます。

WorkManager の最大スレッド数は，J2EE サーバ単位で設定できます。J2EE サーバの設定ファイルに指定してください。詳細については，「9.4　WorkManager の最大スレッド数を変更する」を参照してください。

最大作業件数と WorkManager の最大スレッド数による流量制御の概要を次の図に示します。

図 1-103　流量制御の概要



[説明]

図の例では，Ref1 の最大作業件数が50，WorkManager の最大スレッド数が10 と設定されています。

このとき，Ref1 の実行間隔内で 52 件の作業が作業テーブルに登録されたとすると，呼び出し対象となる作業の件数は最大作業件数の 50 件となります。50 件を超えた作業 2 件は，次回の実行間隔の経過後の呼び出し対象になります。

呼び出し対象となる 50 件の作業に対しては，WorkManager の最大スレッド数 10 のスレッドで同時に呼び出し処理を実行します。

1.8.8　プロセスデータの利用

ここでは，プロセスデータの利用について説明します。

(1)　REST アプリケーションまたは Java オブジェクト呼び出し時のプロセスデータの受け渡し

アプリケーション呼び出しサービスでは，REST アプリケーションや Java オブジェクトを呼び出す際，呼び出し元案件のプロセスデータを REST アプリケーションまたは Java オブジェクトに渡すことができます。また，REST アプリケーションや Java オブジェクトからデータを受け取り，呼び出し元の案件のプロセスデータに登録できます。

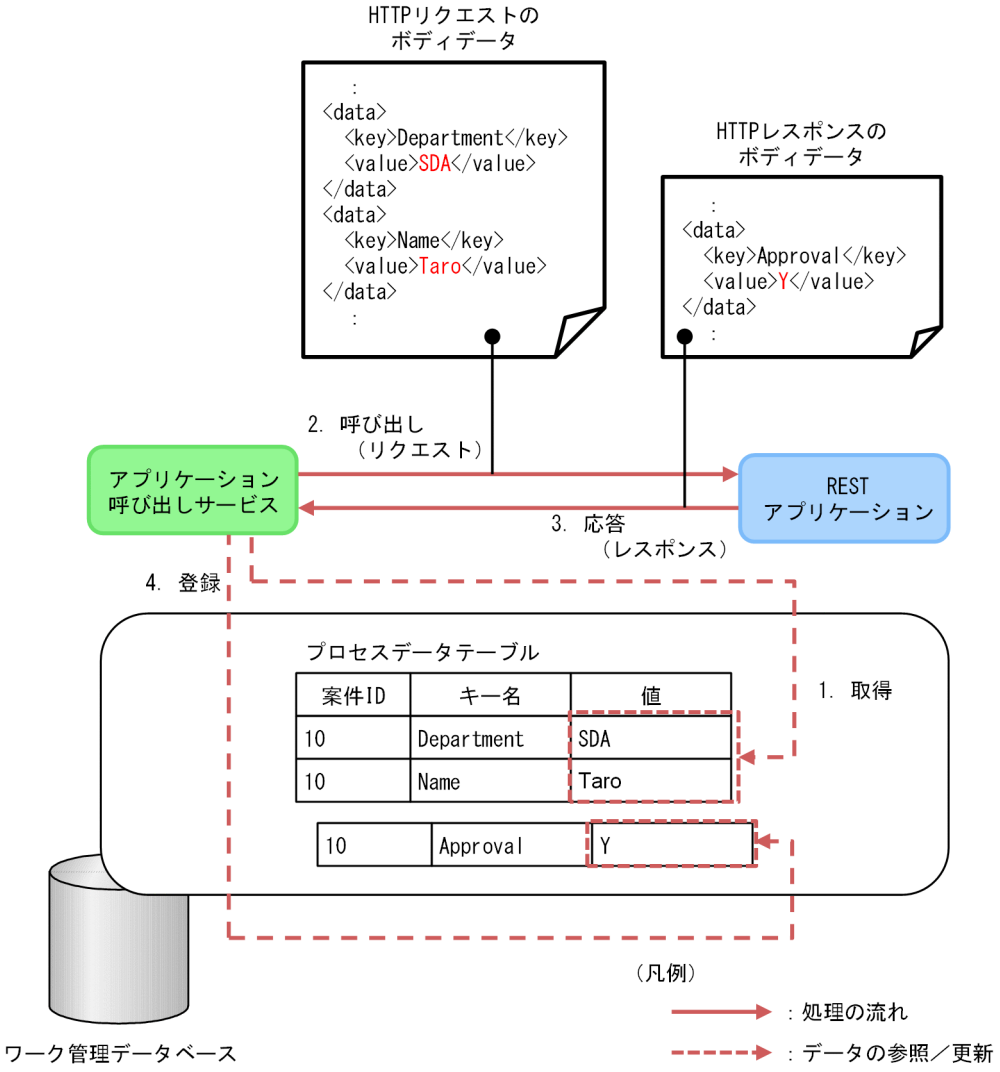
プロセスデータの受け渡しでは，単一型プロセスデータとリスト型プロセスデータの両方を扱うことができます。

プロセスデータを REST アプリケーションや Java オブジェクトに渡す方法，および REST アプリケーションや Java オブジェクトから受け取ったデータをプロセスデータテーブルに登録する方法については，「15.3　アプリケーション呼び出し情報ファイル」を参照してください。

REST アプリケーションを呼び出す場合のプロセスデータの受け渡し

REST アプリケーションを呼び出す場合のプロセスデータの受け渡しの処理の流れを，次の図に示します。

図 1-104　プロセスデータの受け渡し（REST アプリケーション呼び出し時）



[説明]

1. アプリケーション呼び出しサービスがプロセスデータテーブルを検索し，呼び出し元の案件のプロセスデータ値を取得します。図の例では，プロセスデータキー名が"Department"と"Name"であるプロセスデータ値"SDA"と"Taro"を取得しています。
2. アプリケーション呼び出しサービスが REST アプリケーションを呼び出す際に，1.で取得したプロセスデータ値を HTTP リクエストのボディデータとして渡しています。

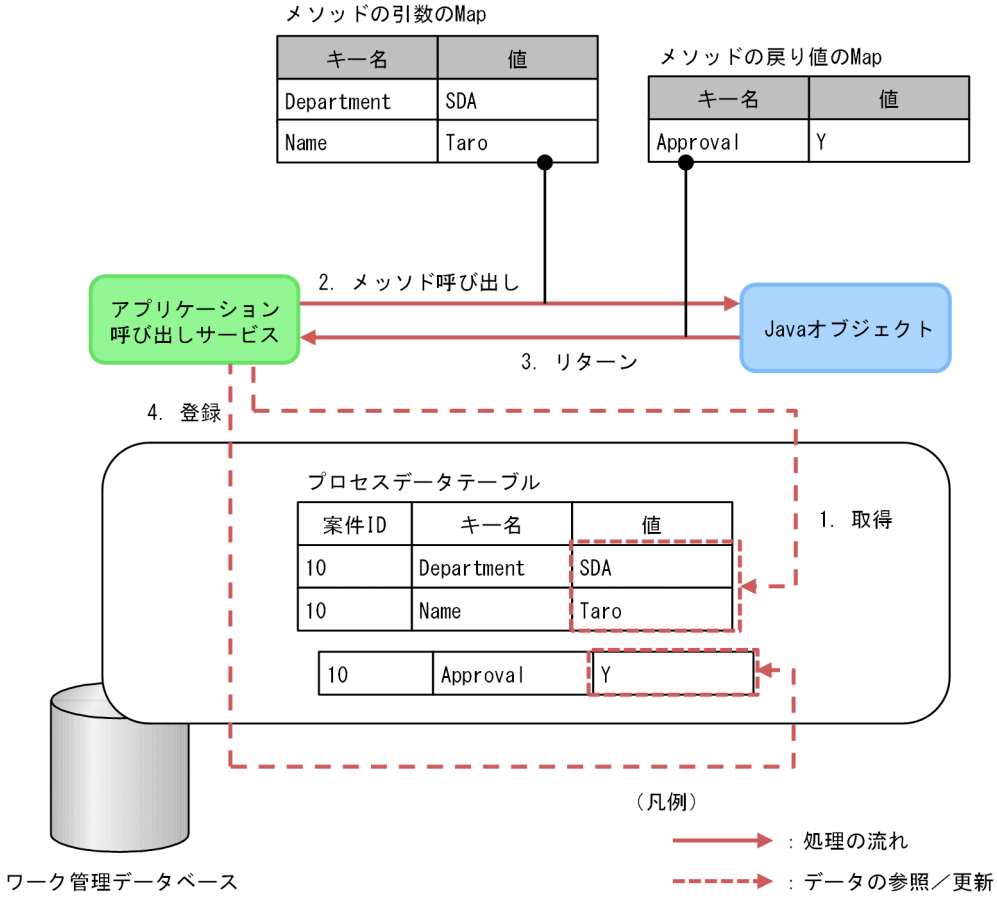
3. REST アプリケーションからの応答がアプリケーション呼び出しサービスに戻ります。図の例では，HTTP レスポンスのボディデータとして，キーを表す"Approval"と，キーに対応するデータ"Y"が含まれているものとします。

4. アプリケーション呼び出しサービスが，3.で REST アプリケーションから受け取ったデータをプロセスデータ値として登録します。図の例では，プロセスデータキー名"Approval"のプロセスデータ値に，3.で受け取ったデータ"Y"を登録しています。

Java オブジェクトを呼び出す場合のプロセスデータの受け渡し

Java オブジェクトを呼び出す場合のプロセスデータの受け渡しの処理の流れを，次の図に示します。

図 1-105　プロセスデータの受け渡し（Java オブジェクト呼び出し時）



[説明]

1. アプリケーション呼び出しサービスがプロセスデータテーブルを検索し，呼び出し元案件のプロセスデータ値を取得します。図の例では，プロセスデータキー名が"Department"と"Name"であるプロセスデータ値"SDA"と"Taro"を取得しています。

2. アプリケーション呼び出しサービスが Java オブジェクトを呼び出す際に，1.で取得したプロセスデータ値を Java オブジェクトのメソッドの引数として渡しています。

3. Java オブジェクトのメソッドの戻り値をアプリケーション呼び出しサービスに渡します。図の例では，キーを表す"Approval"とキーに対応する値"Y"が含まれているものとします。

4. アプリケーション呼び出しサービスが，3.で Java オブジェクトから受け取ったデータをプロセスデータ値として登録します。図の例では，プロセスデータキー名"Approval"のプロセスデータ値に，3.で受け取ったデータ"Y"を登録しています。

関連項目

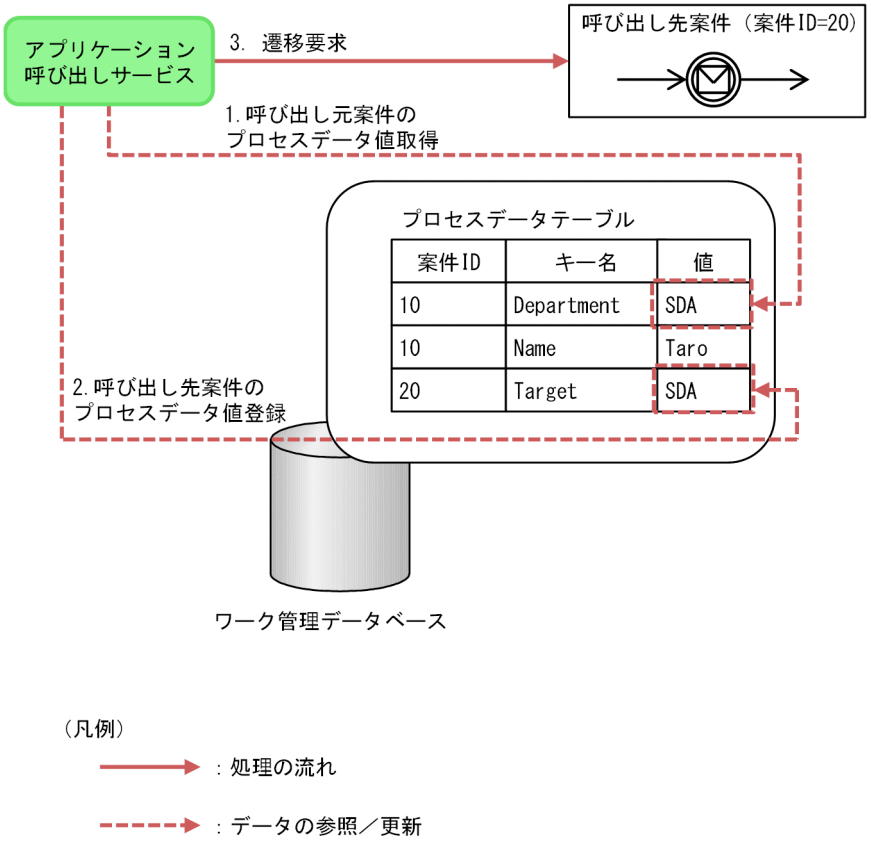
- 1.6　プロセスデータとは

(2)　メッセージ送信時またはエラースロー時の呼び出し先案件へのプロセスデータの登録

メッセージ送信時またはエラースロー時，呼び出し元案件のプロセスデータ値を，呼び出し先案件のプロセスデータ値として登録できます。

呼び出し元案件のプロセスデータ値を，呼び出し先案件のプロセスデータ値として登録する処理の流れを次の図に示します。

図 1-106　呼び出し先案件へのプロセスデータの登録



[説明]

1. アプリケーション呼び出しサービスがプロセスデータテーブルを検索し，呼び出し元案件のプロセスデータ値を取得します。図の例では，呼び出し元案件（案件 ID=10）のプロセスデータキー名が"Department"であるプロセスデータ値"SDA"を取得しています。

2. アプリケーション呼び出しサービスが，呼び出し先案件のプロセスデータを登録します。図の例では，呼び出し先案件（案件 ID=20）のプロセスデータキー名"Target"のプロセスデータ値に，1.で取得したプロセスデータ値"SDA"を登録しています。
3. 呼び出し先案件の BPMN 要素に対して遷移要求を行います。

呼び出し先案件と呼び出し元案件とが同じ案件でも，この例と同様に，呼び出し先案件にプロセスデータを登録できます。プロセスデータを登録する方法については，「15.3.4　プロセスデータの設定方法（アプリケーション呼び出し情報ファイル）」を参照してください。

(3)　メッセージ送信時またはエラースロー時の呼び出し元案件へのプロセスデータの登録

メッセージ送信時またはエラースロー時，呼び出し元の案件にプロセスデータを登録できます。

プロセスデータを登録する方法については，「15.3.4　プロセスデータの設定方法（アプリケーション呼び出し情報ファイル）」を参照してください。

(4)　タイマーイベントの場合

タイマーイベントについては，プロセスデータを受け渡ししません。

1.8.9　障害時の動作

アプリケーション呼び出しの処理中に障害が発生した場合，呼び出しが成功するまでリトライします。アプリケーション呼び出しサービス自身に障害が発生した場合は，アプリケーション呼び出しサービスを多重化していれば，ほかのアプリケーション呼び出しサービスに処理が引き継がれます。

アプリケーション呼び出しサービスでは，大きく分けて次に示す障害の発生が考えられます。

- アプリケーション呼び出し処理での障害
  ・アプリケーション呼び出し情報ファイルの設定不備
  ・呼び出し処理の失敗※
  ・呼び出し元作業の完了の失敗
- アプリケーション呼び出しサービス自身の障害
  J2EE サーバプロセスのダウン

注※
アプリケーション呼び出し処理に失敗するケースを次の表に示します。

表 1-22　アプリケーション呼び出し処理に失敗するケース

BPMN 要素	呼び出し処理に失敗するケース
サービスタスクおよびビジネスルールタスク	• 呼び出し先の REST アプリケーションとの通信障害が発生した場合 • 呼び出し先の REST アプリケーションが返却したステータスコードの判定結果が失敗であった場合 • 呼び出し先の Java オブジェクトのインスタンス生成に失敗した場合 • 呼び出し先の Java オブジェクトで例外が発生した場合
メッセージ	呼び出し処理（案件投入または遷移要求）に失敗した場合
エラー	呼び出し処理（遷移要求）に失敗した場合
タイマー	呼び出し処理（案件投入または遷移要求）に失敗した場合

次に，これらの障害時の動作について説明します。

アプリケーション呼び出し処理での障害

アプリケーション呼び出し処理で障害が発生すると，呼び出しが成功するまでリトライします。

リトライは呼び出しが失敗したあと，リトライ間隔後に行われます。

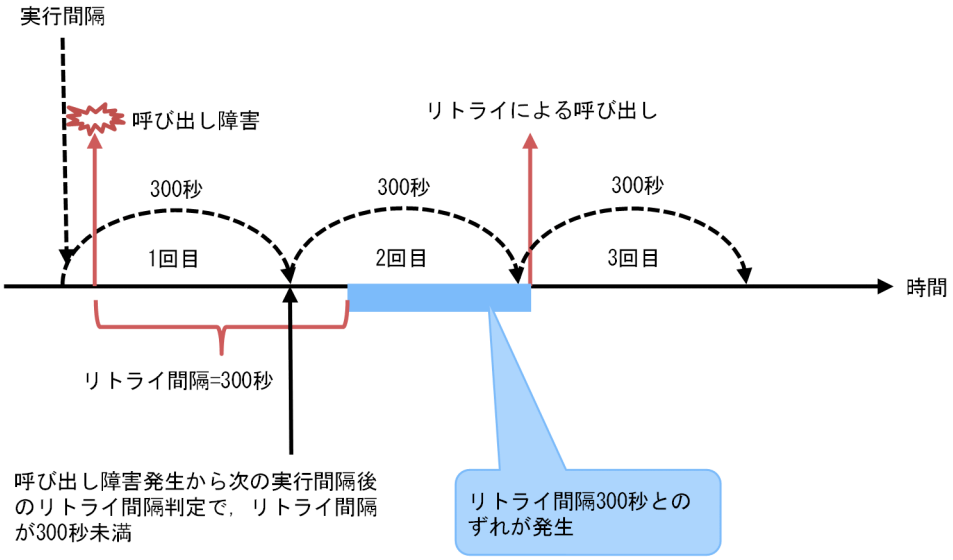
メモ

実際にリトライする間隔は，リトライ間隔の設定値から最大で実行間隔分のずれが発生します。実行間隔を小さくするほど，リトライ間隔の設定値と実際に作業をリトライする間隔とのずれは小さくなります。ただし，呼び出し処理の頻度がその分増えるため，負荷が増加します。

リトライ間隔と実行間隔の関係を次の図に示します。

図の例では，アプリケーション呼び出しサービスは，実行間隔 300 秒ごとに呼び出し処理を行っています。1 回目の実行間隔の途中で呼び出し障害が発生し，呼び出しが失敗した時刻が作業に設定されます。そして，2 回目の実行間隔の最初でリトライ間隔の判定処理が行われます。しかし，判定のタイミングでは，リトライ間隔の 300 秒が経過していません。そのため，リトライによる呼び出しは 3 回目の実行間隔の間に行われ，リトライ間隔の設定値である 300 秒とのずれが発生します。

図 1-107　リトライ間隔と実行間隔のずれ



なお，リトライした回数は，呼び出し元の作業の優先度に設定されます。アプリケーションの呼び出しに失敗した時刻は，呼び出し元の作業の処理期限に設定されます。アプリケーション呼び出しサービスは，設定したリトライ回数だけリトライします。設定した回数分リトライしても失敗した場合は，アプリケーション呼び出しサービスのメッセージファイルに，エラーメッセージを出力します。また，呼び出し元の作業の状態を「作業者実行」に，優先度を「0」に変更します。これ以降，該当する作業はリトライの対象から外れます。

> **メモ**
>
> リトライ対象の作業が BPMN 連携ライブラリの API やコマンドで「実行開始可能」状態に変更された場合，作業の優先度が「0」に変更されます。このとき，設定したリトライ回数よりも多くアプリケーション呼び出しのリトライが行われます。

リトライの対象から外れた作業を再び呼び出し対象とする場合は，障害の原因を取り除いたあとで，該当する作業の状態を「作業者実行」から「実行開始可能」に変更してください。作業の状態を「作業者実行」から「実行開始可能」に変更する方法については，「10.5　リトライの対象から外れた作業に関する運用」を参照してください。

リトライ回数とリトライ間隔は，ref 識別子単位で設定できます。アプリケーション呼び出し制御情報の「リトライ回数（RetryCount）」および「リトライ間隔（RetryInterval）」に設定してください。アプリケーション呼び出し制御情報の詳細については，マニュアル『uCosminexus Service Coordinator Interactive Workflow コマンド』の「ciwmngap（アプリケーション呼び出し制御情報の管理）」を参照してください。

なお，リトライ回数およびリトライ間隔が有効になるかどうかは，BPMN 要素によって異なります。

リトライ回数およびリトライ間隔が有効になる BPMN 要素

- サービスタスク
- ビジネスルールタスク

- メッセージイベント
- エラーイベント

リトライ回数およびリトライ間隔が有効にならない BPMN 要素

- タイマーイベント

アプリケーション呼び出しサービス自身の障害

アプリケーション呼び出しサービス自身が障害になった場合，アプリケーション呼び出しサービスを多重化していれば，ほかのアプリケーション呼び出しサービスが代わりに呼び出し処理を継続できます。

なお，ほかのアプリケーション呼び出しサービスが代わりに呼び出し処理を行うタイミングは，障害復旧間隔が経過したあとになります。

障害復旧間隔は ref 識別子単位に設定できます。アプリケーション呼び出し制御情報の「障害復旧間隔(RecoveryTime)」に設定してください。アプリケーション呼び出し制御情報の詳細については，マニュアル『uCosminexus Service Coordinator Interactive Workflow コマンド』の「ciwmngap（アプリケーション呼び出し制御情報の管理)」を参照してください。

重要

呼び出し先の処理が長時間掛かると，次の図のように障害復旧間隔後にほかのアプリケーション呼び出しサービスが同じ作業の呼び出し処理を行ってしまいます。

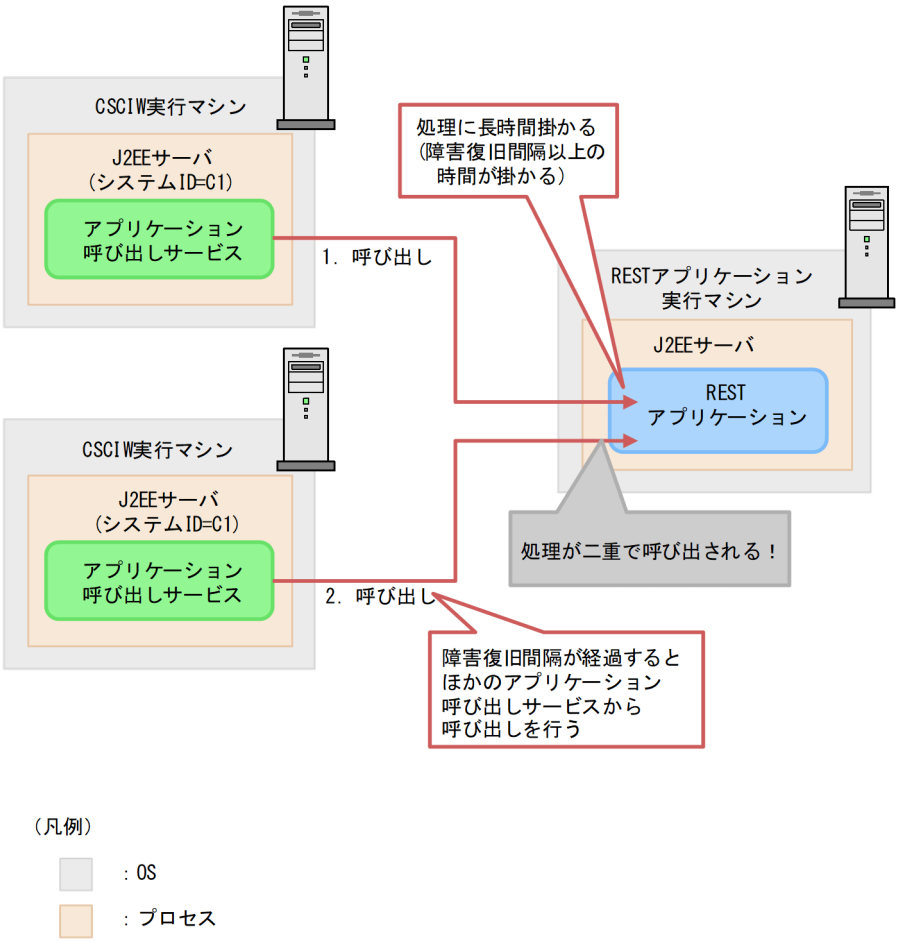
このように，二重に作業が呼び出されてしまうおそれがあるため，呼び出し先のアプリケーションで長時間掛かる処理はしないようにしてください。

障害復旧間隔は，次の式で求められる値よりも大きくしてください。

（〈呼び出し先のアプリケーションの処理時間〉 ×〈最大作業件数〉）÷〈WorkManagerのスレッド数〉

また，アプリケーション呼び出しの実行中に J2EE サーバを強制停止した場合，J2EE サーバの再開始後，障害復旧間隔が経過するまでアプリケーションの呼び出しが行われないおそれがあります。このため通常の運用では J2EE サーバを強制停止したり，障害復旧間隔を大きくし過ぎたりしないでください。

図 1-108　REST アプリケーションの処理時間が障害復旧間隔よりも長い場合の動作



1.8.10　呼び出し処理のタイムアウト

呼び出し先が REST アプリケーションの場合，アプリケーション呼び出しサービスと REST アプリケーションの間の RESTful Web サービスの通信タイムアウトを設定できます。

通信タイムアウトの設定方法を，優先順位が高い順に示します。

1. アプリケーション呼び出し情報ファイルでの設定

   アプリケーション呼び出し情報ファイルでタイムアウトを設定する場合は，ref 識別子ごとに異なるタイムアウト値を指定できます。アプリケーション呼び出し情報ファイルでの設定方法については，「15.3.14　REST アプリケーションの呼び出しの場合にファイルに指定する内容」の「(6)　rest.request.read.timeout」と「(7)　rest.request.connect.timeout」を参照してください。

2. Cosminexus の J2EE サーバ単位の通信タイムアウトの設定

   Cosminexus の J2EE サーバ単位の通信タイムアウトを設定する場合は，J2EE サーバ内の RESTful Web サービスの通信で，最も長くしたいタイムアウト値を設定してください。

   Cosminexus の J2EE サーバ単位の通信タイムアウトの設定方法については，マニュアル『Cosminexus アプリケーションサーバ 機能解説 基本・開発編(Web コンテナ)』の「J2EE サーバ単位の通信タイム

アウトの設定」およびマニュアル『Cosminexus アプリケーションサーバ リファレンス 定義編(サーバ定義)』の「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。

注意事項

通信タイムアウト（クライアントソケットの接続タイムアウト，またはクライアントソケットの読み込みタイムアウト）の設定値が，0 または指定なしの場合，REST アプリケーションが無応答になると，アプリケーション呼び出しサービスは応答を待ち続けます。そのため，上記の 1.または 2.の方法で，通信タイムアウトに 0 以外の値を設定してください。

1.8.11　アプリケーション呼び出しの一時抑止

アプリケーション呼び出しサービスは，障害発生時にアプリケーション呼び出しを一時的に抑止し，障害への対処後に一時抑止を解除できます。

アプリケーション呼び出しを一時抑止した場合の BPMN 要素の動作を，次の表に示します。

表 1-23　アプリケーション呼び出しを一時抑止した場合の BPMN 要素の動作

BPMN 要素	一時抑止した場合の動作
サービスタスクまたはビジネスルールタスク	REST アプリケーションまたは Java オブジェクトを呼び出しません。
メッセージイベント	メッセージをスローしません。
エラーイベント	エラーをスローしません。
タイマーイベント	イベント発火をしません。

また，一時抑止すると，解除するまで，リトライした回数（タイマーを除く BPMN 要素の場合）や，実行回数（タイマーの場合）がカウントアップされることはありません。

一時抑止と解除は次に示す単位で行えます。

- 作業単位
- アプリケーション呼び出し制御情報単位
- ビジネスプロセス定義単位

(1)　作業単位の一時抑止と解除

作業単位の一時抑止と解除は，開始（タイマー）を除く BPMN 要素に適用できます。

作業単位でアプリケーション呼び出しを一時抑止する場合，`ciwchgapwork` コマンドを使用して，作業の状態を「作業者実行」に変更します。また，一時抑止を解除する場合，作業の状態を「実行開始可能」に戻します。

実行例を次に示します。

一時抑止の実行例

1. ciwchgapwork コマンドを実行して「実行開始可能」状態の作業の情報を取得します。

   ciwchgapwork コマンドの指定形式を次に示します。

   ```
   ciwchgapwork -sid システムID -list -ope -s READY > ファイルパス
   ```

   出力結果はファイルにリダイレクトします。

2. 手順 1.でリダイレクトしたファイル中の，アプリケーション呼び出しを一時抑止したい作業の行だけを残して，それ以外の行は削除するか，コメント行に変更します。

3. ciwchgapwork コマンドを実行して，作業を「作業者実行」状態に変更します。

   ciwchgapwork コマンドの指定形式を次に示します。-f オプションには，手順 2.で編集したファイルを指定してください。

   ```
   ciwchgapwork -sid システムID -chg -s PERFORMING -f ファイルパス
   ```

一時抑止の解除の実行例

1. ciwchgapwork コマンドを実行して，作業を「実行開始可能」状態に戻します。

   ciwchgapwork コマンドの指定形式を次に示します。

   ```
   ciwchgapwork -sid システムID -chg -s READY -f ファイルパス
   ```

   -f オプションには，「一時抑止の実行例」の手順 2.で編集したファイルを指定してください。

ciwchgapwork コマンドの詳細については，マニュアル『uCosminexus Service Coordinator Interactive Workflow コマンド』を参照してください。

(2)　アプリケーション呼び出し制御情報単位の一時抑止と解除

アプリケーション呼び出し制御情報単位の一時抑止と解除は，次に示す単位で，どの BPMN 要素に対しても適用できます。

- ref 識別子単位
- グループ単位
- ref 識別子共通設定単位

アプリケーション呼び出し制御情報単位でアプリケーション呼び出しを一時抑止する場合，ciwmngap コマンドを使用して最大作業件数に「0」を指定します。また，一時抑止を解除する場合は，最大作業件数を元の値に戻します。

実行例を次に示します。

一時抑止の実行例

1. アプリケーション呼び出し制御情報ファイルを作成します。一時抑止する対象（以下の例では ref 識別子またはグループ）の最大作業件数に「0」を指定します。
   - アプリケーション呼び出し制御情報ファイルの例（ref 識別子単位の場合）

     ```
     U,ope,ref01,300,300,0,0,1500
     ```

   - アプリケーション呼び出し制御情報ファイルの例（グループ単位の場合）

     ```
     U,grp,group01,300,300,0,0,1500
     ```

2. ciwmngap コマンドを実行します。-apdf オプションには，手順 1.のファイルを指定します。

   ciwmngap コマンドの指定形式を次に示します。

   ```
   ciwmngap -sid システムID -chg -apdf アプリケーション呼び出し制御情報ファイル
   ```

一時抑止の解除の実行例

1. アプリケーション呼び出し制御情報ファイルを作成します。一時抑止を解除する対象（以下の例では ref 識別子またはグループ）の最大作業件数に元の値（以下の例では 10000）を指定します。
   - アプリケーション呼び出し制御情報ファイルの例（ref 識別子単位の場合）

     ```
     U,ope,ref01,300,300,0,10000,1500
     ```

   - アプリケーション呼び出し制御情報ファイルの例（グループ単位の場合）

     ```
     U,grp,group01,300,300,0,10000,1500
     ```

2. ciwmngap コマンドを実行します。-apdf オプションには，手順 1.のファイルを指定します。

   ciwmngap コマンドの指定形式を次に示します。

   ```
   ciwmngap -sid システムID -chg -apdf アプリケーション呼び出し制御情報ファイル
   ```

(3)　ビジネスプロセス定義単位の一時抑止と解除

ビジネスプロセス定義単位の一時抑止と解除は，開始（タイマー）に適用できます。開始（タイマー）以外の BPMN 要素には適用できません。

ビジネスプロセス定義単位でアプリケーション呼び出しを一時抑止する場合，ciwmngbp コマンドを使用して，ビジネスプロセス定義の状態を「非活性」に変更します。また，一時抑止を解除する場合，ビジネスプロセス定義の状態を「活性」に戻します。

実行例を次に示します。

一時抑止の実行例

1. ciwmngbp コマンドを実行して，ビジネスプロセス定義の状態を「非活性」に変更します。

ciwmngbp コマンドの指定形式を次に示します。

```
ciwmngbp -sid システムid -chg -bpn ビジネスプロセス定義名 -bpv ビジネスプロセス定義バージョン -s INACTIVE
```

一時抑止の解除の実行例

1. ciwmngbpコマンドを実行して，ビジネスプロセス定義の状態を「活性」に戻します。

   ciwmngbp コマンドの指定形式を次に示します。

```
ciwmngbp -sid システムid -chg -bpn ビジネスプロセス定義名 -bpv ビジネスプロセス定義バージョン -s ACTIVE
```

1.8.12　アプリケーション呼び出し再実行のための ID 送信

アプリケーション呼び出しサービスは，呼び出し先の REST アプリケーションと CSCIW の案件，作業を紐づけできるように，アプリケーション呼び出しサービスからのリクエストで HTTP ヘッダとして案件 ID，作業 ID を送信します。

案件 ID，作業 ID を送信するかどうかは，システム ID 単位で設定できます。共通設定ファイルの AppCallServiceSendID で設定します。詳細は「15.2.5(30)　AppCallServiceSendID」を参照してください。

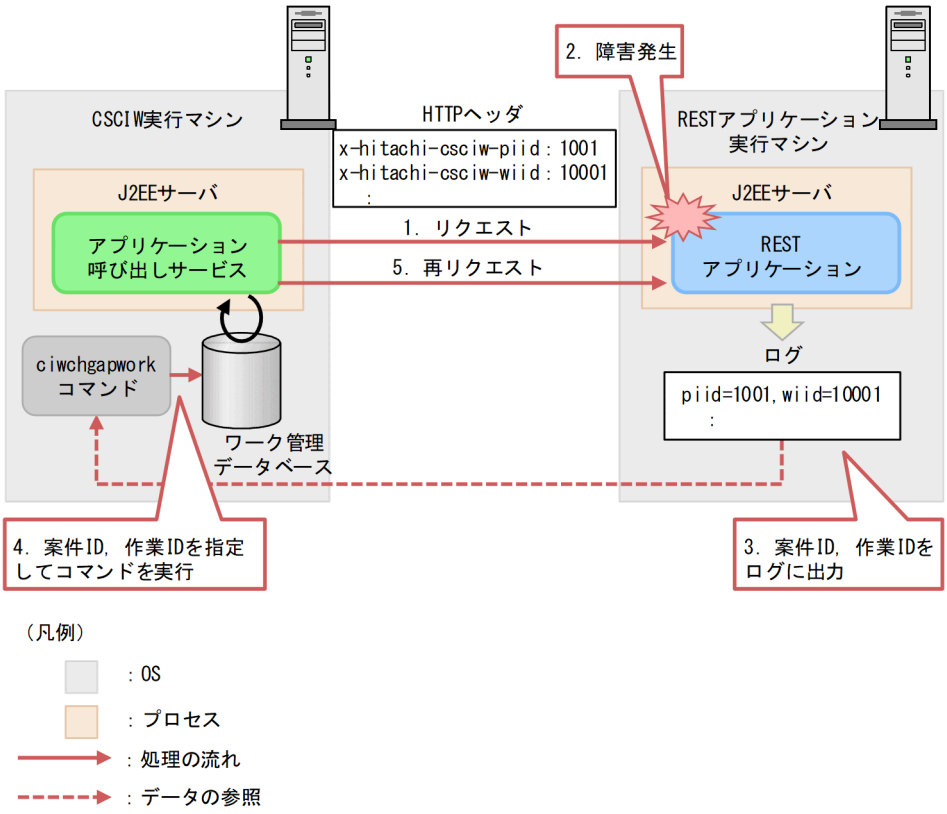
送信する HTTP ヘッダのキー名を次に示します。

- x-hitachi-csciw-piid
- x-hitachi-csciw-wiid

次の図に示すとおり，HTTP ヘッダの案件 ID，作業 ID をログに出力しておくと，呼び出し先の REST アプリケーションで障害が発生した際に，どの案件のリクエストで障害が発生したかの調査に利用できます。また，ログに出力した案件 ID，作業 ID をciwchgapwork コマンドの-f オプションに指定する入力ファイルに記述することで，障害が発生したアプリケーション呼び出しのリクエストを再実行できます。アプリケーション呼び出しを再実行する手順は，「10.5.1　アプリケーション呼び出しを再実行する」を参照してください。

この機能を利用する場合，「10.5.1　アプリケーション呼び出しを再実行する」にあるciwchgapwork コマンドの-list オプションの実行を省略できます。

図 1-109　HTTP ヘッダとして案件 ID，作業 ID を送信する場合の例



1.8.13　タイマーイベント使用時の注意事項

タイマーイベント使用時の，アプリケーション呼び出しサービスに関する注意事項について説明します。

(1)　タイマーイベントの発火時刻の計算方法

ここでは，タイマーイベントの発火時刻の再計算に関する注意事項について説明します。

アプリケーション呼び出しサービスは，タイマーイベントのイベントを発火したあと，実行回数が最大実行回数に達していない場合，次回のイベント発火時刻を再計算※します。

注※

イベント発火時刻の再計算とは，現在時刻を基点にタイマールールを再評価して次回のイベント発火時刻を求めることです。

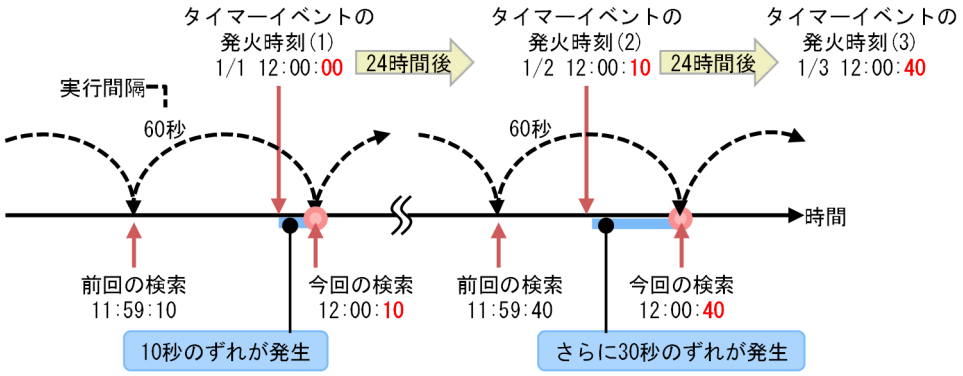
タイマールールが間隔方式の場合，タイマーイベントの次回の発火時刻は，アプリケーション呼び出しサービスが実際にイベント発火を行った時刻を基点として再計算します。

ただし，タイマーイベントの発火時刻と，実際にイベント発火が行われる時刻との間では，最大でアプリケーション呼び出し制御情報の実行間隔分のずれが発生することがあります。このずれは，タイマーイベントの発火を繰り返すことで蓄積します。

実行間隔とタイマーイベントの発火時刻との関係の例を次の図で示します。

図 1-110　発火時刻と実行間隔のずれ（間隔方式の場合）

●タイマールールは「24時間間隔」



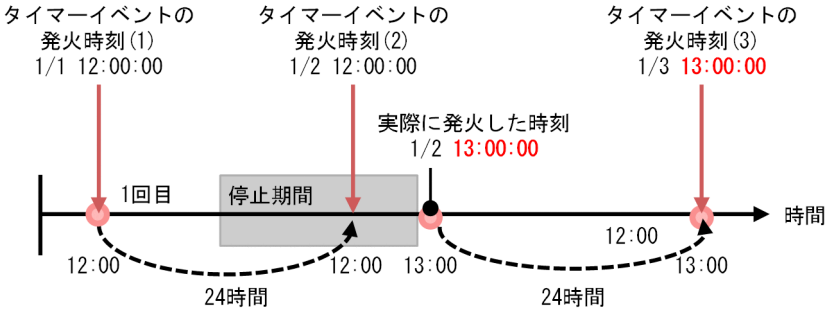
[説明]

1. 図中の「タイマーイベントの発火時刻(1)」には 2018/1/1 12:00:00 が指定されています。しかし，アプリケーション呼び出しサービスによる作業の検索および発火が実際に行われた時刻は，12:00:10 です。
2. 「タイマーイベントの発火時刻(2)」は，前回イベント発火が行われた時刻を基準として計算するため，2018/1/2 12:00:10 となり，前回の発火時刻と比べて 10 秒遅くなります。
3. 2. によって，「タイマーイベントの発火時刻(2)」には 2018/1/2 12:00:10 が指定されています。しかし，アプリケーション呼び出しサービスによる作業の検索と発火が実際に行われる時刻は 12:00:40 になります。
4. 「タイマーイベントの発火時刻(3)」は，前回イベント発火が行われた時刻を基準として計算するため，2018/1/3 12:00:40 となり，前回の発火時刻と比べて 30 秒遅くなります。

また，アプリケーション呼び出しサービスの停止中にタイマーイベントの発火時刻を過ぎた場合，イベント発火時刻に大きなずれが発生することがあります。アプリケーション呼び出しサービスの停止によって「タイマーイベント発火時刻(3)」が 1 時間遅くなる例を，次の図に示します。

図 1-111　アプリケーション呼び出しサービスの停止によるイベント発火時刻のずれ（間隔方式の場合）

●タイマールールは「24時間間隔」

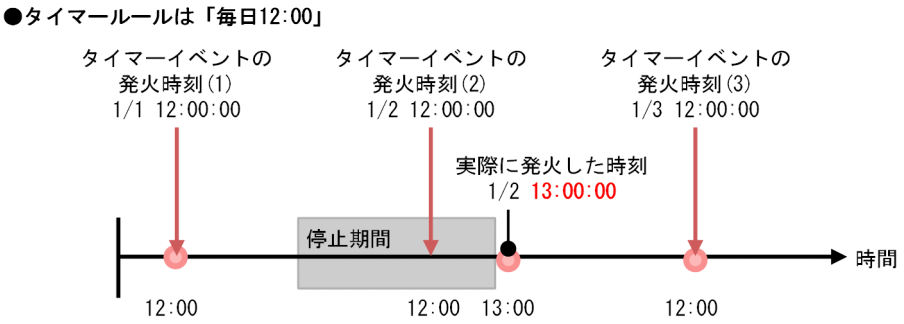


［説明］

1. 図中の「タイマーイベントの発火時刻(1)」には 2018/1/1 12:00:00 が指定されています。発火が実際に行われた時刻は，12:00:00 です。
2. 「タイマーイベントの発火時刻(2)」は，前回イベント発火が行われた時刻を基準として計算するため，2018/1/2 12:00:00 となります。しかし，アプリケーション呼び出しサービスが停止しているため，実際に発火した時刻は，停止期間が過ぎた 2018/1/2 13:00:00 となります。
3. 「タイマーイベントの発火時刻(3)」は，前回イベント発火が行われた時刻を基準として計算するため，2018/1/3 13:00:00 となり，発火時刻が 1 時間遅くなります。

発火時刻のずれを防ぐには，タイマールールを定期日時方式で指定する必要があります。定期日時方式の場合の計算方法を，次の図に示します。

図 1-112　発火時刻の計算方法（定期日時方式の場合）



［説明］

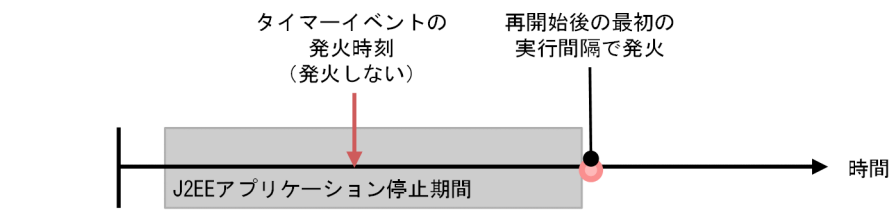
タイマールールが定期日時方式の場合，間隔方式のようなイベント発火時刻のずれは発生しません。例えば，タイマールールが「毎日 12 時」の場合，実際のイベント発火時刻が 1 時間遅れの 13 時になっても，次のイベント発火時刻は翌日の 12 時となります。

(2)　J2EE アプリケーション停止中にイベント発火時刻を過ぎた場合

アプリケーション呼び出しサービスが停止している期間にタイマーイベントの発火時刻を過ぎた場合の動作について説明します。

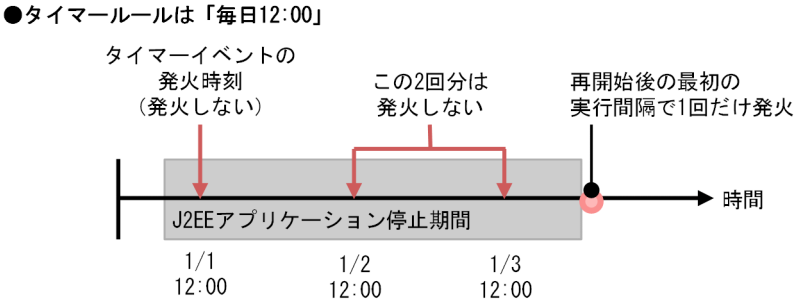
アプリケーション呼び出しサービスが停止している期間にタイマーイベントの発火時刻を過ぎた場合，アプリケーション呼び出しサービスの再開始後，最初の実行間隔で，発火時刻を過ぎたタイマーイベントの発火を行います。J2EE アプリケーション停止中に発火時刻が過ぎた場合の例を，次の図で示します。

図 1-113　J2EE アプリケーション停止中に発火時刻が過ぎた場合（実行回数が 1 回のとき）



最大実行回数が 2 回以上のタイマールールの場合でも，停止していた期間のタイマーイベントを再開始時に複数回まとめては発火しません。例えば，毎日 12:00 に発火するタイマーイベントに案件が遷移した状態でアプリケーション呼び出しサービスを 3 日間停止した場合，アプリケーション呼び出しサービスの再開始後にはタイマーイベントを 1 回だけ発火します。J2EE アプリケーション停止中に発火時刻が複数回過ぎた場合の例を次の図で示します。

図 1-114　J2EE アプリケーション停止中に発火時刻が過ぎた場合（実行回数が複数回のとき）

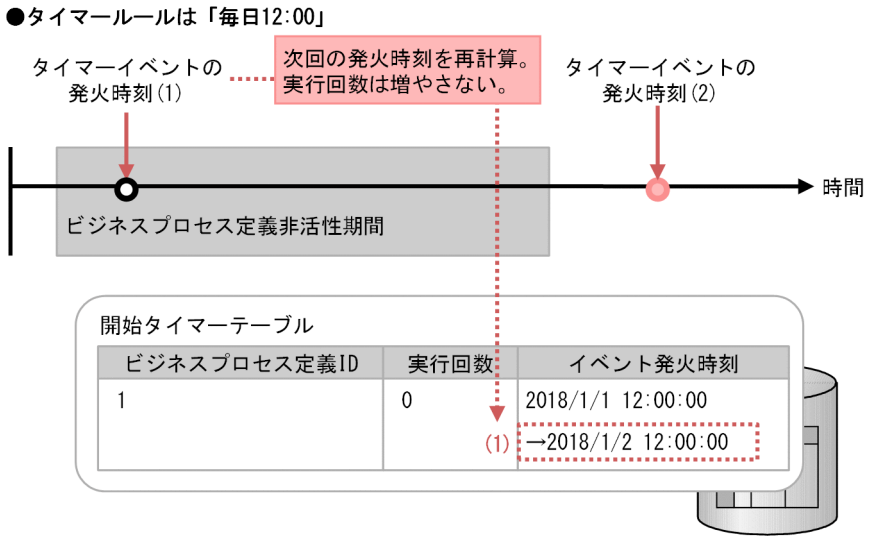


(3)　ビジネスプロセス定義が「非活性」状態中に開始（タイマー）の発火時刻を過ぎた場合

開始（タイマー）イベントを定義したビジネスプロセス定義が「非活性」状態の期間に，タイマーイベントの発火時刻を過ぎた場合の動作について説明します。

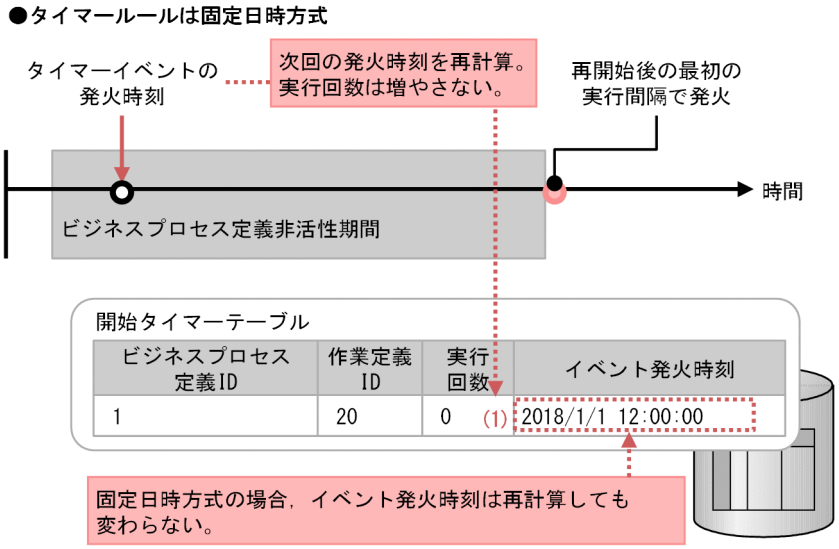
タイマールールが定期日時方式または間隔方式の場合に，「非活性」状態の期間中にイベント発火時刻が過ぎたときの図を次に示します。この場合，イベント発火時刻を過ぎても発火しないで，次回のイベント発火時刻の再計算だけが行われます（図中の(1)）。この再計算時，タイマーの実行回数は増やしません。そして，ビジネスプロセス定義を活性化したあと，次のイベント発火時刻を過ぎた時に発火します（図中の(2)）。

図 1-115　「非活性」状態中に実行時刻が過ぎた場合（定期日時方式／間隔方式）



タイマールールが固定日時方式の場合は，タイマーイベントの発火時刻は再計算しても変わりません。そのため，ビジネスプロセス定義を活性化したあと，最初の実行間隔でイベント発火が行われます。タイマールールが固定日時方式で，「非活性」状態中に実行時刻が過ぎた場合の動作を次の図で示します。

図 1-116　非活性化中に実行時刻が過ぎた場合（固定日時方式）



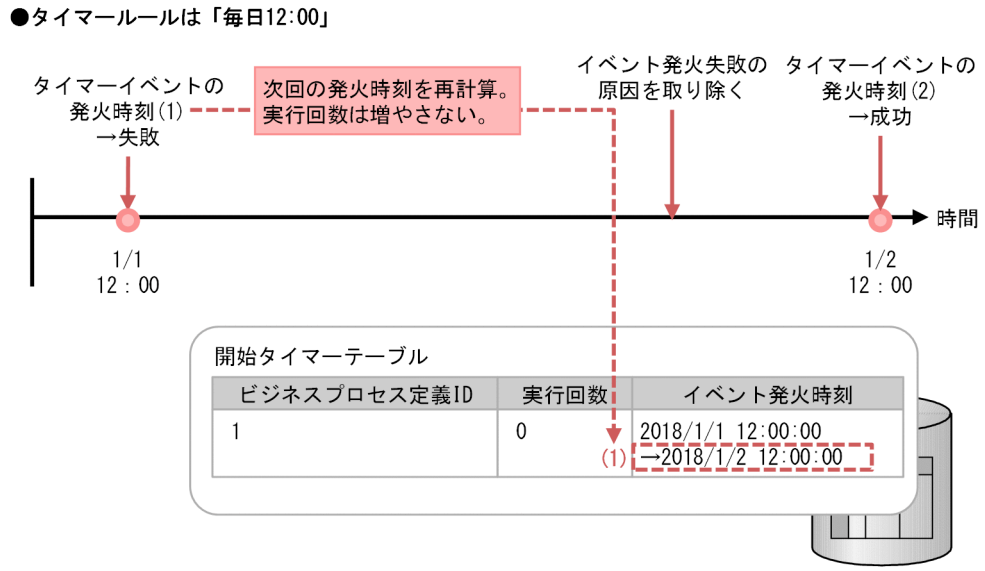
(4)　開始（タイマー）イベントの発火失敗時の動作

開始（タイマー）イベントの発火失敗時の動作について説明します。

タイマールールが間隔方式または定期日時方式の場合

間隔方式または定期日時方式のタイマールールが指定された開始（タイマー）イベントの発火が失敗した場合について説明します。ここでは，定期日時方式の場合の例を図で示します。開始（タイマー）イベントの発火が失敗した場合，アプリケーション呼び出しサービスはタイマーイベントの発火時刻を再計算します（図中(1)）。イベント発火失敗の原因を取り除き，タイマーイベントの発火時刻を再び過ぎたあと，案件の再投入を試みます（図中(2)）。その際，タイマーイベントの発火に失敗した回数は，タイマーの実行回数には含まれません。

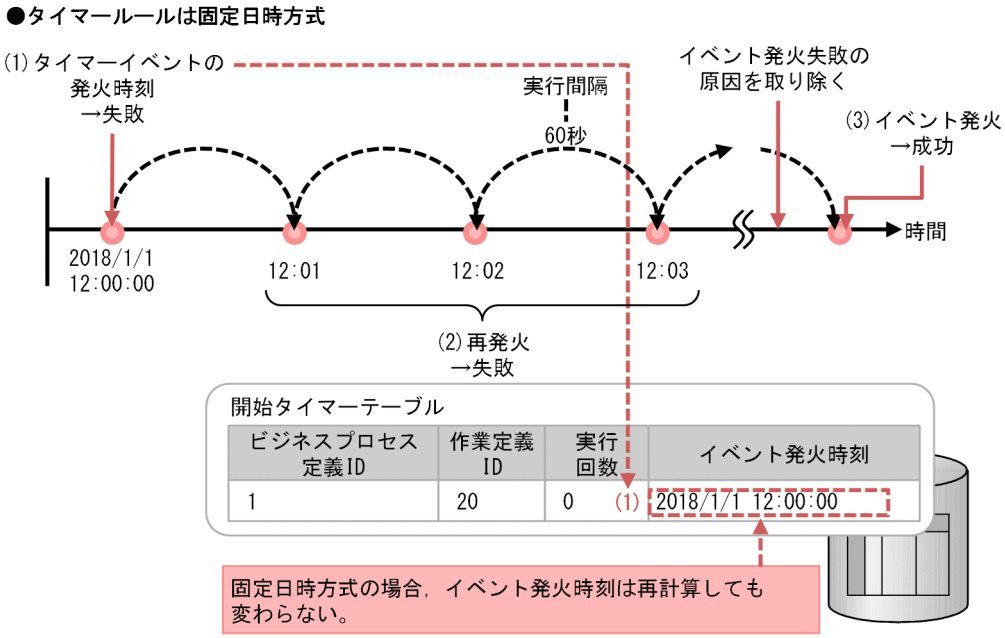
図 1-117　タイマーによる案件投入の失敗と回復（定期日時方式の場合）



タイマールールが固定日時方式の場合

固定日時方式のタイマールールが指定された開始（タイマー）イベントの発火が失敗したときについて，次の図で説明します。開始（タイマー）イベントの発火が失敗した場合，タイマーイベントの発火時刻は再計算しても変わりません（図中(1)）。このため，アプリケーション呼び出し制御情報に指定した実行間隔の経過後，アプリケーション呼び出しサービスは再びイベント発火を試みます（図中(2)）。この動作は，失敗した原因を取り除いてイベント発火が成功するまで繰り返されます（図中(3)）。

図 1-118　タイマーによる案件投入の失敗と回復（固定日時方式の場合）



メモ

タイマーによる案件投入の失敗は，案件投入の延長で次のような問題がある場合にが発生します。

- 分岐条件の評価でエラーが発生した
- コールアクティビティ情報ファイルの読み込み，または解析でエラーが発生した
- マルチインスタンスの完了条件（completionCondition）または繰り返し回数（loopCardinality）の評価でエラーが発生した

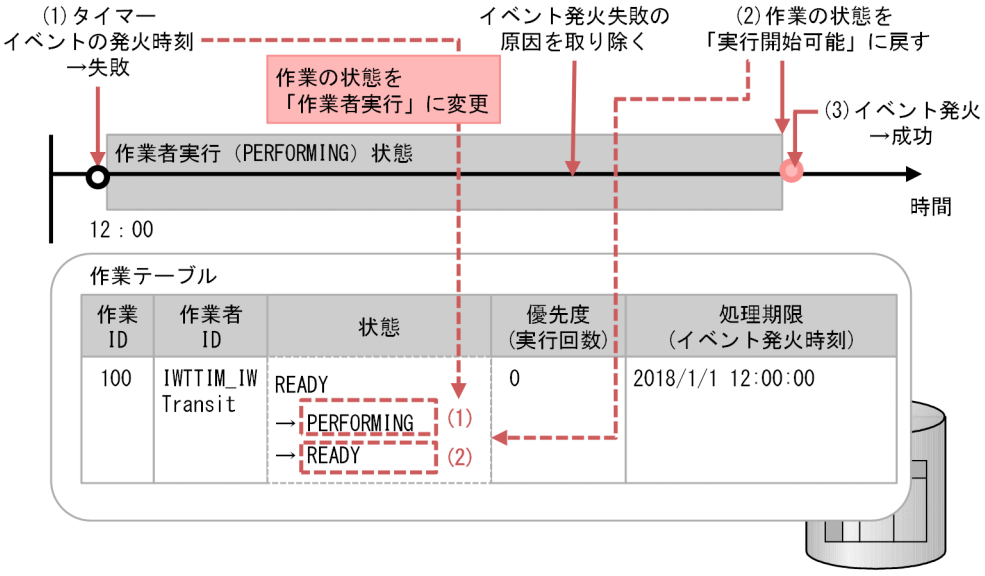
(5) 開始（タイマー）以外のタイマーイベントの発火失敗時の動作

開始（タイマー）以外のタイマーイベントの発火失敗時の動作について説明します。

開始（タイマー）以外のタイマーイベントの発火が失敗した場合について，次の図で説明します。イベントの発火が失敗した場合，アプリケーション呼び出しサービスはリトライしないで，作業の状態を「作業者実行（PERFORMING）」に変更します（図中(1)）。「作業者実行」状態のタイマーイベントは，アプリケーション呼び出しサービスによるイベント発火の対象外となります。タイマーイベントの発火に失敗した回数は，タイマーの実行回数には含めません。

案件の遷移が失敗した原因を取り除いて，作業の状態を「作業者実行」から「実行開始可能（READY）」に変更すると（図中(2)），タイマーイベントは再びアプリケーション呼び出しサービスによるイベント発火の対象となります。そして，次回の実行間隔でイベント発火が行われます（図中(3)）。

図 1-119　タイマーイベントの発火失敗と回復



メモ

タイマーイベントの発火の失敗は，遷移の延長で次のような問題があった場合に発生します。

- 分岐条件の評価でエラーが発生した
- コールアクティビティ情報ファイルの読み込み，または解析でエラーが発生した

- マルチインスタンスの完了条件（completionCondition）または繰り返し回数（loopCardinality）の評価でエラーが発生した

(6)　ビジネスプロセス定義のタイマールールを変更する場合

ビジネスプロセス定義のタイマールールを変更する方法について説明します。

開始（タイマー）のタイマールールの変更

登録済みのビジネスプロセス定義の開始（タイマー）のタイマールールの変更は，サポートしていません。

開始（タイマー）のタイマールールを変更するには，ビジネスプロセス定義のバージョンを変更して新規登録する必要があります。

開始（タイマー）以外のタイマールールの変更

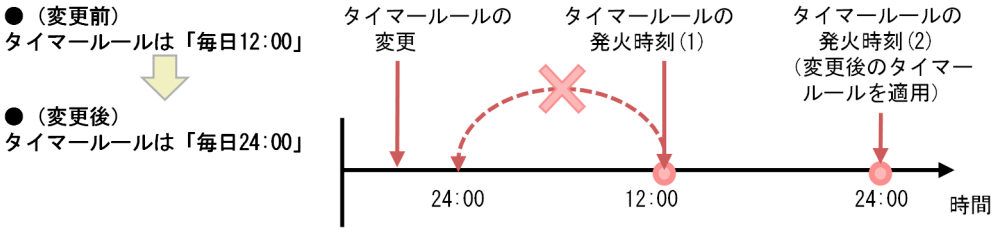
開始（タイマー）以外のタイマーイベントのタイマールールは，ciweditbp コマンドを使用して変更できます。 ただし，タイマーイベントがすでに待ち状態で，かつイベント発火時刻が決定している場合は，タイマールールを変更してもタイマーイベントの発火時刻はすぐには変更されません。

次に，タイマールールの変更，およびその適用タイミングについて説明します。

タイマールールの変更が適用されるケース

タイマールールの変更が適用される場合の例を，次の図で示します。タイマールールが定期日時方式または間隔方式で，かつ最大実行回数が 2 回以上の場合，現在設定されているイベント発火時刻（図中(1)）を経過したあと，その次のイベント発火時刻の再計算時に新しいタイマールールが適用されます（図中(2)）。

図 1-120　タイマールールの変更（最大実行回数 2 回以上）



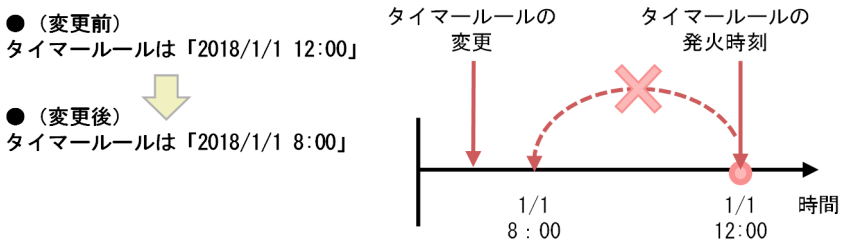
タイマールールの変更が適用されないケース

タイマールールが次のように指定されている場合，タイマーイベント発火時刻を再計算するタイミングがないため，タイマールールの変更は適用されません。

- 固定日時方式の場合
- 最大実行回数が 1 回で定期日時方式の場合
- 最大実行回数が 1 回で間隔方式の場合

タイマールールが固定日時または最大実行回数 1 回の場合の例を，次の図に示します。

図 1-121　タイマールールの変更が適用されない場合（固定日時または最大実行回数 1 回）



1.8.14　アプリケーション呼び出しサービスの使用上の注意事項

アプリケーションが CSCIW の API や BPMN 連携ライブラリの Java API を発行した延長で，アプリケーション呼び出しサービスが実行される場合，API を発行したアプリケーションがトランザクションを決着するまで，アプリケーション呼び出しサービスは該当する案件の排他待ちとなります。そのため，アプリケーションが API を利用する際は，トランザクションの開始から終了までの期間はできるだけ短くしてください。

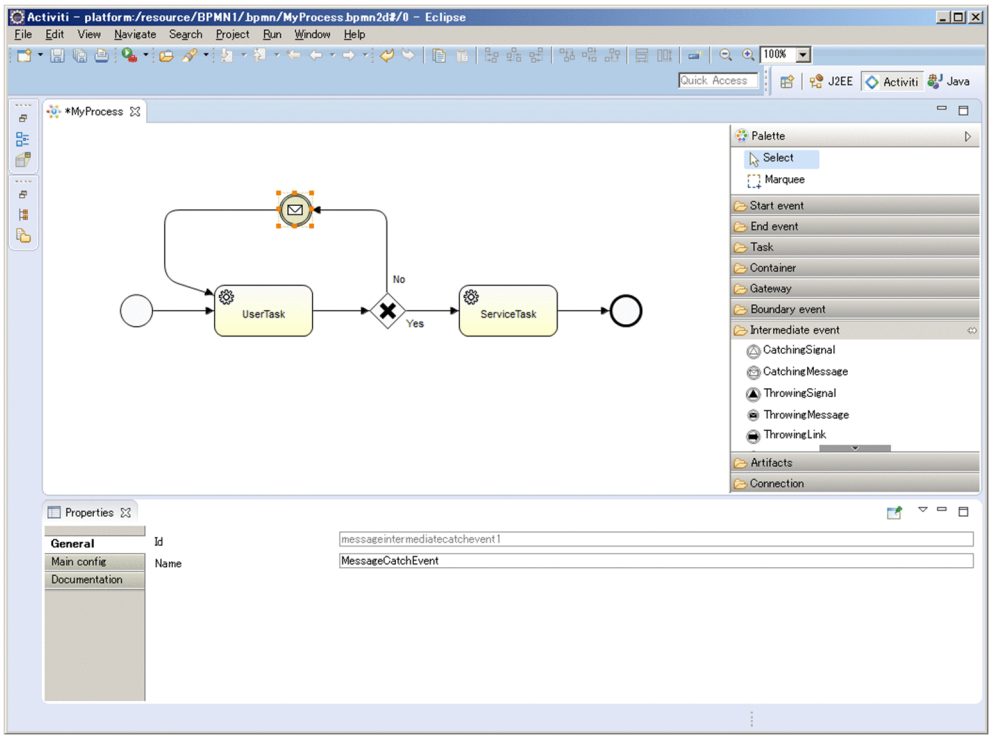
1.9　BPMN エディタとは

BPMN エディタを使用して BPMN に従ったビジネスプロセスを作成できます。

BPMN エディタの画面は，パレット，キャンバスおよびプロパティから構成されています。

- パレット：ビジネスプロセスの定義に必要な BPMN 要素を表示します。
- キャンバス：BPMN 要素を使用してビジネスプロセスを描画できます。
- プロパティ：BPMN 要素のプロパティがタブに表示され，値を設定できます。

図 1-122　BPMN エディタ



また，BPMN エディタには CSCIW-Definer が組み込まれており，次の機能が使用できます。

- ビジネスプロセス定義の登録
- ビジネスプロセス定義の削除
- ビジネスプロセス定義の一覧取得

関連項目

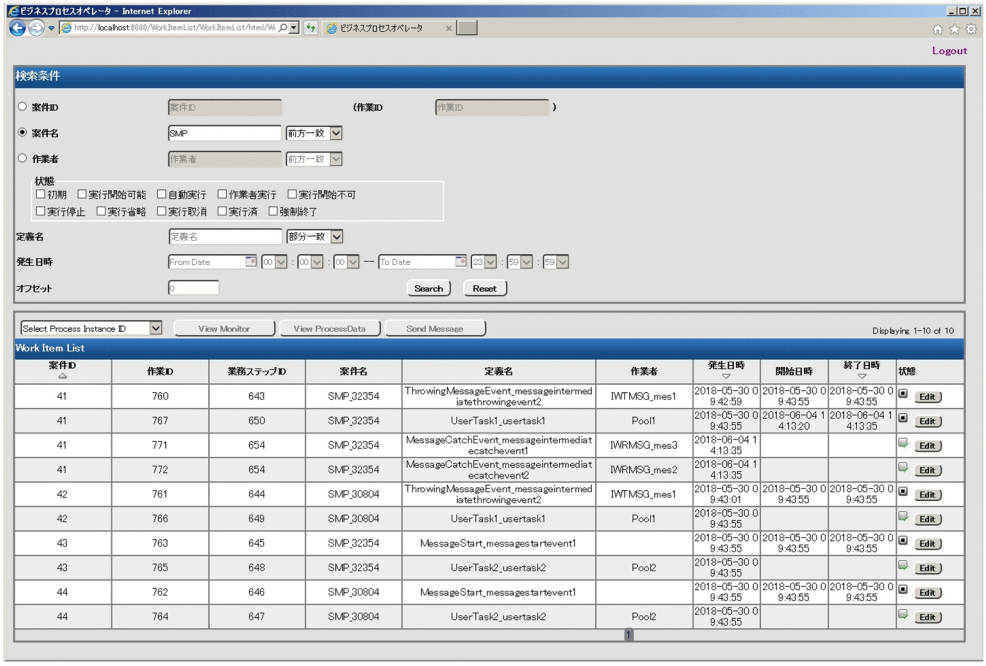
- 1.3.1　BPMN 連携機能で使用できる BPMN 要素

1.10　ビジネスプロセスオペレータとは

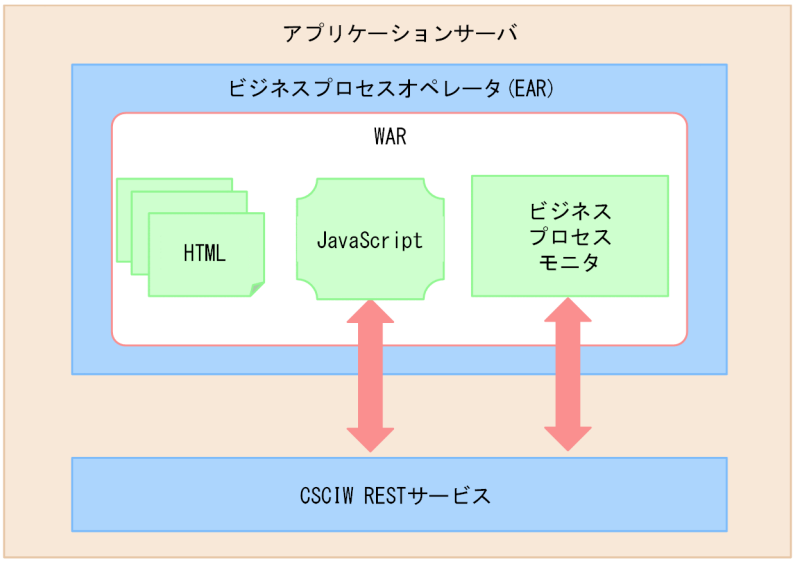
ビジネスプロセスオペレータでできることを，次に示します。

- 案件の進捗状況やプロセスデータの確認
- メッセージイベントへのメッセージ送信
- 作業状態の変更

図 1-123　ビジネスプロセスオペレータの画面



ビジネスプロセスオペレータのアーキテクチャ



（凡例）

JavaEEアプリケーションサーバ　　リクエスト / レスポンス

JavaEEアプリケーション

ビジネスプロセスオペレータは，HTML と JavaScript から構成されていて，ビジネスプロセスモニタを含みます。ビジネスプロセスオペレータは，REST サービスにアクセスすることで，案件を参照および操作します。また，ビジネスプロセスモニタによって，BPMN プロセス図や案件のステータスを表示させることができます。

ビジネスプロセスモニタについては，「付録 D　ビジネスプロセスモニタとは」を参照してください。

1.11　BPMN 連携機能の利用時に開発するアプリケーション

BPMN 連携機能の利用時に，CSCIW で開発する必要があるアプリケーションの種類を次に示します。

項番	アプリケーションの種類	必須／任意	説明
1	業務アプリケーション	必須	案件を実行，運用するためのアプリケーションです。次の 2 種類の開発方法があります。 ・REST API を使用して開発する ・Java API を使用して開発する
2	REST アプリケーション	任意	アプリケーション呼び出しサービスを使用する場合に，アプリケーション呼び出しサービスから呼び出すアプリケーションです。REST に基づいた形式で開発します。
3	Java オブジェクト	任意	アプリケーション呼び出しサービスを使用する場合に，アプリケーション呼び出しサービスから呼び出すアプリケーションです。CSCIW が提供するインタフェースを実装して開発します。

使用する API に従って，必要な章をお読みください。

関連項目

- 4.　REST API を使用して業務アプリケーションを開発する
- 5.　Java API を使用して業務アプリケーションを開発する
- 6.　アプリケーション呼び出しサービスを使用する

1.12　BPMN 要素の状態遷移モデル

BPMN 要素の状態遷移について説明します。

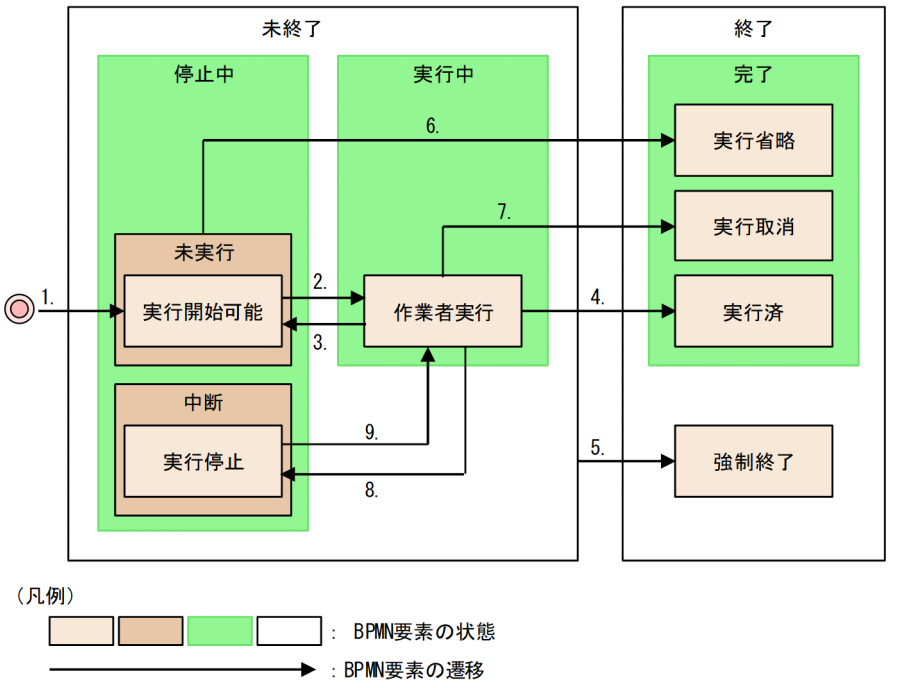
BPMN 要素ごとの，状態遷移についての説明の記載個所を次の表に示します。

表 1-24　BPMN 要素ごとの状態遷移についての説明の記載個所

項番	BPMN 要素	記載個所
1	ユーザタスク	状態遷移の詳細は次の図を参照してください。
	ユーザタスク（シーケンシャルマルチインスタンス）	
	ユーザタスク（パラレルマルチインスタンス）	
	サービスタスク	
	サービスタスク（シーケンシャルマルチインスタンス）	
	サービスタスク（パラレルマルチインスタンス）	
	ビジネスルールタスク	
	ビジネスルールタスク（シーケンシャルマルチインスタンス）	
	ビジネスルールタスク（パラレルマルチインスタンス）	
	コールアクティビティ	
	コールアクティビティ（シーケンシャルマルチインスタンス）	
	コールアクティビティ（パラレルマルチインスタンス）	
	キャッチ（メッセージ）	
	キャッチ（タイマー）	
	スロー（メッセージ）	
	終了（メッセージ）	
	終了（エラー）	
2	折りたたまれたアドホック・サブプロセス	状態遷移の詳細は「1.5.6(1)　アドホック・サブプロセスの状態遷移」を参照してください。
	展開されたアドホック・サブプロセス	
3	上記以外の BPMN 要素	状態遷移の詳細は図の下の「メモ」を参照してください。

上の表の項番 1 に分類した BPMN 要素の状態遷移を次の図に示します。

図 1-124　BPMN 要素の状態遷移



注
大きい枠で示す状態は，小さい枠で示す状態を含んでいます。
例えば，「未終了」状態は，「実行開始可能」状態，「作業者実行」状態および「実行停止」状態を含んでいます。

上の図の番号 1.～9.を付けた状態遷移がどの契機に発生するかについては，「1.12.2　BPMN 要素の状態遷移の契機」を参照してください。

メモ

この説明中の遷移「1.」「2.」などは，「図 1-123　BPMN 要素の状態遷移」の番号と対応しています。

次に示す BPMN 要素については，BPMN 要素に遷移した直後に，ワークフローシステムによって遷移 1.，2.，および 4.が同時に実行されます。そのため，状態は「実行済」だけになります。

- 開始（メッセージ）
- 開始（タイマー）
- 終了（タイプなし）
- 境界中断（メッセージ）
- 境界非中断（メッセージ）
- 境界中断（エラー）
- 境界中断（タイマー）
- 境界非中断（タイマー）
- イベント・サブプロセス中断開始（メッセージ）

- イベント・サブプロセス非中断開始（メッセージ）
- イベント・サブプロセス中断開始（エラー）
- イベント・サブプロセス中断開始（タイマー）
- イベント・サブプロセス非中断開始（タイマー）

次に示す BPMN 要素については，BPMN 要素に遷移した直後に，ワークフローシステムによって遷移 1.，5.が同時に実行されます。そのため，状態は「強制終了」だけになります。

- 強制終了

次に示す BPMN 要素については，BPMN 要素に遷移しても，作業インスタンスが作成されないため，状態遷移しません。

- 開始（タイプなし）
- 排他ゲートウェイ
- 並列ゲートウェイ
- 排他イベントゲートウェイ
- キャッチ（リンク）
- スロー（リンク）
- イベント・サブプロセス
- 折りたたまれたサブプロセス
- 展開されたサブプロセス
- 折りたたまれたサブプロセス（シーケンシャルマルチインスタンス）
- 折りたたまれたサブプロセス（パラレルマルチインスタンス）
- 展開されたサブプロセス（シーケンシャルマルチインスタンス）
- 展開されたサブプロセス（パラレルマルチインスタンス）

1.12.1　BPMN 要素ごとに異なる意味を持つ状態

BPMN 要素ごとに異なる意味を持つ状態について，次の表に示します。

表 1-25　BPMN 要素ごとに異なる意味を持つ状態

状態	説明
実行開始可能	• コールアクティビティの場合 子案件が投入され，子案件を実行している状態を示します。 • キャッチイベントの場合

状態	説明
実行開始可能	イベント受信待ちの状態を示します。 • サービスタスク，ビジネスルールタスク，メッセージイベント，エラーイベント，タイマーイベントの場合 アプリケーション呼び出しサービスの処理対象になった状態を示します。※ • 上記以外の場合 BPMN 要素に対応する処理を実行できる状態を示します。
作業者実行	• サービスタスク，ビジネスルールタスク，メッセージイベント，エラーイベント，タイマーイベントの場合※ アプリケーション呼び出しサービスの処理でリトライ回数分の障害が発生し，かつ処理対象から外れた状態を示します。 • タイマーイベントの場合 アプリケーション呼び出しサービスの処理で障害が発生し，かつ処理対象から外れた状態を示します。※ • 上記以外の場合 BPMN 要素に対応する処理を実行している状態を示します。

注※

アプリケーション呼び出しサービスの呼び出し対象となる BPMN 要素については，「1.8.1　呼び出し対象の BPMN 要素」を参照してください。

1.12.2　BPMN 要素の状態遷移の契機

ここでは，「図 1-123　BPMN 要素の状態遷移」の番号 1.～9.を付けた状態遷移がどの契機に発生するかを次の表に示します。

表 1-26　BPMN 要素の状態遷移の発生契機

遷移する契機	「図 1-123　BPMN 要素の状態遷移」で付与している遷移の番号								
	1.	2.	3.	4.	5.	6.	7.	8.	9.
ワークフローシステムの制御※1	○	○	－	○	○	○	○	－	－
アプリケーション呼び出しサービス※2	－	○	－	○	－	－	－	－	－
ビジネスプロセスオペレータ	－	○	○	○	－	－	－	○	○
Java API	※3								
REST API	※3								
ciwchgapwork コマンド	※3								

（凡例）

○：遷移する

－：遷移しない

注※1

ワークフローシステムの制御による状態遷移の詳細を次に示します。

遷移 1.：BPMN 要素に遷移したとき

遷移 2.，4.が同時に遷移：コールアクティビティの場合に子案件が完了したとき，またはキャッチイベントの場合にイベントを受信したとき

遷移 5.：強制終了イベント，または境界中断イベントが実行されたとき

遷移 6.：マルチインスタンスの場合に完了条件を満たしたとき

遷移 7.：マルチインスタンスの場合に完了条件を満たしたとき

注※2

アプリケーション呼び出しサービスによる状態遷移の詳細を次に示します。

遷移 2.，4.が同時に遷移：呼び出し処理が完了したとき

遷移 2.：呼び出し処理で障害が発生し，かつ処理対象から外れたとき

注※3

REST API，Java API，および`ciwchgapwork` コマンドで発生する遷移については，次の記述を参照してください。

- 「11.5.22　作業の着手」
- 「11.5.23　作業の完了」
- 「11.5.25　作業者を変更して着手」
- 「11.5.26　作業を着手して完了」
- 「11.5.27　作業の状態の変更」
- 「11.5.28　条件に一致する作業の作業者割り当てと着手」
- 「11.5.29　作業の返却」
- 「12.4.7　changeStateWI（作業の状態変更）」
- 「12.4.8　completeWI（作業の完了）」
- 「12.4.9　performAndCompleteWI（作業の着手と完了）」
- 「12.4.10　performWI（作業の着手）」
- 「12.4.11　reassignAndPerformWI（作業の作業者変更および作業の着手）」
- 「12.4.13　allocateWIEx（指定した条件に一致する作業の着手）」
- 「12.4.14　freeWI（作業の返却）」

- マニュアル『uCosminexus Service Coordinator Interactive Workflow コマンド』の「ciwchgapwork（アプリケーション呼び出し作業の状態変更)」

2 ビジネスプロセスを開発する

開発環境でのビジネスプロセスの開発について説明します。

2.1　ビジネスプロセスの開発の流れ

ビジネスプロセスの開発手順および削除の手順を流れ図で示します。

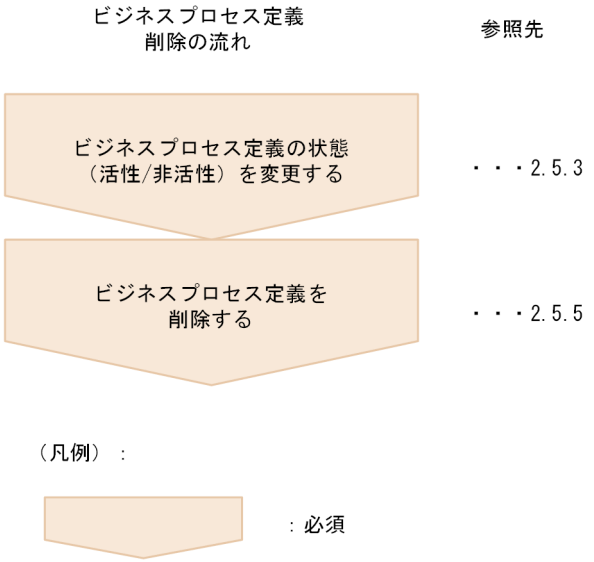
なお，図中の数字は，対応する内容を説明している節，または項の番号を表しています。

ビジネスプロセス定義を作成して登録するまでの流れ

図 2-1　ビジネスプロセス定義の作成から登録までの流れ

ビジネスプロセス定義を削除する流れ

図 2-2　ビジネスプロセス定義を削除する流れ



登録済みのビジネスプロセス定義を変更する流れ

登録済みのビジネスプロセス定義を変更する流れについては，「2.6.3　ビジネスプロセス定義の変更の流れ」を参照してください。

エディタのアンインストール

エディタのアンインストールについては，「2.8　BPMN エディタをアンインストールする」を参照してください。

関連項目

- 2.2　BPMN エディタをインストールする
- 2.3　接続先を設定する
- 2.3.1　BPMN エディタから CSCIWManagementServer にログインする
- 2.4.1　ビジネスプロセスを作成する
- 2.4.2　アプリケーション呼び出し情報ファイルを作成する
- 2.4.3　コールアクティビティ情報ファイルを作成する
- 2.4.4　BPMN ビジネスプロセス定義ファイルを CSCIW のビジネスプロセス定義ファイルに変換する
- 2.5.1　実行環境に定義ファイルを転送する
- 2.5.2　ビジネスプロセス定義を登録する
- 2.5.3　ビジネスプロセス定義の状態（活性/非活性）を変更する
- 2.5.4　ビジネスプロセス定義をバージョンアップする
- 2.5.5　ビジネスプロセス定義を削除する

2.2 BPMN エディタをインストールする

Eclipse に BPMN エディタをインストールします。

前提条件

次に示すソフトウェアをインストールしている。

- uCosminexus Developer
- Eclipse
  Eclipse セットアップ機能が実行済みである必要があります。Eclipse セットアップ機能については，マニュアル『Cosminexus アプリケーションサーバ アプリケーション開発ガイド』を参照してください。
- uCosminexus Service Coordinator Interactive Workflow
- uCosminexus Business Process Developer

ヒント

Activiti BPMN Designer のインストール時に，関連するプラグインも自動でダウンロードおよびインストールされます。そのため，設定画面［有効なソフトウェア・サイト］で有効になっているサイトにアクセスできる環境で，インストールを実施してください。なお，サイトにアクセスする際にプロキシサーバを経由する場合は，Eclipse の設定画面［一般］-［ネットワーク接続］でプロキシの設定を実施してください。

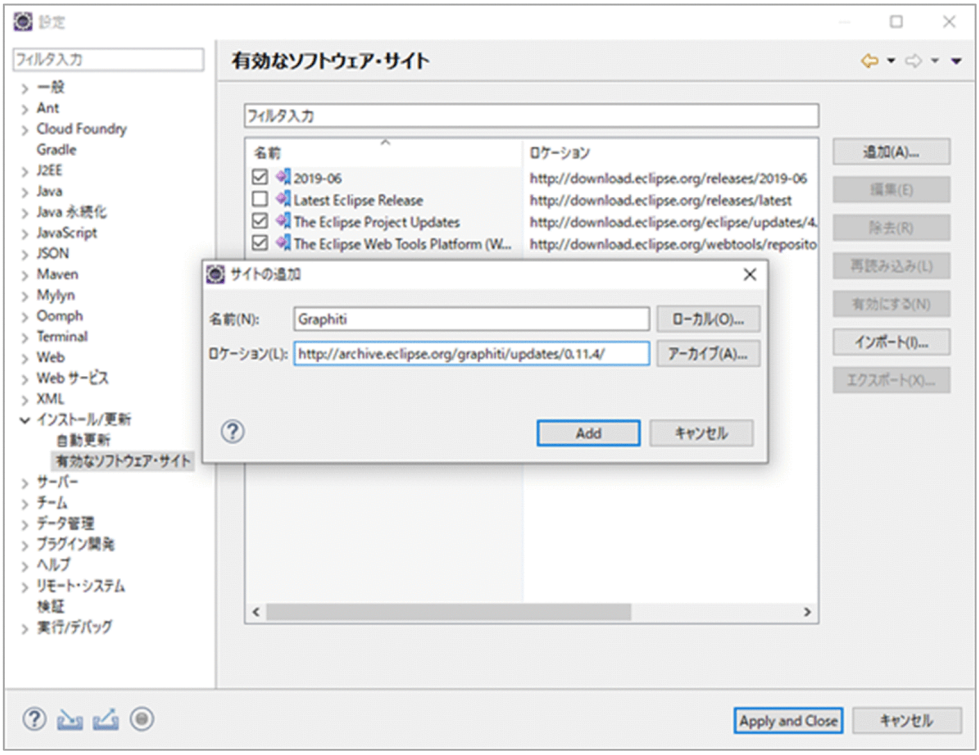
サイトにアクセスできない環境でインストールする場合は，事前に次に示すプラグインをダウンロードし，Activiti BPMN Designer と一緒にインストールしてください。

- Graphiti 0.11.4
- EMF Model Transaction Runtime 1.8.0 以降
- EMF Validation Framework Runtime 1.8.0 以降

操作手順

1. Graphiti のソフトウェアサイトを設定する。
   Eclipse を起動し，設定画面［有効なソフトウェア・サイト］を開きます。［追加］ボタンをクリックし，次の内容で Graphiti のサイトを追加します。

図 2-3　設定画面［有効なソフトウェア・サイト］



- ［名前］

  任意の名称を入力する。

- ［ロケーション］

  http://archive.eclipse.org/graphiti/updates/0.11.4/を入力する。

2. EMF のソフトウェアサイトを設定する。

手順 1 と同様の手順で，EMF Model Transaction および EMF Validation Framework のサイトを追加します。［ロケーション］には次に示すサイトを入力してください。

- EMF Model Transaction

  http://download.eclipse.org/modeling/emf/transaction/updates/releases/

- EMF Validation Framework

  http://download.eclipse.org/modeling/emf/validation/updates/releases/

3. uCosminexus Business Process Developer の eclipse フォルダを設定する。

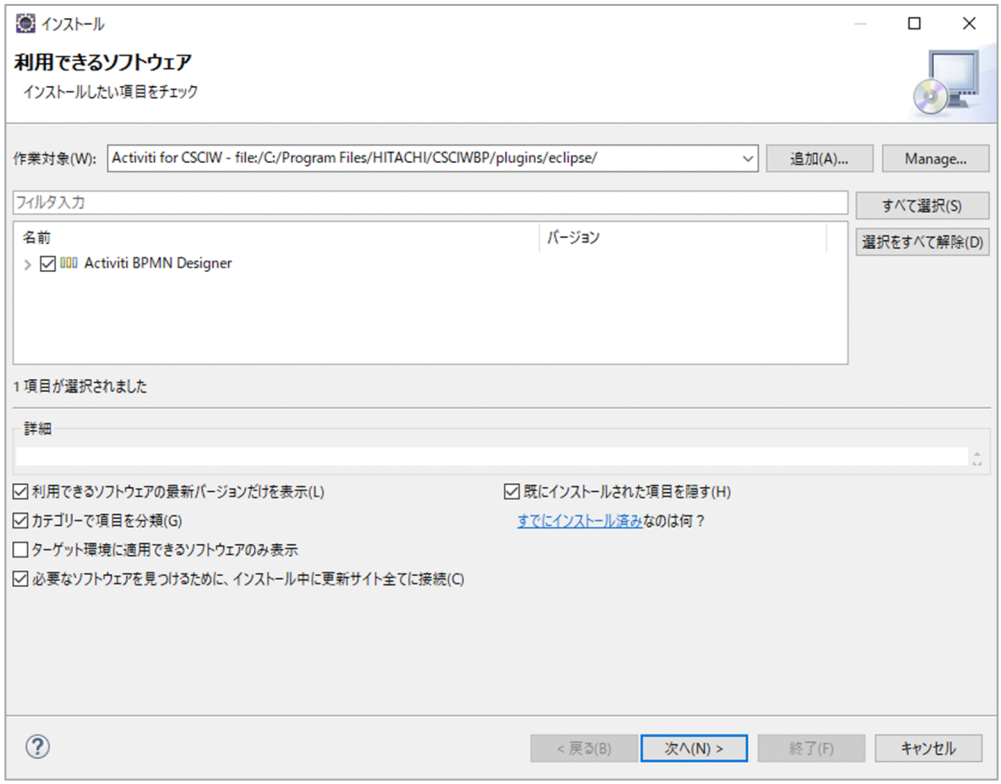
手順 1 と同様の手順で，uCosminexus Business Process Developer のeclipse フォルダを，［有効なソフトウェアサイト］に登録します。［ロケーション］には次に示すディレクトリを入力します。

- file:<uCosminexus Business Process Developerのインストールディレクトリ>/plugins/eclipse/

4. Eclipse に BPMN エディタをインストールする。

メニュー［ヘルプ］－［新規ソフトウェアのインストール］から，画面［利用できるソフトウェア］を開きます。［作業対象］で，手順 3 で追加したサイトを選択し，［Activiti BPMN Designer］にチェックを入れて［次へ］ボタンをクリックし，インストールを進めます。

図 2-4　画面［新規ソフトウェアのインストール］



インストール中に，署名のないコンテンツ※をインストールする旨のメッセージが表示されますが，インストールを継続してください。また，Eclipse Foundation の証明書を確認するメッセージが表示されますが，チェックを入れてインストールを進めてください。

注※

次のファイルが対象になります。

- org.activiti.designer.eclipse_5.18.0.xxxxxxxxxxxx.jar
- org.activiti.designer.feature_5.18.0.xxxxxxxxxxxx.jar
- org.activiti.designer.gui_5.18.0.xxxxxxxxxxxx.jar
- org.activiti.designer.help_5.18.0.xxxxxxxxxxxx.jar
- org.activiti.designer.integration_5.18.0.xxxxxxxxxxxx.jar
- org.activiti.designer.libs_5.18.0.xxxxxxxxxxxx.jar
- org.activiti.designer.util_5.18.0.xxxxxxxxxxxx.jar
- org.activiti.designer.validation.bpmn20_5.18.0.xxxxxxxxxxxx.jar

ヒント

BPMN エディタをアンインストールする場合は，まず，メニュー［ヘルプ］－［インストール詳細］から，画面［Eclipse のインストール詳細］を開きます。その画面で［Activiti Eclipse BPMN 2.0 Designer］を選択したあとに，［アンインストール］ボタンをクリックします。

2.3　接続先を設定する

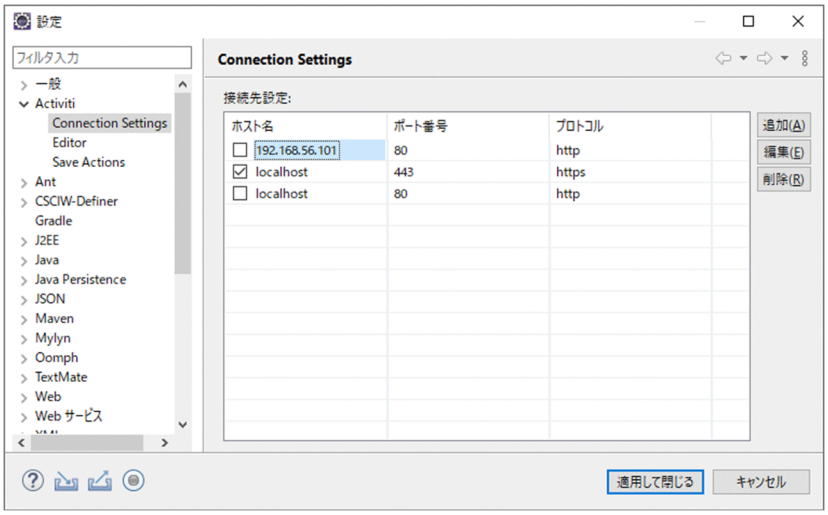
BPMN エディタから CSCIWManagementServer にログインするためには，BPMN エディタから接続できる接続先の設定が必要です。接続先ページ，［接続先設定］ダイアログについて次に示します。

接続先ページ

接続先ページでは，CSCIWManagementServer に接続するための設定をします。

接続先ページは，［ウィンドウ］－［設定］メニューで設定ページを開き，［Activiti］－［Connection Settings］を選択すると表示されます。接続先ページを次に示します。

図 2-5　接続先ページ



接続先ページの各項目について説明します。

- ［ホスト名］
  ［接続先設定］ダイアログで登録した接続先ホストの名称が表示されます。
  登録したデータがない場合は，空欄です。
  接続先ホストを指定する場合は，対象データのホスト名のチェックボックスをチェックします。
- ［ポート番号］
  ［接続先設定］ダイアログで登録済みの接続先ホストのポート番号が表示されます。
  登録したデータがない場合は，空欄です。
- ［プロトコル］
  ［接続先設定］ダイアログで登録した［HTTPS を使用する］チェックボックスのチェック状態によって，次のように表示されます。
  ［HTTPS を使用する］チェックボックスにチェックあり：https
  ［HTTPS を使用する］チェックボックスにチェックなし：http
- ［追加］ボタン

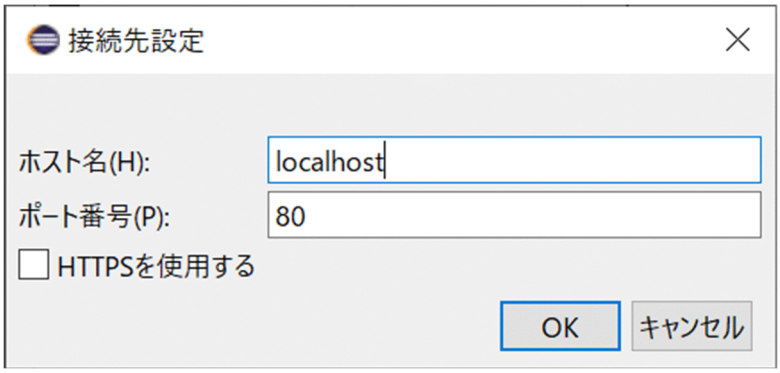
接続先ホストのデータを追加する場合にクリックします。
このボタンをクリックすると，［接続先設定］ダイアログが表示されます。

- ［編集］ボタン
  登録済みの接続先ホストのデータを変更する場合にクリックします。
  データが選択されていない場合は，このボタンは非活性になります。
  このボタンをクリックすると，［接続先設定］ダイアログが表示されます。
- ［削除］ボタン
  登録済みの接続先ホストのデータを削除する場合にクリックします。
  削除する場合は，対象のデータのホスト名を選択してから，このボタンをクリックします。
  データが選択されていない場合は，このボタンは非活性になります。
- ［適用して閉じる］ボタン
  接続先ページを閉じます。各項目で設定した内容が反映されます。
- ［キャンセル］ボタン
  接続先ページを閉じます。各項目で設定した内容は反映されません。

［接続先設定］ダイアログ

［接続先設定］ダイアログを次に示します。

図 2-6　［接続先設定］ダイアログ



［接続先設定］ダイアログの各項目について説明します。

- ［ホスト名］
  接続先ホストの名称を入力します。
- ［ポート番号］
  接続先ホストのポート番号を入力します。
- ［HTTPS を使用する］
  CSCIWManagementServer に接続するためのプロトコルとして HTTPS を使用するかどうかを指定します。HTTPS を使用する場合は，チェックボックスをチェックします。なお，CSCIWManagementServer との通信は，Java の API を使用しています。そのため，HTTPS 通信を

使用する場合は，必要に応じて Eclipse を起動する際に使用している JDK に証明書をインポートする必要があります。証明書のインポートの方法は JDK の説明書を参照してください。

- ［OK］ボタン
  ［接続先設定］ダイアログを閉じます。各項目で設定した内容が反映されます。
- ［キャンセル］ボタン
  ［接続先設定］ダイアログを閉じます。各項目で設定した内容が反映されます。

メモ

ログインした状態で接続先の設定を変更した場合，自動的にログアウトします。

2.3.1 BPMN エディタから CSCIWManagementServer にログインする

BPMN エディタから CSCIWManagementServer にログインする方法を説明します。ログインは，BPMN エディタからビジネスプロセスの登録・削除を実施する際に必要です。

操作手順

1. BPMN エディタから CSCIWManagementServer にログインするため，［ログイン］ダイアログを表示する。
   ［ログイン］ダイアログを表示するには，次の 2 つの方法があります。
   - Activiti Explorer のビジネスプロセス定義ファイル（.hbx）を選択した状態で右クリックして表示されるメニューの［Register Process Definition］をクリックする。
   - Registered Definition View の［ログイン］ボタンをクリックする。または，ツリービューを展開する。
2. ［ログイン］ダイアログで J2EE サーバに登録している［ユーザ ID］と［パスワード］を入力して［OK］ボタンをクリックする。
   ログインが成功すると，Registered Definition View が表示されます。

メモ

Registered Definition View が表示されない場合は，次の操作で Registered Definition View を表示できます。

1. Eclipse のメニューから［ウィンドウ］－［ビューの表示］－［その他］選択する。
   ［ビューの表示］ダイアログが表示されます。
2. ［ビューの表示］ダイアログのツリーから［Activiti］－［Registered Definition View］を選択して，［開く］ボタンをクリックする。

2.4　ビジネスプロセスの開発（作成および変換）

ビジネスプロセスを作成して，CSCIW で使用できるファイル形式（.hbx）に変換します。

2.4.1　ビジネスプロセスを作成する

BPMN エディタを使用してビジネスプロセスを作成します。

関連項目

- 1.3.1　BPMN 連携機能で使用できる BPMN 要素

(1)　BPMN ビジネスプロセス定義ファイルの作成時の規則

BPMN ビジネスプロセス定義ファイルを作成する際の規則について説明します。

規則によっては，プロパティ入力時，または検証時にチェックされます。

ビジネスプロセス定義ファイルのチェックの詳細については，「(2)　BPMN ビジネスプロセス定義ファイルのチェック」を参照してください。

入力必須の項目

BPMN ビジネスプロパティ定義ファイルに入力必須の項目と，チェックのタイミングについて，次の表で示します。

入力必須の項目	チェックのタイミング	
	プロパティ入力時	検証時
各要素のId プロパティ	○	－
Message ref プロパティ	－	○
Error ref プロパティ	－	○
Operation ref プロパティ	－	○
Called element プロパティ	－	○
Condition プロパティ（排他ゲートウェイから流出するシーケンスフローが 2 本以上ある場合）※	－	○
Loop cardinality プロパティ（マルチインスタンスを使用する場合）	－	○

（凡例）

○：チェックする

－：チェックしない

注※

デフォルトシーケンスフローを除きます。

プロパティ値の文字および文字列

プロパティ値の文字および文字列に関する規則と，チェックのタイミングを次に示します。

プロパティ	規則	チェックのタイミング	
		プロパティ入力時	検証時
Id プロパティ値	次に示す文字を使用してください。 • 半角英数字（0x30～0x39，0x41～0x5a，0x61～0x7a） • アンダースコア（0x5f） 「IW」から始まる値を指定しないでください。	○	－
Name プロパティ値	次に示す文字を使用してください。 • 全角文字（ただし,”，＾，および全角空白を除く） • 半角英数字（0x30～0x39，0x41～0x5a，0x61～0x7a） • アンダースコア（0x5f） 「IW」から始まる値を指定しないでください。	○	－
• Operation ref プロパティ値 • Called element プロパティ値	次に示す文字を使用してください。 • 半角英数字（0x30～0x39，0x41～0x5a，0x61～0x7a） • アンダースコア（0x5f）	○	－
	ファイル名として使用できない文字列（例：aux，con）を指定しないでください。	－	－
• Message definitions のId プロパティ値 • Error definitions のId プロパティ値 • Process タブのName プロパティ値	ファイル名として使用できない文字列（例：aux，con）を指定しないでください。	－	－
• Condition プロパティ（シーケンスフロー） • Loop cardinality プロパティ（マルチインスタンスの BPMN 要素） • Completion condition プロパティ（マルチインスタンスの BPMN 要素） • Completion condition プロパティ（アドホック・サブプロセスの BPMN 要素）	正しい XPath を指定してください。	－	－

（凡例）

○：チェックする

－：チェックしない

プロパティ値の長さ

プロパティ値の長さに関する規則，およびチェックのタイミングを次に示します。プロパティ値の長さは，次の範囲にしてください。

プロパティ	プロパティ値の長さの範囲	チェックのタイミング	
		プロパティ入力時	検証時
Process タブのId	59 バイト以下	○	－
Process タブのName	64 バイト以下※	○	－
プールのName	32 バイト以下	○	－
レーンのName	32 バイト以下	○	－
アクティビティのId とName の長さの合計	50 バイト以下	○	－
イベントのId とName の長さの合計	50 バイト以下	○	－
ゲートウェイのId とName の長さの合計	50 バイト以下	○	－
サブプロセスのId とName の長さの合計	50 バイト以下	○	－
イベント・サブプロセスのId とName の長さの合計	50 バイト以下	○	－
アドホック・サブプロセスのId とName の長さの合計	50 バイト以下	○	－
シーケンスフロー，およびデフォルトシーケンスフローのId	60 バイト以下	○	－
コールアクティビティのCalled element	32 バイト以下	○	－
サービスタスクのOperation ref	25 バイト以下	○	－
ビジネスルールタスクのOperation ref	25 バイト以下	○	－
Message definitions のId	25 バイト以下	○	－
Error definitions のId	25 バイト以下	○	－
Process data key name（[Use the process data as the participant] がチェックされているとき）	64 バイト以下	○	－

プロパティ	プロパティ値の長さの範囲	チェックのタイミング	
		プロパティ入力時	検証時
Process data key name（[Use the process data as the timer rule] がチェックされているとき）	64 バイト以下	○	－

（凡例）

○：チェックする

－：チェックしない

注※

レーンのName プロパティおよびプールのName プロパティが 0 バイトの場合は，32 バイト以下となります。

プロパティ値の一意性

プロパティ値の一意性に関する規則，およびチェックのタイミングを次に示します。

対象	規則	チェックのタイミング	
		プロパティ入力時	検証時
General タブおよび Process タブで入力するId プロパティ値	BPMN ビジネスプロセス定義ファイル内のサポート対象要素に対して，すべての要素間で大文字と小文タイマー字を区別しないで一意になるようにしてください。	○	－
「Name プロパティ値_Id プロパティ値」の規則で生成される文字列	すべての要素間で大文字と小文字を区別しないで一意になるようにしてください。	○	－
同一システム内の Process タブの Name プロパティ値	大文字と小文字を区別しないで，同一システム内で一意になるようにしてください。例えば，"SampleBP1"と"samplebp1"は，同一システム内で共存できません。	－	－
同一システム内の次に示すプロパティ • Operation ref • Message definitions のId • Error definitions のId • Called element	大文字と小文字を区別しないで，同一システム内で一意になるようにしてください。例えば，"msgAAA"と"msgaaa"は，同一システム内で共存できません。	－	－

（凡例）

○：チェックする

－：チェックしない

BPMN 要素の定義

排他ゲートウェイ以外の要素が複数の遷移先を持つときは，シーケンスフローの条件式（Condition プロパティ）を指定しないでください。条件分岐するには，排他ゲートウェイを使用する必要があります。

BPMN 要素の配置

BPMN 要素の配置に関する規則，およびチェックのタイミングを次に示します。

要素の種別	規則	チェックのタイミング	
		プロパティ入力時	検証時
開始イベント	次のどちらかの配置をした場合，開始イベントに対して同じ Message ref の値を指定しないでください。 • 開始（メッセージ）イベントを 1 つのプールに 2 個以上配置 • イベント・サブプロセス非中断開始（メッセージ）イベントとイベント・サブプロセス中断開始（メッセージ）イベントのどちらか，または両方を，1 つのイベント・サブプロセスに 2 個以上配置	−	○
	開始（タイプなし）イベントを，1 つのプールに 2 個以上配置しないでください。	−	○
	イベント・サブプロセス中断開始（エラー）イベントを，1 つのイベント・サブプロセスに 2 個以上配置した場合，同じ Error ref の値を指定しないでください。	−	−
サブプロセス / イベント・サブプロセス / アドホック・サブプロセス	イベント・サブプロセスを配置したプール，サブプロセス，またはイベント・サブプロセスの範囲内※1 に，そのイベント・サブプロセス非中断開始（メッセージ）イベントまたはイベント・サブプロセス中断開始（メッセージ）イベントと同じMessage ref の値を持つ次のイベントを配置しないでください。 • イベント・サブプロセス非中断開始（メッセージ）イベント • イベント・サブプロセス中断開始（メッセージ）イベント • キャッチ（メッセージ）イベント • 境界非中断（メッセージ）イベント • 境界中断（メッセージ）イベント	−	○
	イベント・サブプロセスを配置したプール，サブプロセス，またはイベント・サブプロセスの範囲内※1 に，そのイベント・サブプロセス中断開始（エラー）イベントと同じError ref の値を持つ次のイベントを配置しないでください。 • イベント・サブプロセス中断開始（エラー）イベント • 境界中断（エラー）イベント	−	○
	サブプロセス（マルチインスタンス）の範囲内※2 に，マルチインスタンス化した BPMN 要素を配置しないでください。	−	○
	アドホック・サブプロセスに，次の要素を配置しないでください。 • 開始イベント • 終了イベント • プール	○	−

要素の種別	規則	チェックのタイミング	
		プロパティ入力時	検証時
サブプロセス / イベント・サブプロセス / アドホック・サブプロセス	• レーン • サブプロセス • イベント・サブプロセス • アドホック・サブプロセス	○	−
ゲートウェイ	排他イベントゲートウェイの遷移先に，Message ref の値が同じキャッチ（メッセージ）イベントを複数配置しないでください。	−	○
	排他イベントゲートウェイからキャッチイベントにシーケンスフローを接続する場合，キャッチイベントの入力シーケンスフローは，排他イベントゲートウェイから遷移するシーケンスフローの 1 つだけになるようにしてください（そのほかの要素からの入力シーケンスフローを，キャッチイベントに接続しないでください）。	−	○
境界イベント	境界非中断（メッセージ）イベントもしくは境界中断（メッセージ）イベントが付けられたサブプロセスまたはアドホック・サブプロセス内に，その境界非中断（メッセージ）イベントまたは境界中断（メッセージ）イベントと同じMessage ref の値を持つ次のイベントを配置しないでください。 • イベント・サブプロセス非中断開始（メッセージ）イベント • イベント・サブプロセス中断開始（メッセージ）イベント • キャッチ（メッセージ）イベント • 境界非中断（メッセージ）イベント • 境界中断（メッセージ）イベント	−	○
	境界非中断（メッセージ）イベントまたは境界中断（メッセージ）イベントが付けられたアクティビティの境界に，その境界非中断（メッセージ）イベントまたは境界中断（メッセージ）イベントと同じMessage ref の値を持つ次のイベントを配置しないでください。 • 境界非中断（メッセージ）イベント • 境界中断（メッセージ）イベント	−	○
	境界中断（エラー）イベントが付けられたサブプロセスまたはアドホック・サブプロセス内に，その境界中断（エラー）イベントと同じError ref の値を持つ次のイベントを配置しないでください。 • イベント・サブプロセス中断開始（エラー）イベント • 境界中断（エラー）イベント	−	○
	境界中断（エラー）イベントが付けられたアクティビティの境界に，その境界中断（エラー）イベントと同じError ref の値を持つ境界中断（エラー）イベントを配置しないでください。	−	○

要素の種別	規則	チェックのタイミング	
		プロパティ入力時	検証時
その他・共通	プールの中には，次のうち 1 つ以上の要素を配置してください。 • ユーザタスク • サービスタスク • ビジネスルールタスク • コールアクティビティ • サブプロセス • アドホック・サブプロセス • キャッチ（メッセージ）イベント • スロー（メッセージ）イベント • キャッチ（タイマー）イベント	－	○
	サブプロセスの中には，次のうち 1 つ以上の要素を配置してください。 • ユーザタスク • サービスタスク • ビジネスルールタスク • コールアクティビティ • サブプロセス • アドホック・サブプロセス • キャッチ（メッセージ）イベント • スロー（メッセージ）イベント • キャッチ（タイマー）イベント	－	○
	アドホック・サブプロセスの中には，次のうち 1 つ以上の要素を配置してください。 • ユーザタスク • サービスタスク • ビジネスルールタスク • コールアクティビティ	－	○

（凡例）

○：チェックする

－：チェックしない

注※1

サブプロセスまたはイベント・サブプロセス内にサブプロセス，イベント・サブプロセス，またはアドホック・サブプロセスを配置した場合は，内側のサブプロセス，イベント・サブプロセスまたはアドホック・サブプロセス内も範囲に含みます。

注※2

サブプロセス内にサブプロセス，イベント・サブプロセス，またはアドホック・サブプロセスを配置した場合は，内側のサブプロセス，イベント・サブプロセスまたはアドホック・サブプロセス内も範囲に含みます。

BPMN 要素の配置可否

配置する要素の，配置先への配置可否を次の表で示します。

項番	配置する要素		配置先								
			キャンバス	プール/レーン	イベント・サブプロセス	展開されたサブプロセス	展開されたサブプロセス(マルチインスタンス)	展開されたアドホック・サブプロセス	アクティビティ各種	イベント各種	ゲートウェイ各種
1	コンテナ	プール	○	×	×	×	×	×	×	×	×
2		レーン	×	○	×	×	×	×	×	×	×
3		イベント・サブプロセス	×	○	○	○	○	×	×	×	×
4		展開されたサブプロセス	×	○	○	○	○	×	×	×	×
5		折りたたまれたサブプロセス	×	○	○	○	○	×	×	×	×
6		展開されたサブプロセス(マルチインスタンス)	×	○	○	○	×	×	×	×	×
7		折りたたまれたサブプロセス（マルチイ	×	○	○	○	×	×	×	×	×

項番	配置する要素		配置先								
			キャンバス	プール/レーン	イベント・サブプロセス	展開されたサブプロセス	展開されたサブプロセス(マルチインスタンス)	展開されたアドホック・サブプロセス	アクティビティ各種	イベント各種	ゲートウェイ各種
7	コンテナ	ンスタンス)	×	○	○	○	×	×	×	×	×
8		展開されたアドホック・サブプロセス	×	○	○	○	○	×	×	×	×
9		折りたたまれたアドホック・サブプロセス	×	○	○	○	○	×	×	×	×
10	アクティビティ	ユーザタスク	×	○	○	○	○	○	×	×	×
11		サービスタスク	×	○	○	○	○	○	×	×	×
12		ビジネスルールタスク	×	○	○	○	○	○	×	×	×
13		コールアクティビティ	×	○	○	○	○	○	×	×	×
14		ユーザタスク(マルチインスタンス)	×	○	○	○	×	○	×	×	×
15		サービスタスク（マルチイ	×	○	○	○	×	○	×	×	×

項番	配置する要素		配置先								
			キャンバス	プール/レーン	イベント・サブプロセス	展開されたサブプロセス	展開されたサブプロセス(マルチインスタンス)	展開されたアドホック・サブプロセス	アクティビティ各種	イベント各種	ゲートウェイ各種
15	アクティビティ	ンスタンス)	×	○	○	○	×	○	×	×	×
16		ビジネスルールタスク（マルチインスタンス)	×	○	○	○	×	○	×	×	×
17		コールアクティビティ(マルチインスタンス)	×	○	○	○	×	○	×	×	×
18	イベント	開始(タイプなし)	×	○	×	○	○	×	×	×	×
19		開始(メッセージ)	×	○	×	×	×	×	×	×	×
20		開始(タイマー)	×	○	×	×	×	×	×	×	×
21		イベント・サブプロセス非中断開始(メッセージ)	×	×	○	×	×	×	×	×	×
22		イベント・サブプロセス中断開始	×	×	○	×	×	×	×	×	×

項番	配置する要素		配置先								
			キャンバス	プール/レーン	イベント・サブプロセス	展開されたサブプロセス	展開されたサブプロセス(マルチインスタンス)	展開されたアドホック・サブプロセス	アクティビティ各種	イベント各種	ゲートウェイ各種
22	イベント	(メッセージ)	×	×	○	×	×	×	×	×	×
23		イベント・サブプロセス非中断開始（タイマー)	×	×	○	×	×	×	×	×	×
24		イベント・サブプロセス中断開始(タイマー)	×	×	○	×	×	×	×	×	×
25		イベント・サブプロセス中断開始(エラー)	×	×	○	×	×	×	×	×	×
26		終了(タイプなし)	×	○	○	○	○	×	×	×	×
27		終了(メッセージ)	×	○	○	○	○	×	×	×	×
28		終了(エラー)	×	○	○	○	○	×	×	×	×
29		強制終了	×	○	○	○	○	×	×	×	×
30		スロー(メッセージ)	×	○	○	○	○	○	×	×	×

項番	配置する要素		配置先								
			キャンバス	プール/レーン	イベント・サブプロセス	展開されたサブプロセス	展開されたサブプロセス(マルチインスタンス)	展開されたアドホック・サブプロセス	アクティビティ各種	イベント各種	ゲートウェイ各種
31	イベント	スロー(リンク)	×	○	○	○	○	○	×	×	×
32		キャッチ(メッセージ)	×	○	○	○	○	○	×	×	×
33		キャッチ（リンク）	×	○	○	○	○	○	×	×	×
34		キャッチ（タイマー)	×	○	○	○	○	○	×	×	×
35		境界非中断(メッセージ)	×	×	×	○	○	○	○	×	×
36		境界中断(メッセージ)	×	×	×	○	○	○	○	×	×
37		境界非中断(タイマー)	×	×	×	○	○	○	○	×	×
38		境界中断（タイマー)	×	×	×	○	○	○	○	×	×
39		境界中断（エラー)	×	×	×	○	○	○	○	×	×
40	ゲートウェイ	排他ゲートウェイ	×	○	○	○	○	○	×	×	×

項番	配置する要素		配置先								
			キャンバス	プール/レーン	イベント・サブプロセス	展開されたサブプロセス	展開されたサブプロセス(マルチインスタンス)	展開されたアドホック・サブプロセス	アクティビティ各種	イベント各種	ゲートウェイ各種
41	ゲートウェイ	並列ゲートウェイ	×	○	○	○	○	○	×	×	×
42		排他イベントゲートウェイ	×	○	○	○	○	○	×	×	×

（凡例）

○：CSCIW で配置可能

×：CSCIW で配置不可

ユーザタスクの設定

ユーザタスクの設定に関する規則，およびチェックのタイミングを次に示します。

規則	チェックのタイミング	
	プロパティ入力時	検証時
[Use the process data as the participant] がチェックされているとき，[Process data key name] を指定してください。	−	○
[Use the process data as the participant] がチェックされているとき，[Process data key name] を，次の規則で指定してください。 • "$S"から開始する • "$S"のあとに，"IW"から始まる文字列は指定できない • "$S"のあとには，英数字，"{"，および"}"だけが指定できる	○	−

（凡例）

○：チェックする

−：チェックしない

タイマーイベントの設定

タイマーイベントの設定に関する規則，およびチェックのタイミングを次に示します。

規則	チェックのタイミング	
	プロパティ入力時	検証時
[Type] を必ず指定してください。	−	○
[Use the process data as the timer rule] がチェックされているとき，[Process data key name] を指定してください。	−	○
[Use the process data as the timer rule] がチェックされているとき，[Process data key name] を，次の規則で指定してください。 • "$S"から開始する • "$S"のあとに，"IW"から始まる文字列は指定できない • "$S"のあとには，英数字，"{"，および"}"だけが指定できる	○	−
タイマールールの定期日時方式を使用する場合，存在する日付を指定してください（存在しない日付（4/31 など）を指定しないでください）。	−	○
キャッチ（タイマー）イベント，イベント・サブプロセス中断開始（タイマー）イベント，および境界中断（タイマー）イベントの実行回数は，1 以外を指定しても無効です。	−	−
開始（タイマー）イベントの場合，[Use the process data as the timer rule] をチェックしても無効です。	−	−

（凡例）

○：チェックする

−：チェックしない

シーケンスフローの接続

シーケンスフローの接続に関する規則，およびチェックのタイミングを次に示します。

対象	規則	チェックのタイミング	
		プロパティ入力時	検証時
中間イベント	アドホック・サブプロセス内の中間イベント（キャッチ（リンク）イベントは除く）には，入力シーケンスフローを接続してください。	−	○
	アドホック・サブプロセス内の中間イベント（スロー（リンク）イベントは除く）には，出力シーケンスフローを接続してください。	−	○
ゲートウェイ	アドホック・サブプロセス内のゲートウェイには，入力シーケンスフローを接続してください。	−	○
	アドホック・サブプロセス内の排他イベントゲートウェイには，出力シーケンスフローを接続してください。	−	○

（凡例）

○：チェックする

－：チェックしない

(2)　BPMN ビジネスプロセス定義ファイルのチェック

BPMN ビジネスプロセス定義ファイルは，BPMN エディタでの入力時，および検証実行時にチェックされます。

BPMN エディタでプロパティを入力した時のチェック

プロパティタブのテキストボックスの入力値は，次のフィールドにカーソルを移動したときなどの，カーソル移動時にチェックされます。

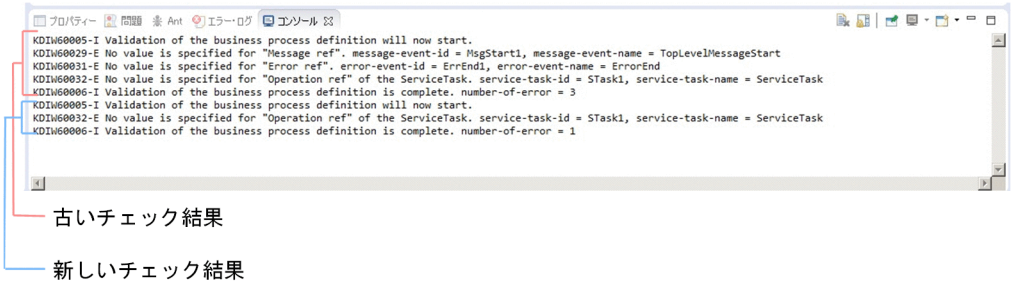
Message 定義，Error 定義の作成ダイアログの入力情報は，入力ダイアログボックスの［OK］をクリックしたときにチェックされます。

エラーがある場合は，エラーメッセージのダイアログが表示されます。

検証実行時のチェック

BPMN エディタのキャンバスを右クリックすると表示されるメニューで，［Validate Diagram］をクリックすると，チェックされます。

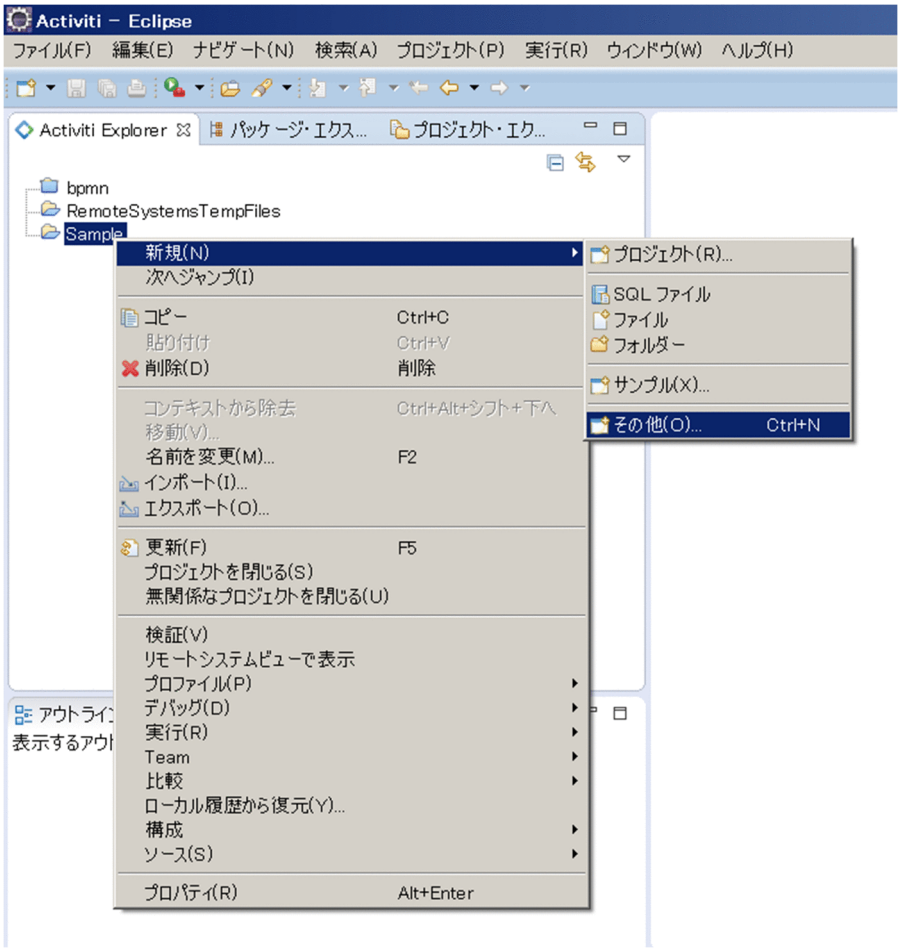
検証結果は，Eclipse コンソールに表示されます。［Validate Diagram］を繰り返し実行した場合，古いチェック結果の下に新しいチェック結果が表示されます。検証開始メッセージと終了メッセージから判断して，最新のチェック結果を確認してください。

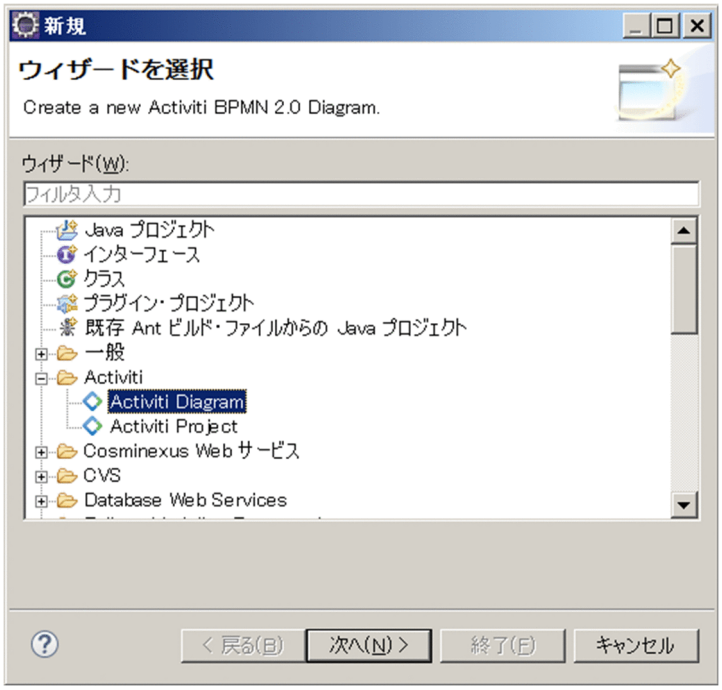


(3)　ビジネスプロセスの作成例

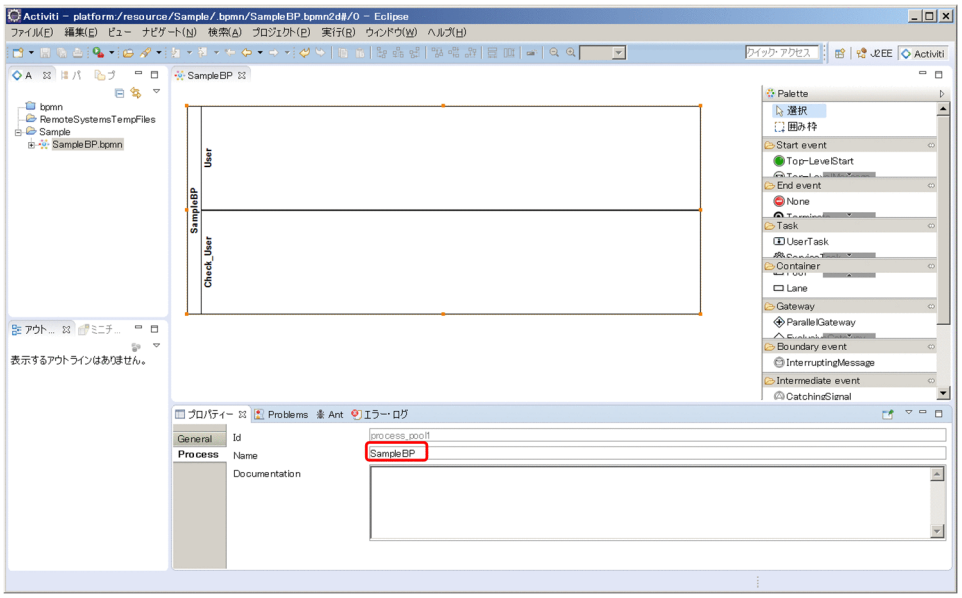
ビジネスプロセスの作成の流れを説明します。

1. BPMN エディタの［新規］メニューから［Activiti Diagram］ウィザードを実行して，BPMN ビジネスプロセス定義ファイルを作成します。



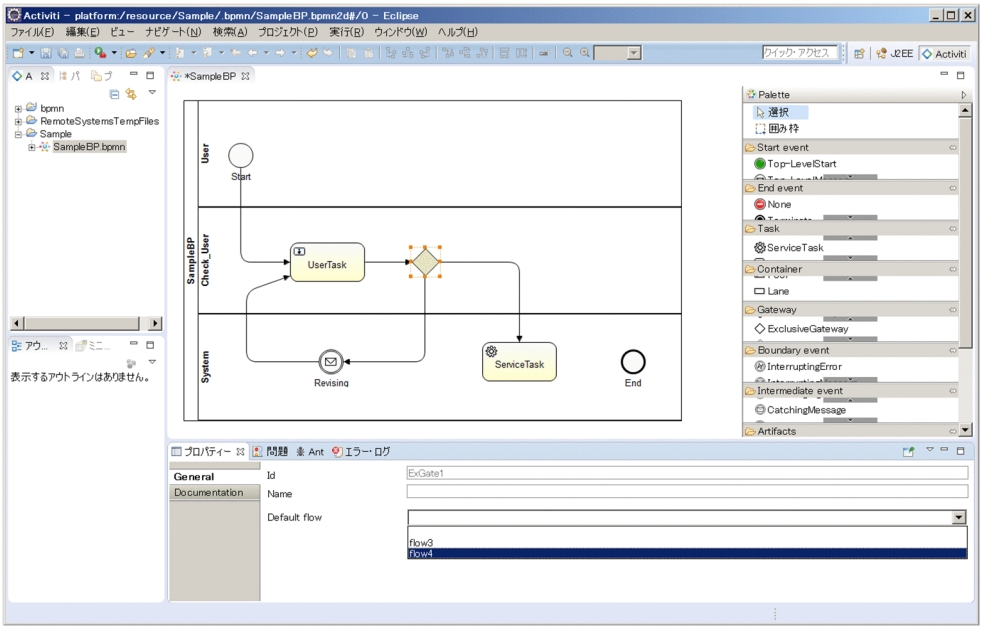


2. プールを選択すると表示される［Process］タブの［Name］テキストボックスに，ビジネスプロセスの名前を入力します。



3. 排他ゲートウェイを定義します。

   a. デフォルトシーケンスフローを描く場合は，排他ゲートウェイを選択して，［Default flow］リストボックスからデフォルトシーケンスフローとするシーケンスフローを設定します。

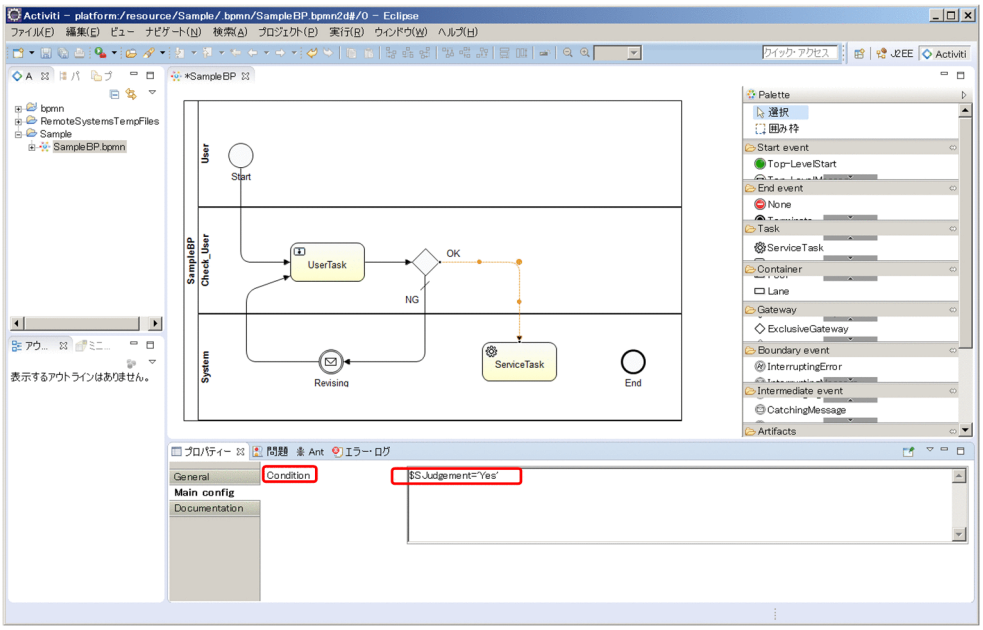


   b. 分岐条件を設定する場合は，シーケンスフローを選択すると表示される［Main config］タブの［Condition］テキストボックスエリアに，XPath 式を入力します。

   XPath 式の入力例

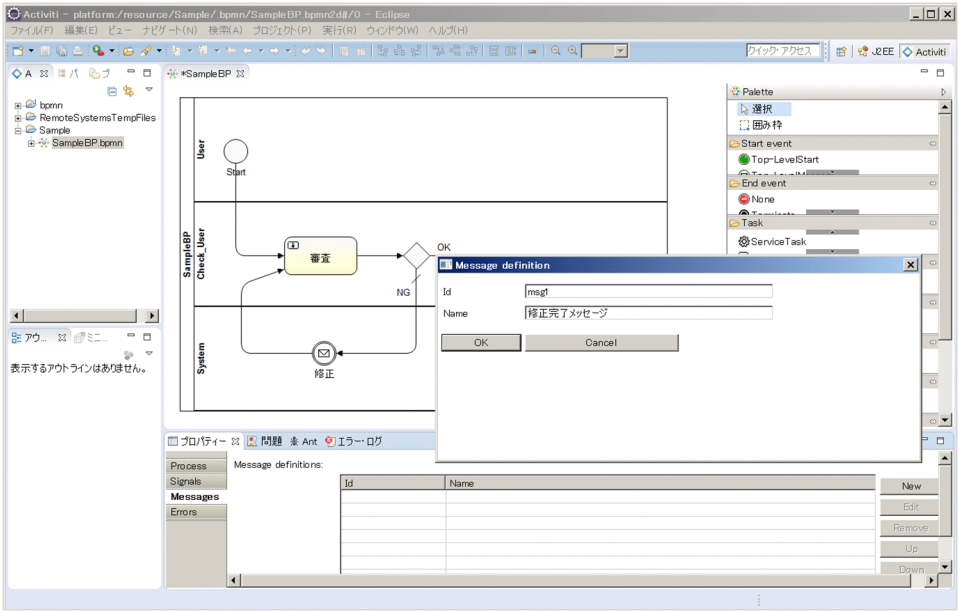
```
$SJudgement='Yes'
```

   "$SJudgement"は，文字列型の"Judgement"という名前のプロセスデータを表しています。

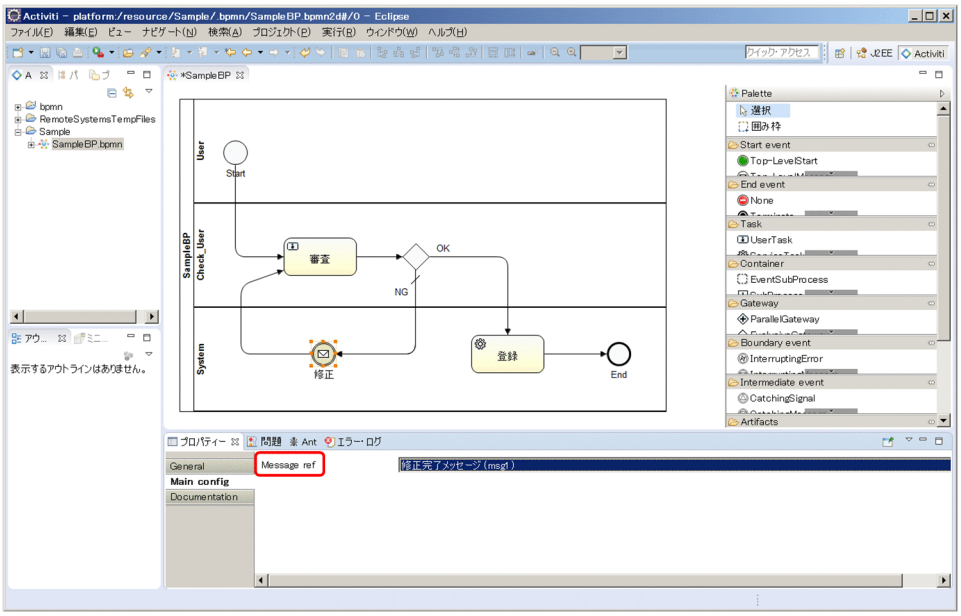


4. messageRef を定義します。

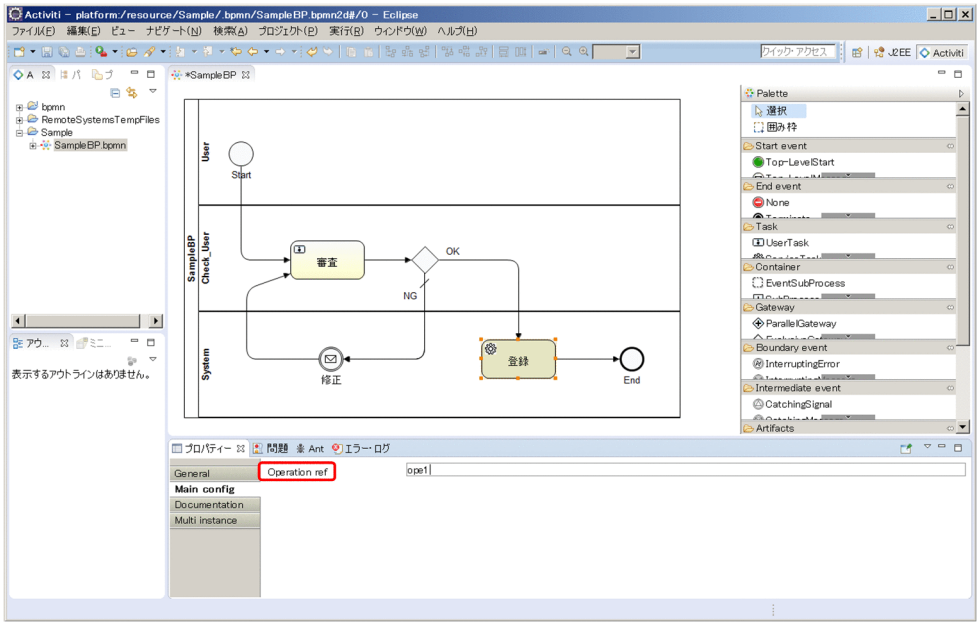
   a. キャンバス上の何もないところをクリックすると［Messages］タブが表示されます。［New］ボタンをクリックして表示される［Message definition］画面で，［Id］と［Name］を入力します。［Id］に入力した値がmessageRef 値になります。



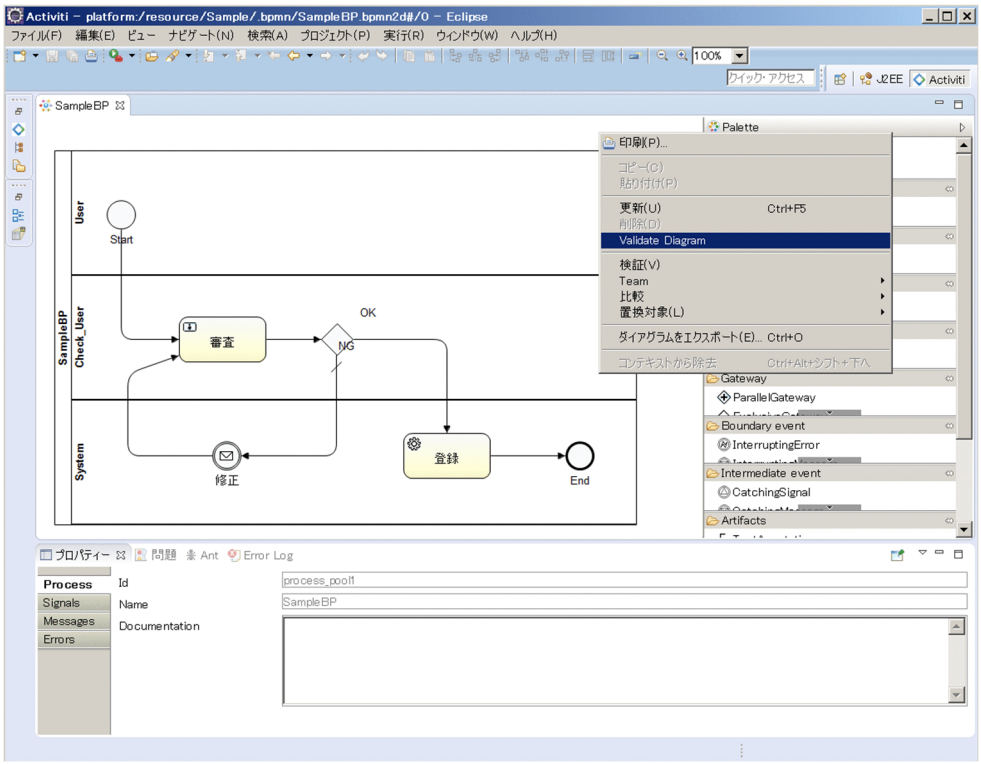
   b. メッセージイベントを選択すると，［Main config］タブが表示されます。［Message ref］リストボックスに，上記で［Id］に入力したメッセージを設定します。



5. サービスタスクでは，［Operation ref］テキストボックスに`operationRef`値（任意）を入力します。この値が，アプリケーション呼び出し情報ファイルのファイル名として使用されます。



6. ビジネスプロセス定義を検証します。キャンバスを右クリックすると表示されるメニューで［Validate Diagram］をクリックします。チェック内容の詳細については「(1)　BPMN ビジネスプロセス定義ファイルの作成時の規則」を参照してください。



検証結果が Eclipse コンソールに表示されます。表示される検証結果については，「(2)　BPMN ビジネスプロセス定義ファイルのチェック」の「検証実行時のチェック」の説明を参照してください。

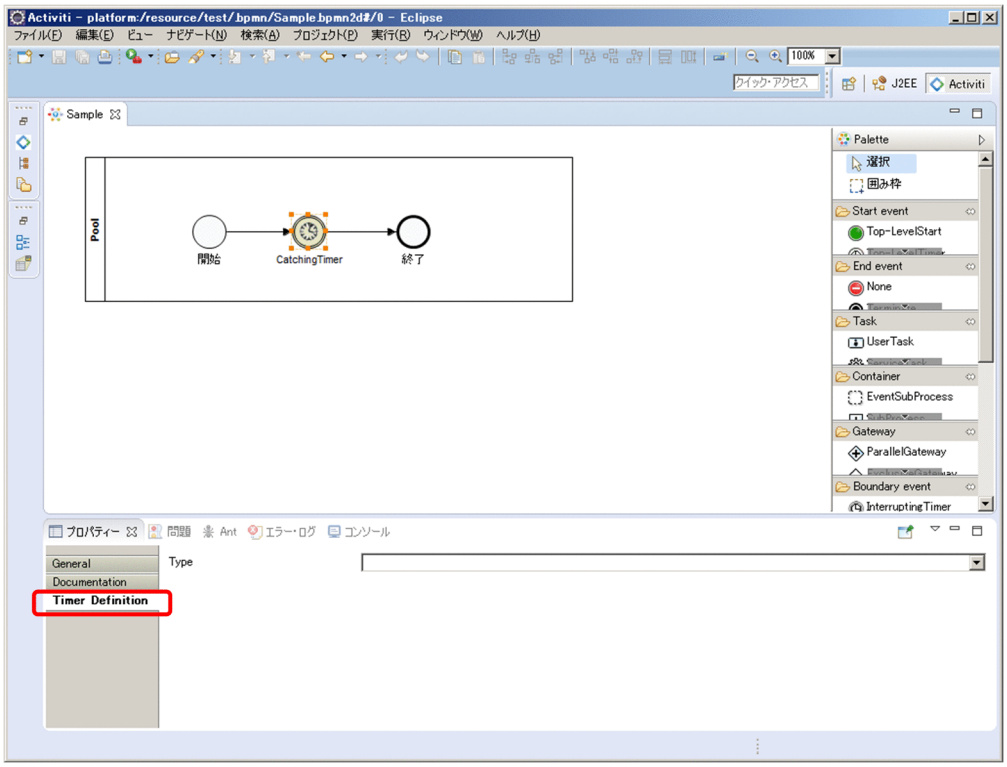
(4)　タイマーイベントの作成方法

タイマーイベントを作成する方法について説明します。

タイマーイベントの新規作成

パレットを使用してタイマーイベントを作成します。作成できるタイマーイベントの種類については，「1.4 タイマーイベントとは」を参照してください。

タイマーイベント作成後，タイマーイベントを選択すると，［Timer Definition］タブが表示されます。



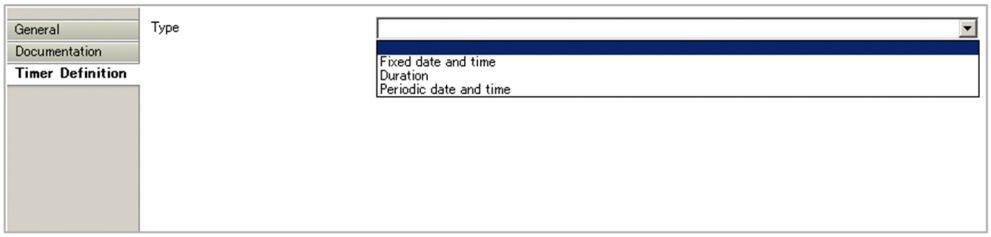
タイマールールの設定方法

[Timer Definition] タブの［Type］リストボックスに，次のどれかのタイマールールを指定します。タイマールールの詳細については，「1.4.2　タイマーイベントのタイマールール」を参照してください。

- 固定日時方式の場合：Fixed date and time
- 間隔方式の場合：Duration
- 定期日時方式の場合：Periodic date and time

[Timer Definition] タブの入力エリアを次の図で示します。

図 2-7　Timer Definition タブの入力エリア

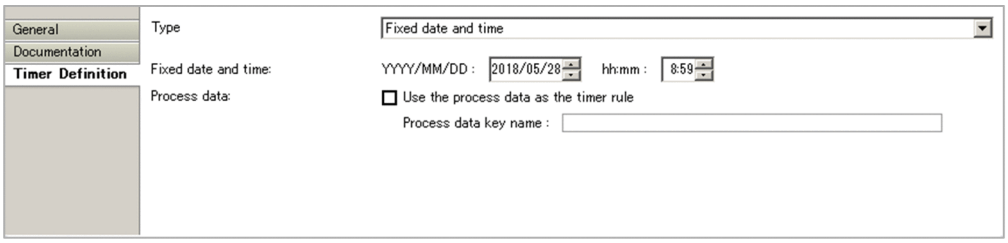


［Type］リストボックスで選択したタイマールールによって，［Timer Definition］タブの入力エリアの表示が切り替わります。以降で，それぞれのタイマールールを選択した場合の入力エリアの表示と，設定方法を説明します。

固定日時方式（Fixed date and time）の設定

固定日時方式（Fixed date and time）を指定した場合の入力エリアの表示を次の図で示します。

図 2-8　固定日時方式（Fixed date and time）の場合の入力エリア



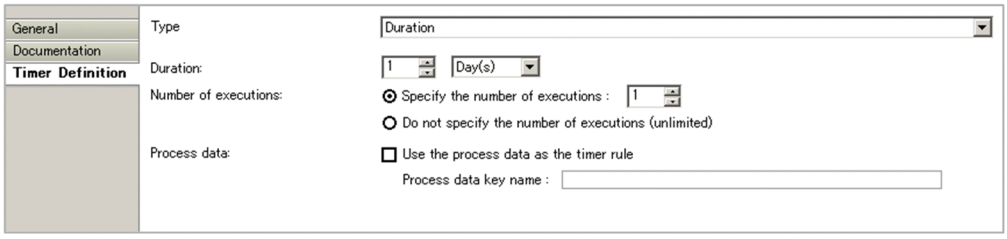
入力エリアに次の設定をします。

- Fixed date and time:
  - [YYYY/MM/DD]　：タイマーイベントが発火する日付を選択。
  - [hh:mm]　：タイマーイベントが発火する時刻を選択。
- Process data:
  - [Use the process data as the timer rule]　：タイマーイベントのタイマールールを動的に変更する場合にチェック。
  - [Process data key name]　：プロセスデータキー名を入力。

間隔方式（Duration）の設定

間隔方式（Duration）を指定した場合の入力エリアの表示を次の図で示します。

図 2-9　間隔方式（Duration）の設定画面



入力エリアに次の設定をします。

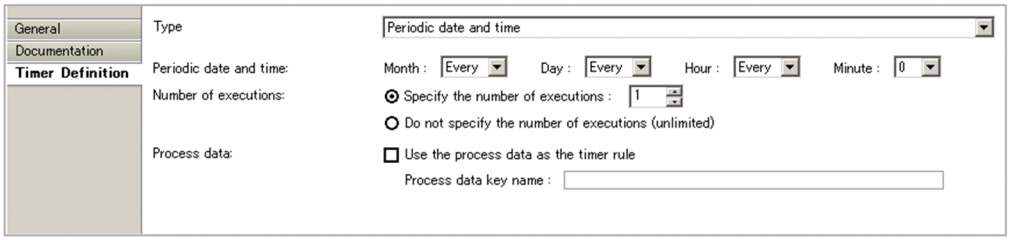
- Duration:
  時間長の数値と単位を選択。
- Number of executions:
  - [Specify the number of executions]　：実行回数を指定する場合に，ラジオボタンを選択。また，実行する回数を選択。
  - [Do not specify the number of executions (unlimited)]　：実行回数を無限とする場合にラジオボタンを選択。
- Process data:
  - [Use the process data as the timer rule]　：タイマーイベントのタイマールールを動的に変更する場合にチェック。

- [Process data key name] ：プロセスデータキー名を入力。

定期日時方式（Periodic date and time）の設定

定期日時方式（Periodic date and time）を指定した場合の入力エリアの表示を次の図で示します。

図 2-10　定期日時方式（Periodic date and time）の設定画面



入力エリアに次の設定をします。

- Periodic date and time:
  - [Month] ： 「月」を選択。任意の月とする場合は「Every」を選択。
  - [Day] ：「日」を選択。任意の日とする場合は「Every」を選択。
  - [Hour] ：「時」を選択。任意の時とする場合は「Every」を選択。
  - [Minute] ：「分」を選択。
- Number of executions:
  - [Specify the number of executions] ：実行回数を指定する場合に，ラジオボタンを選択。また，実行する回数を選択。
  - [Do not specify the number of executions (unlimited)] ：実行回数を無限とする場合にラジオボタンを選択。
- Process data:
  - [Use the process data as the timer rule] ：タイマーイベントのタイマールールを動的に変更する場合にチェック。
  - [Process data key name] ：プロセスデータキー名を入力。

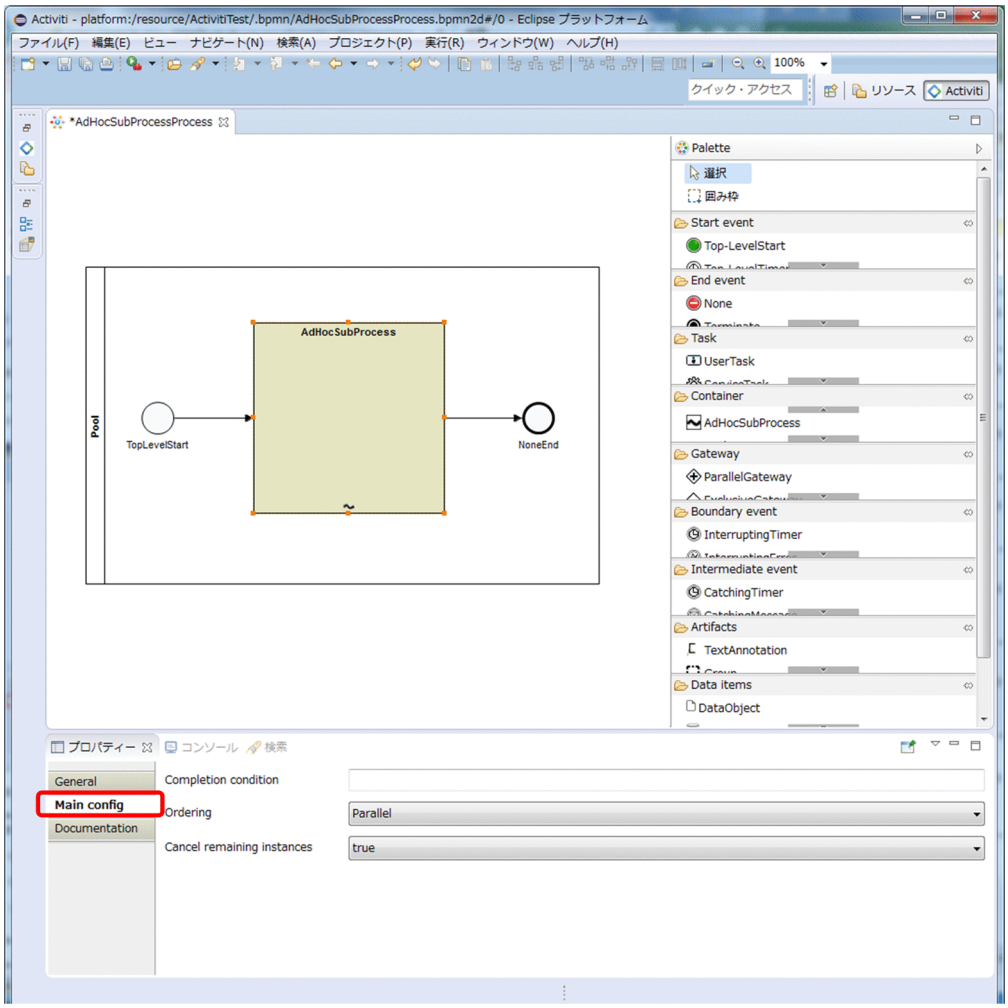
(5)　アドホック・サブプロセスの作成方法

アドホック・サブプロセスを作成する方法について説明します。

アドホック・サブプロセスの新規作成

パレットを使用してアドホック・サブプロセスを作成します。

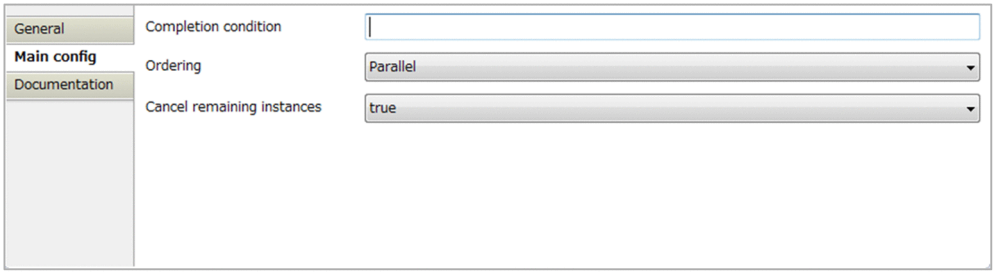
アドホック・サブプロセス作成後，アドホック・サブプロセスを選択すると，[Main config]タブが表示されます。



アドホック・サブプロセスの属性の設定方法

[Main config] タブの入力エリアを次の図で示します。

図 2-11　Main config タブの入力エリア



入力エリアに次の設定をします。

- Completion condition:
  アドホック・サブプロセスの完了条件の評価式を入力。
- Ordering:
  - [Parallel]：アドホック・サブプロセス内のフローノードを並列実行させる場合に選択。

- [Sequential]：アドホック・サブプロセス内のフローノードを逐次実行させる場合に選択。

- Cancel remaining instances:
  - [true]：完了条件（Completion condition）が成立した場合に，実行中のフローノードを強制終了させるときに選択。
  - [false]：完了条件（Completion condition）が成立した場合に，実行中のフローノードを強制終了させないときに選択。

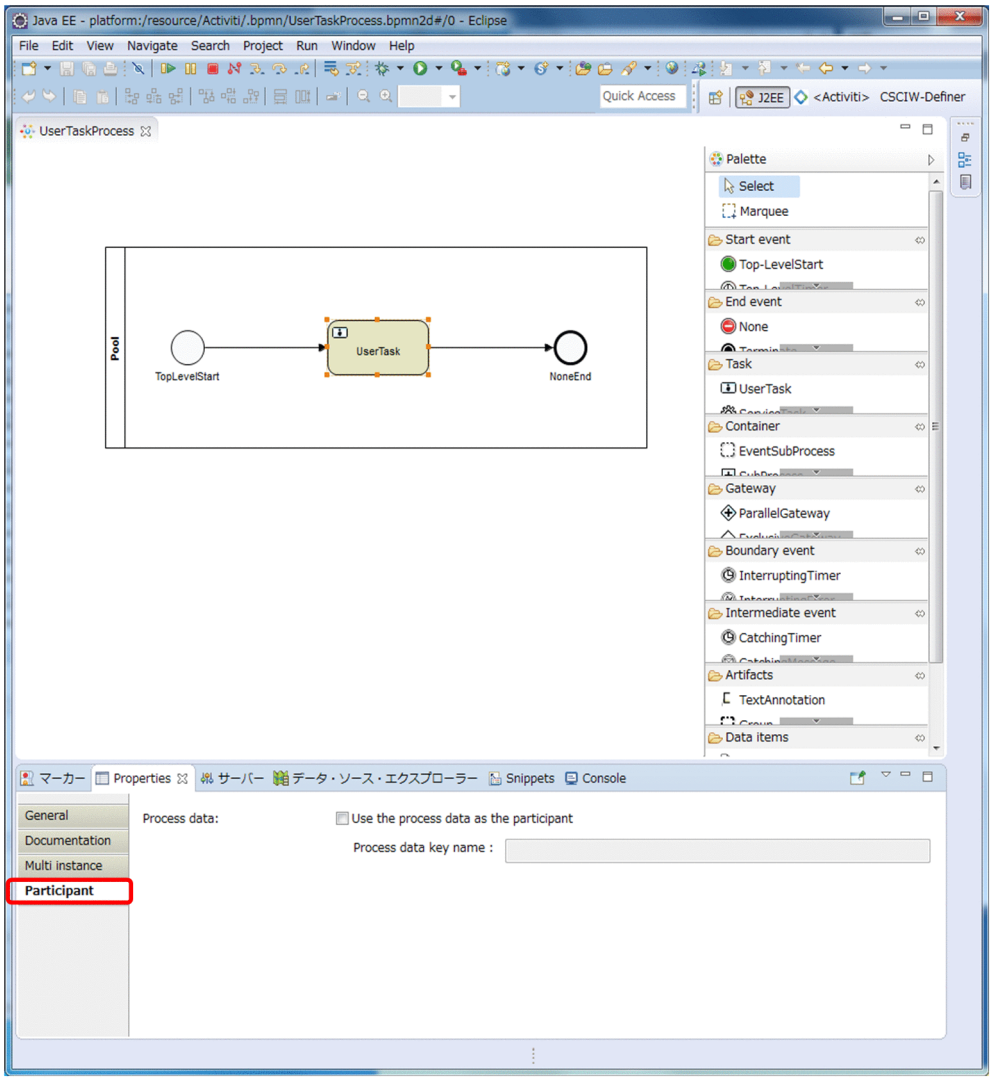
(6)　ユーザタスクの Participant を動的に指定する方法

ユーザタスクの Participant をプロセスデータで動的に指定するための，ユーザタスクの Participant タブの設定方法を説明します。

ユーザタスクの新規作成

パレットを使用してユーザタスクを作成します。

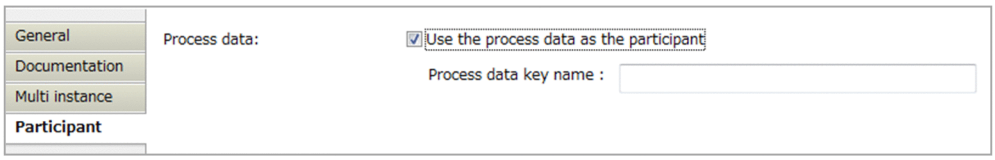
ユーザタスクの作成後，ユーザタスクを選択すると，［Participant］タブが表示されます。



プロセスデータの設定方法

［Participant］タブの入力エリアを次の図で示します。

図 2-12　Participant タブの入力エリア



入力エリアに次の設定をします。

- ［Use the process data as the participant］
  Participant をプロセスデータで動的に指定する場合にチェック。
- ［Process data key name］
  プロセスデータキー名を入力。

(7)　ビジネスプロセス作成時の注意事項

BPMN エディタでビジネスプロセスを作成する際の注意事項を説明します。

- Id プロパティ値には，一意な値が自動的に付与されます。各要素のId プロパティ値を変更する場合は，［ウィンドウ］－［設定］メニューをクリックすると表示される設定ダイアログで，次の項目にチェックを入れてください。

  ［Activiti］－［Editor］

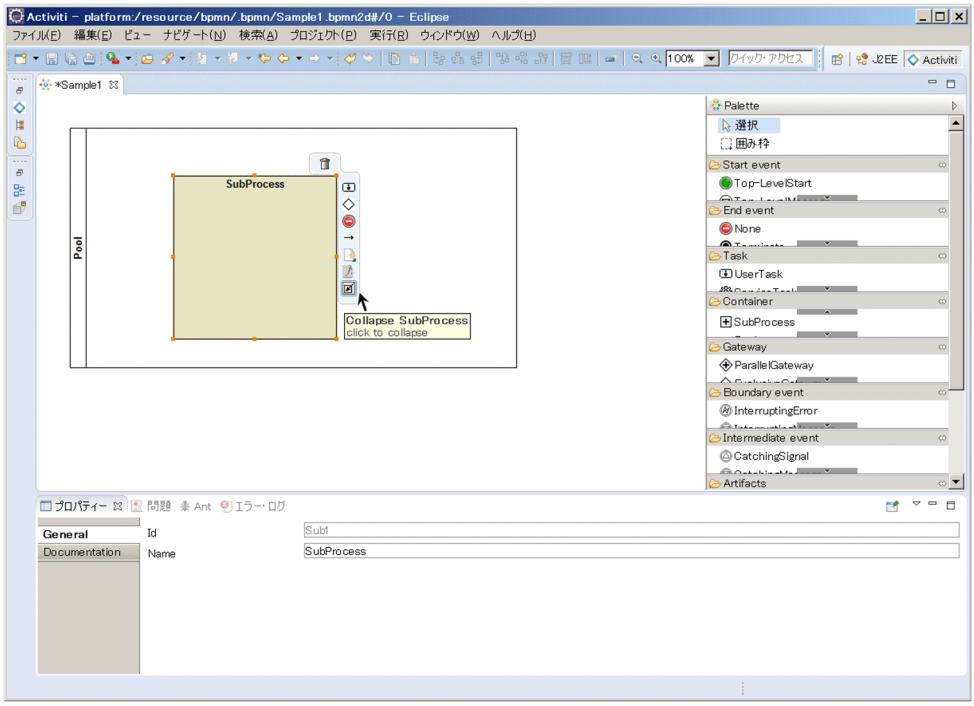
  　［Enable Id for BPMN Elements］のチェックボックス

  Id プロパティ値は，ビジネスプロセス定義ファイル内で一意になるようにしてください。重複するとBPMN エディタ上で予期しない動作をするおそれがあります。
- Id プロパティ値やName プロパティ値の長さをチェックする必要があります。［ウィンドウ］－［設定］メニューをクリックすると表示される設定ダイアログで，次の項目のチェックを入れてください（チェックを外さないでください）。

  ［Activiti］－［Editor］

  　［Apply length validation for Id and Name attributes］のチェックボックス
- 対応するスロー（リンク）イベントとキャッチ（リンク）イベントは，［Main config］タブの［LinkEventDefinition name］プロパティの値を一致させてください。
- イベント・サブプロセス中断開始（メッセージ）イベントを描くには，イベント・サブプロセス非中断開始（メッセージ）イベント［Non-InterruptingMessage］を描き，［Main config］タブの［isInterrupting］プロパティをtrue に変更してください。
- 境界非中断（メッセージ）イベントを描くには，境界中断（メッセージ）イベントを描き，［Main config］タブの［Cancel activity］プロパティをfalse に変更してください。

- イベント・サブプロセス中断開始（エラー）イベントおよび境界中断（エラー）イベントは，Error ref プロパティで設定したエラーを受信します。エラーの定義（Error definitions）では，ErrorCode プロパティの設定は必要ありません。
- マルチインスタンスを描くには，対象となるユーザタスク，サービスタスク，コールアクティビティ，またはサブプロセスの［Multi instance］タブを開き，［Loop cardinality］プロパティに値を設定してください。
- サブプロセスを折りたたむ場合，または展開する場合は，対象のサブプロセスにマウスカーソルを合わせると表示されるメニューの［Collapse SubProcess］または［Expand SubProcess］をクリックしてください。



- イベント・サブプロセス中断開始（タイマー）イベントを描くには，イベント・サブプロセス非中断開始（タイマー）イベント［Non-InterruptingTimer］を描き，［Main config］タブの［isInterrupting］プロパティをtrue に変更してください。
- 境界非中断（タイマー）イベントを描くには，境界中断（タイマー）イベントを描き，［Main config］タブの［Cancel activity］プロパティをfalse に変更してください。

2.4.2　アプリケーション呼び出し情報ファイルを作成する

操作手順

1. 作成したビジネスプロセス定義の中で，次の表で示す BPMN 要素を使用している場合，ref 識別子ごとにアプリケーション呼び出し情報ファイルを作成する。

表 2-1　BPMN 要素と対応する ref 識別子

BPMN 要素	ref 識別子
サービスタスク	operationRef の値
ビジネスルールタスク	operationRef の値
スロー（メッセージ）	messageRef の値
終了（メッセージ）	messageRef の値
終了（エラー）	errorRef の値

なお，同じアプリケーション呼び出し情報ファイルを適用する場合は同じ ref 識別子を設定してください。異なるアプリケーション呼び出し情報ファイルを適用する必要がある場合は，異なる ref 識別子を設定してください。

アプリケーション呼び出し情報ファイルの詳細については，「15.3　アプリケーション呼び出し情報ファイル」を参照してください。

2.4.3　コールアクティビティ情報ファイルを作成する

操作手順

1. 作成したビジネスプロセス定義の中で，コールアクティビティを定義している場合，calledElement ごとにコールアクティビティ情報ファイルを作成する。

コールアクティビティ情報ファイルの詳細については，「15.5　コールアクティビティ情報ファイル」を参照してください。

2.4.4　BPMN ビジネスプロセス定義ファイルを CSCIW のビジネスプロセス定義ファイルに変換する

BPMN ビジネスプロセス定義ファイルを CSCIW のビジネスプロセス定義ファイルに変換するには，BPMN エディタで変換する方法とコマンドで変換する方法があります。

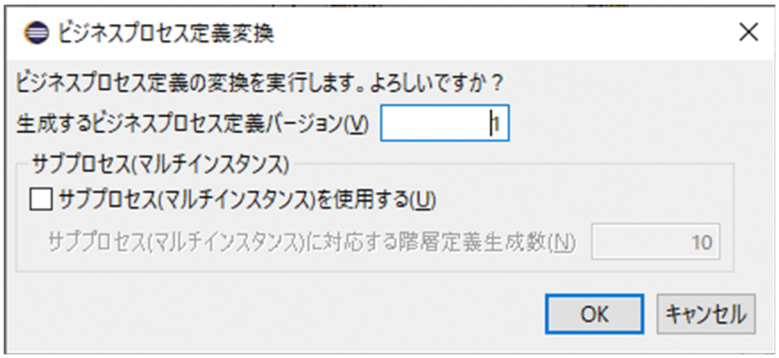
(1) BPMN エディタを使用した変換方法

BPMN エディタ上でciwtransbpmn コマンドの形式 1 を実行し，BPMN ビジネスプロセス定義ファイル（.bpmn）を，CSCIW で使用できるビジネスプロセス定義ファイル（.hbx）の形式に変換します。

操作手順

1. BPMN エディタで Activiti Explorer を起動する。
2. Activiti Explorer から BPMN ビジネスプロセス定義ファイル（.bpmn）を選択して右クリックする。
3. コンテキストメニューの［Transform BPMN Definition File］をクリックする。
   ［ビジネスプロセス定義変換］ダイアログが表示されるので，指定します。

   図 2-13 ［ビジネスプロセス定義変換］ダイアログ



   - ［生成するビジネスプロセス定義バージョン］
     ciwtransbpmn コマンド実行時の-bpv オプションに設定されます。
     1～9999 の整数で指定します。デフォルト値は 1 です。
   - ［サブプロセス(マルチインスタンス)を使用する］
     ［サブプロセス(マルチインスタンス)に対応する階層定義生成数］に値を入力する場合，チェックしてください。
   - ［サブプロセス(マルチインスタンス)に対応する階層定義生成数］
     ciwtransbpmn コマンド実行時の-mimax オプションに設定されます。
     1～999 の整数で指定します。デフォルト値は 10 です。

4. ［OK］ボタンをクリックする。

操作結果

ciwtransbpmn コマンドでの変換結果は，選択した BPMN ビジネスプロセス定義ファイル（.bpmn）と同一のディレクトリに出力されます。

ビジネスプロセス定義の変換に成功した場合，変換結果の CSCIW のビジネスプロセス定義ファイル（.hbx）と BPMN ビジネスプロセス定義ファイル（.bpmn）が Activiti Explorer に表示されます。

ciwtransbpmn コマンドの実行結果は，Eclipse コンソールビューに出力されます。

ciwtransbpmn コマンドの詳細については，マニュアル『uCosminexus Service Coordinator Interactive Workflow コマンド』を参照してください。

注意事項

uCosminexus Business Process Developer はインストール時に，BPMN エディタのプラグインファイルに uCosminexus Business Process Developer のインストールディレクトリを書き込みます。BPMN エディタが変換コマンド実行する場合，インストール時に書き込んだインストールディレクトリを読み込んで変換コマンドを実行します。そのため，uCosminexus Business Process Developer を再インストールした場合は，Eclipse に BPMN エディタを再インストールしてください。

(2) コマンドを使用した変換方法

BPMN ビジネスプロセス定義ファイル（.bpmn）を，CSCIW で使用できるビジネスプロセス定義ファイル（.hbx）の形式にコマンドを使用して変換します。

操作手順

1. ciwtransbpmn コマンドを使用して，BPMN ビジネスプロセス定義ファイル（.bpmn）を CSCIW で使用できるビジネスプロセス定義ファイル（.hbx）の形式に変換する。

操作結果

変換した BPMN ビジネスプロセス定義ファイルの分だけ，ディレクトリが出力されます。ディレクトリ内には，ビジネスプロセス定義ファイル（.hbx），およびビジネスプロセスの登録時に実行環境のマシンに転送する，BPMN ビジネスプロセス定義ファイル（.bpmn）が生成されています。

2.5　ビジネスプロセスの登録と削除

ビジネスプロセス定義をデータベースに登録し，活性化します。また，必要に応じて，ビジネスプロセスの内容や適用状態を変更します。

不要になったビジネスプロセス定義は削除します。

2.5.1　実行環境に定義ファイルを転送する

開発環境のマシンで作成した各定義ファイルを，実行環境のマシンに転送します。

前提条件

開発環境に，次に示すファイルが作成されている。

- アプリケーション呼び出し情報ファイル
- CSCIW のビジネスプロセス定義ファイル
- BPMN ビジネスプロセス定義ファイル
- コールアクティビティ情報ファイル
- REST アプリケーション呼び出し用ヘッダファイル
- REST アプリケーション呼び出しスキーマ変換用スタイルシート

操作手順

1. 開発環境のマシンで，対象ファイルが格納されているディレクトリを確認する。

   ■対象ファイル

   - アプリケーション呼び出し情報ファイル
   - CSCIW のビジネスプロセス定義ファイル※
   - BPMN ビジネスプロセス定義ファイル※
   - コールアクティビティ情報ファイル
   - REST アプリケーション呼び出し用ヘッダファイル
   - REST アプリケーション呼び出しスキーマ変換用スタイルシート

   注※

   コマンドを使用して登録する場合

2. 対象ファイルを実行環境のマシンに転送する。

   - アプリケーション呼び出し情報ファイル

     アプリケーション呼び出し情報ファイルの格納先ディレクトリにファイルを転送してください。

■アプリケーション呼び出し情報ファイルの格納先ディレクトリ（デフォルト）

＜ Windows の場合＞

サービスタスクおよびビジネスルールタスク：%CSCIW_HOME%¥bpmn¥callinfo¥ope

メッセージ：%CSCIW_HOME%¥bpmn¥callinfo¥msg

エラー：%CSCIW_HOME%¥bpmn¥callinfo¥err

＜ UNIX の場合＞

サービスタスクおよびビジネスルールタスク：${CSCIW_HOME}/bpmn/callinfo/ope

メッセージ：${CSCIW_HOME}/bpmn/callinfo/msg

エラー：${CSCIW_HOME}/bpmn/callinfo/err

実行環境でマルチマシン構成またはクラスタ構成を使用している場合は，すべてのマシンにファイルを転送してください。

- CSCIW のビジネスプロセス定義ファイルおよび BPMN ビジネスプロセス定義ファイル

  コマンドを使用して登録する場合は，任意のディレクトリにファイルを転送してください。

  実行環境でマルチマシン構成またはクラスタ構成を使用している場合は，任意の 1 台にファイルを転送してください。

  なお，ファイルの転送に FTP などのファイル転送機能を利用する場合は，バイナリ形式でファイルを転送する必要があります。

- コールアクティビティ情報ファイル

  コールアクティビティ情報ファイルの格納先ディレクトリにファイルを転送してください。

  ■コールアクティビティ情報ファイルの格納先ディレクトリ（デフォルト）

  ＜ Windows の場合＞

  %CSCIW_HOME%¥bpmn¥callinfo¥call

  ＜ UNIX の場合＞

  ${CSCIW_HOME}/bpmn/callinfo/call

  実行環境でマルチマシン構成またはクラスタ構成を使用している場合は，すべてのマシンにファイルを転送してください。

  なお，ファイルの転送に FTP などのファイル転送機能を利用する場合は，テキスト形式でファイルを転送する必要があります。

- REST アプリケーション呼び出し用ヘッダファイル，および REST アプリケーション呼び出しスキーマ変換用スタイルシート

  アプリケーション呼び出し情報ファイルに記載した格納先ディレクトリにファイルを転送してください。

  実行環境でマルチマシン構成またはクラスタ構成を使用している場合は，すべてのマシンにファイルを転送してください。

2.5.2　ビジネスプロセス定義を登録する

ビジネスプロセス定義をワーク管理データベースへ登録するには，BPMN エディタ，案件運用操作，またはコマンドを使用する方法があります。

(1)　BPMN エディタを使用した登録方法

CSCIW で使用できる形式（.hbx）に変換されたビジネスプロセス定義を，BPMN エディタを使用してワーク管理データベースに登録します。

操作手順

1. BPMN エディタで Activiti Explorer を起動する。
2. Activiti Explorer から CSCIW のビジネスプロセス定義ファイル（.hbx）を選択して右クリックする。
3. コンテキストメニューの［Register Process Definition］をクリックする。
4. 登録確認ダイアログで［はい］ボタンをクリックする。
5. ログインしていない場合は，［ログイン］ダイアログからログインする。
   ログインしていない場合は［ログイン］ダイアログが表示されるので，J2EE サーバに登録している［ユーザ ID］と［パスワード］を入力して［OK］ボタンをクリックしてください。

操作結果

選択した CSCIW のビジネスプロセス定義ファイルと同一のディレクトリに存在する同一名・同一バージョンの BPMN ビジネスプロセス定義ファイル（.bpmn）がワーク管理データベースへ登録されます。

なお，BPMN エディタで登録する CSCIW ビジネスプロセス定義ファイル，BPMN ビジネスプロセス定義ファイルは，BPMN エディタまたはciwtransbpmn コマンドで生成したファイルを使用してください。また，ファイルの内容およびファイル名を変更しないでください。

ビジネスプロセス定義の登録に成功した場合，Registered Definition View の表示が更新されます。

ビジネスプロセス定義登録の実行結果は，Eclipse コンソールビューに出力されます。

異常終了する場合

- ビジネスプロセス定義ファイル（.hbx）のファイル名が不正な場合
  指定したビジネスプロセス定義ファイル(.hbx)のファイル名が「<ビジネスプロセス定義名>#<ビジネスプロセス定義バージョン>.hbx」の形式になっていない場合，エラーになります。
- BPMN ビジネスプロセス定義ファイルが存在しない場合
  指定されたビジネスプロセス定義ファイル（.hbx）と同一ディレクトリに同一名・同一バージョンの BPMN ビジネスプロセス定義ファイル（.bpmn）が存在しない場合，エラーになります。

- BPMN ビジネスプロセス定義ファイルのファイルサイズが規定値を超えている場合

  指定されたビジネスプロセス定義ファイル（.hbx）と同一ディレクトリに存在する同一名・同一バージョンの BPMN ビジネスプロセス定義ファイル（.bpmn）のファイルサイズが 4 メガバイト（4,194,304 バイト）を超えている場合，エラーになります。
- ビジネスプロセス定義を上書き登録しようとした場合

  ビジネスプロセス定義の上書き登録はできません。登録対象のビジネスプロセス定義がすでに登録されている場合，エラーメッセージが Eclipse コンソールビューに表示され，BPMN ビジネスプロセス定義は登録しないで，登録処理を終了します。

  再登録をしたい場合は，登録済みのビジネスプロセス定義を削除したあと，再度ビジネスプロセス定義を登録してください。

(2) 案件運用操作を使用した登録方法

CSCIW で使用できる形式（.hbx）に変換されたビジネスプロセス定義を，案件運用操作を使用してワーク管理データベースに登録します。

操作手順

1. 案件運用操作の［ビジネスプロセス定義一覧］画面から［ビジネスプロセス定義登録］画面を起動する。
2. ［参照］ボタンをクリックして表示されるファイル選択ダイアログで CSCIW のビジネスプロセス定義ファイル，および BPMN ビジネスプロセス定義ファイルを選択する。
3. ［登録］ボタンをクリックする。

操作結果

CSCIW のビジネスプロセス定義ファイル，および BPMN ビジネスプロセス定義ファイルがワーク管理データベースに登録されます。

なお，案件運用操作で登録する CSCIW ビジネスプロセス定義ファイル，BPMN ビジネスプロセス定義ファイルは，BPMN エディタまたはciwtransbpmn コマンドで生成したファイルを使用してください。

(3) コマンドを使用した登録方法

CSCIW で使用できる形式（.hbx）に変換されたビジネスプロセス定義をコマンドを使用して，ワーク管理データベースに登録します。

操作手順

1. 実行マシンへ転送した CSCIW ビジネスプロセス定義ファイル，および BPMN ビジネスプロセス定義ファイルに対してciwmngbp コマンドを実行して，ワーク管理データベースへ登録する。

   ciwmngbp コマンドの指定形式を次に示します。

```
ciwmngbp -sid 〈システムID〉 -reg -bpf 〈ビジネスプロセス定義ファイルの絶対パス〉 -bpmnf 〈BPMNビジネスプロセス定義ファイルの絶対パス〉
```

なお，ciwmngbp コマンドで登録する CSCIW ビジネスプロセス定義ファイル，BPMN ビジネスプロセス定義ファイルは，BPMN エディタまたはciwtransbpmn コマンドで生成したファイルを使用してください。

ciwmngbp コマンドの詳細については，マニュアル『uCosminexus Service Coordinator Interactive Workflow コマンド』を参照してください。

ヒント

-list オプションを指定したciwmngbp コマンドを実行することで，データベースに登録されているビジネスプロセス定義，および BPMN ビジネスプロセス定義の一覧を確認できます。

ビジネスプロセス定義，および BPMN ビジネスプロセス定義の一覧を確認する場合のciwmngbp コマンドの指定形式を次に示します。

```
ciwmngbp -sid 〈システムID〉 -list
```

2.5.3 ビジネスプロセス定義の状態（活性/非活性）を変更する

ワーク管理データベースに登録したビジネスプロセス定義の状態を変更します。

背景

ビジネスプロセス定義の状態には，「活性（ACTIVE)」と「非活性（INACTIVE)」があります。

案件を生成および開始するためには，ビジネスプロセス定義を活性状態にする必要があります。

ビジネスプロセス定義を削除する場合には，ビジネスプロセス定義を非活性状態にする必要があります。

なお，BPMN エディタ，または案件運用操作を使用してビジネスプロセス定義を登録した場合，ビジネスプロセス定義は自動的に活性状態になります。

操作手順

1. -chg オプションを指定したciwmngbp コマンドを実行して，ビジネスプロセス定義の状態を変更する。
   - ビジネスプロセス定義を活性化する場合のciwmngbp コマンドの指定形式を次に示します。

     ```
     ciwmngbp -sid 〈システムID〉 -chg -bpn 〈ビジネスプロセス定義名〉 -bpv 〈ビジネスプロセス定義バージョン〉 -s ACTIVE
     ```

   - ビジネスプロセス定義を非活性化する場合のciwmngbp コマンドの指定形式を次に示します。

```
ciwmngbp -sid 〈システムID〉 -chg -bpn 〈ビジネスプロセス定義名〉 -bpv 〈ビジネスプロセス定義バージョン〉 -s INACTIVE
```

ciwmngbp コマンドの詳細については，マニュアル『uCosminexus Service Coordinator Interactive Workflow コマンド』を参照してください。

2. CSCIWManagementServer，および CSCIW を使用している Java アプリケーションを再開始する。

操作結果

ビジネスプロセス定義の状態が変更されます。

重要

- CSCIWManagementServer の停止中に，アプリケーション呼び出しサービスまたは REST サービスが動作した場合，エラーが発生します。ただし，CSCIWManagementServer を再開始した場合は，エラーが発生しても問題ありません。

ヒント

- データベースに登録されているビジネスプロセス定義の状態を確認したい場合は，-list オプションを指定したciwmngbp コマンドを実行してください。実行結果のStateCode 列から，ビジネスプロセス定義の状態を確認できます。

  ビジネスプロセス定義の状態を確認する場合のciwmngbp コマンドの指定形式を次に示します。

  ```
  ciwmngbp -sid 〈システムID〉 -list
  ```

2.5.4 ビジネスプロセス定義をバージョンアップする

ビジネスプロセス定義のバージョン番号を変更する方法を説明します。

背景

CSCIW 内では，ビジネスプロセス定義はビジネスプロセス定義名とバージョン番号で管理されます。つまり，同じ名称のビジネスプロセス定義であっても，バージョンが異なれば別のビジネスプロセス定義として扱われます。

ビジネスプロセス定義のバージョンは，ciwtransbpmn コマンドで任意のバージョン番号を指定できます。

操作手順

1. -bpv オプションを指定したciwtransbpmn コマンドを実行して，バージョン番号を指定する。

   バージョン番号を指定する場合のciwtransbpmn コマンドの指定形式を次に示します。

```
ciwtransbpmn -src 〈変換対象のBPMNビジネスプロセス定義ファイルパス〉-bpv 〈ビジネスプロセス定義バージョン〉-destdir 〈ビジネスプロセス定義ファイル出力先パス〉
```

ciwtransbpmn コマンドの詳細については，マニュアル『uCosminexus Service Coordinator Interactive Workflow コマンド』を参照してください。

操作結果

ビジネスプロセス定義のバージョンが，指定されたバージョン番号に変更されます。

2.5.5　ビジネスプロセス定義を削除する

ワーク管理データベースに登録したビジネスプロセス定義を削除するには，BPMN エディタ，案件運用操作，またはコマンドを使用する方法があります。

(1)　BPMN エディタを使用した削除方法

ワーク管理データベースに登録したビジネスプロセス定義を，BPMN エディタを使用して削除します。

前提条件

- ビジネスプロセス定義に属する案件が削除されている。
  案件を削除する方法については，「10.4　案件およびプロセスデータを削除する」を参照してください。

操作手順

1. BPMN エディタの Registered Definition View を起動する。
2. ログインしていない場合は，［ログイン］ボタンでログインする。
   ログインしていない場合は［ログイン］ダイアログが表示されるので，J2EE サーバに登録している［ユーザ ID］と［パスワード］を入力して［OK］ボタンをクリックしてください。
3. Registered Definition View から削除対象の CSCIW のビジネスプロセス定義を選択する。
4. コンテキストメニューの［削除］をクリックする。
   削除確認ダイアログが表示されます。
5. ［はい］ボタンをクリックする。
6. 削除したビジネスプロセス定義に対応しているアプリケーション呼び出し情報ファイルおよびコールアクティビティ情報ファイルのうち，ほかのビジネスプロセス定義で使用していないファイルは削除する。

操作結果

ビジネスプロセス定義が削除されます。

なお，削除する CSCIW のビジネスプロセス定義ファイルと同一名・同一バージョンの BPMN ビジネスプロセス定義ファイル（.bpmn）がワーク管理データベースに登録されている場合，CSCIW のビジネスプロセス定義ファイルと併せて BPMN ビジネスプロセス定義ファイルが削除されます。

ビジネスプロセス定義の削除に成功した場合，Registered Definition View の表示が更新されます。

ビジネスプロセス定義削除の実行結果は，Eclipse コンソールビューに出力されます。

(2)　案件運用操作を使用した削除方法

CSCIW のデータベースに登録したビジネスプロセス定義を，案件運用操作を使用して削除します。

前提条件

- ビジネスプロセス定義に属する案件が削除されている。
  案件を削除する方法については，「10.4　案件およびプロセスデータを削除する」を参照してください。

操作手順

1. 案件運用操作の［ビジネスプロセス定義一覧］画面から削除対象のビジネスプロセス定義を選択する。
2. メニューの［ビジネスプロセス定義削除］をクリックする。
3. 削除したビジネスプロセス定義に対応しているアプリケーション呼び出し情報ファイルおよびコールアクティビティ情報ファイルのうち，ほかのビジネスプロセス定義で使用していないファイルは削除する。

操作結果

ビジネスプロセス定義が削除されます。

なお，削除する CSCIW のビジネスプロセス定義ファイルと同一名・同一バージョンの BPMN ビジネスプロセス定義ファイル（.bpmn）がワーク管理データベースに登録されている場合，CSCIW のビジネスプロセス定義ファイルと併せて BPMN ビジネスプロセス定義ファイルが削除されます。

(3)　コマンドを使用した削除方法

ワーク管理データベースに登録したビジネスプロセス定義を，コマンドを使用して削除します。

前提条件

- ビジネスプロセス定義ファイルが非活性状態になっている。
- ビジネスプロセス定義に属する案件が削除されている。
  案件を削除する方法については，「10.4　案件およびプロセスデータを削除する」を参照してください。

操作手順

1. -del オプションを指定したciwmngbp コマンドを実行して，ビジネスプロセス定義を削除する。

   ビジネスプロセス定義を削除する場合のciwmngbp コマンドの指定形式を次に示します。

   ```
   ciwmngbp -sid 〈システムID〉 -del -bpn 〈ビジネスプロセス定義名〉 -bpv 〈ビジネスプロセス定義バージョン〉
   ```

   ciwmngbp コマンドの詳細については，マニュアル『uCosminexus Service Coordinator Interactive Workflow コマンド』を参照してください。

2. CSCIWManagementServer，および CSCIW を使用している Java アプリケーションを再開始する。

3. 削除したビジネスプロセス定義に対応しているアプリケーション呼び出し情報ファイルおよびコールアクティビティ情報ファイルのうち，ほかのビジネスプロセス定義で使用していないファイルは削除する。

操作結果

ビジネスプロセス定義が削除されます。

なお，削除する CSCIW のビジネスプロセス定義ファイルと同一名・同一バージョンの BPMN ビジネスプロセス定義ファイル（.bpmn）がワーク管理データベースに登録されている場合，CSCIW のビジネスプロセス定義ファイルと併せて BPMN ビジネスプロセス定義ファイルが削除されます。

ヒント

ビジネスプロセス定義が削除されたかどうかを確認したい場合は，-list オプションを指定したciwmngbp コマンドを実行してください。実行結果で，該当するビジネスプロセス定義の情報が表示されていなければ，削除されています。

ビジネスプロセス定義が削除されたかどうかを確認する場合のciwmngbp コマンドの指定形式を次に示します。

```
ciwmngbp -sid 〈システムID〉 -list
```

2.6　登録済みのビジネスプロセス定義を変更する

実行中のビジネスプロセスに影響を与えることなく，ビジネスプロセス定義の一部を変更します。ビジネスプロセス定義を変更する場合，BPMN エディタで次に示す編集ができます。

- BPMN 要素の設定値の変更
- BPMN 要素の追加
- BPMN 要素の削除

変更した内容はビジネスプロセスの実行されていない個所（これから実行される個所）に反映されます。すでに実行された個所には反映されません。

なお，サポートしていない変更を行った場合の動作は保証しません。

メモ

実行中の作業が存在する状況でも，ビジネスプロセス定義を変更できます。

ただし，実行中の作業には「すでに実行された個所」が含まれるため，変更内容が反映されない場合があります。

例えば，サービスタスクの ref 識別子（operationRef）を変更した場合，実行中の作業の作業者 ID は「IWTOPE_<operationRef>」で確定しているため，この作業に対して変更後の ref 識別子（operationRef）は反映されません。

2.6.1　ビジネスプロセス定義の変更例

ビジネスプロセス定義の一部を変更する例を示します。

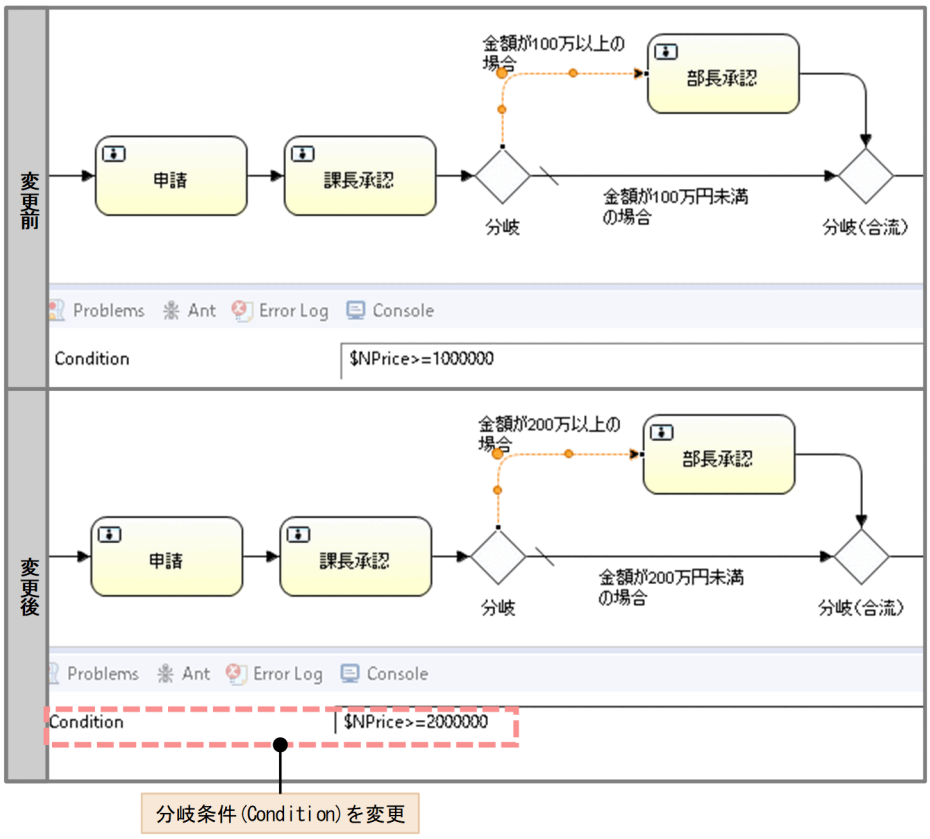
(1)　BPMN 要素の設定値の変更

BPMN 要素の設定値を変更する例を示します。

排他ゲートウェイの分岐条件を変更する

次の図のように，運用中のビジネスプロセス定義の排他ゲートウェイの分岐条件を変更できます。

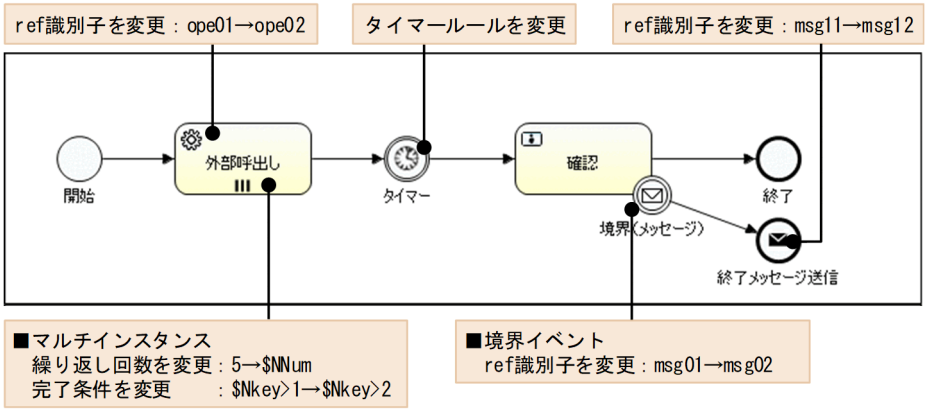
図 2-14　排他ゲートウェイの分岐条件の変更例



タスクやイベントの設定値を変更する

次の図のように，運用中のビジネスプロセス定義のタスクやイベントの設定値を変更できます。

図 2-15　タスクやイベントの設定値を変更する例



なお，ref 識別子やタイマールールの変更後の設定値は遷移済みのタスクやイベントには反映されません。設定値の変更後に遷移したタスクやイベントに反映されます。完了条件については，設定値の変更後に完了条件を評価するタイミングから反映されます。

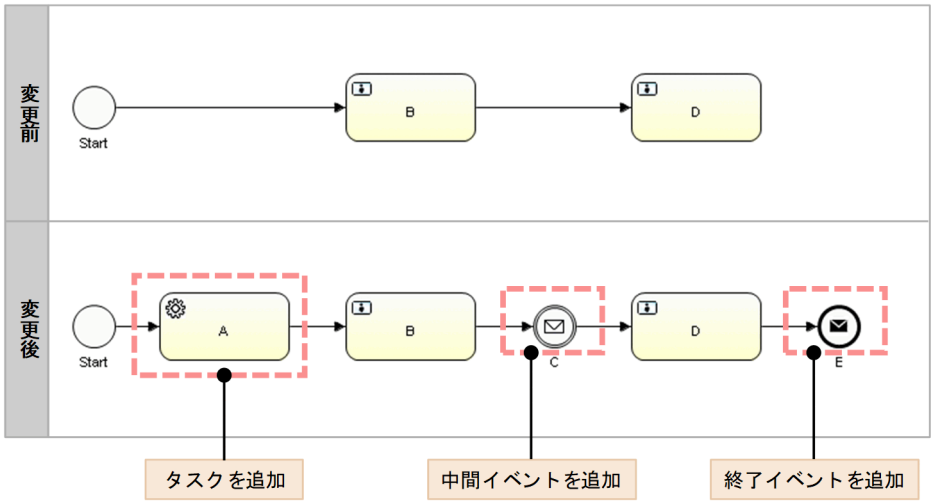
(2)　BPMN 要素の追加

BPMN 要素を追加する例を示します。

タスクやイベントを追加する

次の図のように，運用中のビジネスプロセス定義にタスクやイベントを追加できます。

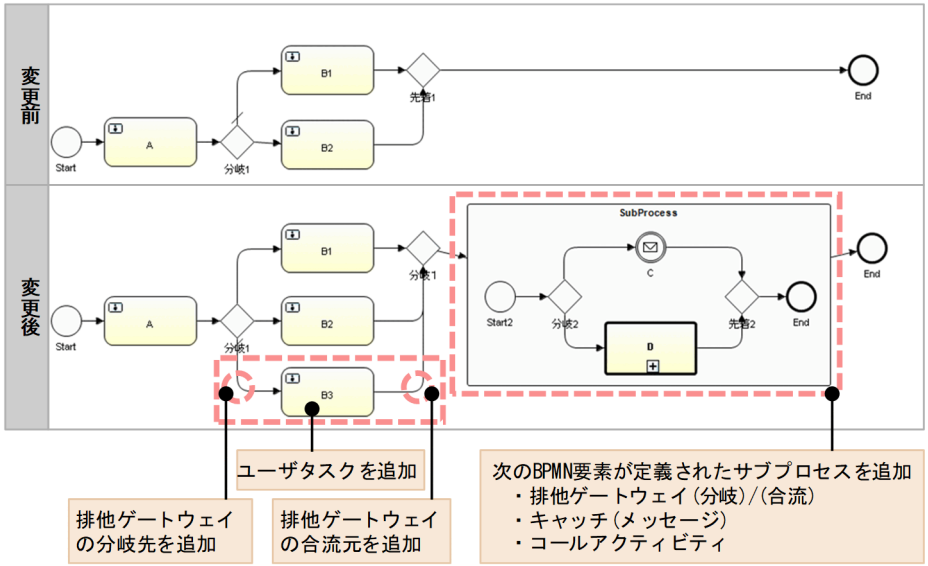
図 2-16　タスクやイベントを追加する例



排他ゲートウェイやサブプロセスを追加する

次の図のように，運用中のビジネスプロセス定義に排他ゲートウェイやサブプロセスを追加できます。

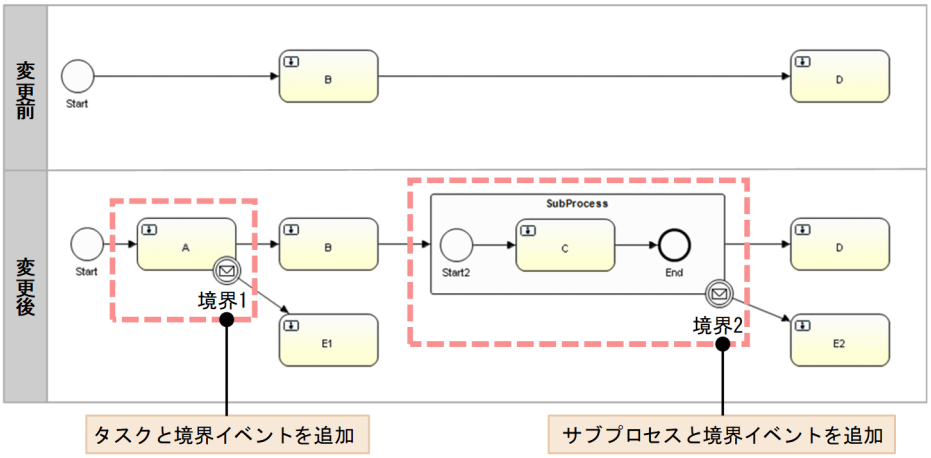
図 2-17　排他ゲートウェイやサブプロセスを追加する例



境界イベントを追加する

次の図のように，運用中のビジネスプロセス定義のタスクやサブプロセスと一緒に境界イベントを追加できます。ただし，既存のタスクやサブプロセスに境界イベントは追加できません。

図 2-18　境界イベントを追加する例



(3)　BPMN 要素の削除

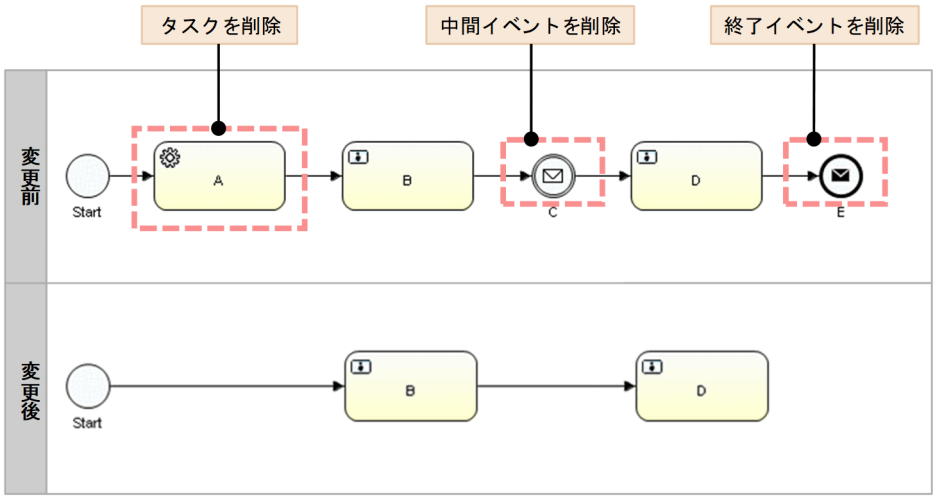
BPMN 要素を削除する例を示します。

タスクやイベントを削除する

次の図のように，運用中のビジネスプロセス定義のタスクやイベントを削除できます。

実行中のインスタンスがないタスクやイベントだけ削除できます。1 つでも削除対象のタスクやイベントが実行中の案件がある場合は削除できません。

図 2-19　タスクやイベントを削除する例

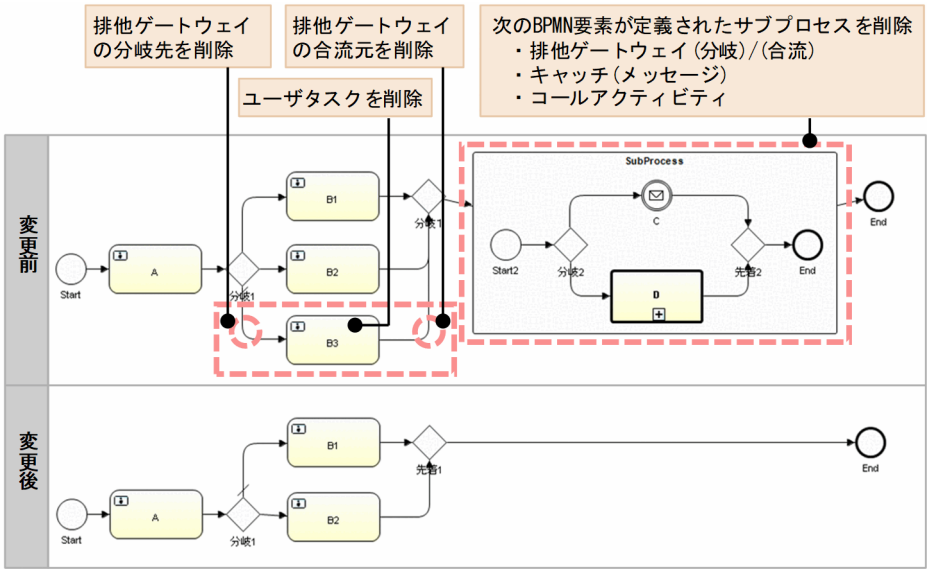


排他ゲートウェイやサブプロセスを削除する

次の図のように，運用中のビジネスプロセス定義の排他ゲートウェイやサブプロセスを削除できます。

サブプロセス内が実行中でない場合だけ削除できます。1 つでも削除対象のサブプロセス内のタスクやイベントが実行中の案件がある場合は削除できません。

図 2-20　排他ゲートウェイやサブプロセスを削除する例



(4)　並列ゲートウェイに関する変更・追加・削除

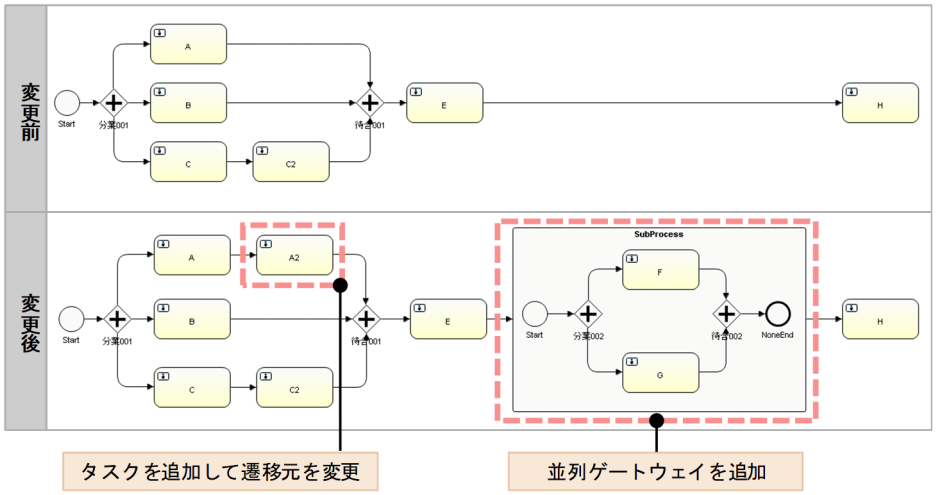
-ejp オプションを指定したciwchgenv コマンドを実行してEditJoinPermission にtrue を設定している場合，並列ゲートウェイに関する変更・追加・削除ができます。環境構築時のデフォルト設定（EditJoinPermission にfalse を設定）では，並列ゲートウェイに関する変更・追加・削除はできません。

ciwchgenv コマンドの詳細については，マニュアル『uCosminexus Service Coordinator Interactive Workflow コマンド』を参照してください。

並列ゲートウェイの遷移元の変更や並列ゲートウェイの追加をする

次の図のように，運用中のビジネスプロセス定義の並列ゲートウェイの遷移元の変更や，並列ゲートウェイの追加ができます。

図 2-21　並列ゲートウェイに関する変更や追加を行う例



2.6.2　BPMN エディタで変更可能なビジネスプロセス定義の詳細

BPMN エディタでビジネスプロセス定義を変更する場合の，規則や BPMN 要素ごとの編集可否について説明します。

(1)　BPMN 要素の追加または削除の可否

BPMN 要素を追加または削除する時の共通規則と，追加または削除できる BPMN 要素を次に示します。

BPMN 要素を追加または削除する時の共通規則

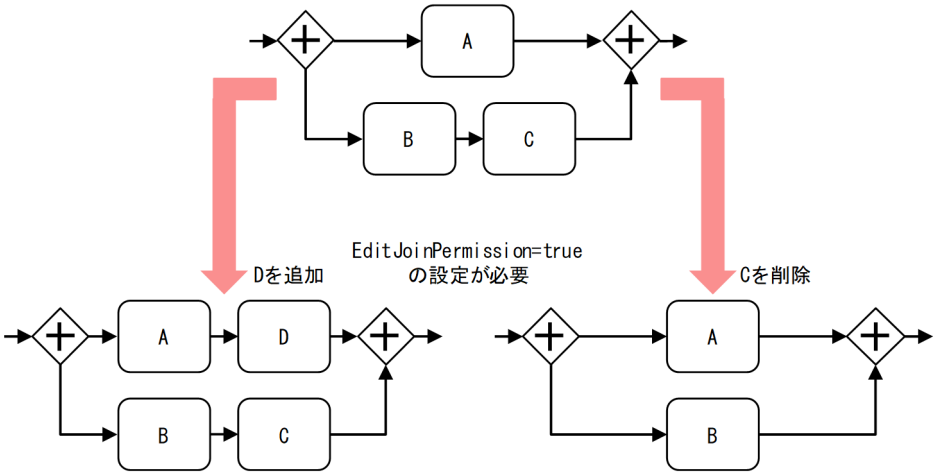
BPMN 要素を追加，または削除する時は，次の規則に従ってください。

- 削除対象の BPMN 要素が実行中の案件が 0 件であること
- 追加または削除する BPMN 要素の遷移先に並列ゲートウェイがある場合，-ejp オプションを指定した ciwchgenv コマンドを実行して EditJoinPermission に true を設定していること

  ciwchgenv コマンドの詳細については，マニュアル『uCosminexus Service Coordinator Interactive Workflow コマンド』を参照してください。

  並列ゲートウェイについては，「(3)　並列ゲートウェイに関する変更・追加・削除の可否」を参照してください。

図 2-22　追加または削除する BPMN 要素の遷移先に並列ゲートウェイがある場合



開始イベントの追加または削除（その 1）

開始（タイプなし）※

- 新規にサブプロセス，サブプロセス（マルチインスタンス）を追加する場合，その内部に開始（タイプなし）を追加できます。ただし，開始（タイプなし）を，省略形の有無に関わらず複数個追加する場合は，-ejp オプションを指定した ciwchgenv コマンドで EditJoinPermission に true を設定する必要があります。
- 既存のサブプロセス，サブプロセス（マルチインスタンス）を削除する場合，その内部の開始（タイプなし）を削除できます。ただし，開始（タイプなし）を，省略形の有無に関わらず複数個削除する場合

は，-ejp オプションを指定したciwchgenv コマンドでEditJoinPermission にtrue を設定する必要があります。

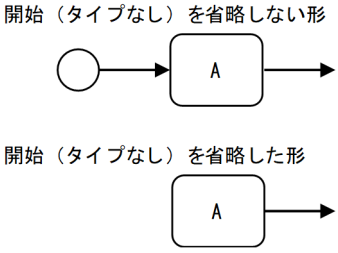
- トップレベル（レーン含む）に開始イベントの追加または削除はできません。
- 既存のサブプロセス，サブプロセス（マルチインスタンス）を残す場合，その内部への開始（タイプなし）の追加または削除はできません。

注※

省略形を含みます。

次の図の上段が開始（タイプなし）を省略しない形，下段が開始（タイプなし）を省略した形で，どちらも意味は同じです。

図 2-23　開始（タイプなし）の省略形



開始イベントの追加または削除（その 2）

```
開始（メッセージ）
開始（タイマー）
```

開始イベントの追加または削除はできません。

イベント・サブプロセス開始イベントの追加または削除（その 1）

```
イベント・サブプロセス非中断開始（メッセージ）
イベント・サブプロセス中断開始（メッセージ）
イベント・サブプロセス中断開始（エラー）
```

- 新規にイベント・サブプロセスを追加する場合，その内部にイベント・サブプロセス開始イベントを追加できます。
- 既存のイベント・サブプロセスを削除する場合，その内部のイベント・サブプロセス開始イベントを削除できます。
- 既存のイベント・サブプロセスを残す場合，その内部へのイベント・サブプロセス開始イベントの追加または削除はできません。

イベント・サブプロセス開始イベントの追加または削除（その 2）

```
イベント・サブプロセス非中断開始（タイマー）
イベント・サブプロセス中断開始（タイマー）
```

- 新規にイベント・サブプロセスを追加する場合，その内部にイベント・サブプロセス開始（タイマー）を追加できます。ただし，イベント・サブプロセス開始（タイマー）を追加する場合は，-ejp オプションを指定したciwchgenv コマンドでEditJoinPermission にtrue を設定する必要があります。また，新規イベント・サブプロセスの追加先は，サブプロセス，サブプロセス（マルチインスタンス），イベント・サブプロセスである必要があります。トップレベル（レーン含む）への追加はできません。また，追加先のサブプロセス，サブプロセス（マルチインスタンス），イベント・サブプロセスが実行中の案件が 0 件である必要があります。
- 既存のイベント・サブプロセスを削除する場合，その内部のイベント・サブプロセス開始（タイマー）を削除できます。ただし，イベント・サブプロセス開始（タイマー）を削除する場合は，-ejp オプションを指定したciwchgenv コマンドでEditJoinPermission にtrue を設定する必要があります。また，既存イベント・サブプロセスの削除元は，サブプロセス，サブプロセス（マルチインスタンス），イベント・サブプロセスである必要があります。トップレベル（レーン含む）からの削除はできません。また，削除元のサブプロセス，サブプロセス（マルチインスタンス），イベント・サブプロセスが実行中の案件が 0 件である必要があります。
- 既存のイベント・サブプロセスを残す場合，その内部へのイベント・サブプロセス開始（タイマー）の追加または削除はできません。

境界イベントの追加または削除（その 1）

```
境界非中断（メッセージ）
境界中断（メッセージ）
境界中断（エラー）
```

- 新規にアクティビティ，アクティビティ（マルチインスタンス）を追加する場合，その境界に境界イベントを追加（アタッチ）できます。
- 既存のアクティビティ，アクティビティ（マルチインスタンス）を削除する場合，その境界の境界イベントを削除（アタッチ解除）できます。
- 既存のアクティビティ，アクティビティ（マルチインスタンス）を残す場合，その境界への境界イベントの追加（アタッチ）または削除（アタッチ解除）はできません。
- 新規にサブプロセス，サブプロセス（マルチインスタンス），アドホック・サブプロセスを追加する場合，その境界に境界イベントを追加（アタッチ）できます。
- 既存のサブプロセス，サブプロセス（マルチインスタンス），アドホック・サブプロセスを削除する場合，その境界の境界イベントを削除（アタッチ解除）できます。
- 既存のサブプロセス，サブプロセス（マルチインスタンス），アドホック・サブプロセスを残す場合，その境界への境界イベントの追加（アタッチ）または削除（アタッチ解除）はできません。

境界イベントの追加または削除（その 2）

```
境界非中断（タイマー）
境界中断（タイマー）
```

- 新規にアクティビティ，アクティビティ（マルチインスタンス）を追加する場合，その境界に境界（タイマー）を追加（アタッチ）できます。
- 既存のアクティビティ，アクティビティ（マルチインスタンス）を削除する場合，その境界の境界（タイマー）を削除（アタッチ解除）できます。
- 既存のアクティビティ，アクティビティ（マルチインスタンス）を残す場合，その境界への境界（タイマー）の追加（アタッチ）または削除（アタッチ解除）はできません。
- 新規にサブプロセス，サブプロセス（マルチインスタンス），アドホック・サブプロセスを追加する場合，その境界に境界（タイマー）を追加（アタッチ）できます。ただし，境界（タイマー）を追加（アタッチ）する場合は，-ejp オプションを指定したciwchgenv コマンドでEditJoinPermission にtrue を設定する必要があります。
- 既存のサブプロセス，サブプロセス（マルチインスタンス），アドホック・サブプロセスを削除する場合，その境界の境界（タイマー）を削除（アタッチ解除）できます。ただし，境界（タイマー）を削除（アタッチ解除）する場合は，-ejp オプションを指定したciwchgenv コマンドでEditJoinPermission に true を設定する必要があります。
- 既存のサブプロセス，サブプロセス（マルチインスタンス），アドホック・サブプロセスを残す場合，その境界への境界（タイマー）の追加（アタッチ）または削除（アタッチ解除）はできません。

終了イベントの追加または削除

終了（タイプなし）※
終了（メッセージ）
終了（エラー）
強制終了

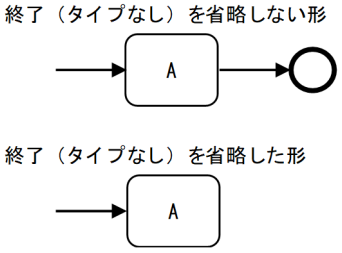
トップレベル（レーン含む），サブプロセス，サブプロセス（マルチインスタンス），イベント・サブプロセスに終了イベントの追加または削除ができます。

注※

省略形を含みます。

次の図の上段が終了（タイプなし）を省略しない形，下段が終了（タイプなし）を省略した形で，どちらも意味は同じです。

図 2-24　終了（タイプなし）の省略形



アクティビティの追加または削除

ユーザタスク
サービスタスク
ビジネスルールタスク
コールアクティビティ

- トップレベル（レーン含む），サブプロセス，サブプロセス（マルチインスタンス），イベント・サブプロセス，アドホック・サブプロセスにアクティビティの追加または削除ができます。
- 追加または削除するアクティビティに遷移元がなく，開始（タイプなし）の省略形も追加・削除される場合，「開始イベントの追加または削除（その 1）」も参照してください。

アクティビティ（マルチインスタンス）の追加または削除

ユーザタスク（マルチインスタンス）
サービスタスク（マルチインスタンス）
ビジネスルールタスク（マルチインスタンス）
コールアクティビティ（マルチインスタンス）

- トップレベル（レーン含む），サブプロセス，イベント・サブプロセス，アドホック・サブプロセスにアクティビティ（マルチインスタンス）の追加または削除ができます。
- 追加または削除するアクティビティ（マルチインスタンス）に遷移元がなく，開始（タイプなし）の省略形も追加・削除される場合，「開始イベントの追加または削除（その 1）」も参照してください。

キャッチイベント，スローイベントの追加または削除

キャッチ（メッセージ）
キャッチ（タイマー）
キャッチ（リンク）
スロー（メッセージ）
スロー（リンク）

トップレベル（レーン含む），サブプロセス，サブプロセス（マルチインスタンス），イベント・サブプロセス，アドホック・サブプロセスにキャッチイベント，スローイベントの追加または削除ができます。

ゲートウェイの追加または削除

排他ゲートウェイ※1
並列ゲートウェイ※2
排他イベントゲートウェイ

トップレベル（レーン含む），サブプロセス，サブプロセス（マルチインスタンス），イベント・サブプロセス，アドホック・サブプロセスにゲートウェイの追加または削除ができます。ただし，並列ゲートウェイ（省略形を含む）を追加または削除する場合は，-ejp オプションを指定したciwchgenv コマンドで EditJoinPermission にtrue を設定する必要があります。

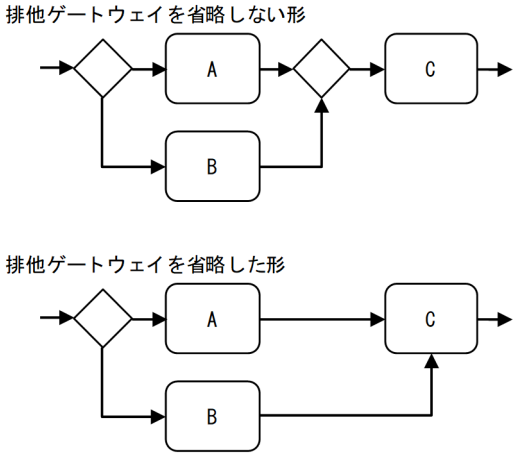
並列ゲートウェイについては，「(3)　並列ゲートウェイに関する変更・追加・削除の可否」を参照してください。

注※1

省略形を含みます。

次の図の上段が排他ゲートウェイを省略しない形，下段が排他ゲートウェイを省略した形で，どちらも意味は同じです。なお，合流する側は省略できますが，分岐する側は省略できません。

図 2-25　排他ゲートウェイの省略形

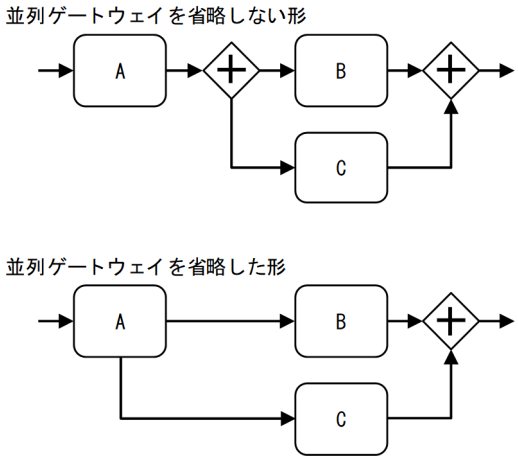


注※2

省略形を含みます。

次の図の上段が並列ゲートウェイを省略しない形，下段が並列ゲートウェイを省略した形で，どちらも意味は同じです。なお，分岐する側は省略できますが，合流する側は省略できません。

図 2-26　並列ゲートウェイの省略形



サブプロセスの追加または削除

- サブプロセスの内部や境界に追加または削除できる BPMN 要素を含む場合にだけ，サブプロセスを追加または削除できます。

- トップレベル（レーン含む），サブプロセス，サブプロセス（マルチインスタンス），イベント・サブプロセスにサブプロセスの追加または削除ができます。
- 追加または削除するサブプロセスに遷移元がなく，開始（タイプなし）の省略形も追加・削除される場合，「開始イベントの追加または削除（その 1）」も参照してください。

サブプロセス（マルチインスタンス）の追加または削除

- サブプロセス（マルチインスタンス）の内部や境界に追加または削除できる BPMN 要素を含む場合にだけ，サブプロセス（マルチインスタンス）を追加または削除できます。
- トップレベル（レーン含む），サブプロセス，イベント・サブプロセスにサブプロセス（マルチインスタンス）の追加または削除ができます。
- 追加または削除するサブプロセス（マルチインスタンス）に遷移元がなく，開始（タイプなし）の省略形も追加・削除される場合，「開始イベントの追加または削除（その 1）」も参照してください。

イベント・サブプロセスの追加または削除

- イベント・サブプロセスの内部に追加または削除できる BPMN 要素を含む場合にだけ，イベント・サブプロセスを追加または削除できます。
- トップレベル（レーン含む），サブプロセス，サブプロセス（マルチインスタンス），イベント・サブプロセスにイベント・サブプロセスの追加または削除ができます。
- イベント・サブプロセスがイベント・サブプロセス開始（タイマー）を含み，それらを追加または削除する場合，イベント・サブプロセスの追加先または削除元に制限があります。制限の詳細は「イベント・サブプロセス開始イベントの追加または削除（その 2）」を参照してください。

アドホック・サブプロセスの追加または削除

- アドホック・サブプロセスの内部や境界に追加または削除できる BPMN 要素を含む場合にだけ，アドホック・サブプロセスを追加または削除できます。
- トップレベル（レーン含む），サブプロセス，サブプロセス（マルチインスタンス），イベント・サブプロセスにアドホック・サブプロセスの追加または削除ができます。
- 追加または削除するアドホック・サブプロセスに遷移元がなく，開始（タイプなし）の省略形も追加・削除される場合，「開始イベントの追加または削除（その 1）」も参照してください。

レーンの追加または削除

レーンの内部に追加または削除できる BPMN 要素を含む場合にだけ，レーンを追加または削除できます。

(2)　BPMN 要素の属性値の変更可否

BPMN 要素の属性値の変更可否を示します。

Id，Name，Documentation 以外の属性値の変更

BPMN 要素ごとの Id，Name，Documentation 以外の属性値の変更可否を次の表に示します。

表 2-2　Id，Name，Documentation 以外の属性値の変更可否

BPMN 要素		属性値（括弧内は BPMN エディタの［プロパティ］タブ上の表示）	
		変更できる属性値	変更できない属性値
イベント	開始（タイプなし）	－	－
	開始（メッセージ）	－	ref 識別子（Message ref）
	開始（タイマー）	－	タイマールール（［Timer Definition］タブのすべての設定）
	イベント・サブプロセス非中断開始（メッセージ） イベント・サブプロセス中断開始（メッセージ）	ref 識別子（Message ref）	中断イベントか非中断イベントかの属性（isInterrupting）
	イベント・サブプロセス中断開始（エラー）	ref 識別子（Error ref）	－
	イベント・サブプロセス非中断開始（タイマー） イベント・サブプロセス中断開始（タイマー）	タイマールール（［Timer Definition］タブのすべての設定）	中断イベントか非中断イベントかの属性（isInterrupting）
	境界非中断（メッセージ） 境界中断（メッセージ）	ref 識別子（Message ref）	中断イベントか非中断イベントかの属性（Cancel activity）
	境界中断（エラー）	ref 識別子（Error ref）	－
	境界非中断（タイマー） 境界中断（タイマー）	タイマールール（［Timer Definition］タブのすべての設定）	中断イベントか非中断イベントかの属性（Cancel activity）
	キャッチ（メッセージ）	ref 識別子（Message ref）	－
	キャッチ（タイマー）	タイマールール（［Timer Definition］タブのすべての設定）	－
	キャッチ（リンク）	－	リンク名（LinkEventDefinition name）
	スロー（メッセージ）	ref 識別子（Message ref）	－
	スロー（リンク）	－	リンク名（LinkEventDefinition name）
	終了（メッセージ）	ref 識別子（Message ref）	－
	終了（エラー）	ref 識別子（Error ref）	－
	終了（タイプなし）	－	－
	強制終了	－	－

BPMN 要素		属性値（括弧内は BPMN エディタの［プロパティ］タブ上の表示）	
		変更できる属性値	変更できない属性値
アクティビティ	ユーザタスク	作業者（[Participan］タブのすべての設定）	－
	サービスタスク ビジネスルールタスク	ref 識別子（Operation ref）	－
	コールアクティビティ	－	calledElement 属性（Called element）
アクティビティ（マルチインスタンス）	ユーザタスク（マルチインスタンス） サービスタスク（マルチインスタンス） ビジネスルールタスク（マルチインスタンス） コールアクティビティ（マルチインスタンス）	パラレルかシーケンシャルか（Sequential）	－
		繰り返し回数（Loop cardinality）	
		完了条件（Completion condition）	
ゲートウェイ	排他ゲートウェイ	デフォルト遷移先（Default flow）	－
	並列ゲートウェイ	－	－
	排他イベントゲートウェイ	－	－
接続オブジェクト	シーケンスフロー	分岐条件（Condition）	－
コンテナ	レーン	－	－
	サブプロセス	－	－
	イベント・サブプロセス	－	－
	アドホック・サブプロセス	インスタンスキャンセル属性（Cancel remaining instances）	フローノード実行方式（Ordering）
		完了条件（Completion condition）	
	サブプロセス（マルチインスタンス）	繰り返し回数（Loop cardinality）※	パラレルかシーケンシャルか（Sequential）
		完了条件（Completion condition）	

（凡例）

－：該当する属性値はありません。

注※

Loop cardinality を変更できるかどうかは，サブプロセス（マルチインスタンス）に次の BPMN 要素を含むかどうかに依存します。

- 複数の開始（タイプなし）（省略形を含む）
- 並列ゲートウェイ（省略形を含む）
- イベント・サブプロセス開始（タイマー）
- サブプロセス，アドホック・サブプロセスにアタッチされた境界（タイマー）

これらの BPMN 要素のどれも含まない場合

Loop cardinality を変更できます。

これらの BPMN 要素のどれかを含み，かつEditJoinPermission がtrue の場合

Loop cardinality を変更できます。

これらの BPMN 要素のどれかを含み，かつEditJoinPermission がfalse（デフォルト）の場合

- Loop cardinality が固定値のとき

  Loop cardinality を変更できません。
- Loop cardinality が固定値でないとき

  ciwtransbpmn コマンドの，次の 2 つのオプションに同じ値を指定すれば Loop cardinality を変更できます。
  - -src オプションまたは-srcdir オプションを指定して実行した際の-mimax オプション（ビジネスプロセス登録当初の値）
  - -update オプションを指定して実行した際の-mimax オプション（ビジネスプロセス変更時の値）

Id，Name の変更

対象の BPMN 要素が追加できて，かつ削除できる場合に Id，Name を変更できます（Id，Name の変更は内部的に削除および追加と同等になるため）。BPMN 要素の追加，または削除の可否については，「(1) BPMN 要素の追加または削除の可否」を参照してください。

Id，Name を変更できる例

- ユーザタスクの Id，Name を変更できます。ただし，ユーザタスクの遷移先に並列ゲートウェイがある場合は，-ejp オプションを指定したciwchgenv コマンドでEditJoinPermission にtrue を設定する必要があります。また，ユーザタスクが実行中の案件が 0 件である必要があります。
- 並列ゲートウェイの Id，Name を変更できます。ただし，-ejp オプションを指定したciwchgenv コマンドでEditJoinPermission にtrue を設定する必要があります。
- イベント・サブプロセスがイベント・サブプロセス開始（メッセージ），サービスタスク，並列ゲートウェイ，終了（タイプなし）で構成されている場合，イベント・サブプロセスの Id，Name を変更できます。ただし，-ejp オプションを指定したciwchgenv コマンドでEditJoinPermission にtrue を設定する必要があります。また，イベント・サブプロセス内部が実行中の案件が 0 件である必要があります。

Documentation の変更

任意の BPMN 要素の Documentation に設定する内容を変更できます。

(3) 並列ゲートウェイに関する変更・追加・削除の可否

-ejp オプションを指定したciwchgenv コマンドを実行してEditJoinPermission にtrue を設定している場合，並列ゲートウェイに関する変更・追加・削除ができます。環境構築時のデフォルト設定（EditJoinPermission にfalse を設定）では，並列ゲートウェイに関する変更・追加・削除はできません。

ciwchgenv コマンドの詳細については，マニュアル『uCosminexus Service Coordinator Interactive Workflow コマンド』を参照してください。

EditJoinPermission の設定値ごとの，並列ゲートウェイに関する変更・追加・削除の可否を次に示します。

表 2-3　並列ゲートウェイの変更可否

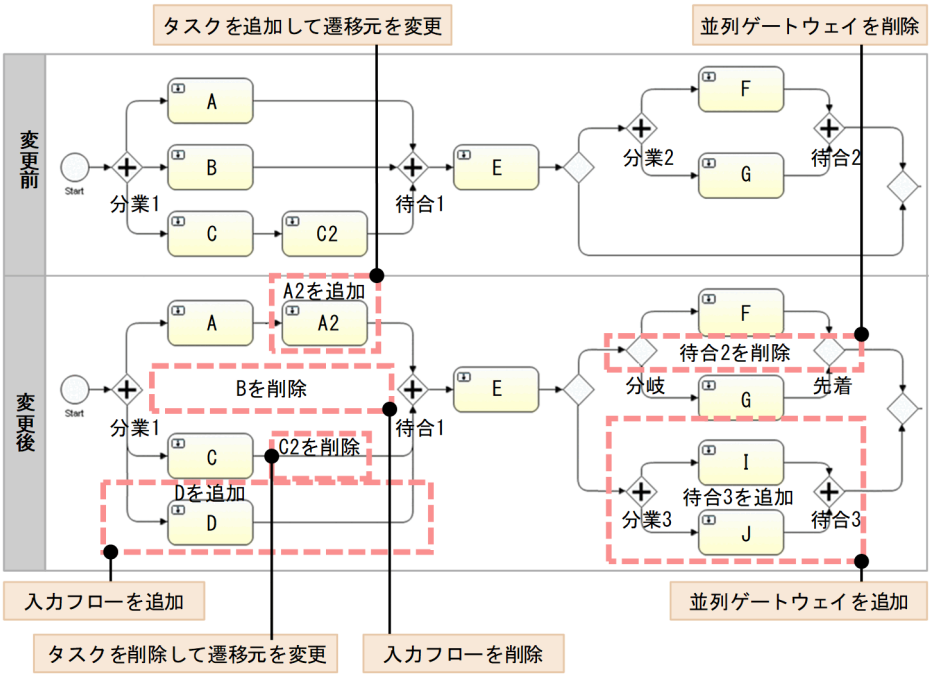
項番	並列ゲートウェイの変更・追加・削除	EditJoinPermission の値	
		False（デフォルト）	true
1	並列ゲートウェイ（分岐）の追加	×	○
2	並列ゲートウェイ（分岐）の削除	×	○
3	並列ゲートウェイ（分岐）の遷移先の変更 • 並列ゲートウェイ（分岐）と遷移先の間に追加 • 並列ゲートウェイ（分岐）の遷移先を削除	○	○
4	並列ゲートウェイ（分岐）からの出力フローの追加	○	○
5	並列ゲートウェイ（分岐）からの出力フローの削除	○	○
6	並列ゲートウェイ（合流）の追加	×	○
7	並列ゲートウェイ（合流）の削除	×	○
8	並列ゲートウェイ（合流）の遷移元の変更 • 遷移元と並列ゲートウェイ（合流）の間に追加 • 並列ゲートウェイ（合流）の遷移元を削除	×	○
9	並列ゲートウェイ（合流）への入力フローの追加	×	○
10	並列ゲートウェイ（合流）への入力フローの削除	×	○

（凡例）

○：並列ゲートウェイに関する変更・追加・削除ができる

×：並列ゲートウェイに関する変更・追加・削除ができない

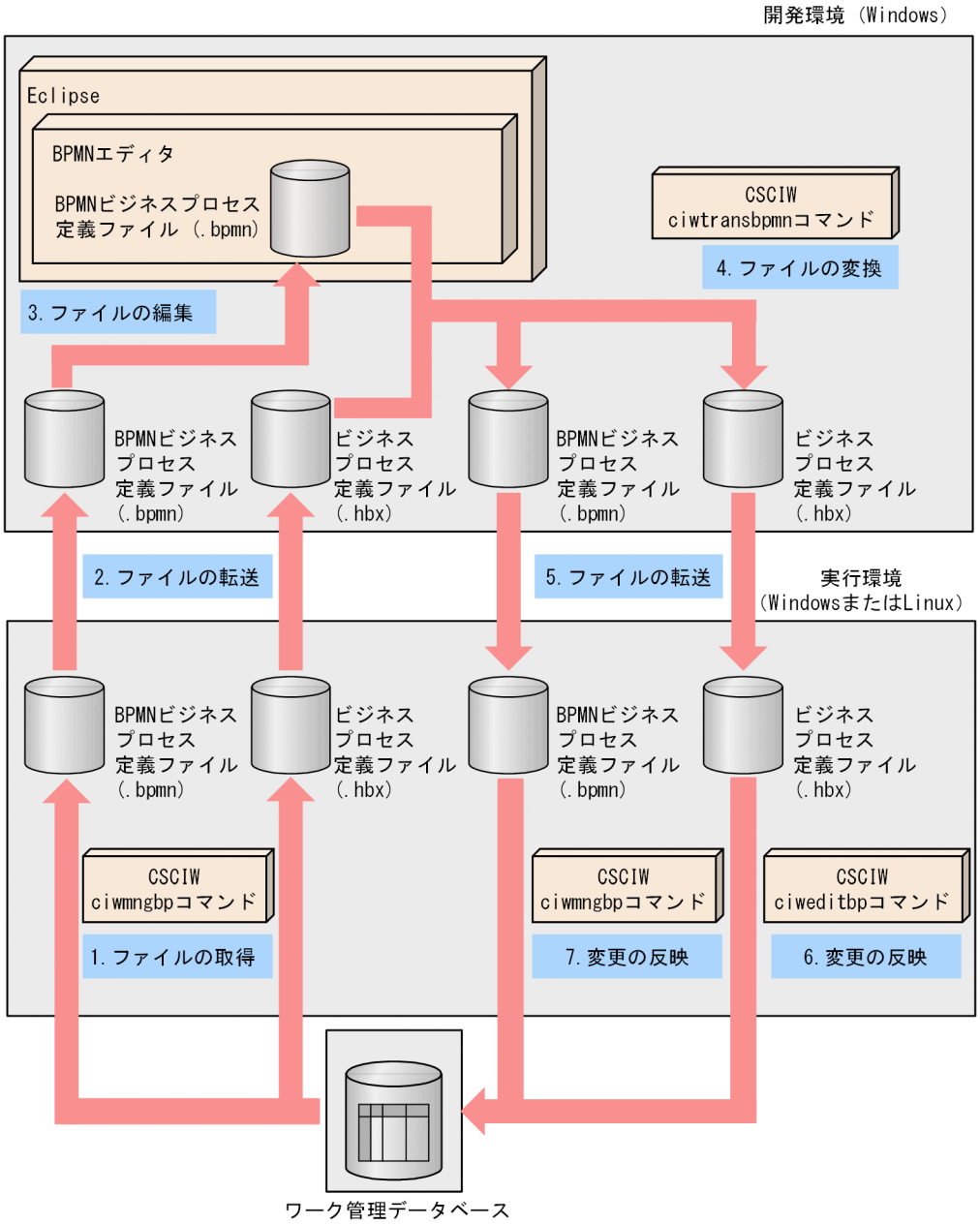
図 2-27　並列ゲートウェイに関する変更を行う例



2.6.3　ビジネスプロセス定義の変更の流れ

ビジネスプロセス定義の変更手順の流れを説明します。

図 2-28　ビジネスプロセス定義変更の流れ



1. -get オプションを指定したciwmngbp コマンドを実行して，データベースに登録された CSCIW のビジネスプロセス定義ファイルおよび BPMN ビジネスプロセス定義ファイルを取得する。

   ciwmngbp コマンドの詳細については，マニュアル『uCosminexus Service Coordinator Interactive Workflow コマンド』を参照してください。

2. 1.で取得した CSCIW のビジネスプロセス定義ファイルおよび BPMN ビジネスプロセス定義ファイルを，開発環境に転送する。

3. BPMN エディタで，2.で転送した BPMN ビジネスプロセス定義ファイルを編集する。

4. -update オプションを指定したciwtransbpmn コマンドを実行して，編集済みの BPMN ビジネスプロセス定義を変換して CSCIW のビジネスプロセス定義を生成する。

   -update オプションには，3.で編集した BPMN ビジネスプロセス定義ファイルを指定します。

-bpf オプションには，2.で転送した CSCIW のビジネスプロセス定義ファイルを指定します。

実行結果として，BPMN ビジネスプロセス定義ファイルと CSCIW のビジネスプロセス定義ファイルが生成されます。

ciwtransbpmn コマンドの詳細については，マニュアル『uCosminexus Service Coordinator Interactive Workflow コマンド』を参照してください。

5. 実行環境の任意のディレクトリに 4.で生成された CSCIW のビジネスプロセス定義ファイルおよび BPMN ビジネスプロセス定義ファイルを転送する。
6. -edt オプションを指定したciweditbp コマンドを実行して，データベースのビジネスプロセス定義を更新する。

   -bpf オプションには，5.で転送した CSCIW のビジネスプロセス定義ファイルを指定します。

   ciweditbp コマンドの詳細については，マニュアル『uCosminexus Service Coordinator Interactive Workflow コマンド』を参照してください。
7. -regbpmn オプションを指定したciwmngbp コマンドを実行して，データベースの BPMN ビジネスプロセス定義を更新する。

   -bpmnf オプションには，5.で転送した BPMN ビジネスプロセス定義ファイルを指定します。

2.6.4 複数のビジネスプロセスを修正する場合の注意事項

1 つの BPMN ビジネスプロセス定義ファイルに含まれる複数のビジネスプロセスを修正する場合の注意事項を説明します。

「2.6.3 ビジネスプロセス定義の変更の流れ」の手順 2.で開発環境に定義ファイルを転送する際は，修正対象のビジネスプロセスに対応した，次に示すファイル名の BPMN ビジネスプロセス定義ファイルを取得する必要があります（複数のビジネスプロセスを修正する場合は複数の BPMN ビジネスプロセス定義ファイルを取得し，編集する必要があります）。

<ビジネスプロセス名>#<ビジネスプロセス定義バージョン>.bpmn

ただし，次に示す条件をすべて満たす場合は，複数の BPMN ビジネスプロセス定義ファイルを取得しないで，修正対象のビジネスプロセスをすべて含む BPMN ビジネスプロセス定義ファイルを 1 ファイルだけ取得して編集することもできます。

- 同時に複数のビジネスプロセスを修正する
- 修正対象のビジネスプロセスに対応した BPMN ビジネスプロセス定義ファイルの内容がすべて同一である
- BPMN エディタでの修正時にプロセス（プール）を追加しない
- ciwtransbpmn コマンドの-src オプションに，修正した BPMN ビジネスプロセス定義ファイルを指定しない
- ciwtransbpmn コマンドの-srcdir オプションに，修正した BPMN ビジネスプロセス定義ファイルが含まれるディレクトリを指定しない

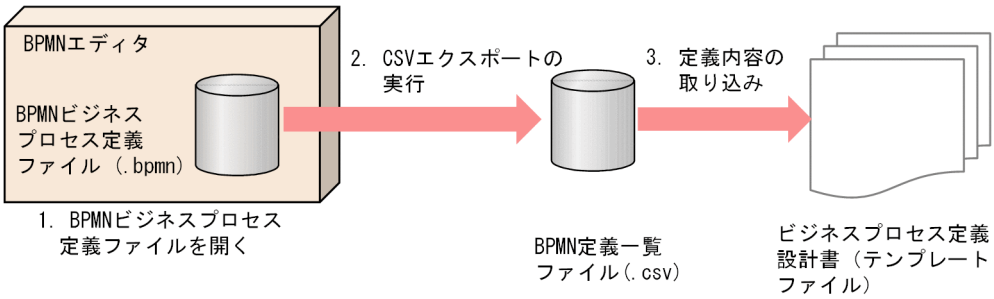
なお，BPMN ビジネスプロセス定義ファイルを 1 つだけ編集して複数のビジネスプロセスを修正した場合も，-update オプションで生成されたすべての BPMN ビジネスプロセス定義ファイルを実行環境に転送する必要があります。

2.7　ビジネスプロセス定義のエクスポート

ビジネスプロセス定義の定義内容は，BPMN エディタから CSV ファイル（BPMN 定義一覧ファイル）にエクスポートできます。この機能を CSV エクスポートと呼びます。エクスポートした CSV ファイルをビジネスプロセス定義設計書などの作成に利用できます。

ビジネスプロセス定義の定義内容を CSV ファイル（BPMN 定義一覧ファイル）にエクスポートして，ビジネスプロセス定義設計書を作成する流れを次の図で示します。

図 2-29　ビジネスプロセス定義をエクスポートしてビジネスプロセス定義設計書を作成する流れ



[説明]

1. BPMN エディタで，ビジネスプロセス定義設計書を作成したい BPMN ビジネスプロセス定義ファイルを開きます。
2. CSV エクスポートを実行すると，ビジネスプロセス定義の定義内容が，BPMN 定義一覧ファイルに CSV 形式で出力されます。
3. Microsoft Excel や Microsoft Word のマクロなどを使用して，ビジネスプロセス定義の定義内容を，BPMN 定義一覧ファイルからビジネスプロセス定義設計書に取り込みます。

2.7.1　ビジネスプロセス定義を CSV ファイルにエクスポートする

CSV エクスポートの実行方法と実行結果について説明します。

CSV エクスポートの実行方法

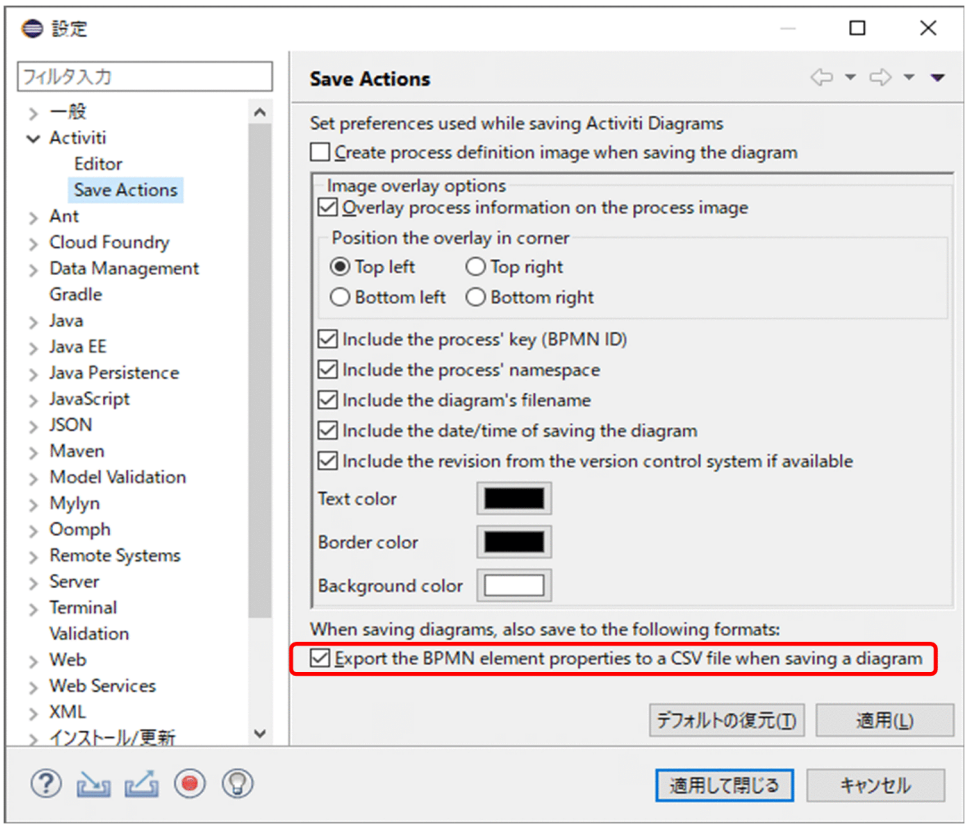
次に示すどちらかの方法で，ビジネスプロセス定義の定義内容を CSV ファイルにエクスポートします。

- BPMN エディタのメニューから実行

  BPMN エディタ上でキャンバスを右クリックすると表示されるメニュー［Export CSV］を実行します。メニュー実行時にキャンバスに表示されているビジネスプロセス定義の定義内容が，CSV ファイルに出力されます。
- BPMN エディタで BPMN ビジネスプロセス定義ファイルを保存する時に自動実行

  BPMN エディタの［ウィンドウ］メニューから［設定］－［Activiti］－［Save Actions］を選択します。[Export the BPMN element properties to a CSV file when saving a diagram］チェック

ボックスにチェックを入れると，CSV ファイルへのエクスポートを自動実行する設定が有効になります。この設定を有効にした状態で BPMN ビジネスプロセス定義ファイルを保存すると，ビジネスプロセス定義の定義内容は CSV ファイルに自動でエクスポートされます。



CSV ファイルへのエクスポートを自動実行したくない場合は，［Export the BPMN element properties to a CSV file when saving a diagram］チェックボックスのチェックを外してください。

CSV エクスポートの実行結果

CSV エクスポートの実行結果は，Eclipse コンソールに出力されます。Eclipse コンソールが BPMN エディタの画面上に表示されていない場合は，Eclipse コンソールが表示されます。CSV エクスポートの実行が失敗した場合は，エラーメッセージが表示されます。

なお，エクスポートを自動実行する設定をして BPMN ビジネスプロセス定義ファイルを保存した場合に，CSV エクスポートの自動実行に失敗しても，BPMN ビジネスプロセス定義ファイルは保存されます。

2.7.2　BPMN 定義一覧ファイルの格納先ディレクトリとファイル名

BPMN 定義一覧ファイルは，BPMN ビジネスプロセス定義ファイルを保存するディレクトリに格納されます。ファイルの格納先にすでに同名のファイルがある場合は，上書きされます。

BPMN 定義一覧ファイルのファイル名は，BPMN ビジネスプロセス定義ファイルと同じ名前で拡張子が「.csv」です。

2.7.3　BPMN 定義一覧ファイルの詳細

BPMN 定義一覧ファイルの詳細を次に示します。

(1)　BPMN 定義一覧ファイルの記述形式

BPMN 定義一覧ファイルの記述形式を次に示します。

- プロパティの値は「列」，BPMN 要素または定義種別ごとの値は「行」に記述されます。
- 1 行目にはヘッダ情報が記述され，2 行目以降には各ヘッダ情報に対応するプロパティの値が記述されます。
- データの区切りはコンマです。
- 値はすべて，半角ダブルクォーテーション「"」で囲まれます。ただし，プロパティの値が未設定，または空文字列が設定されている場合には，そのプロパティについては何も記述されません。

  また，プロパティの値の中に半角ダブルクォーテーション「"」が含まれる場合は，プロパティの値の中の「"」の直前に「"」が 1 つ付加されて，「""」となります。
- 改行コードは「CRLF」です。
- 文字コードは UTF-8 です。

> **ヒント**
>
> BPMN 定義一覧ファイルを Excel で開いた場合，文字コードが UTF-8 であっても S-JIS として認識されます。この状態で Excel で編集作業をすると，BPMN 定義一覧ファイル中に UTF-8 と S-JIS の文字コードが混在します。 Excel で BPMN 定義一覧ファイルの文字コード UTF-8 としてを開く方法の例を次に示します。
>
> ［データ］－［データの取得と変換］グループの［テキストまたは CSV から］で「Unicode (UTF-8)」を指定

BPMN 定義一覧ファイルの記述例を次に示します。この例では，「"""スタートイベント""です"」は，プロパティ値が「"スタートイベント"です」であることを表しています。

```
"Id","Type","Name","Documentation","Default flow","MessageRef","OperationRef","ErrorRef", …
"pool1","Pool","プール1","プール1です",,,,, …
"lane1","Lane","レーン1","レーン1です",,,,, …
"UTask1","UserTask","ユーザタスク1","ユーザタスク1です",,,,, …
"STask1","ServiceTask","サービスタスク1","サービスタスク1です","flow1",,"ope1",, …
"STask2","ServiceTask","サービスタスク2","サービスタスク2です",,,"ope2",, …
"Start1","TopLevelStart","スタート","""スタートイベント""です",,,,, …
```

(2)　BPMN 定義一覧ファイルのプロパティ値

BPMN 定義一覧ファイルのプロパティ値は，BPMN エディタのタブで設定できる値です。BPMN 定義一覧ファイルのプロパティ値は，ここで示す記述形式でエクスポートされます。

BPMN 定義一覧ファイルのプロパティ値の記述形式，およびプロパティ値を設定できるタブを次に示します。なお，BPMN 定義一覧ファイルの見出し行には，次の表のプロパティ名が記述されます。

表 2-4　プロパティ名とプロパティ値の記述形式および設定できるタブ

プロパティ名	プロパティ値の記述形式	設定できるタブ
Id	文字列	• General • Process • Messages • Errors
Type	文字列 BPMN 要素または定義種別ごとのType 列の値については，「表 2-5　BPMN 要素または定義種別ごとのプロパティ値と Type 列の値およびプロパティを設定できるタブの関係」を参照してください。	—
Name	文字列	• General • Process • Messages • Errors
Documentation	文字列	• Documentation • Process
Default flow	文字列	General
Message ref	文字列	Main config
Operation ref	文字列	Main config
Error ref	文字列	Main config
Called element	文字列	Main config
Text	文字列	Main config
Condition	文字列	Main config
LinkEventDefinition name	文字列	Main config
ErrorCode	文字列	Errors
Sequential	Boolean 次のどちらかの文字列になります。 • true • false	Multi instance
Loop cardinality	文字列	Multi instance
Completion condition	文字列	• Main config • Multi instance
Ordering	次のどちらかの文字列になります。	Main config

プロパティ名	プロパティ値の記述形式	設定できるタブ
Ordering	• Parallel • Sequential	Main config
Cancel remaining instances	Boolean 次のどちらかの文字列になります。 • true • false	Main config
Timer Definition Type	BPMN エディタ上でのタイマールールの設定によって，次のどれかの文字列になります。 • Fixed date and time タイマールールの設定が「Fixed date and time」の場合 • Duration タイマールールの設定が「Duration」の場合 • Periodic date and time タイマールールの設定が「Periodic date and time」の場合 BPMN エディタ上でタイマールールが設定されていない場合は空になります。	TimerDefinition
Fixed date and time	次の形式の文字列です。※ yyyy-MM-ddThh:mm:00 BPMN エディタ上でタイマールールが設定されていない場合，または「Fixed date and time」以外を指定している場合は空になります。 BPMN エディタで指定する場合は，秒フィールドは「00」固定です。	TimerDefinition
Duration	次のどれかの形式の文字列です。※ • Pn_yY • Pn_mM • Pn_dD • PTn_hH • $PTn_{mi}M$ BPMN エディタ上でタイマールールが設定されていない場合，または「Duration」以外を指定している場合は空になります。	TimerDefinitionx
Periodic date and time	次の形式の文字列です。※ *-M-dTh:m BPMN エディタ上でタイマールールが設定されていない場合，または「Periodic date and time」以外を指定している場合は空になります。	TimerDefinition
Number of executions	Boolean • true	TimerDefinition

プロパティ名	プロパティ値の記述形式	設定できるタブ
Number of executions	BPMN エディタ上で［Specify the number of executions］ラジオボタンが選択されている場合 • false BPMN エディタ上で［Specify the number of executions］ラジオボタンが選択されていない場合 BPMN エディタ上でタイマールールが設定されていない場合，または「Fixed date and time」が指定されている場合は空になります。	TimerDefinition
Specify the number of executions	1～9999 の整数です。 BPMN エディタ上でタイマールールが設定されていない場合，「Fixed date and time」が指定されている場合，または［Specify the number of executions］ラジオボタンが選択されていない場合は空になります。	TimerDefinition
Use the process data	Boolean ［TimerDefinition］タブの場合 • true BPMN エディタ上で［Use the process data as the timer rule］チェックボックスがチェックされているとき • false BPMN エディタ上で［Use the process data as the timer rule］チェックボックスがチェックされていないとき BPMN エディタ上でタイマールールが設定されていない場合，空になります。 ［Participant］タブの場合 • true BPMN エディタ上で［Use the process data as the participant］チェックボックスがチェックされているとき • false BPMN エディタ上で［Use the process data as the participant］チェックボックスがチェックされていないとき BPMN エディタ上でユーザタスクが設定されていない場合，空になります。	• TimerDefinition • Participant
Process data key name	文字列 BPMN エディタ上で，［TimerDefinition］タブの［Use the process data as the timer rule］ラジオボタン，または［Participant］タブの［Use the process data as the participant］ラ	• TimerDefinition • Participant

プロパティ名	プロパティ値の記述形式	設定できるタブ
Process data key name	ジオボタンが選択されていない場合は空になります。	• TimerDefinition • Participant

（凡例）

－：該当なし

注※

文字列の形式については，「1.4.3　タイマーイベントのタイマールール動的変更」を参照してください。

BPMN 要素または定義種別ごとに，エクスポートされるプロパティ値とそのときにType 列に出力される値，およびプロパティ値を設定できるタブについて次に示します。

表 2-5　BPMN 要素または定義種別ごとのプロパティ値と Type 列の値およびプロパティを設定できるタブの関係

カテゴリ	BPMN 要素・定義種別	Type 列の値	設定できるタブ	プロパティ値
イベント	開始（タイプなし）	Top-LevelStart	General	• Id • Name
			Documentation	Documentation
	開始（メッセージ）	Top-LevelMessage	General	• Id • Name
			Documentation	Documentation
			Main config	Message ref
	イベント・サブプロセス非中断開始（メッセージ）※1	EvSub-Non-InterruptingMessage	General	• Id • Name
			Documentation	Documentation
			Main config	Message ref
	イベント・サブプロセス中断開始（メッセージ）※1	EvSub-InterruptingMessage	General	• Id • Name
			Documentation	Documentation
			Main config	Message ref
	イベント・サブプロセス中断開始（エラー）	EvSub-InterruptingError	General	• Id • Name
			Documentation	Documentation
			Main config	Error ref
	開始（タイマー）	Top-LevelTimer	General	• Id • Name

カテゴリ	BPMN 要素・定義種別	Type 列の値	設定できるタブ	プロパティ値
イベント	開始（タイマー）	Top-LevelTimer	Documentation	Documentation
			Timer Definition	• Timer Definition Type • Fixed date and time • Duration • Periodic date and time • Number of executions • Specify the number of executions • Use the process data as the timer rule • Process data key name
	イベント・サブプロセス非中断開始（タイマー） ※1	EvSub-Non-InterruptingTimer	General	• Id • Name
			Documentation	Documentation
			Timer Definition	• Timer Definition Type • Fixed date and time • Duration • Periodic date and time • Number of executions • Specify the number of executions • Use the process data as the timer rule • Process data key name
	イベント・サブプロセス中断開始（タイマー） ※1	EvSub-InterruptingTimer	General	• Id • Name
			Documentation	Documentation
			Timer Definition	• Timer Definition Type

カテゴリ	BPMN 要素・定義種別	Type 列の値	設定できるタブ	プロパティ値
イベント	イベント・サブプロセス中断開始（タイマー） ※1	EvSub-InterruptingTimer	Timer Definition	• Fixed date and time • Duration • Periodic date and time • Number of executions • Specify the number of executions • Use the process data as the timer rule • Process data key name
	キャッチ（メッセージ）	CatchingMessage	General	• Id • Name
			Documentation	Documentation
			Main config	Message ref
	キャッチ（リンク）	CatchingLink	General	• Id • Name
			Documentation	Documentation
			Main config	LinkEventDefinition name
	キャッチ（タイマー）	CatchingTimer	General	• Id • Name
			Documentation	Documentation
			Timer Definition	• Timer Definition Type • Fixed date and time • Duration • Periodic date and time • Number of executions • Specify the number of executions • Use the process data as the timer rule

カテゴリ	BPMN 要素・定義種別	Type 列の値	設定できるタブ	プロパティ値
イベント	キャッチ（タイマー）	CatchingTimer	Timer Definition	• Process data key name
	スロー（メッセージ）	ThrowingMessage	General	• Id • Name
			Documentation	Documentation
			Main config	Message ref
	スロー（リンク）	ThrowingLink	General	• Id • Name
			Documentation	Documentation
			Main config	LinkEventDefinition name
	境界非中断（メッセージ） ※2	BoundEv-Non-InterruptingMessage	General	• Id • Name
			Documentation	Documentation
			Main config	Message ref
	境界中断（メッセージ） ※2	BoundEv-InterruptingMessage	General	• Id • Name
			Documentation	Documentation
			Main config	Message ref
	境界中断（エラー）	BoundEv-InterruptingError	General	• Id • Name
			Documentation	Documentation
			Main config	Error ref
	境界非中断（タイマー） ※2	BoundEv-Non-InterruptingTimer	General	• Id • Name
			Documentation	Documentation
			Timer Definition	• Timer Definition Type • Fixed date and time • Duration • Periodic date and time • Number of executions

カテゴリ	BPMN 要素・定義種別	Type 列の値	設定できるタブ	プロパティ値
イベント	境界非中断（タイマー） ※2	BoundEv-Non-InterruptingTimer	Timer Definition	• Specify the number of executions • Use the process data as the timer rule • Process data key name
	境界中断（タイマー） ※2	BoundEv-InterruptingTimer	General	• Id • Name
			Documentation	Documentation
			Timer Definition	• Timer Definition Type • Fixed date and time • Duration • Periodic date and time • Number of executions • Specify the number of executions • Use the process data as the timer rule • Process data key name
	終了（タイプなし）	EndEvent-None	General	• Id • Name
			Documentation	Documentation
	終了（メッセージ）	EndEvent-Message	General	• Id • Name
			Documentation	Documentation
			Main config	Message ref
	終了（エラー）	EndEvent-Error	General	• Id • Name
			Documentation	Documentation
			Main config	Error ref
	強制終了	EndEvent-Terminate	General	• Id • Name

カテゴリ	BPMN 要素・定義種別	Type 列の値	設定できるタブ	プロパティ値
イベント	強制終了	EndEvent-Terminate	Documentation	Documentation
アクティビティ	ユーザタスク	UserTask	General	• Id • Name • Default flow
			Documentation	Documentation
			Multi instance	• Sequential • Loop cardinality • Completion condition
			Participant	• Use the process data as the participant • Process data key name
	サービスタスク	ServiceTask	General	• Id • Name • Default flow
			Documentation	Documentation
			Main config	Operation ref
			Multi instance	• Sequential • Loop cardinality • Completion condition
	ビジネスルールタスク	BusinessRuleTask	General	• Id • Name • Default flow
			Documentation	Documentation
			Main config	Operation ref
			Multi instance	• Sequential • Loop cardinality • Completion condition
	コールアクティビティ	CallActivity-Collapsed	General	• Id • Name • Default flow
			Documentation	Documentation
			Main config	Called element

カテゴリ	BPMN 要素・定義種別	Type 列の値	設定できるタブ	プロパティ値
アクティビティ	コールアクティビティ	CallActivity-Collapsed	Multi instance	• Sequential • Loop cardinality • Completion condition
ゲートウェイ	並列ゲートウェイ	ParallelGateway	General	• Id • Name
			Documentation	Documentation
	排他ゲートウェイ	ExclusiveGateway	General	• Id • Name • Default flow
			Documentation	Documentation
	排他イベントゲートウェイ	Event-BasedGateway	General	• Id • Name
			Documentation	Documentation
接続オブジェクト	シーケンスフロー	SequenceFlow	General	• Id • Name
			Documentation	Documentation
			Main config	Condition
	デフォルトシーケンスフロー	DefaultSequenceFlow	General	• Id • Name
			Documentation	Documentation
	メッセージフロー	MessageFlow	General	• Id • Name
	関連	Association	General	Id
	データの関連	DataAssociation	General	• Id • Name
			Documentation	Documentation
データ	データオブジェクト	DataObject	General	• Id • Name
			Documentation	Documentation
	データストア	DataStore	General	• Id • Name
			Documentation	Documentation
コンテナ	プール	Pool	General	• Id

カテゴリ	BPMN 要素・定義種別	Type 列の値	設定できるタブ	プロパティ値
コンテナ	プール	Pool	General	• Name
	レーン	Lane	General	• Id • Name
	サブプロセス※3	SubProcess	General	• Id • Name • Default flow
			Documentation	Documentation
			Multi instance	• Sequential • Loop cardinality • Completion condition
	イベント・サブプロセス	EventSubProcess	General	• Id • Name
			Documentation	Documentation
	アドホック・サブプロセス ※4	AdHocSubProcess	General	• Id • Name • Default flow
			Documentation	Documentation
			Main config	• Completion condition • Ordering • Cancel remaining instances
その他	テキスト注釈	TextAnnotation	General	Id
			Main config	Text
	グループ	Group	General	• Id • Name
			Documentation	Documentation
	プロセス	Process	Process	• Id • Name • Documentation
	メッセージ定義	MessageDefinitions	Messages	• Id • Name
	エラー定義	ErrorDefinitions	Errors	• Id • Name • ErrorCode

注※1

Type 列の値は，isInterrupting プロパティの値によって変わり，isInterrupting プロパティの値が true の場合は中断イベント，false の場合は非中断イベントとして出力されます。なお，isInterrupting は BPMN 定義一覧ファイルには出力されません。

注※2

Type 列の値は，Cancel activity プロパティの値によって変わり，Cancel activity プロパティの値が true の場合は中断イベント，false の場合は非中断イベントとして出力されます。なお，Cancel activity は BPMN 定義一覧ファイルには出力されません。

注※3

展開されたサブプロセスと折りたたまれたサブプロセスは区別しません。

注※4

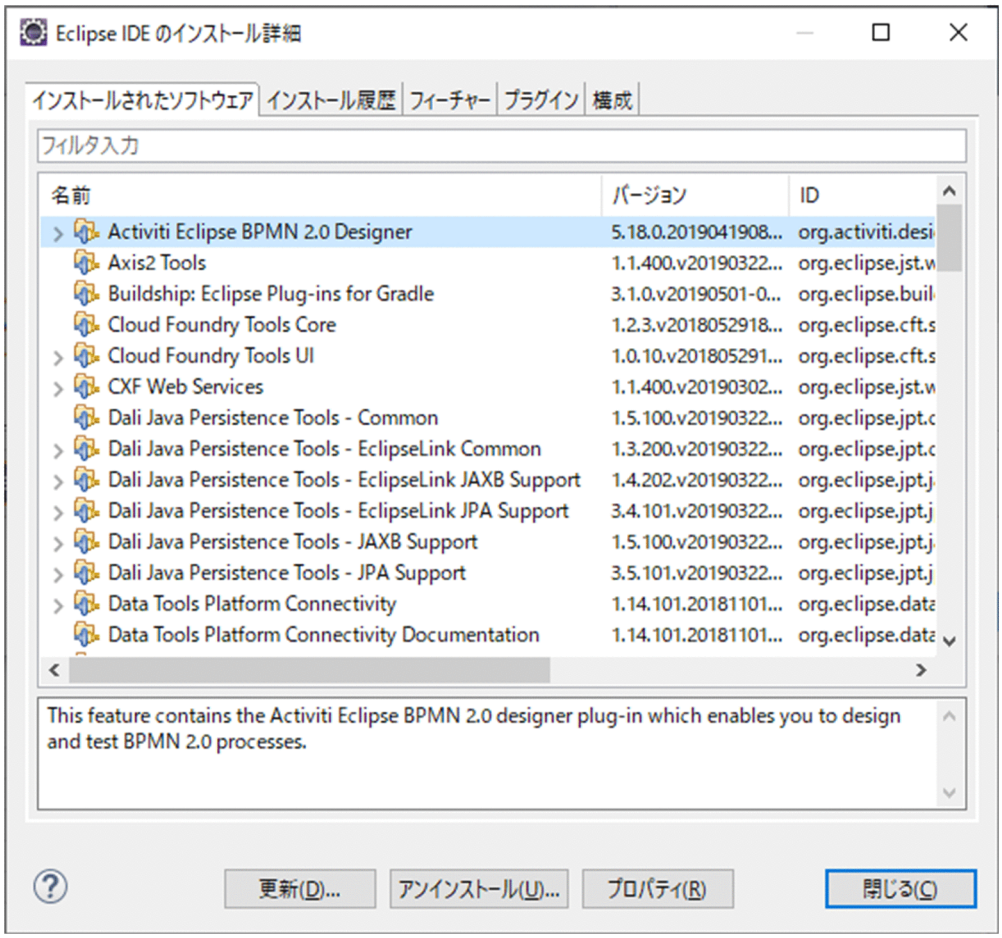
展開されたアドホック・サブプロセスと折りたたまれたアドホック・サブプロセスは区別しません。

2.8　BPMN エディタをアンインストールする

BPMN エディタをアンインストールする手順を示します。

操作手順

1. Eclipse のメニューの［ヘルプ］－［Eclipse IDE について］から［インストール詳細］ボタンをクリックする。
   ［Eclipse IDE のインストール詳細］ダイアログが表示されます。
2. ［インストールされたソフトウェア］タブを選択する。
3. 「Activiti Eclipse BPMN 2.0 Designer」を選択し，［アンインストール］ボタンをクリックする。



4. 画面に従ってアンインストールを進める。

3 ワークフローシステムを設計する

実行環境での見積もりや，運用手順の設計について説明します。

3.1　実行環境に設定する値を見積もる

実行環境に設定する各値を見積もります。

ヒント

必要に応じて，アプリケーション呼び出しサービスの性能チューニングの実施を検討してください。

関連項目

- 付録 C　アプリケーション呼び出しサービスの性能チューニング

3.1.1　BPMN 連携機能に使用するワーク管理データベース容量を見積もる

BPMN 連携機能に使用するワーク管理データベース容量を見積もります。

操作手順

1. マニュアル『uCosminexus Service Coordinator Interactive Workflow システム構築・運用ガイド』の「付録 B　テーブル容量の見積もり」に記載されているテーブルの容量について，容量を見積もる。
2. 「付録 A　テーブル容量の見積もり」に記載されているテーブル容量について，容量を見積もる。

重要

データベースの容量に大きく影響する次のテーブルは他のテーブルより詳細に見積もってください。

- インスタンステーブル

  レコード数が多くなるため，テーブル容量が大きくなります。
- BPMN ビジネスプロセス定義テーブル

  `BpmnProcessDef` カラムの列長が 4 メガバイトと大きいため，テーブル容量が大きくなります。

  `BpmnProcessDef` カラムには BPMN ビジネスプロセス定義ファイルが格納されます。ワーク管理データベースに登録する BPMN ビジネスプロセス定義ファイルのサイズを基に見積もってください。

3.1.2　データベースコネクション数を見積もる

データベースコネクション数を見積もります。

操作手順

1. マニュアル『uCosminexus Service Coordinator Interactive Workflow システム構築・運用ガイド』の「4.5.1　事前準備」を参考に，データベースコネクション数を見積もる。

2. BPMN 連携機能に関連するサービスで必要となるデータベースコネクション数を見積もる。

   BPMN 連携機能に関連するサービスで必要とするコネクション数については，次に示す計算式から求めてください。

   REST サービス

   ```
   RESTサービスで使用するデータベースコネクション数 ＝〈REST APIの同時実行数〉
   ```

4 REST API を使用して業務アプリケーションを開発する

REST API を使用して業務アプリケーションを開発する場合の，REST API の使い方について説明します。

4.1　REST API を使用した業務処理の実装

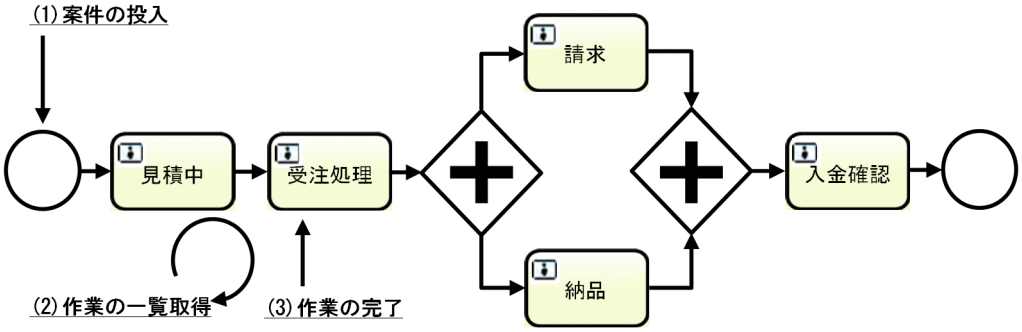
販売業務ビジネスプロセスの例を使用して，業務処理を実装する際の REST API の使い方を示します。

ここで説明する例には，RESTful Web サービス用クライアント API を使用します。また，実装例として記述しているサンプルプログラムは，推奨モードを使用した場合のものです。

RESTful Web サービス用クライアント API の詳細は，マニュアル『Cosminexus アプリケーションサーバ Web サービス開発ガイド』およびマニュアル『Cosminexus 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）』を参照してください。

次の図で示す販売業務ビジネスプロセスを，BPMN エディタで定義した場合について説明します。このビジネスプロセスは，案件投入すると開始されます。業務アプリケーションの案件投入の REST API を使って，ビジネスプロセスを操作します。

図 4-1　販売業務ビジネスプロセス



4.1.1　案件の投入（REST API の使い方 1）

REST API を使用した案件の投入について説明します。

案件を投入するには，/v1/process-instance/create-and-start の URL にPOST を発行します。案件のパラメタをリクエストボディに XML 形式で指定して実行します。実行が成功すると，生成された案件が返却されます。XML で返却された案件データは，Java のオブジェクトにマッピングされます。

案件投入 URL

```
String postURL = new String(targetURL + "/v1/process-instance/create-and-start");
```

案件投入パラメタ

```
CreateAndStartProcessInstance parameter = new CreateAndStartProcessInstance();
parameter.setName("process1");
parameter.setDeadline("2020-12-31T12:00:00+09:00");
parameter.setPriority("10");
parameter.setProcessDataList(null);
parameter.setDefinitionName("販売業務");
```

```
parameter.setDefinitionVersion("1");
List<ProcessData> processData = new ArrayList<ProcessData>();
ProcessData data1 = new ProcessData();
data1.setKey("$SAPP");
data1.setValue("Y");
processData.add(data1);
ProcessData data2 = new ProcessData();
data2.setKey("$NID");
data2.setValue("1");
processData.add(data2);
parameter.setProcessDataList(processData);

Entity<?> entity = Entity.entity(parameter, MediaType.APPLICATION_XML);
```

案件投入

```
WebTarget targetPost = client.target(postURL);
Response postResponse = targetPost
                        .request()
                        .accept(MediaType.APPLICATION_XML)
                        .post(entity);
```

案件 ID の取得

```
processInstance = postResponse.readEntity(ProcessInstance.class);
String piid = processInstance.getID();
```

4.1.2　作業の一覧取得（REST API の使い方 2）

REST API を使用した作業の一覧取得について説明します。

作業の一覧を取得するには，/v1/work-item の URL にGET を発行します。パラメタに案件 ID を指定して実行すると，案件の作業一覧が返却されます。XML で返却された案件データは，Java のオブジェクトにマッピングされます。

作業の一覧取得 URL

```
String getURL = new String(targetURL + "/v1/work-item"
                                     + "?filter=ProcessInstanceID%3D" + piid);
```

作業の一覧取得リクエスト

```
WebTarget targetGet= client.target(getURL);
ClientResponse getResponse = targetGet
                             .request()
                             .accept(MediaType.APPLICATION_XML)
                             .get();
```

作業の一覧を取得

```
WorkItemList workItemList = getResponse.readEntity(WorkItemList.class);

List<WorkItem> workItems = workItemList.getWorkItems();
Iterator<WorkItem> it = workItems.iterator();
while (it.hasNext()) {
    WorkItem wi = it.next();
    String WIID = wi.getID();
    :
}
```

4.1.3　作業の完了（REST API の使い方 3）

REST API を使用した作業の完了について説明します。

作業を完了するには，/v1/work-item/(PIID)/(WKID)/perform-and-complete の URL にPUT を発行します。リクエストボディにパラメタを指定して実行すると，作業が完了します。XML で返却された作業は，Java のオブジェクトにマッピングされます。

作業の完了 URL

```
String putURL = new String(targetURL + "/v1/work-item/" + piid + "/" + wi.getID()
                                     + "/perform-and-complete");
```

作業の完了

```
entity = Entity.entity( "<Parameter />", MediaType.APPLICATION_XML);
WebTarget targetPut = client.target(putURL);
ClientResponse putResponse = targetPut
                             .request()
                             .accept(MediaType.APPLICATION_XML)
                             .put(entity);
```

作業の取得

```
WorkItem targetWorkItem = putResponse.readEntity(WorkItem.class);
```

4.2 Web リソースクライアントの実装例

販売業務ビジネスプロセスを実行する Web リソースクライアントの実装例を示します。

案件を投入したあと，その案件の作業一覧を取得し，作業を完了させる例です。

SampleRESTClient.java

```
package rest.sample;

import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

import javax.ws.rs.core.MediaType;
import javax.ws.rs.core.Response;
import javax.ws.rs.client.Client;
import javax.ws.rs.client.ClientBuilder;
import javax.ws.rs.client.Entity;
import javax.ws.rs.client.WebTarget;

public class SampleRESTClient {

    public static void main(final String[] args) {
        try {
            SampleRESTClient sampleClient = new SampleRESTClient();
            sampleClient.demonstration("restserver", "80", "csciwws");
        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
    }

    private void demonstration(final String host, final String port, final String contextroo
t) {
        System.out.println("Demonstration started.");

        String targetURL = "http://" + host + ":" + port + "/" + contextroot;

        // Invoke resource
        Client client = ClientBuilder.newClient();

        // Set the target resource's URL
        String postURL = new String(targetURL + "/v1/process-instance/create-and-start");
        System.out.println("The target URL is ¥"POST " + postURL + "¥".");

        // ProcessInstance RequestBody
        CreateAndStartProcessInstance parameter = new CreateAndStartProcessInstance();
        parameter.setName("process1");
        parameter.setDeadline("2020-12-31T12:00:00+09:00");
        parameter.setPriority("10");
        parameter.setProcessDataList(null);
        parameter.setDefinitionName("APCS3process");
        parameter.setDefinitionVersion("1");

        // ProcessData
```

```
        List<ProcessData> processData = new ArrayList<ProcessData>();
        ProcessData data1 = new ProcessData();
        data1.setKey("$SAPP");
        data1.setValue("Y");
        processData.add(data1);
        ProcessData data2 = new ProcessData();
        data2.setKey("$NID");
        data2.setValue("1");
        processData.add(data2);
        parameter.setProcessDataList(processData);

        Entity<?> entity = Entity.entity(parameter, MediaType.APPLICATION_XML);

        WebTarget targetPost = client.target(postURL);
        Response postResponse = targetPost
                              .request()
                              .accept(MediaType.APPLICATION_XML)
                              .post(entity);

        ProcessInstance processInstance = null;
        int statusCode = postResponse.getStatus();
        if (statusCode == 201) {
            processInstance = postResponse.readEntity(ProcessInstance.class);
        } else {
            String error = postResponse.readEntity(String.class);
            System.out.println("Status Code=" + statusCode);
            client.close();
            throw new RuntimeException(error);
        }

        System.out.println("Response of ProcessInstance is"
                + "¥r¥n  Name:" + processInstance.getProcessDefinitionName()
                + "¥r¥n  PIID:" + processInstance.getID()
                + "¥r¥n  StateCode:" + processInstance.getStateCode());

        // Set the target resource's URL
        String piid = processInstance.getID();
        String getURL = new String(targetURL + "/v1/work-item"
                                           + "?filter=ProcessInstanceID%3D" + piid);
        System.out.println("The target URL is ¥"GET " + getURL + "¥".");

        WebTarget targetGet = client.target(getURL);
        Response getResponse = targetGet
                             .request()
                             .accept(MediaType.APPLICATION_XML)
                             .get();

        statusCode = getResponse.getStatus();
        if (statusCode != 200) {
            String error = getResponse.readEntity(String.class);
            System.out.println("Status Code=" + statusCode);
            client.close();
            throw new RuntimeException(error);
        }

        WorkItemList workItemList = getResponse.readEntity(WorkItemList.class);
        System.out.println("The number of WorkItems is " + workItemList.getWorkItems().size(
));
```

```
        List<WorkItem> workItems = workItemList.getWorkItems();
        if (workItems != null) {
            Iterator<WorkItem> it = workItems.iterator();
            while (it.hasNext()) {
                WorkItem wi = it.next();

                // Set the target resource's URL
                String putURL = new String(targetURL + "/v1/work-item/" + piid + "/" + wi.ge
tID()
                                                    + "/perform-and-complete");
                System.out.println("The target URL is ¥"PUT " + putURL + "¥".");

                entity = Entity.entity( "<Parameter />", MediaType.APPLICATION_XML);

                WebTarget targetPut = client.target(putURL);
                Response putResponse = targetPut
                                       .request()
                                       .accept(MediaType.APPLICATION_XML)
                                       .put(entity);

                WorkItem workItem = null;
                statusCode = putResponse.getStatus();
                if (statusCode == 200) {
                    workItem = putResponse.readEntity(WorkItem.class);
                } else {
                    String error = putResponse.readEntity(String.class);
                    System.out.println("Status Code=" + statusCode);
                    client.close();
                    throw new RuntimeException(error);
                }

                System.out.println("WorkItem is"
                        + "¥r¥n  Name:" + workItem.getWorkDefinitionName()
                        + "¥r¥n  WIID:" + workItem.getID()
                        + "¥r¥n  State:" + workItem.getStateCode());
            }
        }

        client.close();
        System.out.println("Demonstration ended.");
    }
}
```

ProcessInstance.java

```
package rest.sample;

import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlRootElement;

@XmlRootElement(name="ProcessInstance")
class ProcessInstance {
    private String closedDate;
    private String creator;
    private String deadline;
    private String ID;
```

```
    private String movedDate;
    private String name;
    private String priority;
    private String processDefinitionID;
    private String processDefinitionName;
    private String startDate;
    private String stateCode;

    public ProcessInstance() {}
    @XmlElement(name="ClosedDate")
    public String getClosedDate() { return closedDate; }
    public void setClosedDate(String closedDate) { this.closedDate = closedDate; }
    @XmlElement(name="Creator")
    public String getCreator() { return creator; }
    public void setCreator(String creator) { this.creator = creator; }
    @XmlElement(name="Deadline")
    public String getDeadline() { return deadline; }
    public void setDeadline(String deadline) { this.deadline = deadline; }
    @XmlElement(name="ID")
    public String getID() { return ID; }
    public void setID(String iD) { ID = iD; }
    @XmlElement(name="MovedDate")
    public String getMovedDate() { return movedDate; }
    public void setMovedDate(String movedDate) { this.movedDate = movedDate; }
    @XmlElement(name="Name")
    public String getName() { return name; }
    public void setName(String name) { this.name = name; }
    @XmlElement(name="Priority")
    public String getPriority() { return priority; }
    public void setPriority(String priority) { this.priority = priority; }
    @XmlElement(name="ProcessDefinitionID")
    public String getProcessDefinitionID() { return processDefinitionID; }
    public void setProcessDefinitionID(String processDefinitionID) {
            this.processDefinitionID = processDefinitionID; }
    @XmlElement(name="ProcessDefinitionName")
    public String getProcessDefinitionName() { return processDefinitionName; }
    public void setProcessDefinitionName(String processDefinitionName) {
            this.processDefinitionName = processDefinitionName; }
    @XmlElement(name="StartDate")
    public String getStartDate() { return startDate; }
    public void setStartDate(String startDate) { this.startDate = startDate; }
    @XmlElement(name="StateCode")
    public String getStateCode() { return stateCode; }
    public void setStateCode(String stateCode) { this.stateCode = stateCode; }
}
```

ProcessData.java

```
package rest.sample;

import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlRootElement;

@XmlRootElement(name = "ProcessData")
public class ProcessData {
    private String key;
    private String value;
```

```
    public ProcessData() {}

    @XmlElement(name = "Key")
    public String getKey() { return key; }
    public void setKey(String aKey) { key = aKey; }

    @XmlElement(name = "Value")
    public String getValue() { return value; }
    public void setValue(String aValue) { value = aValue; }
}
```

CreateAndStartProcessInstance.java

```
package rest.sample;

import java.util.List;

import javax.xml.bind.annotation.XmlAccessType;
import javax.xml.bind.annotation.XmlAccessorType;
import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlElementWrapper;
import javax.xml.bind.annotation.XmlRootElement;

@XmlAccessorType(XmlAccessType.FIELD)
@XmlRootElement(name = "Parameter")
public class CreateAndStartProcessInstance {
    @XmlElement(name = "Name")
    private String name;

    @XmlElement(name = "Deadline")
    private String deadline;

    @XmlElement(name = "Priority")
    private String priority;

    @XmlElementWrapper(name = "ProcessDataList")
    @XmlElement(name = "ProcessData")
    private List<ProcessData> processDataList;

    @XmlElement(name = "DefinitionName")
    private String definitionName;

    @XmlElement(name = "DefinitionVersion")
    private String definitionVersion;

    public String getName() { return name; }
    public void setName(String name) { this.name = name; }
    public String getDeadline() { return deadline; }
    public void setDeadline(String deadline) { this.deadline = deadline; }
    public String getPriority() { return priority; }
    public void setPriority(String priority) { this.priority = priority; }
    public List<ProcessData> getProcessDataList() { return processDataList; }
    public void setProcessDataList(List<ProcessData> processes) { this.processDataList = pro
cesses; }
    public String getDefinitionName() { return definitionName; }
    public void setDefinitionName(String definitionName) { this.definitionName = definitionN
```

```
ame; }
    public String getDefinitionVersion() { return definitionVersion; }
    public void setDefinitionVersion(String definitionVersion) { this.definitionVersion = de
finitionVersion; }
}
```

WorkItemList.java

```
package rest.sample;

import java.util.List;

import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlRootElement;

@XmlRootElement(name="WorkItems")
class WorkItemList {
    private List<WorkItem> workItems;

    @XmlElement(name="WorkItem")
    public List<WorkItem> getWorkItems() { return workItems; }
    public void setWorkItems(List<WorkItem> workItems) { this.workItems = workItems; }
}
```

WorkItem.java

```
package rest.sample;

import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlRootElement;

@XmlRootElement(name="WorkItem")
class WorkItem {
    private String activityInstanceID;
    private String closedDate;
    private String creationDate;
    private String deadline;
    private String ID;
    private String name;
    private String participant;
    private String priority;
    private String processDefinitionID;
    private String processInstanceID;
    private String processInstanceName;
    private String startDate;
    private String stateCode;
    private String workTypeCode;
    private String workDefinitionID;
    private String workDefinitionName;

    public WorkItem() {}
    @XmlElement(name="ActivityInstanceID")
    public String getActivityInstanceID() { return activityInstanceID; }
    public void setActivityInstanceID(String activityInstanceID) {
            this.activityInstanceID = activityInstanceID; }
```

```
    @XmlElement(name="ClosedDate")
    public String getClosedDate() { return closedDate; }
    public void setClosedDate(String closedDate) { this.closedDate = closedDate; }
    @XmlElement(name="CreationDate")
    public String getCreationDate() { return creationDate; }
    public void setCreationDate(String creationDate) { this.creationDate = creationDate; }
    @XmlElement(name="Deadline")
    public String getDeadline() { return deadline; }
    public void setDeadline(String deadline) { this.deadline = deadline; }
    @XmlElement(name="ID")
    public String getID() { return ID; }
    public void setID(String iD) { ID = iD; }
    @XmlElement(name="Name")
    public String getName() { return name; }
    public void setName(String name) { this.name = name; }
    @XmlElement(name="Paticipant")
    public String getParticipant() { return participant; }
    public void setParticipant(String participant) { this.participant = participant; }
    @XmlElement(name="Priority")
    public String getPriority() { return priority; }
    public void setPriority(String priority) { this.priority = priority; }
    @XmlElement(name="ProcessDefinitionID")
    public String getProcessDefinitionID() { return processDefinitionID; }
    public void setProcessDefinitionID(String processDefinitionID) {
            this.processDefinitionID = processDefinitionID; }
    @XmlElement(name="ProcessInstanceID")
    public String getProcessInstanceID() { return processInstanceID; }
    public void setProcessInstanceID(String processInstanceID) {
            this.processInstanceID = processInstanceID; }
    @XmlElement(name="ProcessInstanceName")
    public String getProcessInstanceName() { return processInstanceName; }
    public void setProcessInstanceName(String processInstanceName) {
            this.processInstanceName = processInstanceName; }
    @XmlElement(name="StartDate")
    public String getStartDate() { return startDate; }
    public void setStartDate(String startDate) { this.startDate = startDate; }
    @XmlElement(name="StateCode")
    public String getStateCode() { return stateCode; }
    public void setStateCode(String stateCode) { this.stateCode = stateCode; }
    @XmlElement(name="WorkTypeCode")
    public String getWorkTypeCode() { return workTypeCode; }
    public void setWorkTypeCode(String workTypeCode) { this.workTypeCode = workTypeCode; }
    @XmlElement(name="WorkDefinitionID")
    public String getWorkDefinitionID() { return workDefinitionID; }
    public void setWorkDefinitionID(String workDefinitionID) { this.workDefinitionID = workD
efinitionID; }
    @XmlElement(name="WorkDefinitionName")
    public String getWorkDefinitionName() { return workDefinitionName; }
    public void setWorkDefinitionName(String workDefinitionName) {
            this.workDefinitionName = workDefinitionName; };
}
```

5 Java API を使用して業務アプリケーションを開発する

BPMN 連携ライブラリの Java API を使用して業務アプリケーションを開発する場合の，業務アプリケーションの作成について説明します。なお，CSCIW で使用する API の特長については，マニュアル『uCosminexus Service Coordinator Interactive Workflow AP 開発ガイド』の「2.1　API の特長」を参照してください。

5.1　Java API を使用した業務アプリケーションの開発

BPMN 連携ライブラリの Java API を使用する業務アプリケーションを作成します。

5.1.1　BPMN 連携機能で使用できる API

CSCIW で使用する API のうち，BPMN 連携機能で使用できる API を示します。

マニュアル『uCosminexus Service Coordinator Interactive Workflow AP 開発ガイド』に記載している，CSCIW で使用する API には，BPMN 連携機能を使用した場合に使用できる API と使用できない API があります。BPMN 連携機能を使用した場合に使用できる API と使用できない API を次の表で示します。

表 5-1　BPMN 連携機能使用時の API の使用可否

CSCIW で使用する API の種類	BPMN 連携機能使用時の API の使用可否
参照系 API	使用できる
更新系 API	使用できない BPMN 連携機能が提供する REST API および Java API を使用して，案件の投入や削除，業務ステップや作業の状態変更などをする必要があります。
属性取得 API	使用できる

参照系 API と更新系 API の詳細については，マニュアル『uCosminexus Service Coordinator Interactive Workflow AP 開発ガイド』の「付録 A　更新系 API と参照系 API」を参照してください。

5.1.2　パッケージをインポートする

BPMN 連携ライブラリの Java API を使用する場合は，次に示すパッケージをすべてインポートする必要があります。

- CSCIW が提供する Java API のパッケージ
  「jp.co.Hitachi.soft.csciw」
- BPMN 連携ライブラリが提供する Java API のパッケージ
  「jp.co.Hitachi.soft.csciw.bpmn」
  「jp.co.Hitachi.soft.csciw.bpmn.processdata」

次に示すパッケージのインポート文をソースコードに記述してください。

```
import jp.co.Hitachi.soft.csciw.*;
import jp.co.Hitachi.soft.csciw.bpmn.*;
import jp.co.Hitachi.soft.csciw.bpmn.processdata.*;
```

5.1.3　業務アプリケーションをコンパイルする

コンパイル

- Java コンパイラ
  業務アプリケーションは JDK8 以降を使用してコンパイルします。
- クラスパス
  業務アプリケーションをコンパイルする場合は，次に示す JAR ファイルをクラスパスに設定します。
  - CSCIW をインストールした環境の場合

    ＜ Windows のとき＞

    ```
    <CSCIWインストールディレクトリ>¥lib¥csciw.jar
    <CSCIWインストールディレクトリ>¥lib¥csciwcmn.jar
    <CSCIWインストールディレクトリ>¥lib¥csciwbpmn.jar
    ```

    ＜ UNIX のとき＞

    ```
    /opt/hitachi/CSCIW/lib/csciw.jar
    /opt/hitachi/CSCIW/lib/csciwcmn.jar
    /opt/hitachi/CSCIW/lib/csciwbpmn.jar
    ```

  - uCosminexus Business Process Developer をインストールした環境の場合

    ```
    <uCosminexus Business Process Developerのインストールディレクトリ>¥lib¥csciw.jar
    <uCosminexus Business Process Developerのインストールディレクトリ>¥lib¥csciwcmn.jar
    <uCosminexus Business Process Developerのインストールディレクトリ>¥lib¥csciwbpmn.jar
    ```

注意事項

各業務アプリケーションのear に上記の JAR ファイルは組み込まないでください。

5.2　Java API を使用した業務アプリケーションの処理の流れ

Java API を使用した業務アプリケーションの処理の流れを示します。

5.2.1　CSCIW を初期化する

CSCIW の初期化については，マニュアル『uCosminexus Service Coordinator Interactive Workflow AP 開発ガイド』の「2.3.1　CSCIW の初期化」を参照してください。

5.2.2　BPMN 連携ライブラリを初期化する

BPMN 連携ライブラリを使用する場合，事前に BPMN 連携ライブラリの初期化を実行しておく必要があります。プロセスの開始時に，BPMN 連携ライブラリを初期化してください。BPMN 連携ライブラリを初期化する場合の業務アプリケーションの実装例を次に示します。

例

```
try {
    // コネクションの取得および自動コミットのOFF
    …
    CIWBPMNLibAdmin.initializeCIWBPMNLib(connection);
    …
    // トランザクションの終了処理
}
catch(CIWFatalException e) {
    // 例外処理を記述
}
```

BPMN 連携ライブラリを初期化する前に，CSCIW を初期化しておく必要があります。CSCIW を初期化する方法は J2EE サーバで使用する場合と，Java アプリケーションで使用する場合で異なります。

J2EE サーバで使用する場合，CSCIWManagementServer※を開始します。

Java アプリケーションで使用する場合は，CIWAdmin クラスの initializeCIWFactory メソッドを実行します。

注※

アプリケーションサーバでの CSCIW の初期化・終了処理を行う，製品で提供している J2EE アプリケーションです。

重要

BPMN 連携ライブラリの Java API 実行時にはトランザクションを明示的に開始しません。そのため，業務プログラムでは，CSCIW の Java API を呼び出す前に，トランザクションを開始しておく必要があります。

また，BPMN 連携ライブラリの Java API で利用したトランザクションを終了しません。業務プログラムでトランザクションを終了してください。

詳細については，マニュアル『uCosminexus Service Coordinator Interactive Workflow AP 開発ガイド』の「2.10　業務プログラムで管理するトランザクション」を参照してください。

5.2.3　CIWFactory オブジェクトを取得する

CIWFactory オブジェクトの取得については，マニュアル『uCosminexus Service Coordinator Interactive Workflow AP 開発ガイド』の「2.3.2　CIWFactory オブジェクトの取得」を参照してください。

5.2.4　CIWServer オブジェクトを生成する

CIWServer オブジェクトの生成については，マニュアル『uCosminexus Service Coordinator Interactive Workflow AP 開発ガイド』の「2.3.3　CIWServer オブジェクトの生成」を参照してください。

5.2.5　CIWBPMNLib オブジェクトを生成する

CIWBPMNLib オブジェクトは，CIWBPMNLibFactory クラスのcreateCIWBPMNLib メソッドを使用して生成します。CIWBPMNLib オブジェクトはスレッドごとに生成してください。CIWBPMNLib オブジェクトを生成する場合の業務アプリケーションの実装例を次に示します。

例

```
try {
    CIWBPMNLib bpmnlib = CIWBPMNLibFactory.createCIWBPMNLib();
    …
}
catch(CIWFatalException e) {
    // 例外処理を記述
}
```

5.2.6　CIWServer オブジェクトに対して Connection オブジェクトを関連づける

CIWServer オブジェクトに対する Connection オブジェクトの関連づけについては，マニュアル『uCosminexus Service Coordinator Interactive Workflow AP 開発ガイド』の「2.3.4　CIWServer オブジェクトに対する Connection オブジェクトの関連づけ」を参照してください。

5.2.7　BPMN 連携ライブラリを利用した業務処理を実装する

BPMN 連携ライブラリを利用した業務処理の実装例を示します。

案件などの検索は CSCIW の**CIWServer** クラスなどの参照系 API を使用します。案件の投入や作業の完了など更新系の処理をする場合は，BPMN 連携ライブラリの API を使用します。

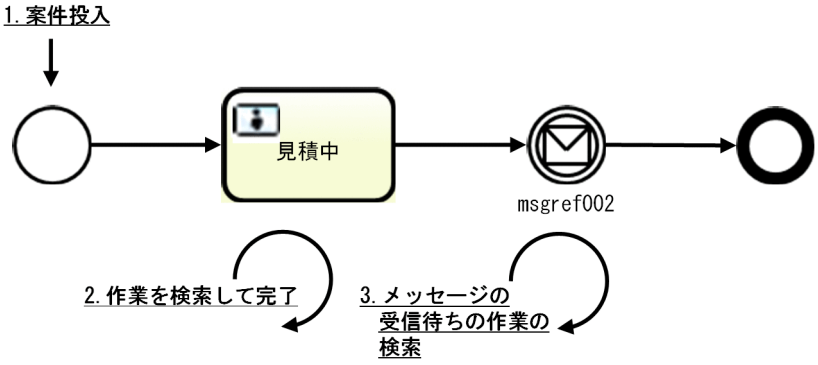
(1)　実装例 1

CSCIW の**CIWServer** クラスなどの参照系 API を使用した場合の実装例を示します。

案件の投入や作業の完了などの更新系 API は使用しないで，BPMN 連携ライブラリの API を使用します。

次の図のような販売業務ビジネスプロセスを BPMN エディタで定義した場合，案件を投入すると，このビジネスプロセスは開始されます。

図 5-1　販売業務ビジネスプロセス



例

1. 案件の投入

   案件を投入する場合の業務アプリケーションの実装例を次に示します。

```
try {
    // コネクションの取得および自動コミットのOFF
    …
    CIWServer server = factory.createCIWServer("User");
    CIWConnectionCoordinator coordinator = server.getConnectionCoordinator();
    coordinator.attachDatabaseConnection(connection);
    …
```

```
    // 以下を指定してビジネスプロセス定義「販売業務ビジネスプロセス」の案件を投入
    // 案件名「案件1」
    // プロセスデータ（プロセスデータキー名「$NPrice」，プロセスデータ値「50000」）
    Map<CIWProcessInstance.AttributeName, Object> attributes
        = new HashMap<CIWProcessInstance.AttributeName, Object>();
    attributes.put(
        CIWProcessInstance.AttributeName.NAME,
        "案件1");
    Collection<CIWBPMNProcessData<?>> pdlist
        = new ArrayList<CIWBPMNProcessData<?>>();
    pdlist.add(CIWBPMNProcessDataFactory.createProcessData(
        "$NPrice ",
        Integer.valueOf(50000)));
    CIWProcessInstance processInstance = bpmnlib.createAndStartPI(
        connection,
        server,
        "販売業務ビジネスプロセス",
        null,
        attributes,
        pdlist);
    …
    coordinator.detachDatabaseConnection();
    // トランザクションの終了処理
    …
}
catch(CIWException e) {
    // 例外処理を記述
}
```

2. 作業を検索して完了

作業を検索して，検索した作業の着手と完了を実行する場合の業務アプリケーションの実装例を次に示します。

```
try {
    // コネクションの取得および自動コミットのOFF
    …
    CIWServer server = factory.createCIWServer("User");
    CIWConnectionCoordinator coordinator = server.getConnectionCoordinator();
    coordinator.attachDatabaseConnection(connection);
    …
    // 作業者IDが「見積担当者」かつ「実行開始可能」状態の作業を取得
    EnumSet<CIWWorkItem.AttributeName> attributeNames
        = EnumSet.allOf(CIWWorkItem.AttributeName.class);
    List<CIWWorkItem> wiList = server.getWorkItemsList(
        "Participant = '見積担当者' AND StateCode = '"
            + CIWWorkItem.State.READY.toStateCode() + "'",
        null,
        0,
        1,
        attributeNames);
    // 以下を指定して作業の着手と完了
    // プロセスデータ（プロセスデータキー名「$SExaminationResult」，プロセスデータ値「O
K」）
    Collection<CIWBPMNProcessData<?>> pdlist
        = new ArrayList<CIWBPMNProcessData<?>>();
    pdlist.add(
        CIWBPMNProcessDataFactory.createProcessData(
```

```
            "$SExaminationResult ",
            "OK"));
    if(wiList.size() > 0) {
        bpmnlib.performAndCompleteWI(
            connection,
            server,
            wiList.get(0).getProcessInstanceID(),
            wiList.get(0).getID(),
            pdlist);
    }
    …
    coordinator.detachDatabaseConnection();
    // トランザクションの終了処理
    …
}
catch(CIWException e) {
    // 例外処理を記述
}
```

3. メッセージの受信待ちの作業の検索

メッセージの受信待ちの作業一覧を取得する場合の業務アプリケーションの実装例を次に示します。

```
try {
    // コネクションの取得および自動コミットのOFF
    …
    CIWServer server = factory.createCIWServer("User");
    CIWConnectionCoordinator coordinator = server.getConnectionCoordinator();
    coordinator.attachDatabaseConnection(connection);
    …
    // messageRefが「msgref002」で受信待ちしている作業一覧を取得
    EnumSet<CIWWorkItem.AttributeName> attributeNames
        = EnumSet.allOf(CIWWorkItem.AttributeName.class);
    List<CIWWorkItem> wiList = server.getWorkItemsList(
        "Participant = 'IWRMSG_msgref002' AND " +
        "StateCode = '"+ CIWWorkItem.State.READY.toStateCode() + "'",
        null,
        0,
        -1,
        attributeNames);
    …
    coordinator.detachDatabaseConnection();
    // トランザクションの終了処理
    …
}
catch(CIWException e) {
    // 例外処理を記述
}
```

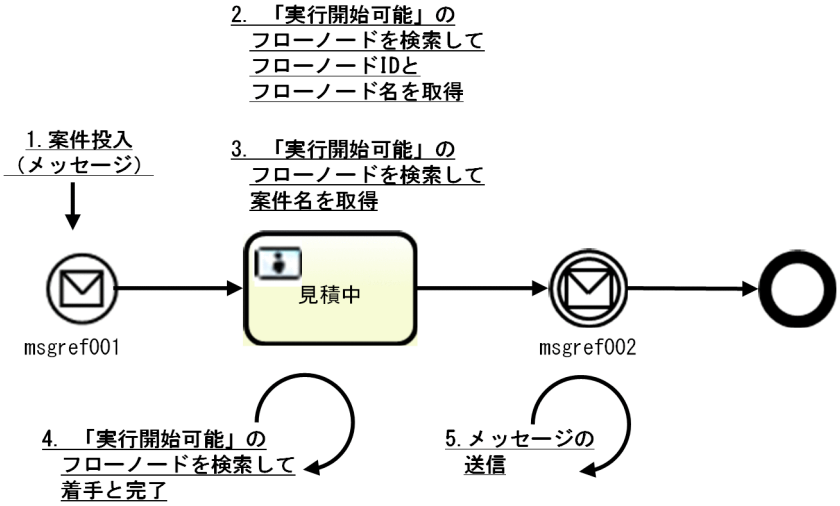
(2) 実装例 2

BPMN 連携ライブラリの参照系 API を使用した場合の実装例を示します。

案件の投入や作業の完了などの更新系 API は使用しないで，BPMN 連携ライブラリの API を使用します。

次の図のような販売業務ビジネスプロセスを BPMN エディタで定義した場合，案件（メッセージ）を投入すると，このビジネスプロセスは開始されます。

図 5-2　販売業務ビジネスプロセス



例

1. 案件の投入（メッセージ）

案件投入（メッセージ）をする場合の業務アプリケーションの実装例を次に示します。

業務処理の入力項目は，次のとおりです。

- ビジネスプロセス定義名
- 案件名（案件キー）
- フローノード ID（BPMN 要素の id 属性値）

```
try {
    // コネクションの取得および自動コミットのOFF
    …
    CIWServer server = factory.createCIWServer(”User”);
    CIWConnectionCoordinator coordinator = server.getConnectionCoordinator();
    coordinator.attachDatabaseConnection(connection);
    …
    // 以下を指定して該当するフローノードのref識別子を取得
    // ビジネスプロセス定義名「販売業務ビジネスプロセス」
    // フローノードID「MsgStart1」
    List<CIWBPMNFlowNodeDefinition> fnDefinitionList = bpmnlib.getFlowNodeDefinitionsLis
t(
        connection,
        server,
        ”販売業務ビジネスプロセス”,
        ”MsgStart1”,
        null,
        null);
    …
    switch(fnDefinitionList.get(0).getFlowNodeType()) {
        case MESSAGE_TOPSTART:
            // フローノードが開始（メッセージ）の場合，以下を指定して
            // 案件投入（メッセージ）を実行
```

```
            // ビジネスプロセス定義名「販売業務ビジネスプロセス」
            // 案件名「案件1」
            // MessageRef（取得したref識別子）
            Map<CIWProcessInstance.AttributeName, Object> attributes
                = new HashMap<CIWProcessInstance.AttributeName, Object>();
            attributes.put(
                CIWProcessInstance.AttributeName.NAME,
                ”案件1”);
            Collection<CIWBPMNProcessData<?>> pdlist
                = new ArrayList<CIWBPMNProcessData<?>>();
            pdlist.add(CIWBPMNProcessDataFactory.createProcessData(
                ”$NPrice”,
                Integer.valueOf(50000)));
            CIWProcessInstance processInstance = bpmnlib.startMessage(
                connection,
                server,
                ”販売業務ビジネスプロセス”,
                null,
                attributes,
                null,
                fnDefinitionList.get(0).getFlowNodeRefID());
    }
    …
    coordinator.detachDatabaseConnection();
    // トランザクションの終了処理
    …
}
catch(CIWException e) {
    // 例外処理を記述
}
```

2.「実行開始可能」のフローノードを検索してフローノード ID とフローノード名を取得

「実行開始可能」のフローノードを検索して，検索したフローノードのフローノード ID（BPMN 要素の id 属性値）とフローノード名（BPMN 要素の name 属性値）を取得する場合の業務アプリケーションの実装例を次に示します。

業務処理の入力項目は，次のとおりです。

- ビジネスプロセス定義名
- 案件名（案件キー）

```
try {
    // コネクションの取得および自動コミットのOFF
    …
    CIWServer server = factory.createCIWServer(”User”);
    CIWConnectionCoordinator coordinator = server.getConnectionCoordinator();
    coordinator.attachDatabaseConnection(connection);
    …
    // 以下を指定して「実行中」の案件を取得
    // ビジネスプロセス定義名「販売業務ビジネスプロセス」
    // 案件名「案件1」
    EnumSet<CIWProcessInstance.State> pistates = EnumSet.of(
        CIWProcessInstance.State.RUNNING);
    List<CIWProcessInstance> pilist = bpmnlib.getProcessInstancesListByPIName(
        connection,
        server,
```

```
        ”販売業務ビジネスプロセス”,
        ”案件1”,
        pistates,
        null);
    // 以下を指定して「実行開始可能」状態のフローノードを取得
    // 案件ID（取得した案件のID）
    EnumSet<CIWWorkItem.State> wistates
        = EnumSet.of(CIWWorkItem.State.READY);
    List<CIWBPMNFlowNodeInstance> fnlist = bpmnlib.getFlowNodeInstancesListByPIID(
        connection,
        server,
        pilist.get(0).getID(),
        null,
        null,
        wistates,
        null,
        null);
    // フローノードIDとフローノード名を取得
    fnlist.get(0).getFlowNodeID();
    fnlist.get(0).getFlowNodeName();
    …
    coordinator.detachDatabaseConnection();
    // トランザクションの終了処理
    …
}
catch(CIWException e) {
    // 例外処理を記述
}
```

3.「実行開始可能」のフローノードを検索して案件名を取得

「実行開始可能」のフローノードを検索して，検索したフローノードが所属する案件の案件名を取得する場合の業務アプリケーションの実装例を次に示します。

業務処理の入力項目は，次のとおりです。

- ビジネスプロセス定義名
- フローノード名（BPMN 要素の name 属性値）

```
try {
    // コネクションの取得および自動コミットのOFF
    …
    CIWServer server = factory.createCIWServer(”User”);
    CIWConnectionCoordinator coordinator = server.getConnectionCoordinator();
    coordinator.attachDatabaseConnection(connection);
    …
    // 以下を指定して「実行開始可能」状態のフローノードを取得
    // ビジネスプロセス定義名「販売業務ビジネスプロセス」
    // フローノード名「見積中」
    EnumSet<CIWWorkItem.State> wistates
        = EnumSet.of(CIWWorkItem.State.READY);
    EnumSet<CIWBPMNFlowNodeInstance.AttributeName> attributeNames
        = EnumSet.of(
                CIWBPMNFlowNodeInstance.AttributeName.PROCESS_INSTANCE_NAME);
    List<CIWBPMNFlowNodeInstance> fnlist = bpmnlib.getFlowNodeInstancesListByPDName(
        connection,
        server,
        ”販売業務ビジネスプロセス”,
```

```
        null,
        ”見積中”,
        wistates,
        null,
        attributeNames);
    // フローノードの案件名を取得
    fnlist.get(0).getProcessInstanceName();
    …
    coordinator.detachDatabaseConnection();
    // トランザクションの終了処理
    …
}
catch(CIWException e) {
    // 例外処理を記述
}
```

4.「実行開始可能」のフローノードを検索して着手と完了

「実行開始可能」のフローノードを検索して，検索したフローノードに対応する作業の着手と完了をする場合の業務アプリケーションの実装例を次に示します。

業務処理の入力項目は，次のとおりです。

- ビジネスプロセス定義名
- 案件名（案件キー）
- フローノード名（BPMN 要素の name 属性値）

```
try {
    // コネクションの取得および自動コミットのOFF
    …
    CIWServer server = factory.createCIWServer(”User”);
    CIWConnectionCoordinator coordinator = server.getConnectionCoordinator();
    coordinator.attachDatabaseConnection(connection);
    …
    // 以下を指定して「実行中」状態の案件を取得
    // ビジネスプロセス定義名「販売業務ビジネスプロセス」
    // 案件名「案件1」
    EnumSet<CIWProcessInstance.State> pistates
        = EnumSet.of(CIWProcessInstance.State.RUNNING);
    List<CIWProcessInstance> pilist = bpmnlib.getProcessInstancesListByPIName(
        connection,
        server,
        ”販売業務ビジネスプロセス”,
        ”案件1”,
        pistates,
        null);
    // 以下を指定して「実行開始可能」状態のフローノードを取得
    // 案件ID（取得した案件のID）
    // フローノード名「見積中」
    EnumSet<CIWWorkItem.State> wistates
        = EnumSet.of(CIWWorkItem.State.READY);
    List<CIWBPMNFlowNodeInstance> fnlist = bpmnlib.getFlowNodeInstancesListByPIID(
        connection,
        server,
        pilist.get(0).getID(),
        null,
        ”見積中”,
```

```
        wistates,
        null,
        null);
    // 以下を指定して作業の着手と完了
    // 案件ID（取得したフローノードの案件ID）
    // 作業ID（取得したフローノードの作業ID）
    // プロセスデータ（プロセスデータキー名「$SExaminationResult」,
    //                 プロセスデータ値「OK」）
    Collection<CIWBPMNProcessData<?>> pdlist
        = new ArrayList<CIWBPMNProcessData<?>>();
    pdlist.add(
        CIWBPMNProcessDataFactory.createProcessData(
            "$SExaminationResult",
            "OK"));
    bpmnlib.performAndCompleteWI(
        connection,
        server,
        fnlist.get(0).getProcessInstanceID(),
        fnlist.get(0).getWorkItemID(),
        pdlist);
    …
    coordinator.detachDatabaseConnection();
    // トランザクションの終了処理
    …
}
catch(CIWException e) {
    // 例外処理を記述
}
```

5. メッセージの送信

フローノードの検索をして，検索したフローノードに対してメッセージを送信する場合の業務アプリケーションの実装例を次に示します。

業務処理の入力項目は，次のとおりです。

- ビジネスプロセス定義名
- 案件名（案件キー）
- フローノード ID（BPMN 要素の id 属性値）

```
try {
    // コネクションの取得および自動コミットのOFF
    …
    CIWServer server = factory.createCIWServer("User");
    CIWConnectionCoordinator coordinator = server.getConnectionCoordinator();
    coordinator.attachDatabaseConnection(connection);
    …
    // 以下を指定して該当するフローノード定義のref識別子を取得
    // ビジネスプロセス定義名「販売業務ビジネスプロセス」
    // フローノードID「MsgCatch1」
    List<CIWBPMNFlowNodeDefinition> fnDefinitionList = bpmnlib.getFlowNodeDefinitionsLis
t(
        connection,
        server,
        "販売業務ビジネスプロセス",
        "MsgCatch1",
        null,
```

```
        null);
    …
    switch(fnDefinitionList.get(0).getFlowNodeType()) {
        case MESSAGE_CATCH:
            // フローノードがキャッチ（メッセージ）の場合，以下を指定して
            // 「実行中」の案件を取得
            // ビジネスプロセス定義名「販売業務ビジネスプロセス」
            // 案件名「案件1」
            EnumSet <CIWProcessInstance.State> pistates = EnumSet.of(
                CIWProcessInstance.State.RUNNING);
            List<CIWProcessInstance> pilist =
                                    bpmnlib.getProcessInstancesListByPIName(
                connection,
                server,
                ”販売業務ビジネスプロセス”,
                ”案件1”,
                pistates,
                null);
            // 取得した案件に対して，以下を指定してメッセージを送信
            // 案件ID（取得した案件のID）
            // プロセスデータ（プロセスデータキー名「$NPrice」，
            //                  プロセスデータ値「50000」）
            // MessageRef（取得したフローノード定義のref識別子）
            Collection<CIWBPMNProcessData<?>> pdlist
                = new ArrayList<CIWBPMNProcessData<?>>();
            pdlist.add(CIWBPMNProcessDataFactory.createProcessData(
                ”$NPrice”,
                Integer.valueOf(50000)));
            bpmnlib.sendMessage (
                connection,
                server,
                pilist.get(0).getID(),
                pdlist,
                fnDefinitionList.get(0).getFlowNodeRefID());
    }
    …
    coordinator.detachDatabaseConnection();
    // トランザクションの終了処理
    …
}
catch(CIWException e) {
    // 例外処理を記述
}
```

5.2.8 CIWServer オブジェクトに対する Connection オブジェクトの関連づけを解除する

CIWServer オブジェクトに対する Connection オブジェクトの関連づけの解除については，マニュアル『uCosminexus Service Coordinator Interactive Workflow AP 開発ガイド』の「2.3.6　CIWServer オブジェクトに対する Connection オブジェクトの関連づけの解除」を参照してください。

5.2.9　BPMN 連携ライブラリを終了する

BPMN 連携ライブラリを使用する業務アプリケーションを終了する際に，BPMN 連携ライブラリの終了処理を行います。BPMN 連携ライブラリの終了処理はプロセスの終了時に行ってください。BPMN 連携ライブラリの終了処理を行う場合の業務アプリケーションの実装例を次に示します。

例

```
CIWBPMNLibAdmin.finalizeCIWBPMNLib();
```

5.2.10　CSCIW を終了する

CSCIW の終了については，マニュアル『uCosminexus Service Coordinator Interactive Workflow AP 開発ガイド』の「2.3.7　CSCIW の終了」を参照してください。

5.3 Java API 利用時の注意事項

ここでは，BPMN 連携ライブラリ独自の注意事項について説明します。

重要

BPMN 連携ライブラリを使用する場合にも，マニュアル『uCosminexus Service Coordinator Interactive Workflow AP 開発ガイド』の次の説明が該当しますので，参照してください。

- 「2.9　CSCIW で管理するトランザクション」
- 「2.10　業務プログラムで管理するトランザクション」
- 「2.11　業務プログラムとデータベース間での文字コード」
- 「2.12　API 利用時の注意事項」

スレッドからの利用について

スレッド利用時の留意点を次に示します。

- CIWBPMNLibFactory は，複数スレッドからアクセスできます。
- CIWBPMNLib およびCIWBPMNLib から生成する各種オブジェクトは，1 つのスレッドからだけアクセスできます。複数のスレッドから関連するオブジェクトに同時にアクセスしないでください。

べき等性について

種別がPUT の API は同じ引数で複数回呼び出された場合，2 回目以降は実行済みと判定され，実行されないでfalse が返されます。

同じ引数が指定された 2 回目以降の実行かどうかの判定条件を次の表に示します。実行済みと判定されない場合，実行されて成功したときはtrue が返されます。

種別がPOST の API は，呼び出されるごとに新たに実行されます。ただし，createFlowNodeInstanceForAdHocSubProcess（アドホック・サブプロセスのフローノードを生成）は，実行済みと判定された場合，実行されないで生成済みのフローノードが返されます。実行済みと判定されない場合，実行されて成功したときは生成したフローノードが返されます。

表 5-2　API の実行済みの判定条件

項番	API の説明	種別	実行済み判定
1	createAndStartPI 案件の投入	POST	なし
2	deletePI 案件の削除	PUT	指定された案件が存在しない場合
3	terminatePI	PUT	指定された案件の状態が強制終了（u）の場合

項番	APIの説明	種別	実行済み判定
3	案件の強制終了	PUT	指定された案件の状態が強制終了（u）の場合
4	adhocCreateAndMakeTransitionAI 強制的に任意の業務ステップに遷移させるアドホック処理	PUT	次の条件が両方成立した場合 • 指定された遷移元の業務ステップの状態が強制終了（u）の場合 • 指定された遷移先に実行開始不可（l）/実行中（d）/実行停止（m）/遷移済（t）/強制終了（u）のどれかの状態の業務ステップが存在する場合
5	changeStateAI 業務ステップの状態変更	PUT	指定された業務ステップの状態が変更後の業務ステップの状態の場合
6	makeBackwardTransitionAI 差し戻し/引き戻し	PUT	次の条件が両方成立した場合 • 指定された遷移元の業務ステップの状態が強制終了（u）の場合 • 指定された遷移先に実行開始不可（l）/実行中（d）/実行停止（m）/遷移済（t）/強制終了（u）のどれかの状態の業務ステップが存在する場合
7	changeStateWI 作業の状態変更	PUT	指定された作業の状態が変更後の作業の状態の場合
8	completeWI 作業の完了	PUT	指定された作業の状態が実行済（r）の場合
9	performAndCompleteWI 作業の着手と完了	PUT	指定された作業の状態が実行済（r）の場合
10	performWI 作業の着手	PUT	指定された作業の状態が作業者実行（f）の場合
11	reassignAndPerformWI 作業の作業者変更および作業の着手	PUT	次の条件が両方成立した場合 • 指定された作業の作業者が変更後の作業者と同じ場合 • 指定された作業の状態が作業者実行（f）の場合
12	reassignWI 作業の作業者変更	PUT	指定された作業の作業者が変更後の作業者と同じ場合
13	allocateWIEx 指定した条件に一致する作業の着手	POST	なし
14	freeWI 作業の返却	PUT	次の条件が両方成立した場合 • 指定された作業の状態が実行開始（j）の場合 • 指定された作業の作業者がレーン名と同じ場合。プロセスデータが指定されている場合はプロセスデータの値と同じとき。※
15	setProcessData プロセスデータの登録	POSTまたはPUT	なし
16	sendMessage メッセージイベントの送信	POST	なし

項番	APIの説明	種別	実行済み判定
17	startMessage 案件の投入（メッセージ）	POST	なし
18	createAndStartPIForTimer 案件の投入（タイマー）	POST	なし
19	setDeadlineForTimer タイマーの処理期限を変更	PUT	なし
20	createFlowNodeInstanceForAdHocSubProcess アドホック・サブプロセスのフローノードを生成	POST	指定されたフローノードが示す業務ステップ定義に実行中（d）の状態の業務ステップが存在する場合
21	changeStateAdHocSubProcess アドホック・サブプロセスの状態の変更	PUT	指定されたアドホック・サブプロセスに対応する作業の状態が変更後の作業の状態の場合

注※

レーン名やプロセスデータの値がParticipantカラムのバイト数を超える場合は，Participantカラムのバイト数を超える部分を切り捨てた値と比較します。

6 アプリケーション呼び出しサービスを使用する

アプリケーション呼び出しサービスを使用する場合の，REST アプリケーションの開発，Java オブジェクトの開発，およびアプリケーション呼び出しサービスに関する各種設定について説明します。

6.1　アプリケーション呼び出しサービスから呼び出す REST アプリケーションを開発する

6.1.1　リクエストデータ

REST アプリケーションの開発時に従うリクエストデータの規定について説明します。

(1)　リクエストライン

リクエストラインには，次に示す内容を指定できます。

Method：”GET” ¦ “ POST” ¦ “PUT” ¦ “DELETE”

Request-URI：任意の URI

リクエストラインの指定の詳細については，次の項目を参照してください。

- Method：「15.3.14　REST アプリケーションの呼び出しの場合にファイルに指定する内容」の「(2)　rest.request.method」
- Request-URI：「15.3.14　REST アプリケーションの呼び出しの場合にファイルに指定する内容」の「(3)　rest.request.url」

(2)　リクエストヘッダ

リクエストヘッダには，任意の内容を指定できます。

リクエストヘッダの指定の詳細については，「15.3.14　REST アプリケーションの呼び出しの場合にファイルに指定する内容」の「(4)　rest.request.header.filepath」を参照してください。

(3)　リクエストボディ

リクエストボディは，「6.1.3　ボディデータスキーマ（XML）」または「6.1.4　ボディデータスキーマ（JSON）」に示す XML または JSON のデータスキーマの内容に従って，作成してください。

6.1.2　レスポンスデータ

REST アプリケーションから返却されるレスポンスデータの規定について説明します。

(1)　ステータスライン

REST アプリケーションから返却されるステータスコードの判定結果を，次の表に示します。

表 6-1　REST アプリケーションから返却されるステータスコードの判定結果

項番	ステータスコード	判定結果
1	Informational 1xx	成功
2	Successful 2xx	成功
3	Redirection 3xx	失敗
4	Client Error 4xx	失敗
5	Server Error 5xx	失敗

ステータスコードの判定結果が失敗の場合のアプリケーション呼び出しサービスの動作については，「1.8.9 障害時の動作」を参照してください。

(2)　レスポンスヘッダ

レスポンスヘッダのContent-Type ヘッダは，次のどれかである必要があります。

- text/xml
- application/xml
- application/*+xml
- text/json
- application/json
- application/*+json

Content-Type ヘッダのcharset 引数は，XML の符号化宣言（encoding 属性）と同じ値にしてください。charset 引数を省略した場合は UTF-8 として扱われます。

なお，セッションは継続されないため，Set-Cookie ヘッダの Cookie 情報は保持されません。

(3)　レスポンスボディ

レスポンスボディは，「6.1.3　ボディデータスキーマ（XML）」または「6.1.4　ボディデータスキーマ（JSON）」に示す，XML または JSON のデータスキーマの内容と同じです。

レスポンスボディを省略する場合，ステータスコードは「204 No Content」を指定してください。

6.1.3　ボディデータスキーマ（XML）

HTTP ボディ部分のボディデータスキーマの内容

アプリケーション呼び出しサービスの送受信で使用される HTTP ボディ部分のボディデータスキーマの内容を，次に示します。

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<xs:schema elementFormDefault="qualified" version="1.0"
 xmlns:xs="http://www.w3.org/2001/XMLSchema">

  <xs:element name="csciwRestBody" type="csciwRestBodyType"/>

  <xs:complexType name="csciwRestBodyType">
    <xs:sequence>
      <xs:element name="data" type="dataType" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="dataType">
    <xs:sequence>
      <xs:element name="key" type="xs:string"/>
      <xs:element name="value" type="xs:string" minOccurs="0" />
    </xs:sequence>
  </xs:complexType>
</xs:schema>
```

ボディデータスキーマ定義の要素の階層構造

項番	要素名および属性名※			型	型種別	数
1	csciwRestBody			csciwRestBodyType	混合型	1
2		data		dataType	混合型	0 以上
3			key	xs:string	－	1
4			value	xs:string	－	0～1

(凡例)

－：該当なし

注※

「要素名および属性名」列のインデントは，要素の階層構造を表しています。

ボディデータスキーマ宣言

項番	宣言	属性	値	説明
1	XML 宣言	version	1.0	XML のバージョンを指定します。 1.0 が固定値です。
2		encoding	UTF-8	エンコードを指定します。
3		standalone	yes	外部の markup 宣言の有無について指定します。 スタンドアロン文書のため，yes を指定してください。

項番	宣言	属性	値	説明
4	schema 宣言	elementFormDefault	qualified	ローカル要素にも名前空間接頭辞を付与するために，指定します。
5		version	1.0	XML スキーマのバージョンを指定します。 1.0 が固定値です。
6		xmlns:xs	http://www.w3.org/2001/XMLSchema	名前空間"xs"を宣言するために，指定します。

ボディデータの出力例

上記のスキーマが適用されたボディデータの出力例を次に示します。

```
<?xml version=”1.0” encoding=”UTF-8” standalone=”yes”?>
<csciwRestBody>
    <data>
        <key>key1</key>
        <value>value1</value>
    </data>
    <data>
        <key>key2</key>
        <value>2</value>
    </data>
    <data>
        <key>ListData</key>
        <value>AAA</value>
    </data>
    <data>
        <key>ListData</key>
        <value>BBB</value>
    </data>
</csciwRestBody>
```

6.1.4 ボディデータスキーマ（JSON）

HTTP ボディ部分のボディデータスキーマの内容

アプリケーション呼び出しサービスの送受信で使用される HTTP ボディ部分のボディデータスキーマの内容を，次に示します。

```
{
  ”type”: ”object”,
  ”properties”: {
    ”data”: {
      ”type”: ”array”,
      ”items”: {
        ”properties”: {
```

```
          "key": {
            "type": "string",
            "required": true
          },
          "value": {
            "type": ["string", "null"]
          }
        }
      }
    }
  }
}
```

ボディデータスキーマ定義の要素の階層構造

項番	要素名および属性名※		型	型種別	数	制約
1	data		array	配列	0 以上	－
2		key	string	文字列	1	null 不可
3		value	string	文字列	0～1	null 可

（凡例）

－：該当なし

注※

「要素名および属性名」列のインデントは，要素の階層構造を表しています。

ボディデータの出力例

上記のスキーマが適用されたボディデータの出力例を次に示します。

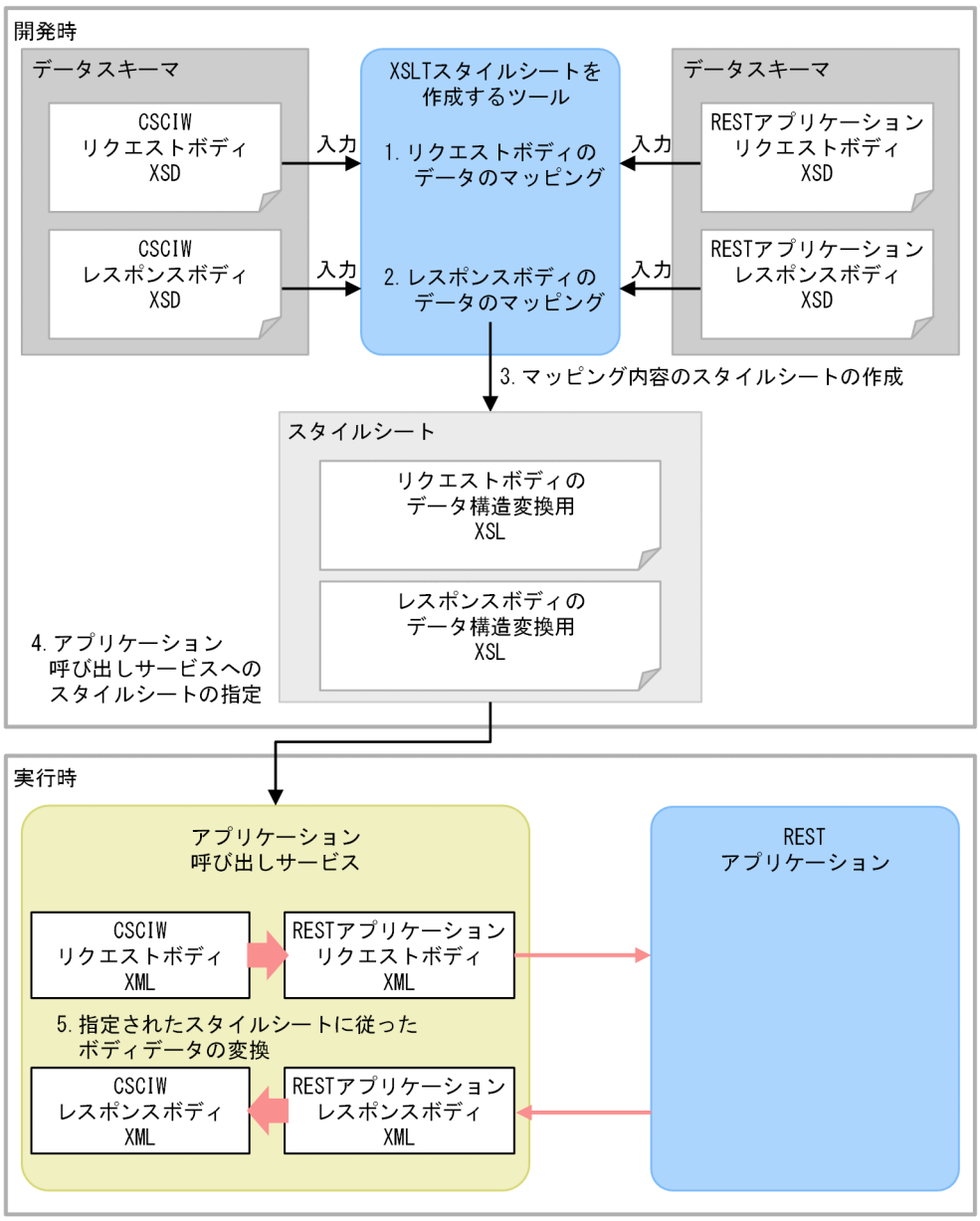
```
{
  "data":[
    { "key":"K1", "value":"V1"},
    { "key":"K2", "value":"V2"}
  ]
}
```

6.1.5　XML 形式のボディデータのスキーマ変換

アプリケーション呼び出しサービスのボディデータスキーマと異なるデータスキーマを持つ REST アプリケーションを呼び出したい場合は，各データスキーマにデータ構造変換のためのスタイルシートを指定する必要があります。

ボディデータのスキーマ変換の手順を次の図に示します。

図 6-1　ボディデータのスキーマ変換



[説明]

■開発時

1. リクエストボディのデータのマッピング

XSLT スタイルシートを作成するツールを使用して，入力スキーマおよび出力スキーマに次に示すボディデータスキーマを指定して，リクエストボディデータをマッピングします。

・入力スキーマ：CSCIW のリクエストのボディデータスキーマ

・出力スキーマ：REST アプリケーションのリクエストのボディデータスキーマ

2. レスポンスボディのデータのマッピング

XSLT スタイルシートを作成するツールを使用して，入力スキーマおよび出力スキーマに次に示すボディデータスキーマを指定して，レスポンスボディデータをマッピングします。

・入力スキーマ：REST アプリケーションのレスポンスのボディデータスキーマ
・出力スキーマ：CSCIW のレスポンスのボディデータスキーマ

3. マッピング内容のスタイルシートの作成

   XSLT スタイルシートを作成するツールを使用して，マッピングした内容のスタイルシートを作成します。

4. アプリケーション呼び出しサービスへのスタイルシートの指定

   作成したスタイルシートをアプリケーション呼び出し情報ファイルに指定します。

   指定方法の詳細については，「15.3.14　REST アプリケーションの呼び出しの場合にファイルに指定する内容」を参照してください。

■実行時

5. 指定されたスタイルシートに従ったボディデータの変換

   アプリケーション呼び出しサービスが REST アプリケーションにリクエストを送信する際，指定されたスタイルシートの内容に従って，CSCIW のリクエストボディのデータが REST アプリケーションのリクエストボディのスキーマに変換されます。

   また，アプリケーション呼び出しサービスが REST アプリケーションからレスポンスを受信する際，指定されたスタイルシートの内容に従って，REST アプリケーションのレスポンスボディのデータが CSCIW のレスポンスボディのスキーマに変換されます。

関連項目

- 6.1.3　ボディデータスキーマ（XML）

6.1.6　REST アプリケーションの開発時の注意事項

アプリケーション呼び出しサービスは，REST アプリケーションの呼び出しに失敗した場合，リトライによって REST アプリケーションを複数回呼び出します。また，アプリケーション呼び出しサービスを強制停止するなどして直後の作業が完了しなかった場合，REST アプリケーションの呼び出しに成功しても，再度 REST アプリケーションを呼び出します。この場合は，リトライ回数の設定に関係なく再度呼び出します。そのため，REST アプリケーションが複数回呼び出されても正常に動作するように，べき等性を保証した実装をしてください。

6.2 アプリケーション呼び出しサービスから呼び出す Java オブジェクトを開発する

アプリケーション呼び出しサービスから呼び出す Java オブジェクトの規定について説明します。アプリケーション呼び出しサービスは，ここで説明する規定の内容に従った Java オブジェクトの呼び出しをサポートします。

6.2.1 Java オブジェクトの作成

Java クラスのインタフェース

アプリケーション呼び出しサービスが Java オブジェクトを呼び出す場合，呼び出される Java クラスに CSCIW が提供するインタフェース**CIWBpmnWorkApplication** を実装してください。

インタフェースと Java クラスのインポート

Java オブジェクト呼び出しで使用する Java クラスのソースには，次に示すインポート文を記述してください。

```
import jp.co.Hitachi.soft.csciw.bpmn.callback.CIWBpmnWorkApplication;
```

また，Java オブジェクト呼び出しで使用する Java クラスのメソッドが**CIWUserException** をスローする場合，次に示すインポート文も記述してください。

```
import jp.co.Hitachi.soft.csciw.callback.CIWUserException;
```

Java クラスのコンパイル

CSCIW が提供するインタフェース**CIWBpmnWorkApplication** を実装した Java クラスを含む JAR ファイルを作成してください。

CSCIW では，JDK11 の機能を使用しています。このため，Java オブジェクト呼び出しで使用する Java クラスは，JDK11 の機能を利用したソースとしてコンパイルしてください。

Java オブジェクト呼び出しで使用する Java クラスのソースをコンパイルする場合，次に示す JAR ファイルをクラスパスに設定します。

- CSCIW をインストールした環境の場合

  ＜ Windows のとき＞

  ```
  <CSCIWのインストールディレクトリ>¥lib¥csciwbpmn.jar
  <CSCIWのインストールディレクトリ>¥lib¥csciwcmn.jar
  ```

＜UNIX のとき＞

```
/opt/hitachi/CSCIW/lib/csciwbpmn.jar
/opt/hitachi/CSCIW/lib/csciwcmn.jar
```

- uCosminexus Business Process Developer をインストールした環境の場合

```
<uCosminexus Business Process Developerのインストールディレクトリ>¥lib¥csciwbpmn.jar
<uCosminexus Business Process Developerのインストールディレクトリ>¥lib¥csciwcmn.jar
```

関連項目

- 6.2.2　Java オブジェクトの作成時の注意事項
- 6.5　Java オブジェクトの作成例
- 13.2　CIWBpmnWorkApplication（Java オブジェクト呼び出しのインタフェース）

6.2.2　Java オブジェクトの作成時の注意事項

- Java クラスには，CSCIW が提供するインタフェースを必ず実装してください。
- Java クラスには，引数のないコンストラクタだけを実装してください。
- Java クラスのパッケージには，「jp.co.Hitachi.soft.csciw」を使用しないでください。
- Java オブジェクトの呼び出しは，複数スレッドから同時に実行される場合があるため，static フィールドを使用しないでください。
- Java オブジェクトは呼び出し元の作業ごとにインスタンス化しメソッドを呼び出します。メソッドの終了後，インスタンスは破棄されます。
- Java オブジェクトでは，CSCIW が提供する更新系 API を発行しないでください。
- アプリケーション呼び出しサービスは，Java オブジェクトの呼び出しに失敗した場合，リトライによって複数回呼び出します。また，アプリケーション呼び出しサービスを強制停止するなどして直後の作業が完了しなかった場合，Java オブジェクトの呼び出しに成功しても，再度 Java オブジェクトを呼び出します。この場合は，リトライ回数の設定に関係なく再度呼び出します。そのため，Java オブジェクトが複数回呼び出されても正常に動作するように実装してください。

6.2.3　Java オブジェクトの組み込み

Java クラスの組み込み

Java オブジェクト呼び出しで使用する Java クラスを含む JAR ファイルは，J2EE サーバのコンテナ拡張ライブラリに設定する必要があります。Cosminexus の J2EE サーバ用オプション定義ファイルの add.class.path に，Java クラスを含む JAR ファイルのファイルパスを設定してください。

Java クラスの指定

アプリケーション呼び出しサービスが呼び出す Java オブジェクトの Java クラス名は，アプリケーション呼び出し情報ファイルで指定します。Java クラス名はパッケージ名を含めて指定してください。

アプリケーション呼び出し情報ファイルの指定例

```
type=JAVA
java.invoke.class=sample.appcall.JavaCallTest
```

関連項目

- 7.9.1　コンテナ拡張ライブラリに JAR ファイルを取り込む
- 15.3.15　Java オブジェクトの呼び出しの場合にファイルに指定する内容

6.2.4　アプリケーション呼び出しサービスから Java オブジェクトに渡すデータ

アプリケーション呼び出しサービスから Java オブジェクトに，プロセスデータなどのデータを渡すことができます。

アプリケーション呼び出しサービスから Java オブジェクトに渡すデータは，アプリケーション呼び出し情報ファイルのプロパティ「java.invoke.key.<key 要素値>」に記述してください。記述したプロパティは，Java オブジェクトのメソッドの引数（java.util.Map 型）として Java オブジェクトに渡されます。

引数（Map 型）のキーには，<key 要素値>が格納されます。また，引数（Map 型）の値には，プロパティの値に指定した組み込み変数やプロセスデータキー名を解決した結果が格納されます。

引数（Map 型）の値はjava.lang.Object 型として定義されており，格納される型はプロパティの値の形式によって異なります。プロパティの値の形式と引数（Map 型）の値の型の対応を，次の表に示します。

表 6-2　プロパティの値の形式と引数（Map 型）の値の型の対応

java.invoke.key.<key 要素値>プロパティの値の形式		引数（Map 型）の値の型
<$変数（形式 1）>のプロセスデータ	次に示すどれかの場合 • $Sxxx • $Sxxx{<リスト内識別子>} • $Sxxx{@MIIndex}	String 型
	次に示すどれかの場合 • $Nxxx • $Nxxx{<リスト内識別子>} • $Nxxx{@MIIndex}	Integer 型
<$変数（形式 2）>のプロセスデータ	$Sxxx{}	List<String>型

java.invoke.key.<key 要素値>プロパティの値の形式		引数（Map 型）の値の型
<$変数（形式 2）>のプロセスデータ	$Nxxx{}	List<Integer>型
組み込み変数		String 型
その他		String 型

注

$Sxxx および$Nxxx の「xxx」は，プロセスデータキー名に使用する任意の文字列を表します。

関連項目

- 1.8.8　プロセスデータの利用
- 13.2　CIWBpmnWorkApplication（Java オブジェクト呼び出しのインタフェース）
- 15.3.10　アプリケーション呼び出し情報ファイルと Java オブジェクトに渡すデータの関係

6.2.5　Java オブジェクトからアプリケーション呼び出しサービスに渡すデータ

Java オブジェクトの実行結果のデータをアプリケーション呼び出しサービスに渡し，プロセスデータとして登録できます。

Java オブジェクトからアプリケーション呼び出しサービスに渡すデータは，Java オブジェクトのメソッドの戻り値（java.util.Map 型）に格納してください。アプリケーション呼び出しサービスは，アプリケーション呼び出し情報ファイルに記述されたプロパティ「java.return.pi.<プロセスデータキー名>=java.return.key.<key 要素値>」に従い，戻り値（Map 型）の値をプロセスデータに登録します。

次に示す例のように，アプリケーション呼び出し情報ファイルに記述したすべての<key 要素値>を，戻り値（Map 型）のキーに格納してください。また，戻り値（Map 型）の値には，<プロセスデータキー名>に対応するプロセスデータの値を格納してください。

アプリケーション呼び出し情報ファイルの記述例

```
java.return.pi.$Sresult1=java.return.key.key1
java.return.pi.$Sresult2=java.return.key.key2
java.return.pi.$Nresult3=java.return.key.key3
```

戻り値の Map の作成例

Map のキー	Map の値
"key1"	"AAA"
"key2"	"BBB"
"key3"	123

戻り値（Map 型）の値はjava.lang.Object 型として定義されており，実際に格納する型は<プロセスデータキー名>の形式によって異なります。<プロセスデータキー名>の形式と戻り値（Map 型）の値の型の対応を，次の表に示します。

表 6-3　プロセスデータキー名の形式と戻り値（Map 型）の値の型の対応

プロセスデータキー名の形式		戻り値（Map 型） の値の型
<$変数（形式 1）>	次に示すどれかの場合 • $Sxxx • $Sxxx{<リスト内識別子>} • $Sxxx{@MIIndex}	String 型
	次に示すどれかの場合 • $Nxxx • $Nxxx{<リスト内識別子>} • $Nxxx{@MIIndex}	String 型 Integer 型
<$変数（形式 2）>	$Sxxx{}	List<String>型
	$Nxxx{}	List<String>型 List<Integer>型

注

「xxx」は，プロセスデータキー名に使用する任意の文字列を表します。

関連項目

- 1.8.8　プロセスデータの利用
- 13.2　CIWBpmnWorkApplication（Java オブジェクト呼び出しのインタフェース）
- 15.3.11　アプリケーション呼び出し情報ファイルと Java オブジェクトから渡すデータの関係

6.3　アプリケーション呼び出しサービスを設定する

アプリケーション呼び出し情報ファイルなどを作成して，アプリケーション呼び出しサービスを使用するための設定をします。

6.3.1　アプリケーション呼び出し情報ファイルの設定

アプリケーション呼び出し情報ファイルでは，アプリケーション呼び出しサービスの呼び出し先の情報や呼び出す際に送受信するデータの情報を，ref 識別子ごとに設定します。

関連項目

- 15.3　アプリケーション呼び出し情報ファイル

6.3.2　アプリケーション呼び出し制御情報の設定

アプリケーション呼び出し制御情報は，ciwmngap コマンドを使用して設定します。

アプリケーション呼び出し制御情報の設定の詳細については，マニュアル『uCosminexus Service Coordinator Interactive Workflow コマンド』の「ciwmngap（アプリケーション呼び出し制御情報の管理)」を参照してください。

6.3.3　アプリケーション呼び出しグループ定義の設定

アプリケーション呼び出しグループ定義は，ciwmngapgrp コマンドを使用して設定できます。ciwmngapgrp コマンドは，アプリケーション呼び出しサービスの性能チューニング時に使用します。

アプリケーション呼び出しグループ定義の設定の詳細については，マニュアル『uCosminexus Service Coordinator Interactive Workflow コマンド』の「ciwmngapgrp（アプリケーション呼び出しグループの管理)」を参照してください。

関連項目

- 付録 C　アプリケーション呼び出しサービスの性能チューニング

6.3.4　ポーリング間隔の設定

アプリケーション呼び出しサービスのポーリング間隔は，共通設定ファイルに設定します。

詳細については，「15.2.5　共通設定ファイルに指定する内容」の「(4)　AppCallServicePollingInterval」を参照してください。

6.3.5　WorkManager の最大スレッド数の設定

WorkManager の最大スレッド数は，J2EE サーバ用ユーザプロパティファイルに設定します。

関連項目

- 9.4　WorkManager の最大スレッド数を変更する

6.3.6　REST 通信に関する設定

アプリケーション呼び出しサービスの REST 通信に関する設定は，Cosminexus アプリケーションサーバで設定します。

Cosminexus アプリケーションサーバでは，次に示す項目などを設定できます。

- 通信タイムアウト（ソケット接続，ソケット読み込み）
- ログ（Cosminexus V11 の V9 互換モードを使用する場合だけ設定できます）

詳細については，マニュアル『Cosminexus アプリケーションサーバ 機能解説 基本・開発編(Web コンテナ)』の「J2EE サーバ単位の通信タイムアウトの設定」および「NIO HTTP サーバ」を参照してください。

メモ

プロキシサーバ経由の接続，SSL プロトコルによる接続などの設定はサポート対象外です。

6.4 REST アプリケーションの実装例およびアプリケーション呼び出しの設定例

ここでは，アプリケーション呼び出しサービスが呼び出す REST アプリケーションの実装例，およびアプリケーション呼び出しの設定例を示します。

6.4.1 REST アプリケーションの実装例

REST アプリケーションを構成する Java プログラムの実装例を示します。この例では，REST アプリケーションは 3 つの Java プログラムで構成されています。

(1) PartsInfoService.java

PartsInfoService.java は，REST アプリケーションのルートリソースクラスのソースファイルです。「<コンテキストルート> ＋ "/parts/order"」の URL に対する POST リクエストを受信するリソースメソッドorder を持っています。

PartsInfoService.java の実装例

```
package xxx.usrapp;

import java.util.List;
import java.util.Calendar;

import javax.ws.rs.POST;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;

/**
 * BPMSから呼び出されるRESTアプリケーションです。
 */
@Path("/parts")
public class PartsInfoService {

  /**
   * BPMSから呼び出されるメソッドです。
   * @param reqData リクエストデータ
   * @return レスポンスデータ
   */
  @POST
  @Produces({"application/xml;charset=UTF-8"})
  @Path("order")
  public CSCIWRestBody order(CSCIWRestBody reqData) {

    //リクエストデータの取得
    if (reqData != null) {
      List<CSCIWData> list = reqData.getDataList();
      if (list != null) {
        for (CSCIWData d : list) {
```

```
          System.out.println(
            String.format("AppInfo: key=%1$s, value=%2$s", d.getKey(), d.getValue())
          );
        }
      }
    }

    //レスポンスデータの作成
    Calendar cal = Calendar.getInstance();
    cal.set(Calendar.DATE, cal.get(Calendar.DATE)+5);
    CSCIWData data = new CSCIWData("schedule", cal.getTime().toString());
    CSCIWRestBody res = new CSCIWRestBody();
    res.add(data);

    return res;
  }
}
```

(2) CSCIWRestBody.java

CSCIWRestBody.java は，REST アプリケーションが送受信するエンティティのcsciwRestBody 要素のソースファイルです。CSCIWRestBody.java は，data 要素を子要素として持っています。data 要素は，任意の数を指定できます。

CSCIWRestBody.java の実装例

```
package xxx.usrapp;

import java.util.ArrayList;
import java.util.List;

import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlRootElement;

/**
 * RESTアプリケーション送受信データのクラスです。
 */
@XmlRootElement(name="csciwRestBody")
public class CSCIWRestBody {

  private List<CSCIWData> list = new ArrayList<CSCIWData>();

  @XmlElement(name="data")
  public List<CSCIWData> getDataList() { return list; }
  public void setDataList(List<CSCIWData> l) { list = l; }

  /**
   * 指定されたデータをリストに追加します。
   * @param data 追加するデータ
   */
  public void add(CSCIWData data) {
    list.add(data);
  }
}
```

(3) CSCIWData.java

CSCIWData.java は，REST アプリケーションが送受信するエンティティのcsciwRestBody 要素の子要素である，data 要素のソースファイルです。data 要素は，子要素としてkey 要素およびvalue 要素を 1 つずつ持っています。key 要素およびvalue 要素のデータ型は，string 型です。

CSCIWData.java の実装例

```
package xxx.usrapp;

import javax.xml.bind.annotation.XmlElement;
import javax.xml.bind.annotation.XmlRootElement;

/**
 * RESTアプリケーション送受信データのクラスです。
 */
@XmlRootElement(name="data")
public class CSCIWData {

  private String key = "";   //キー
  private String value = ""; //値

  /** コンストラクタ */
  public CSCIWData(){}
  public CSCIWData(String k, String v){ key = k; value = v; }

  @XmlElement(name="key")
  public String getKey() { return key; }
  public void setKey(String k) { key = k; }

  @XmlElement(name="value")
  public String getValue() { return value; }
  public void setValue(String v) { value = v; }
}
```

6.4.2 アプリケーション呼び出しの設定例

アプリケーション呼び出し情報ファイルの記述例，およびアプリケーション呼び出し制御情報の設定例を示します。

(1) アプリケーション呼び出し情報ファイルの記述例

アプリケーション呼び出し情報ファイルの記述例を示します。この例での ref 識別子の値は，「ope1」とします。

```
type=REST
rest.request.url=http://localhost/CSPWCOMockApp/parts/order
rest.request.method=POST
rest.request.header.filepath=/home/csciw/ope1header.properties
```

```
rest.request.body.key.offset=0
rest.response.pi.$Sschedule=rest.response.body.key.schedule
```

アプリケーション呼び出し情報ファイルのファイルパスは，「<BpmnCallInformationFileDirプロパティの指定値>/ope/ope1.properties」です。

なお，アプリケーション呼び出し情報ファイルのrest.request.header.filepath プロパティで設定しているHTTP ヘッダファイルの記述例は，次のとおりです。

```
Content-type: application/xml;charset=UTF-8
```

関連項目

- 15.3.14　REST アプリケーションの呼び出しの場合にファイルに指定する内容

(2)　アプリケーション呼び出し制御情報の設定例

この例では，operationRef の値がope1 のサービスタスクに対して，アプリケーション呼び出し制御情報を設定します。

ヒント

アプリケーション呼び出し制御情報は，環境構築時のデフォルトの動作でも問題ない場合，設定は不要です。環境構築時のデフォルトの動作とは，ref 識別子共通設定のデフォルト値での動作のことです。

アプリケーション呼び出し制御情報および ref 識別子共通設定については，マニュアル『uCosminexus Service Coordinator Interactive Workflow コマンド』の「ciwmngap（アプリケーション呼び出し制御情報の管理)」を参照してください。

この例でのアプリケーション呼び出し制御情報に設定する値，およびciwmngap コマンドを使用したアプリケーション呼び出し制御情報の設定例を次に示します。

■アプリケーション呼び出し制御情報に設定する値

- BPMN 要素：サービスタスク
- ref 識別子：ope1
- 実行間隔：180（単位：秒）
- リトライ間隔：900（単位：秒）
- リトライ回数：0（単位：回）
- 最大作業件数：10000（単位：件）
- 障害復旧間隔：1500（単位：秒）

■アプリケーション呼び出し制御情報の設定例

1. アプリケーション呼び出し制御情報ファイルの作成

   アプリケーション呼び出し制御情報を更新するための入力用ファイルである，アプリケーション呼び出し制御情報ファイルを作成します。

   次に示すアプリケーション呼び出し制御情報ファイルの記入例では，「■アプリケーション呼び出し制御情報に設定する値」で示した各値を記入しています。

   アプリケーション呼び出し制御情報ファイルの記入例

   ```
   #UPDATEOPTION,REFTYPE,REF,EXECUTEINTERVAL,RETRYINTERVAL,RETRYCOUNT,WORKITEMMAX,RECOVERYTIME
   U,ope,ope1,180,900,0,10000,1500
   ```

2. ciwmngap コマンドの実行

   作成したアプリケーション呼び出し制御情報ファイルの内容を適用するために，アプリケーション呼び出し制御情報ファイルのファイルパスを指定したciwmngap コマンドを実行します。

   ciwmngap コマンドの指定形式

   ```
   ciwmngap -sid 〈システムID〉 -chg -apdf 〈アプリケーション呼び出し制御情報ファイルのファイルパス〉
   ```

ヒント

アプリケーション呼び出し制御情報ファイルおよびciwmngap コマンドについては，マニュアル『uCosminexus Service Coordinator Interactive Workflow コマンド』の「ciwmngap（アプリケーション呼び出し制御情報の管理)」を参照してください。

関連項目

- 6.3.2　アプリケーション呼び出し制御情報の設定

6.5　Java オブジェクトの作成例

Java オブジェクト呼び出しに使用する Java クラスの作成例を示します。

Java クラスの作成例

```
public class UserWorkApplication implements CIWBpmnWorkApplication {
    /**
    * Javaオブジェクトの業務処理の実装
    */
    @Override
    public Map<String,Object> execute(
        Map<String, Object> aParameters
        )
        throws CIWUserException{
        // ---------------------------------------------------
        //  （１）情報の取得
        // ---------------------------------------------------
        // アプリケーション呼び出し情報ファイルのパラメタ参照
        String param1 = aParameters.get(”key1”).toString();
        // ---------------------------------------------------
        //  （２）業務処理
        // ---------------------------------------------------
        List<String> list = new ArrayList<String>();
        list.add(param1);
        list.add(null);
        list.add(”ccc”);
        // ---------------------------------------------------
        //  （３）処理結果を戻り値に設定
        // ---------------------------------------------------
        return Map.ofEntries(
           Map.entry(”singlekey”, ”OK”),
           Map.entry(”listkey”, list) );
    }
}
```

7 ワークフローシステムを構築する

実行環境でのワークフローシステムの構築について説明します。

7.1　ワークフローシステムの構築の流れ

ワークフローシステムの構築の流れを説明します。

ここでは，ワークフローシステムの構築の流れを次の図で示します。

なお，図中の数字は，対応する内容を説明している節，または項の番号を表しています。

重要

- この節で説明する設定方法は，バージョンごとに異なります。対応するバージョンのマニュアルをご確認ください。
- ワークフローシステムを構築する前に，前提プログラムおよびCSCIWをインストールしておく必要があります。

  前提プログラムおよびCSCIWのインストール方法については，マニュアル『uCosminexus Service Coordinator Interactive Workflow システム構築・運用ガイド』の「4.2　プログラムのインストール」を参照してください。
- UNIXの場合，CSCIWの業務アプリケーションやコマンドは，rootユーザで実行する必要があります。

  rootユーザ以外の任意のOSユーザ（例えばComponent Container管理者）で実行する場合は，CSCIWインストールディレクトリ（/opt/hitachi/CSCIW）配下のすべてのディレクトリとファイルの所有者およびグループを，chownコマンドで変更してください。コマンドの指定方法を次に示します。

```
chown -R <owner>:<group> /opt/hitachi/CSCIW
```

図 7-1　ワークフローシステムの構築の流れ

関連項目

- 7.2　ワーク管理データベースを作成する（HiRDB の場合）
- 7.3　ワーク管理データベースを作成する（ORACLE の場合）
- 7.4　ワーク管理データベースを作成する（PostgreSQL の場合）
- 7.5　データベースへのアクセス権限を付与する（HiRDB の場合）

7.2　ワーク管理データベースを作成する（HiRDB の場合）

SQL スクリプトファイルを使用して，テーブルやインデクスなどを HiRDB に作成します。

HiRDB の SQL 最適化についての注意事項

HiRDB の SQL 最適化オプション，SQL 拡張最適化オプションはデフォルトで指定される最適化方法を無効にしないでください。無効にした場合，性能が劣化するおそれがあります。

該当するクライアント環境定義

PDSQLOPTLVL：SQL 最適化オプション

PDADDITIONALOPTLVL：SQL 拡張最適化オプション

該当するシステム定義のオペランド

pd_optimize_level：SQL 最適化オプション

pd_additional_optimize_level：SQL 拡張最適化オプション

操作手順

1. テキストエディタを使用して，SQL スクリプトファイル中の文字列を編集する。

■対象の SQL スクリプトファイル

- <CSCIWのインストールディレクトリ>¥sql¥createtable_hirdb.sql
- <CSCIWのインストールディレクトリ>¥sql¥createtableex_hirdb.sql
- <CSCIWのインストールディレクトリ>¥sql¥insertex_hirdb.sql

表 7-1　テーブルやインデクスの作成時に書き換えが必要な SQL スクリプトファイル中の文字列（HiRDB の場合）

項番	書き換えが必要な文字列	書き換える内容
1	<SYSTEMID>	システム ID ワーク管理データベースを一意に識別するためのシステム ID に置換してください。システム ID に指定できる文字は，半角英大文字で始まる半角英大文字および半角数字で，5 文字以内です。
2	<RDDATA>	テーブルを格納する RD エリア名 ワーク管理データベースのテーブルを格納する RD エリア名に置換してください。
3	<RDINDEX>	インデクスを格納する RD エリア名 ワーク管理データベースのインデクスを格納する RD エリア名に置換してください。

注

テーブルごとに格納先の RD エリアを変えて指定するなど，格納する RD エリアをカスタマイズできます。RD エリアの指定方法については，マニュアル『HiRDB SQL リファレンス』を参照してください。

2. HiRDB SQL Executer のpdsql コマンドを使用して，編集した SQL スクリプトファイルを実行する。

編集した SQL スクリプトファイルを実行すると，接続する認可識別子のスキーマにテーブルが作成されます。HiRDB SQL Executer の使用方法については，HiRDB SQL Executer 付属のヘルプを参照してください。

pdsql コマンドの指定形式を示します。

```
pdsql -u 接続認可識別子/パスワード
      -h HiRDBサーバのホスト名またはIPアドレス
      -n HiRDBサーバのポート番号
       く編集したSQLスクリプトファイルのパス
```

注

接続認可識別子には，CONNECT 権限のある認可識別子を指定してください。

7.3　ワーク管理データベースを作成する（ORACLE の場合）

SQL スクリプトファイルを使用して，テーブルやインデクスなどを ORACLE データベースに作成します。

操作手順

1. テキストエディタを使用して，SQL スクリプトファイル中の文字列を編集する。

■対象の SQL スクリプトファイル

- <CSCIWのインストールディレクトリ>¥sql¥createtable_oracle.sql
- <CSCIWのインストールディレクトリ>¥sql¥createtableex_oracle.sql
- <CSCIWのインストールディレクトリ>¥sql¥insertex_oracle.sql

表 7-2　テーブルやインデクスの作成時に書き換えが必要な SQL スクリプトファイル中の文字列（ORACLE の場合）

項番	書き換えが必要な文字列	書き換える内容
1	<SYSTEMID>	システム ID ワーク管理データベースを一意に識別するためのシステム ID に置換してください。システム ID に指定できる文字は，半角英大文字で始まる半角英大文字および半角数字で，5 文字以内です。
2	<DATASPACE>	テーブルを格納する表領域名 ワーク管理データベースのテーブルを格納する表領域名に置換してください。
3	<INDEXSPACE>	インデクスを格納する表領域名 ワーク管理データベースのインデクスを格納する表領域名に置換してください。

注

テーブルごとに格納先の表領域を変えて指定するなど，格納する表領域をカスタマイズできます。表領域の指定方法については，ORACLE のマニュアルを参照してください。

2. SQL*Plus を使用して，編集した SQL スクリプトファイルを実行する。

編集した SQL スクリプトファイルを実行すると，接続するユーザのスキーマにテーブルが作成されます。SQL*Plus の使用方法については，ORACLE のマニュアルを参照してください。

SQL*Plus の指定形式を示します。

```
sqlplus　接続ユーザ名/パスワード@Oracle Net接続識別子　@編集したSQLスクリプトファイルの絶対パス
```

注

次の権限を付与したユーザで接続してください。

- CREATE SESSION システム権限

- CREATE TABLE システム権限
- CREATE VIEW システム権限

7.4　ワーク管理データベースを作成する（PostgreSQL の場合）

SQL スクリプトファイルを使用して，テーブルやインデクスなどを PostgreSQL データベースに作成します。

操作手順

1. テキストエディタを使用して，SQL スクリプトファイル中の文字列を編集する。

■対象の SQL スクリプトファイル

- <CSCIWのインストールディレクトリ>/sql/createtable_postgresql.sql
- <CSCIWのインストールディレクトリ>/sql/createtableex_postgresql.sql
- <CSCIWのインストールディレクトリ>/sql/insertex_postgresql.sql

表 7-3　テーブルやインデクスの作成時に書き換えが必要な SQL スクリプトファイル中の文字列（PostgreSQL の場合）

項番	書き換えが必要な文字列	書き換える内容
1	<SYSTEMID>	システム ID ワーク管理データベースを一意に識別するためのシステム ID に置換してください。システム ID に指定できる文字は，半角英大文字で始まる半角英大文字および半角数字で，5 文字以内です。
2	<SCHEMANAME>	CSCIW の管理用のスキーマ名 ワーク管理データベースのテーブルやインデクスを作成するスキーマ名に置換してください。
3	<DATASPACE>	テーブルを格納するテーブル空間名 ワーク管理データベースのテーブルを格納するテーブル空間名に置換してください。
4	<INDEXSPACE>	インデクスを格納するテーブル空間名 ワーク管理データベースのインデクスを格納するテーブル空間名に置換してください。

注

テーブルごとに格納先のテーブル空間を変えて指定するなど，格納するテーブル空間をカスタマイズできます。テーブル空間の指定方法については，PostgreSQL のマニュアルを参照してください。

2. psql コマンドを使用して，編集した SQL スクリプトファイルを実行する。

編集した SQL スクリプトファイルを実行すると，接続するユーザのスキーマにテーブルが作成されます。psql コマンドの使用方法については，PostgreSQL のマニュアルを参照してください。

psql コマンドの指定形式を示します。

```
psql -U 接続ユーザ名 -d 接続データベース名 -f 編集したSQLスクリプトファイルの絶対パス
```

注 1

次の権限を付与したユーザで接続してください。

データベースに対する権限

- CONNECT 権限

スキーマに対する権限

- CREATE 権限
- USAGE 権限

テーブル空間に対する権限

- CREATE 権限

注 2

PostgreSQL 設定ファイル（postgresql.conf）について，standard_conforming_strings のプロパティは設定しないでください。

7.5　データベースへのアクセス権限を付与する（HiRDB の場合）

業務アプリケーションが接続するための認可識別子に，ワーク管理データベースのテーブルやインデクスなどへアクセスするための権限を付与します。ワーク管理データベースのテーブルやインデクスを作成した認可識別子と，業務アプリケーションが接続するための認可識別子とが異なっている場合に，アクセス権限を付与する必要があります。ワーク管理データベースのテーブルやインデクスを作成した認可識別子と同じ認可識別子を使用する場合は，アクセス権限の付与は不要です。

操作手順

1. テキストエディタを使用して，SQL スクリプトファイル中の文字列を編集する。

■対象の SQL スクリプトファイル

- <CSCIWのインストールディレクトリ>/sql/granttable_hirdb.sql
- <CSCIWのインストールディレクトリ>/sql/granttableex_hirdb.sql

表 7-4　アクセス権限を付与する際に書き換えが必要な SQL スクリプトファイル中の文字列（HiRDB の場合）

項番	書き換えが必要な文字列	書き換える内容
1	<SCHEMANAME>	CSCIW の管理用の認可識別子 ワーク管理データベースのテーブルやインデクスを作成した認可識別子に置換してください。
2	<SYSTEMID>	システム ID ワーク管理データベースを一意に識別するためのシステム ID に置換してください。システム ID に指定できる文字は，半角英大文字で始まる半角英大文字および半角数字で，5 文字以内です。
3	<USERNAME>	業務アプリケーション用の認可識別子 業務アプリケーションが接続するための認可識別子に置換してください。

2. HiRDB SQL Executer の pdsql コマンドを使用して，編集した SQL スクリプトファイルを実行する。

編集した SQL スクリプトファイルを実行すると，アクセス権限が付与されます。SQL スクリプトファイルの実行方法は，ワーク管理データベースを作成するときと同じです。

関連項目

- 7.2　ワーク管理データベースを作成する（HiRDB の場合）

7.6　データベースへのアクセス権限を付与する（ORACLE の場合）

業務アプリケーションが接続するためのユーザに，ワーク管理データベースのテーブルやインデクスなどへアクセスするための権限を付与します。ワーク管理データベースのテーブルやインデクスを作成したユーザと，業務アプリケーションが接続するためのユーザとが異なっている場合に，アクセス権限を付与する必要があります。ワーク管理データベースのテーブルやインデクスを作成したユーザと同じユーザを使用する場合は，アクセス権限の付与は不要です。

操作手順

1. テキストエディタを使用して，SQL スクリプトファイル中の文字列を編集する。

■対象の SQL スクリプトファイル

- <CSCIWのインストールディレクトリ>/sql/granttable_oracle.sql
- <CSCIWのインストールディレクトリ>/sql/granttableex_oracle.sql

表 7-5　アクセス権限を付与する際に書き換えが必要な SQL スクリプトファイル中の文字列（ORACLE の場合）

項番	書き換えが必要な文字列	書き換える内容
1	<SCHEMANAME>	CSCIW の管理用のユーザ名 ワーク管理データベースのテーブルやインデクスを作成したユーザ名に置換してください。
2	<SYSTEMID>	システム ID ワーク管理データベースを一意に識別するためのシステム ID に置換してください。システム ID に指定できる文字は，半角英大文字で始まる半角英大文字および半角数字で，5 文字以内です。
3	<USERNAME>	業務アプリケーション用のユーザ名 業務アプリケーションが接続するためのユーザ名に置換してください。

2. SQL*Plus を使用して，編集した SQL スクリプトファイルを実行する。

編集した SQL スクリプトファイルを実行すると，アクセス権限が付与されます。SQL スクリプトファイルの実行方法は，ワーク管理データベースを作成するときと同じです。

関連項目

- 7.3　ワーク管理データベースを作成する（ORACLE の場合）

7.7　データベースへのアクセス権限を付与する（PostgreSQL の場合）

業務アプリケーションが接続するためのユーザに，ワーク管理データベースのテーブルやインデクスなどへアクセスするための権限を付与します。ワーク管理データベースのテーブルやインデクスを作成したユーザと，業務アプリケーションが接続するためのユーザとが異なっている場合に，アクセス権限を付与する必要があります。ワーク管理データベースのテーブルやインデクスを作成したユーザと同じユーザを使用する場合は，アクセス権限の付与は不要です。

操作手順

1. テキストエディタを使用して，SQL スクリプトファイル中の文字列を編集する。

■対象の SQL スクリプトファイル

- <CSCIWのインストールディレクトリ>/sql/granttable_postgresql.sql
- <CSCIWのインストールディレクトリ>/sql/granttableex_postgresql.sql

表 7-6　アクセス権限を付与する際に書き換えが必要な SQL スクリプトファイル中の文字列（PostgreSQL の場合）

項番	書き換えが必要な文字列	書き換える内容
1	<SCHEMANAME>	CSCIW の管理用のスキーマ名 ワーク管理データベースのテーブルやインデクスを作成したスキーマ名に置換してください。
2	<SYSTEMID>	システム ID ワーク管理データベースを一意に識別するためのシステム ID に置換してください。システム ID に指定できる文字は，半角英大文字で始まる半角英大文字および半角数字で，5 文字以内です。
3	<USERNAME>	業務アプリケーション用のユーザ名 業務アプリケーションが接続するためのユーザ名に置換してください。

2. psql コマンドを使用して，編集した SQL スクリプトファイルを実行する。

編集した SQL スクリプトファイルを実行すると，アクセス権限が付与されます。SQL スクリプトファイルの実行方法は，ワーク管理データベースを作成するときと同じです。

3. 業務アプリケーションが接続するためのユーザに，スキーマに対する USAGE 権限を付与する。

なお，スキーマが「public」，または接続ユーザがオーナーの場合は，スキーマに対する権限の付与は不要です。

ユーザ名に，スキーマ名に対する USAGE 権限を付与する SQL コマンドの指定形式を次に示します。

```
GRANT USAGE ON SCHEMA スキーマ名 TO ユーザ名;
```

関連項目

- 7.4　ワーク管理データベースを作成する（PostgreSQL の場合）

7.8 CSCIW の実行環境を初期化する

CSCIW の実行環境を初期化します。

操作手順

1. CSCIW_HOME 環境変数を設定する。

   COSMINEXUS_HOME 環境変数に Cosminexus のインストールフォルダが正しく設定されていることを確認してください。UNIX の場合，確認は不要です。

   CSCIW_HOME 環境変数は，次のように設定してください。

   - Windows の場合

   ```
   CSCIW_HOME=<CSCIWインストールフォルダ>
   ```

   重要

   CSCIW のインストールフォルダ以外は設定しないでください。次のように，CSCIW_HOME 環境変数に，誤った設定（ほかの環境変数の参照先フォルダを設定するなど）をした場合，動作は保証できません。

   ＜誤った設定例＞

   ```
   CSCIW_HOME=%PROGRAMFILES%¥HITACHI¥CSCIW
   ```

   - UNIX の場合

   ```
   CSCIW_HOME=/opt/hitachi/CSCIW
   ```

2. CSCIW のセットアッププロパティファイルを環境に合わせて編集する。

   セットアッププロパティファイルの設定内容については，マニュアル『uCosminexus Service Coordinator Interactive Workflow システム構築・運用ガイド』の「付録 A.2　セットアッププロパティファイル」を参照してください。

   セットアッププロパティファイルの格納パスは，次のとおりです。

   - Windows の場合

   ```
   %CSCIW_HOME%¥conf¥csciwsetup.properties
   ```

   - UNIX の場合

   ```
   ${CSCIW_HOME}/conf/csciwsetup.properties
   ```

   重要

   BPMN 連携機能を使用する場合

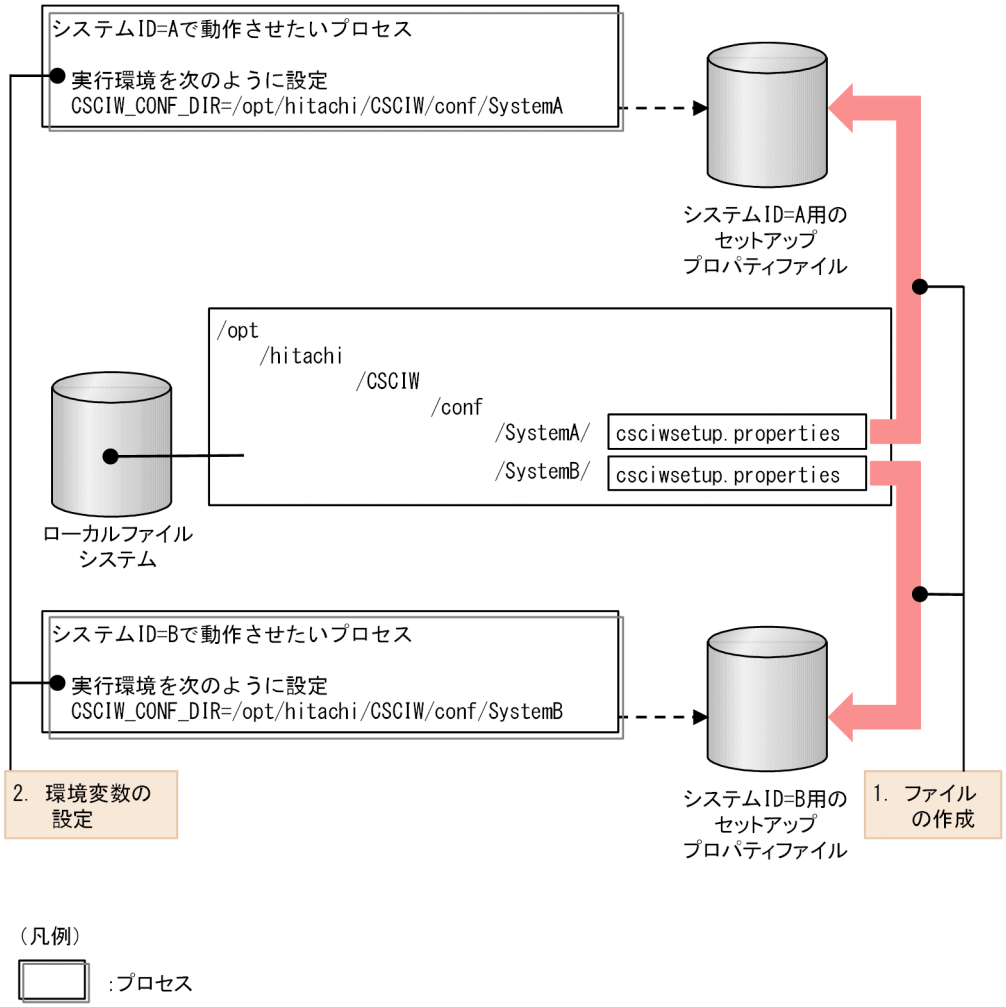
- セットアッププロパティファイル（csciwsetup.properties）に，「BpmnMode=true」を追加する必要があります。
- 「UseApplicationCallService=true」の指定は不要です。

セットアッププロパティファイルの指定例を示します。

```
#############################################################################
## All Rights Reserved. Copyright (C) 2006, 2013, Hitachi, Ltd.            ##
##                                                                         ##
## Licensed Material of Hitachi, Ltd.                                      ##
## Reproduction, use, modification or disclosure otherwise than            ##
## permitted in the License Agreement is strictly prohibited.              ##
##                                                                         ##
#############################################################################
SystemID=SYS1
BpmnMode=true
```

なお，CSCIW をマルチインスタンス構成にする場合は，次のようにセットアッププロパティファイルを設定してください。

図 7-2　CSCIW をマルチインスタンス構成にする場合のセットアッププロパティファイルの設定



［説明］

1. システム ID ごとにセットアッププロパティファイルを作成する
2. 動作させたいシステム ID のセットアッププロパティファイルの格納パスをCSCIW_CONF_DIR 環境変数に設定する

3. コマンド用環境設定ファイルを環境に合わせて編集する。

コマンド用環境設定ファイルの設定内容については，マニュアル『uCosminexus Service Coordinator Interactive Workflow システム構築・運用ガイド』の「付録 A.3　コマンド用環境設定ファイル」を参照してください。

コマンド用環境設定ファイルの格納パスは，次のとおりです。

- Windows の場合

```
%CSCIW_HOME%¥conf¥csciwcmdconf.bat
```

- UNIX の場合

```
${CSCIW_HOME}/conf/csciwcmdconf
```

なお，CSCIW をマルチインスタンス構成にする場合は，次のようにコマンド用環境設定ファイルを設定してください。

1. システム ID ごとにコマンド用環境設定ファイルを作成する
2. 動作させたいシステム ID のコマンド用環境設定ファイルの格納パスをCSCIW_CONF_DIR 環境変数に設定する

4. ciwsetenv コマンドで，CSCIW の実行環境を初期化する。

次に示す形式でciwsetenv コマンドを実行すると，CSCIW の実行環境に必要な設定項目にデフォルト値を設定し，システム ID の設定情報を登録します。

```
ciwsetenv -sid 〈システムID〉 -f 〈環境構築ファイル名〉
```

ciwsetenv コマンドの詳細については，マニュアル『uCosminexus Service Coordinator Interactive Workflow コマンド』を参照してください。

7.9　アプリケーションサーバを設定する

アプリケーションサーバ（Cosminexus）上で CSCIW を使用するための設定をします。

次の 2 とおりの手順が記載されている場合は，どちらかの手順を実施してください。

- 運用管理ポータルを使用して設定する場合
- 定義ファイルおよびサーバ管理コマンドを使用して設定する場合

ヒント

Cosminexus の J2EE サーバを V9 互換モードで動作させる場合は，アプリケーションサーバの設定を開始する前に V9 互換モード用のファイルへの入れ替えが必要です。

マニュアル『uCosminexus Service Coordinator Interactive Workflow システム構築・運用ガイド』の「付録 J.1　互換モード用のファイルへの入れ替え」を参照して，ファイルを入れ替えてください。

7.9.1　コンテナ拡張ライブラリに JAR ファイルを取り込む

J2EE サーバ用オプション定義とサーバ管理コマンド用オプション定義を設定し，コンテナ拡張ライブラリへの取り込みを行います。

操作手順

1. J2EE サーバ用オプション定義ファイルの内容を設定する。

   BPMN 連携機能を使用する場合，Cosminexus の J2EE サーバ用オプション定義ファイルの add.class.path に次のファイルを設定してください。

   - CSCIW ライブラリ用のファイル（csciw.jar および csciwbpmn.jar）
   - JAX-RS ライブラリ用のファイル
     - Cosminexus J2EE サーバを V11 推奨モードで動作させる場合
       jaxrs-impl.jar および jaxrs-jackson.jar
     - Cosminexus J2EE サーバを V9 互換モードで動作させる場合
       cjjaxws.jar および cjjaxrs.jar
   - JDBC ドライバのライブラリ用のファイル
   - 日立ネットワークオブジェクトプラザトレース共通ライブラリ用のファイル（hntrlib2j64.jar）

   J2EE サーバ用オプション定義ファイルの指定形式を示します。

   - Cosminexus J2EE サーバを V11 推奨モードで動作させる場合

＜Windows のとき＞

```
add.class.path=<CSCIWのインストールディレクトリ>¥lib¥csciw.jar
add.class.path=<CSCIWのインストールディレクトリ>¥lib¥csciwbpmn.jar
add.class.path=<cosminexus.home>¥CC¥javaee¥1100¥lib¥jaxrs-impl.jar
add.class.path=<cosminexus.home>¥CC¥javaee¥1100¥lib¥jaxrs-jackson.jar
add.class.path=<JDBCドライバのライブラリのパス>
add.class.path=<プログラムファイルフォルダ>¥Hitachi¥HNTRLib2¥classes¥hntrlib2j64.jar
```

＜UNIX のとき＞

```
add.class.path=/opt/hitachi/CSCIW/lib/csciw.jar
add.class.path=/opt/hitachi/CSCIW/lib/csciwbpmn.jar
add.class.path=/opt/Cosminexus/CC/javaee/1100/lib/jaxrs-impl.jar
add.class.path=/opt/Cosminexus/CC/javaee/1100/lib/jaxrs-jackson.jar
add.class.path=<JDBCドライバのライブラリのパス>
add.class.path=/opt/hitachi/HNTRLib2/classes/hntrlib2j64.jar
```

- Cosminexus J2EE サーバを V9 互換モードで動作させる場合

＜Windows のとき＞

```
add.class.path=<CSCIWのインストールディレクトリ>¥lib¥csciw.jar
add.class.path=<CSCIWのインストールディレクトリ>¥lib¥csciwbpmn.jar
add.class.path=<cosminexus.home>¥jaxws¥lib¥cjjaxws.jar
add.class.path=<cosminexus.home>¥jaxrs¥lib¥cjjaxrs.jar
add.class.path=<JDBCドライバのライブラリのパス>
add.class.path=<プログラムファイルフォルダ>¥Hitachi¥HNTRLib2¥classes¥hntrlib2j64.jar
```

＜UNIX のとき＞

```
add.class.path=/opt/hitachi/CSCIW/lib/csciw.jar
add.class.path=/opt/hitachi/CSCIW/lib/csciwbpmn.jar
add.class.path=/opt/Cosminexus/jaxws/lib/cjjaxws.jar
add.class.path=/opt/Cosminexus/jaxrs/lib/cjjaxrs.jar
add.class.path=<JDBCドライバのライブラリのパス>
add.class.path=/opt/hitachi/HNTRLib2/classes/hntrlib2j64.jar
```

ヒント

J2EE サーバ用オプション定義は，次のどちらかの手順で設定できます。

＜運用管理ポータルを使用して設定する場合＞

1. 運用管理ポータルで［論理サーバの環境設定］アンカーをクリックする。
2. ツリーペインで次の順にクリックする。
   ［<操作対象の運用管理ドメイン名>］－［論理 J2EE サーバ］－［J2EE サーバ］－［<サーバ名称>］
3. 右ペインの［コンテナ］タブ，［J2EE］タブをクリックし，［J2EE コンテナの設定］画面を表示する。
4. ［拡張パラメタ］の欄に，上で示した「J2EE サーバ用オプション定義ファイルの指定形式」を参考にコンテナ拡張ライブラリを設定する。

詳細については，マニュアル『Cosminexus アプリケーションサーバ 運用管理ポータル操作ガイド』を参照してください。

<定義ファイルおよびサーバ管理コマンドを使用して設定する場合>

1. usrconf.cfg（J2EE サーバ用オプション定義ファイル）を編集する。

詳細については，マニュアル『Cosminexus アプリケーションサーバ リファレンス定義編(サーバ定義)』の「usrconf.cfg（J2EE サーバ用オプション定義ファイル)」を参照してください。

2. サーバ管理コマンド用オプション定義ファイルの内容を設定する。

BPMN 連携機能を使用する場合，サーバ管理コマンド用オプション定義ファイルに CSCIW ライブラリ用のファイル（csciw.jar，csciwbpmn.jar，およびhntrlib2j64.jar）を設定してください。

- Windows の場合

  サーバ管理コマンド用オプション定義ファイルのUSRCONF_JVM_CLASSPATH キーに，各ファイル名を設定します。

  すでにUSRCONF_JVM_CLASSPATH キーにほかのファイル名が指定されている場合は，セミコロンで区切ってファイル名を追加してください。

  Windows の場合のサーバ管理コマンド用オプション定義ファイルの指定例を示します。

```
set USRCONF_JVM_CLASSPATH=<CSCIWのインストールディレクトリ>¥lib¥csciw.jar;<CSCIWのインストールディレクトリ>¥lib¥csciwbpmn.jar;<プログラムファイルフォルダ>¥Hitachi¥HNTRLib2¥classes¥hntrlib2j64.jar
```

- UNIX の場合

  サーバ管理コマンド用オプション定義ファイルのUSRCONF_JVM_CLPATH キーに，各ファイル名を設定します。

  すでにUSRCONF_JVM_CLPATH キーにほかのファイル名が指定されている場合は，コロンで区切ってファイル名を追加してください。

  UNIX の場合のサーバ管理コマンド用オプション定義ファイルの指定例を示します。

```
set USRCONF_JVM_CLPATH=/opt/hitachi/CSCIW/lib/csciw.jar:/opt/hitachi/CSCIW/lib/csciwbpmn.jar:/opt/hitachi/HNTRLib2/classes/hntrlib2j64.jar
```

重要

次に示す J2EE サーバ単位の REST（JAX-RS）の通信タイムアウト値のデフォルトは 0（タイムアウトなし）のため，REST アプリケーションが無応答になった場合などに，アプリケーション呼び出しサービスは，応答を待ち続けます。

- ejbserver.javaee.jaxrs.config.client.connectTimeout（クライアントソケットの接続タイムアウト）

- ejbserver.javaee.jaxrs.config.client.readTimeout（クライアントソケットの読み込みタイムアウト）

そのため，どちらの通信タイムアウト値も 0 以外の値を設定してください。また，クライアントソケットの読み込みタイムアウトは REST アプリケーションの処理時間より大きな値を設定してください。

J2EE サーバ単位の JAX-RS の通信タイムアウトの設定については，マニュアル『Cosminexus アプリケーションサーバ 機能解説 基本・開発編(Web コンテナ)』の「J2EE サーバ単位の通信タイムアウトの設定」を参照してください。

次の場合は，アプリケーション呼び出し情報ファイルを使用して設定します。

- ref 識別子ごとに通信タイムアウト値を設定したい場合
- Cosminexus に同梱されている JAX-RS ライブラリをコンテナ拡張ライブラリに設定しない場合

  この場合，J2EE サーバ単位の REST（JAX-RS）の通信タイムアウト値は無効になります。

詳細については，「15.3.14　REST アプリケーションの呼び出しの場合にファイルに指定する内容」の「(6)　rest.request.read.timeout」と「(7)　rest.request.connect.timeout」を参照してください。

7.9.2　環境変数を取り込む

環境変数を設定します。設定する環境変数を次に示します。

- Windows の場合

```
CSCIW_HOME=<CSCIWインストールフォルダ>
```

- UNIX の場合

```
CSCIW_HOME=/opt/hitachi/CSCIW
```

なお，CSCIW_CONF_DIR 環境変数でシステム ID を切り替えたい場合は，CSCIW_CONF_DIR 環境変数も設定してください。

(1)　運用管理ポータルを使用して設定する場合

運用管理ポータルを使用して設定する場合に，環境変数を取り込む方法を説明します。

操作手順

1. 運用管理ポータルにログインする。

2. 運用管理ポータルで［論理サーバの環境設定］アンカーをクリックする。

3. ツリーペインで次の順にクリックする。
［<操作対象の運用管理ドメイン名>］－［論理 J2EE サーバ］－［J2EE サーバ］－［<操作対象の J2EE サーバ名>］

4. 右ペインの［環境変数］タブをクリックし，［環境変数の設定］画面を表示する。

5. ［環境変数］の欄に，環境変数を設定する。
詳細は，マニュアル『Cosminexus 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「J2EE サーバの環境変数の設定」を参照してください。

(2) 定義ファイルおよびサーバ管理コマンドを使用して設定する場合

定義ファイルおよびサーバ管理コマンドを使用して設定する場合に，環境変数を取り込む方法を説明します。

操作手順

1. cjstartsv コマンドを実行するシェルに環境変数を設定する。
cjstartsv コマンドの詳細は，マニュアル『Cosminexus アプリケーションサーバ　リファレンス コマンド編』を参照してください。

7.9.3 Cosminexus を起動する

Cosminexus を起動します。

(1) 運用管理ポータルを使用して設定する場合

運用管理ポータルを使用して設定する場合に，Cosminexus を起動する方法を説明します。

操作手順

1. 運用管理ポータルで［論理サーバの起動／停止］アンカーをクリックし，対象の J2EE サーバを起動する。
詳細は，マニュアル『Cosminexus 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「論理サーバの起動／停止の設定」を参照してください。

(2) 定義ファイルおよびサーバ管理コマンドを使用して設定する場合

定義ファイルおよびサーバ管理コマンドを使用して設定する場合に，Cosminexus を起動する方法を説明します。

操作手順

1. cjstartsv コマンドで J2EE サーバを起動する。

   cjstartsv コマンドの指定形式を次に示します。

   ```
   cjstartsv 〈J2EEサーバ名〉
   ```

   cjstartsv コマンドの詳細は，マニュアル『Cosminexus アプリケーションサーバ　リファレンス コマンド編』の「J2EE サーバを操作するコマンド」を参照してください。

7.9.4　DB Connector を設定する

データベースと接続するために DB Connector を設定します。

(1)　運用管理ポータルを使用して設定する場合

運用管理ポータルを使用して設定する場合に，データベースと接続するための設定方法を説明します。

操作手順

1. DB Connector をインポートする。
   a. 運用管理ポータルにログインする。
   b. 運用管理ポータルで［論理サーバのアプリケーション管理］アンカーをクリックする。
   c. ツリーペインで次の順にクリックする。

      ［<操作対象の運用管理ドメイン名>］－［論理 J2EE サーバ］－［J2EE サーバ］－［<操作対象の J2EE サーバ名>］－［リソース］－［リソースアダプタ］
   d. 右ペインの［インポート］タブをクリックし，［リソースアダプタのインポート］画面を表示する。
   e. インポートするリソースアダプタの指定方法として，［DB Connector］を選択する。
   f. メニューに表示される DB Connector（RAR ファイル）から，接続するデータベースに対応したリソースアダプタを指定する。
   g. ［リソースアダプタ名称］に「DB_Connector_for_CSCIW」を入力する。
   h. ［実行］ボタンをクリックする。
2. DB Connector のプロパティを設定する。
   a. 右ペインの［プロパティ設定］タブをクリックし，［リソースアダプタのプロパティ設定］画面を表示する。
   b. ［リソース名］が「DB_Connector_for_CSCIW」になっていることを確認する。
   c. プロパティを設定するリソースアダプタの［基本設定］アンカーをクリックする。
   d. ［リソースアダプタの基本設定］画面で，プロパティを設定する。

e. [設定] ボタンをクリックする。

3. DB Connector の接続テストを実施する。

a. 右ペインの［開始／停止］タブをクリックし，［リソースアダプタの開始／停止］画面を表示する。

b. 接続テストを実行するリソースアダプタの［接続テスト］アンカーをクリックする。

c. 内容を確認して，［はい］ボタンをクリックする。

4. DB Connector を開始する。

a. 右ペインの［開始／停止］タブをクリックし，［リソースアダプタの開始／停止］画面を表示する。

b. 開始するリソースアダプタの［開始］アンカーをクリックする。

c. 内容を確認して，［はい］ボタンをクリックする。

(2) 定義ファイルおよびサーバ管理コマンドを使用して設定する場合

定義ファイルおよびサーバ管理コマンドを使用して設定する場合に，データベースと接続するための設定方法を説明します。

操作手順

1. cjimportres コマンドを使用して，DB Connector をインポートする。

インポート方法については，マニュアル『Cosminexus 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「DB Connector のインポート」または「リソースアダプタのインポート」の説明を参照してください。

cjimportres コマンドの指定形式を次に示します。

- HiRDB の場合

```
cjimportres <J2EEサーバ名> -type rar -f <Cosminexusインストールディレクトリ>/CC/DBConnector/DBConnector_HiRDB_Type4_CP.rar
```

- Oracle の場合

```
cjimportres <J2EEサーバ名> -type rar -f <Cosminexusインストールディレクトリ>/CC/DBConnector/DBConnector_Oracle_CP.rar
```

- PostgreSQL の場合

```
cjimportres <J2EEサーバ名> -type rar -f <Cosminexusインストールディレクトリ>/CC/DBConnector/DBConnector_PostgreSQL_CP.rar
```

cjimportres コマンドの詳細については，マニュアル『Cosminexus アプリケーションサーバ リファレンス コマンド編』を参照してください。

2. cjgetresprop コマンドおよびcjsetresprop コマンドを使用して，DB Connector のプロパティを設定する。

cjgetresprop コマンドで取得した Connector 属性ファイルの，DB Connector のリソースアダプタの表示名に「DB_Connector_for_CSCIW」（CSCIW のデフォルト）を設定してください。cjsetresprop コマンドで属性を設定します。

```
<display-name xml:lang="en">DB_Connector_for_CSCIW</display-name>
```

設定方法については，マニュアル『Cosminexus 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「DB Connector のプロパティ定義」または「リソースアダプタのプロパティ定義」の説明を参照してください。

項目の詳細については，マニュアル『Cosminexus アプリケーションサーバ リファレンス 定義編（アプリケーション／リソース定義)』の「Connector 属性ファイル」の説明を参照してください。

cjgetresprop コマンドの指定形式を次に示します。

- HiRDB の場合

```
cjgetresprop <J2EEサーバ名> -type rar -resname DB_Connector_for_HiRDB_Type4 -c <属性ファイルパス>
```

- Oracle の場合

```
cjgetresprop <J2EEサーバ名> -type rar -resname DB_Connector_for_Oracle -c <属性ファイルパス>
```

- PostgreSQL の場合

```
cjgetresprop <J2EEサーバ名> -type rar -resname DB_Connector_for_PostgreSQL -c <属性ファイルパス>
```

cjsetresprop コマンドの指定形式を次に示します。

- HiRDB の場合

```
cjsetresprop <J2EEサーバ名> -type rar -resname DB_Connector_for_HiRDB_Type4 -c <属性ファイルパス>
```

- Oracle の場合

```
cjsetresprop <J2EEサーバ名> -type rar -resname DB_Connector_for_Oracle -c <属性ファイルパス>
```

- PostgreSQL の場合

```
cjsetresprop <J2EEサーバ名> -type rar -resname DB_Connector_for_PostgreSQL -c <属性ファイルパス>
```

cjgetresprop コマンドおよびcjsetresprop コマンドの詳細については，マニュアル『Cosminexus アプリケーションサーバ　リファレンス コマンド編』を参照してください。

3. cjdeployrar コマンドを使用して，DB Connector をデプロイする。

設定方法については，マニュアル『Cosminexus 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「DB Connector のデプロイ」または「リソースアダプタのデプロイ」の説明を参照してください。

cjdeployrar コマンドの指定形式を次に示します。

```
cjdeployrar <J2EEサーバ名> -resname DB_Connector_for_CSCIW
```

cjdeployrar コマンドの詳細については，マニュアル『Cosminexus アプリケーションサーバ　リファレンス コマンド編』を参照してください。

4. cjtestres コマンドを使用して，DB Connector の接続テストを実施する。

接続テストの実施方法については，マニュアル『Cosminexus 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「DB Connector の接続テスト」または「J2EE リソースアダプタの接続テスト」の説明を参照してください。

cjtestres コマンドの指定形式を次に示します。

```
cjtestres <J2EEサーバ名> -type rar -resname DB_Connector_for_CSCIW
```

cjtestres コマンドの詳細については，マニュアル『Cosminexus アプリケーションサーバ　リファレンス コマンド編』を参照してください。

5. cjstartrar コマンドを使用して，DB Connector を開始する。

開始方法については，マニュアル『Cosminexus 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「DB Connector の開始」または「J2EE リソースアダプタの開始」の説明を参照してください。

cjstartrar コマンドの指定形式を次に示します。

```
cjstartrar <J2EEサーバ名> -resname DB_Connector_for_CSCIW
```

cjstartrar コマンドの詳細については，マニュアル『Cosminexus アプリケーションサーバ　リファレンス コマンド編』を参照してください。

7.9.5　CSCIWManagementServer に関する設定をする

CSCIWManagementServer を Cosminexus に組み込みます。

操作手順

1. セキュリティロールを追加する。

CSCIWManagementServer にセキュリティロールを設定する方法を示します。

- セキュリティロールの追加

CSCIWManagementServer には，BPMN エディタの認証をするために，Cosminexus のセキュリティロールとして，csciwdef ロールが必要です。次に示すコマンドを実行すると，セキュリティロールを追加できます。

```
cjaddsec <サーバ名称> -type role -name csciwdef
```

cjaddsec コマンドについては，マニュアル『Cosminexus アプリケーションサーバ リファレンス コマンド編』を参照してください。

- ユーザの追加

  セキュリティロールと関連づけるためのユーザを作成してください。ユーザ名およびパスワードには，「:」（半角コロン）は使用できません。

  次に示すコマンドを実行すると，ユーザを追加できます。なお，すでに登録しているユーザを関連づける場合は，ユーザの追加は不要です。

  ```
  cjaddsec <サーバ名称> -type user -name <ユーザ名> -password <パスワード>
  ```

  cjaddsec コマンドについては，マニュアル『Cosminexus アプリケーションサーバ リファレンス コマンド編』を参照してください。

- セキュリティロールとユーザの関連づけ

  追加したセキュリティロールとユーザの関連づけが必要です。次に示すコマンドを実行すると，セキュリティロールとユーザの関連づけができます。

  ```
  cjmapsec <サーバ名称> -role csciwdef -user <ユーザ名>
  ```

  注

  　ユーザ名には，関連づけるユーザ名を 32 バイト以内の文字列で指定してください。

  cjmapsec コマンドについては，マニュアル『Cosminexus アプリケーションサーバ リファレンス コマンド編』を参照してください。

2. CSCIWManagementServer をインポートする。

CSCIWManagementServer のファイルの格納パスは，次のとおりです。

- Windows の場合

  ```
  <CSCIWインストールフォルダ>¥lib¥csciw.ear
  ```

- UNIX の場合

  ```
  /opt/hitachi/CSCIW/lib/csciw.ear
  ```

次のどちらかの手順を実行して，CSCIWManagementServer をインポートしてください。

＜運用管理ポータルを使用して設定する場合＞

1. 運用管理ポータルにログインする。
2. 運用管理ポータルで［論理サーバのアプリケーション管理］アンカーをクリックする。
3. ツリーペインで次の順にクリックする。

［<操作対象の運用管理ドメイン名>］－［論理 J2EE サーバ］－［J2EE サーバ］－［<操作対象の J2EE サーバ名>］－［アプリケーション］

4. 右ペインの［インポート］タブをクリックし，［J2EE アプリケーションのインポート］画面を表示する。

5. ［インポートディレクトリ］に CSCIWManagementServer のファイルが格納されているパスを指定して［適用］ボタンをクリックする。

6. ［J2EE アプリケーションファイル］で「csciw.ear」を選択して［実行］ボタンをクリックする。

詳細は，マニュアル『Cosminexus 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「J2EE アプリケーションの設定」を参照してください。

＜定義ファイルおよびサーバ管理コマンドを使用して設定する場合＞

1. cjimportapp コマンドを使用して，インポートする。

インポート方法については，マニュアル『Cosminexus 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「J2EE アプリケーションのインポート」の説明を参照してください。

cjimportapp コマンドの指定形式を次に示します。

```
cjimportapp <J2EEサーバ名> -f <csciw.earファイルのパス>
```

cjimportapp コマンドの詳細については，マニュアル『Cosminexus アプリケーションサーバ　リファレンス コマンド編』を参照してください。

3. CSCIWManagementServer が使用するデータソース表示名を変更する。

データベースと接続するための DB Connector のリソースアダプタの表示名に「DB_Connector_for_CSCIW」を設定した場合，この手順は不要です。「DB_Connector_for_CSCIW」以外を設定した場合は，データソース表示名をリソースアダプタと同じ表示名に変更してください。

4. CSCIWManagementServer を開始する。

次のどちらかの手順を実行して，CSCIWManagementServer を開始してください。

＜運用管理ポータルを使用して設定する場合＞

1. 運用管理ポータルにログインする。

2. 運用管理ポータルで［論理サーバのアプリケーション管理］アンカーをクリックする。

3. ツリーペインで次の順にクリックする。
［<操作対象の運用管理ドメイン名>］－［論理 J2EE サーバ］－［J2EE サーバ］－［<操作対象の J2EE サーバ名>］－［アプリケーション］

4. 右ペインの［開始／停止］タブをクリックし，［J2EE アプリケーションの開始／停止］画面を表示する。

5. アプリケーション名「CSCIWManagementServer」の［開始］アンカーをクリックする。

6. 内容を確認して，［はい］ボタンをクリックする。

詳細は，マニュアル『Cosminexus 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「J2EE アプリケーションの設定」を参照してください。

＜定義ファイルおよびサーバ管理コマンドを使用して設定する場合＞

1. cjstartapp コマンドを使用して，開始する。

開始方法については，マニュアル『Cosminexus 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「J2EE アプリケーションの開始」の説明を参照してください。

cjstartapp コマンドの指定形式を次に示します。

```
cjstartapp <J2EEサーバ名> -name CSCIWManagementServer
```

cjstartapp コマンドの詳細については，マニュアル『Cosminexus アプリケーションサーバ　リファレンス コマンド編』を参照してください。

5. コンテキストルートを設定する。

CSCIWManagementServer のコンテキストルートのデフォルトは「csciw」です。

J2EE サーバのフロントエンドにリバースプロキシを配置する場合，コンテキストルート「csciw」のリクエストを転送する設定をしてください。

関連項目

- 7.9.4　DB Connector を設定する
- 9.1.4　CSCIWManagementServer が使用するデータソース表示名を変更する

7.9.6　アプリケーション呼び出しサービスに関する設定をする

アプリケーション呼び出しサービスを Cosminexus に組み込みます。

操作手順

1. アプリケーション呼び出しサービスのファイルをインポートする。

アプリケーション呼び出しサービスのファイルの格納パスを示します。

- Windows の場合

```
<CSCIWのインストールディレクトリ>¥lib¥csciwapsrv.ear
```

- UNIX の場合

```
/opt/hitachi/CSCIW/lib/csciwapsrv.ear
```

次のどちらかの手順を実行して，アプリケーション呼び出しサービスをインポートしてください。

＜運用管理ポータルを使用して設定する場合＞

1. 運用管理ポータルにログインする。
2. 運用管理ポータルで［論理サーバのアプリケーション管理］アンカーをクリックする。
3. ツリーペインで次の順にクリックする。

［<操作対象の運用管理ドメイン名>］－［論理 J2EE サーバ］－［J2EE サーバ］－［<操作対象の J2EE サーバ名>］－［アプリケーション］

4. 右ペインの［インポート］タブをクリックし，［J2EE アプリケーションのインポート］画面を表示する。
5. ［インポートディレクトリ］にアプリケーション呼び出しサービスのファイルが格納されているパスを指定して［適用］ボタンをクリックする。
6. ［J2EE アプリケーションファイル］で「csciwapsrv.ear」を選択して［実行］ボタンをクリックする。

詳細は，マニュアル『Cosminexus 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「J2EE アプリケーションの設定」を参照してください。

＜定義ファイルおよびサーバ管理コマンドを使用して設定する場合＞

1. cjimportapp コマンドを使用して，インポートする。

インポート方法については，マニュアル『Cosminexus 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「J2EE アプリケーションのインポート」の説明を参照してください。

cjimportapp コマンドの指定形式を次に示します。

```
cjimportapp 〈サーバ名称〉 -f 〈csciwapsrv.earファイルのパス〉
```

cjimportapp コマンドの詳細については，マニュアル『Cosminexus アプリケーションサーバ　リファレンス コマンド編』を参照してください。

2. アプリケーション呼び出しサービスが使用するデータソース表示名を変更する。

データベースと接続するための DB Connector のリソースアダプタの表示名に「DB_Connector_for_CSCIW」を設定した場合，この手順は不要です。「DB_Connector_for_CSCIW」以外を設定した場合は，データソース表示名をリソースアダプタと同じ表示名に変更してください。

3. アプリケーション呼び出しサービスを開始する。

次のどちらかの手順を実行して，プリケーション呼び出しサービスを開始してください。

＜運用管理ポータルを使用して設定する場合＞

1. 運用管理ポータルにログインする。
2. 運用管理ポータルで［論理サーバのアプリケーション管理］アンカーをクリックする。
3. ツリーペインで次の順にクリックする。
   ［<操作対象の運用管理ドメイン名>］－［論理 J2EE サーバ］－［J2EE サーバ］－［<操作対象の J2EE サーバ名>］－［アプリケーション］
4. 右ペインの［開始／停止］タブをクリックし，［J2EE アプリケーションの開始／停止］画面を表示する。
5. アプリケーション名「CSCIWBpmnAPService01」の［開始］アンカーをクリックする。
6. 内容を確認して，［はい］ボタンをクリックする。

詳細は，マニュアル『Cosminexus 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「J2EE アプリケーションの設定」を参照してください。

＜定義ファイルおよびサーバ管理コマンドを使用して設定する場合＞

1. cjstartapp コマンドを使用して，開始する。

開始方法については，マニュアル『Cosminexus 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「J2EE アプリケーションの開始」の説明を参照してください。

cjstartapp コマンドの指定形式を次に示します。

```
cjstartapp <サーバ名称> -name CSCIWBpmnAPService01
```

cjstartapp コマンドの詳細については，マニュアル『Cosminexus アプリケーションサーバ　リファレンス コマンド編』を参照してください。

操作結果

アプリケーション呼び出しサービスが Cosminexus に組み込まれます。

メモ

同一の J2EE サーバ上にアプリケーション呼び出しサービスと REST アプリケーションを配備している場合は，REST アプリケーションの開始時の順番をアプリケーション呼び出しサービスの開始時の順番（10）より小さくしてください。

開始時の順番は，アプリケーション属性ファイルのhitachi-application-property/start-orderで設定します。

関連項目

- 7.9.4　DB Connector を設定する
- 9.1.5　アプリケーション呼び出しサービスが使用するデータソース表示名を変更する

7.9.7　REST サービスに関する設定をする

REST サービスを Cosminexus に組み込みます。

操作手順

1. REST サービスのファイルをインポートする。

REST サービスのファイルの格納パスを示します。

- Windows の場合

```
<CSCIWのインストールディレクトリ>¥lib¥csciwws.ear
```

- UNIX の場合

```
/opt/hitachi/CSCIW/lib/csciwws.ear
```

次のどちらかの手順を実行して，REST サービスのファイルをインポートしてください。

＜運用管理ポータルを使用して設定する場合＞

1. 運用管理ポータルにログインする。
2. 運用管理ポータルで［論理サーバのアプリケーション管理］アンカーをクリックする。
3. ツリーペインで次の順にクリックする。

   ［<操作対象の運用管理ドメイン名>］－［論理 J2EE サーバ］－［J2EE サーバ］－［<操作対象の J2EE サーバ名>］－［アプリケーション］
4. 右ペインの［インポート］タブをクリックし，［J2EE アプリケーションのインポート］画面を表示する。
5. ［インポートディレクトリ］に REST サービスのファイルが格納されているパスを指定して［適用］ボタンをクリックする。
6. ［J2EE アプリケーションファイル］で「csciwws.ear」を選択して［実行］ボタンをクリックする。

詳細は，マニュアル『Cosminexus 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「J2EE アプリケーションの設定」を参照してください。

＜定義ファイルおよびサーバ管理コマンドを使用して設定する場合＞

1. cjimportapp コマンドを使用して，インポートする。

インポート方法については，マニュアル『Cosminexus 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「J2EE アプリケーションのインポート」の説明を参照してください。

cjimportapp コマンドの指定形式を次に示します。

```
cjimportapp <サーバ名称> -f <csciwws.earファイルのパス>
```

cjimportapp コマンドの詳細については，マニュアル『Cosminexus アプリケーションサーバ　リファレンス コマンド編』を参照してください。

2. REST サービスが使用するデータソース表示名を変更する。

データベースと接続するための DB Connector のリソースアダプタの表示名に「DB_Connector_for_CSCIW」を設定した場合，この手順は不要です。「DB_Connector_for_CSCIW」以外を設定した場合は，データソース表示名をリソースアダプタと同じ表示名に変更してください。

3. REST サービスを開始する。

次のどちらかの手順を実行して，REST サービスを開始してください。

＜運用管理ポータルを使用して設定する場合＞

1. 運用管理ポータルにログインする。
2. 運用管理ポータルで［論理サーバのアプリケーション管理］アンカーをクリックする。
3. ツリーペインで次の順にクリックする。

［<操作対象の運用管理ドメイン名>］－［論理 J2EE サーバ］－［J2EE サーバ］－［<操作対象の J2EE サーバ名>］－［アプリケーション］

4. 右ペインの［開始／停止］タブをクリックし，［J2EE アプリケーションの開始／停止］画面を表示する。
5. アプリケーション名「CSCIWBpmnRESTService」の［開始］アンカーをクリックする。
6. 内容を確認して，［はい］ボタンをクリックする。

詳細は，マニュアル『Cosminexus 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「J2EE アプリケーションの設定」を参照してください。

＜定義ファイルおよびサーバ管理コマンドを使用して設定する場合＞

1. cjstartapp コマンドを使用して，開始する。

開始方法については，マニュアル『Cosminexus 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「J2EE アプリケーションの開始」の説明を参照してください。

cjstartapp コマンドの指定形式を次に示します。

```
cjstartapp <サーバ名称> -name CSCIWBpmnRESTService
```

cjstartapp コマンドの詳細については，マニュアル『Cosminexus アプリケーションサーバ　リファレンス コマンド編』を参照してください。

4. コンテキストルートを設定する。

J2EE サーバのフロントエンドにリバースプロキシを配置する場合，コンテキストルート「csciwws」のリクエストを転送する設定をしてください。

操作結果

REST サービスが Cosminexus に組み込まれます。

関連項目

- 7.9.4　DB Connector を設定する
- 9.1.6　REST サービスが使用するデータソース表示名を変更する

7.9.8　ビジネスプロセスオペレータに関する設定をする

ビジネスプロセスオペレータをアプリケーションサーバに組み込みます。

操作手順

1. ビジネスプロセスオペレータのファイルをインポートする。

ビジネスプロセスオペレータのファイルの格納パスを示します。

- Windows の場合

```
<CSCIWのインストールディレクトリ>¥lib¥csciwwilist.ear
```

- UNIX の場合

```
/opt/hitachi/CSCIW/lib/csciwwilist.ear
```

次のどちらかの手順を実行して，ビジネスプロセスオペレータのファイルをインポートしてください。

＜運用管理ポータルを使用して設定する場合＞

1. 運用管理ポータルにログインする。
2. 運用管理ポータルで［論理サーバのアプリケーション管理］アンカーをクリックする。
3. ツリーペインで次の順にクリックする。

   ［<操作対象の運用管理ドメイン名>］－［論理 J2EE サーバ］－［J2EE サーバ］－［<操作対象の J2EE サーバ名>］－［アプリケーション］
4. 右ペインの［インポート］タブをクリックし，［J2EE アプリケーションのインポート］画面を表示する。
5. ［インポートディレクトリ］にビジネスプロセスオペレータのファイルが格納されているパスを指定して［適用］ボタンをクリックする。
6. ［J2EE アプリケーションファイル］で「csciwwilist.ear」を選択して［実行］ボタンをクリックする。

詳細は，マニュアル『Cosminexus 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「J2EE アプリケーションの設定」を参照してください。

＜定義ファイルおよびサーバ管理コマンドを使用して設定する場合＞

1. cjimportapp コマンドを使用して，インポートする。

インポート方法については，マニュアル『Cosminexus 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「J2EE アプリケーションのインポート」の説明を参照してください。

cjimportapp コマンドの指定形式を次に示します。

```
cjimportapp <サーバ名称> -f <csciwwilist.earファイルのパス>
```

cjimportapp コマンドの詳細については，マニュアル『Cosminexus アプリケーションサーバ　リファレンス コマンド編』を参照してください。

2. ビジネスプロセスオペレータを開始する。

次のどちらかの手順を実行して，ビジネスプロセスオペレータを開始してください。

＜運用管理ポータルを使用して設定する場合＞

1. 運用管理ポータルにログインする。
2. 運用管理ポータルで［論理サーバのアプリケーション管理］アンカーをクリックする。
3. ツリーペインで次の順にクリックする。

［<操作対象の運用管理ドメイン名>］－［論理 J2EE サーバ］－［J2EE サーバ］－［<操作対象の J2EE サーバ名>］－［アプリケーション］

4. 右ペインの［開始／停止］タブをクリックし，［J2EE アプリケーションの開始／停止］画面を表示する。
5. アプリケーション名「CSCIWWIListApp」の［開始］アンカーをクリックする。
6. 内容を確認して，［はい］ボタンをクリックする。

詳細は，マニュアル『Cosminexus 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「J2EE アプリケーションの設定」を参照してください。

＜定義ファイルおよびサーバ管理コマンドを使用して設定する場合＞

1. cjstartapp コマンドを使用して，開始する。

開始方法については，マニュアル『Cosminexus 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「J2EE アプリケーションの開始」の説明を参照してください。

cjstartapp コマンドの指定形式を次に示します。

```
cjstartapp <サーバ名称> -name CSCIWWIListApp
```

cjstartapp コマンドの詳細については，マニュアル『Cosminexus アプリケーションサーバ　リファレンス コマンド編』を参照してください。

3. コンテキストルートを設定する。

J2EE サーバのフロントエンドにリバースプロキシを配置する場合，コンテキストルート「WorkItemList」のリクエストを転送する設定をしてください。

7.9.9　案件運用操作に関する設定をする

案件運用操作を Cosminexus に組み込みます。

操作手順

1. セキュリティロールを設定する。

- セキュリティロールの追加

  案件運用操作には，Cosminexus のセキュリティロールとして，csciwadmin ロールが必要です。次に示すコマンドを実行すると，セキュリティロールを追加できます。

  ```
  cjaddsec <サーバ名称> -type role -name csciwadmin
  ```

- ユーザの追加

  セキュリティロールと関連づけるためのユーザを作成してください。次に示すコマンドを実行すると，ユーザを追加できます。なお，すでに登録しているユーザを関連づける場合は，ユーザの追加は不要です。

```
cjaddsec 〈サーバ名称〉 -type user -name 〈ユーザ名※〉 -password 〈パスワード※〉
```

注※

ユーザ名およびパスワードには，マルチバイト文字は使用できません。

- セキュリティロールとユーザの関連づけ

  追加したセキュリティロールとユーザの関連づけが必要です。次に示すコマンドを実行すると，セキュリティロールとユーザの関連づけができます。

```
cjmapsec 〈サーバ名称〉 -role csciwadmin -user 〈ユーザ名※〉
```

注※

ユーザ名には，関連づけるユーザ名を 32 バイト以内の文字列で指定してください。なお，ユーザ名には，マルチバイト文字は使用できません。

cjaddsec コマンドおよびcjmapsec コマンドの詳細については，マニュアル『Cosminexus アプリケーションサーバ リファレンス コマンド編』を参照してください。

2. 案件運用操作のファイルをインポートする。

案件運用操作のファイルの格納パスを示します。

- Windows の場合

```
<CSCIWインストールフォルダ>¥lib¥csciwadmin.ear
```

- UNIX の場合

```
/opt/hitachi/CSCIW/lib/csciwadmin.ear
```

次のどちらかの手順を実行して，案件運用操作のファイルをインポートしてください。

＜運用管理ポータルを使用して設定する場合＞

1. 運用管理ポータルにログインする。
2. 運用管理ポータルで［論理サーバのアプリケーション管理］アンカーをクリックする。
3. ツリーペインで次の順にクリックする。

   ［<操作対象の運用管理ドメイン名>］－［論理 J2EE サーバ］－［J2EE サーバ］－［<操作対象の J2EE サーバ名>］－［アプリケーション］
4. 右ペインの［インポート］タブをクリックし，［J2EE アプリケーションのインポート］画面を表示する。
5. ［インポートディレクトリ］に案件運用操作のファイルが格納されているパスを指定して［適用］ボタンをクリックする。
6. ［J2EE アプリケーションファイル］で「csciwadmin.ear」を選択して［実行］ボタンをクリックする。

詳細は，マニュアル『Cosminexus 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「J2EE アプリケーションの設定」を参照してください。

＜定義ファイルおよびサーバ管理コマンドを使用して設定する場合＞

1. cjimportapp コマンドを使用して，インポートする。

インポート方法については，マニュアル『Cosminexus 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「J2EE アプリケーションのインポート」の説明を参照してください。

cjimportapp コマンドの指定形式を次に示します。

```
cjimportapp <サーバ名称> -f <csciwadmin.earファイルのパス>
```

cjimportapp コマンドの詳細については，マニュアル『Cosminexus アプリケーションサーバ　リファレンス コマンド編』を参照してください。

3. 案件運用操作が使用するデータソース表示名を変更する。

データベースと接続するための DB Connector のリソースアダプタの表示名に「DB_Connector_for_CSCIW」を設定した場合，この手順は不要です。「DB_Connector_for_CSCIW」以外を設定した場合は，データソース表示名をリソースアダプタと同じ表示名に変更してください。

4. 案件運用操作を開始する。

次のどちらかの手順を実行して，案件運用操作を開始してください。

＜運用管理ポータルを使用して設定する場合＞

1. 運用管理ポータルにログインする。
2. 運用管理ポータルで［論理サーバのアプリケーション管理］アンカーをクリックする。
3. ツリーペインで次の順にクリックする。

   ［<操作対象の運用管理ドメイン名>］－［論理 J2EE サーバ］－［J2EE サーバ］－［<操作対象の J2EE サーバ名>］－［アプリケーション］
4. 右ペインの［開始／停止］タブをクリックし，［J2EE アプリケーションの開始／停止］画面を表示する。
5. アプリケーション名「CSCIWAdminServlet」の［開始］アンカーをクリックする。
6. 内容を確認して，［はい］ボタンをクリックする。

詳細は，マニュアル『Cosminexus 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「J2EE アプリケーションの設定」を参照してください。

＜定義ファイルおよびサーバ管理コマンドを使用して設定する場合＞

1. cjstartapp コマンドを使用して，開始する。

開始方法については，マニュアル『Cosminexus 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「J2EE アプリケーションの開始」の説明を参照してください。

cjstartapp コマンドの指定形式を次に示します。

```
cjstartapp <サーバ名称> -name CSCIWAdminServlet
```

注

cjstartapp コマンドを使用する場合，-jspc オプションを指定しないで実行してください。

cjstartapp コマンドの詳細については，マニュアル『Cosminexus アプリケーションサーバ　リファレンス コマンド編』を参照してください。

5. コンテキストルートを設定する。

J2EE サーバのフロントエンドにリバースプロキシを配置する場合，コンテキストルート「CSCIWAdminServlet」および「CSCIWAdminServletBPMN」のリクエストを転送する設定をしてください。

操作結果

案件運用操作が Cosminexus に組み込まれます。

関連項目

- 7.9.4　DB Connector を設定する
- 9.1.7　案件運用操作が使用するデータソース表示名を変更する

7.10　Java アプリケーション実行時の設定

Java アプリケーション実行時の設定について，説明します。

CSCIW の Java API を使用した Java アプリケーション実行時の設定については，マニュアル『uCosminexus Service Coordinator Interactive Workflow システム構築・運用ガイド』の「4.5.4 Java アプリケーションの設定」を参照してください。

また，Java アプリケーションが BPMN 連携ライブラリを使用している場合は，クラスパスに次のファイルの設定もする必要があります。

- Windows の場合

```
<CSCIWインストールディレクトリ>¥lib¥csciwbpmn.jar
```

- UNIX の場合

```
/opt/hitachi/CSCIW/lib/csciwbpmn.jar
```

8 ワークフローシステムを削除する

実行環境からのワークフローシステムの削除について説明します。

8.1　ワークフローシステムの削除の流れ

この章での作業は，任意の順序で実施できます。

ただし，次の作業を事前に実施してから，「8.7　CSCIWManagementServer を停止および削除する」の作業をしてください。

- 8.2　業務アプリケーションを停止および削除する
- 8.3　案件運用操作を停止および削除する
- 8.5　REST サービスを停止および削除する
- 8.6　アプリケーション呼び出しサービスを停止および削除する

重要

CSCIW のアンインストール方法については，マニュアル『uCosminexus Service Coordinator Interactive Workflow システム構築・運用ガイド』の次の個所を参照してください。

- 「5.5.2　OS からのアンインストール」
- 「5.6　注意事項」

なお，アンインストール時には環境設定ファイルも削除されます。環境設定ファイルが必要な場合は，アンインストール前に退避してください。

関連項目

- 8.2　業務アプリケーションを停止および削除する
- 8.3　案件運用操作を停止および削除する
- 8.4　ビジネスプロセスオペレータを停止および削除する
- 8.5　REST サービスを停止および削除する
- 8.6　アプリケーション呼び出しサービスを停止および削除する
- 8.7　CSCIWManagementServer を停止および削除する
- 8.8　DB Connector を停止および削除する
- 8.9　J2EE サーバを停止する
- 8.10　環境変数を削除する
- 8.11　コンテナ拡張ライブラリから JAR ファイルを削除する
- 8.12　CSCIW の実行環境を削除する
- 8.13　ワーク管理データベースを削除する（HiRDB の場合）
- 8.14　ワーク管理データベースを削除する（ORACLE の場合）
- 8.15　ワーク管理データベースを削除する（PostgreSQL の場合）

8.2　業務アプリケーションを停止および削除する

次のどちらかの手順を実行して，業務アプリケーションを停止および削除してください。

運用管理ポータルを使用する場合

1. 運用管理ポータルにログインする。
2. 運用管理ポータルで［論理サーバのアプリケーション管理］アンカーをクリックする。
3. ツリーペインで次の順にクリックする。
   ［<操作対象の運用管理ドメイン名>］－［論理 J2EE サーバ］－［J2EE サーバ］－［<操作対象の J2EE サーバ名>］－［アプリケーション］
4. 右ペインの［開始／停止］タブをクリックし，［J2EE アプリケーションの開始／停止］画面を表示する。
5. 停止する業務アプリケーションの［停止］アンカーをクリックする。
6. 内容を確認して，［はい］ボタンをクリックする。
7. 右ペインの［削除］タブをクリックし，［J2EE アプリケーションの削除］画面を表示する。
8. 削除する業務アプリケーションの［削除］アンカーをクリックする。
9. 内容を確認して，［はい］ボタンをクリックする。

サーバ管理コマンドを使用する場合

1. cjstopapp コマンドを使用して，CSCIW を使用している業務アプリケーションを停止する。
   業務アプリケーションの停止方法については，マニュアル『Cosminexus 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「J2EE アプリケーションの停止」の説明を参照してください。
   cjstopapp コマンドの指定形式を次に示します。

   ```
   cjstopapp 〈サーバ名称〉 -name 〈業務アプリケーション名〉
   ```

   cjstopapp コマンドの詳細については，マニュアル『Cosminexus アプリケーションサーバ　リファレンス コマンド編』を参照してください。
2. cjdeleteapp コマンドを使用して，業務アプリケーションを削除する。
   業務アプリケーションの削除方法については，マニュアル『Cosminexus 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「J2EE アプリケーションの削除」の説明を参照してください。
   cjdeleteapp コマンドの指定形式を次に示します。

   ```
   cjdeleteapp 〈サーバ名称〉 -name 〈業務アプリケーション名〉
   ```

   cjdeleteapp コマンドの詳細については，マニュアル『Cosminexus アプリケーションサーバ　リファレンス コマンド編』を参照してください。

8.3　案件運用操作を停止および削除する

次のどちらかの手順を実行して案件運用操作を停止および削除してください。

運用管理ポータルを使用する場合

1. 運用管理ポータルにログインする。
2. 運用管理ポータルで［論理サーバのアプリケーション管理］アンカーをクリックする。
3. ツリーペインで次の順にクリックする。
   ［<操作対象の運用管理ドメイン名>］－［論理 J2EE サーバ］－［J2EE サーバ］－［<操作対象の J2EE サーバ名>］－［アプリケーション］
4. 右ペインの［開始／停止］タブをクリックし，［J2EE アプリケーションの開始／停止］画面を表示する。
5. アプリケーション名「CSCIWAdminServlet」の［停止］アンカーをクリックする。
6. 内容を確認して，［はい］ボタンをクリックする。
7. 右ペインの［削除］タブをクリックし，［J2EE アプリケーションの削除］画面を表示する。
8. アプリケーション名「CSCIWAdminServlet」の［削除］アンカーをクリックする。
9. 内容を確認して，［はい］ボタンをクリックする。

サーバ管理コマンドを使用する場合

1. cjstopapp コマンドを使用して，案件運用操作を停止する。
   cjstopapp コマンドの指定形式を次に示します。

   ```
   cjstopapp 〈サーバ名称〉 -name CSCIWAdminServlet
   ```

   cjstopapp コマンドの詳細については，マニュアル『Cosminexus アプリケーションサーバ　リファレンス コマンド編』を参照してください。
2. cjdeleteapp コマンドを使用して，案件運用操作を削除する。
   cjdeleteapp コマンドの指定形式を次に示します。

   ```
   cjdeleteapp 〈サーバ名称〉 -name CSCIWAdminServlet
   ```

   cjdeleteapp コマンドの詳細については，マニュアル『Cosminexus アプリケーションサーバ　リファレンス コマンド編』を参照してください。

8.4　ビジネスプロセスオペレータを停止および削除する

次のどちらかの手順を実行してビジネスプロセスオペレータを停止および削除してください。

運用管理ポータルを使用する場合

1. 運用管理ポータルにログインする。
2. 運用管理ポータルで［論理サーバのアプリケーション管理］アンカーをクリックする。
3. ツリーペインで次の順にクリックする。
   ［<操作対象の運用管理ドメイン名>］－［論理 J2EE サーバ］－［J2EE サーバ］－［<操作対象の J2EE サーバ名>］－［アプリケーション］
4. 右ペインの［開始／停止］タブをクリックし，［J2EE アプリケーションの開始／停止］画面を表示する。
5. アプリケーション名「CSCIWWIListApp」の［停止］アンカーをクリックする。
6. 内容を確認して，［はい］ボタンをクリックする。
7. 右ペインの［削除］タブをクリックし，［J2EE アプリケーションの削除］画面を表示する。
8. アプリケーション名「CSCIWWIListApp」の［削除］アンカーをクリックする。
9. 内容を確認して，［はい］ボタンをクリックする。

サーバ管理コマンドを使用する場合

1. cjstopapp コマンドを使用して，ビジネスプロセスオペレータを停止する。
   cjstopapp コマンドの指定形式を次に示します。

   ```
   cjstopapp 〈サーバ名称〉 -name CSCIWWIListApp
   ```

   cjstopapp コマンドの詳細については，マニュアル『Cosminexus アプリケーションサーバ　リファレンス コマンド編』を参照してください。
2. cjdeleteapp コマンドを使用して，ビジネスプロセスオペレータを削除する。
   cjdeleteapp コマンドの指定形式を次に示します。

   ```
   cjdeleteapp 〈サーバ名称〉 -name CSCIWWIListApp
   ```

   cjdeleteapp コマンドの詳細については，マニュアル『Cosminexus アプリケーションサーバ　リファレンス コマンド編』を参照してください。

8.5　REST サービスを停止および削除する

次のどちらかの手順を実行して REST サービスを停止および削除してください。

運用管理ポータルを使用する場合

1. 運用管理ポータルにログインする。
2. 運用管理ポータルで［論理サーバのアプリケーション管理］アンカーをクリックする。
3. ツリーペインで次の順にクリックする。
   ［<操作対象の運用管理ドメイン名>］－［論理 J2EE サーバ］－［J2EE サーバ］－［<操作対象の J2EE サーバ名>］－［アプリケーション］
4. 右ペインの［開始／停止］タブをクリックし，［J2EE アプリケーションの開始／停止］画面を表示する。
5. アプリケーション名「CSCIWBpmnRESTService」の［停止］アンカーをクリックする。
6. 内容を確認して，［はい］ボタンをクリックする。
7. 右ペインの［削除］タブをクリックし，［J2EE アプリケーションの削除］画面を表示する。
8. アプリケーション名「CSCIWBpmnRESTService」の［削除］アンカーをクリックする。
9. 内容を確認して，［はい］ボタンをクリックする。

サーバ管理コマンドを使用する場合

1. cjstopapp コマンドを使用して，REST サービスを停止する。
   cjstopapp コマンドの指定形式を次に示します。

   ```
   cjstopapp 〈サーバ名称〉 -name CSCIWBpmnRESTService
   ```

   cjstopapp コマンドの詳細については，マニュアル『Cosminexus アプリケーションサーバ　リファレンス コマンド編』を参照してください。
2. cjdeleteapp コマンドを使用して，REST サービスを削除する。
   cjdeleteapp コマンドの指定形式を次に示します。

   ```
   cjdeleteapp 〈サーバ名称〉 -name CSCIWBpmnRESTService
   ```

   cjdeleteapp コマンドの詳細については，マニュアル『Cosminexus アプリケーションサーバ　リファレンス コマンド編』を参照してください。

8.6　アプリケーション呼び出しサービスを停止および削除する

次のどちらかの手順を実行してアプリケーション呼び出しサービスを停止および削除してください。

運用管理ポータルを使用する場合

1. 運用管理ポータルにログインする。
2. 運用管理ポータルで［論理サーバのアプリケーション管理］アンカーをクリックする。
3. ツリーペインで次の順にクリックする。
   ［<操作対象の運用管理ドメイン名>］－［論理 J2EE サーバ］－［J2EE サーバ］－［<操作対象の J2EE サーバ名>］－［アプリケーション］
4. 右ペインの［開始／停止］タブをクリックし，［J2EE アプリケーションの開始／停止］画面を表示する。
5. アプリケーション名「CSCIWBpmnAPService01」の［停止］アンカーをクリックする。
6. 内容を確認して，［はい］ボタンをクリックする。
7. 右ペインの［削除］タブをクリックし，［J2EE アプリケーションの削除］画面を表示する。
8. アプリケーション名「CSCIWBpmnAPService01」の［削除］アンカーをクリックする。
9. 内容を確認して，［はい］ボタンをクリックする。

サーバ管理コマンドを使用する場合

1. cjstopapp コマンドを使用して，アプリケーション呼び出しサービスを停止する。
   cjstopapp コマンドの指定形式を次に示します。

   ```
   cjstopapp 〈サーバ名称〉 -name CSCIWBpmnAPService01
   ```

   cjstopapp コマンドの詳細については，マニュアル『Cosminexus アプリケーションサーバ　リファレンス コマンド編』を参照してください。
2. cjdeleteapp コマンドを使用して，アプリケーション呼び出しサービスを削除する。
   cjdeleteapp コマンドの指定形式を次に示します。

   ```
   cjdeleteapp 〈サーバ名称〉 -name CSCIWBpmnAPService01
   ```

   cjdeleteapp コマンドの詳細については，マニュアル『Cosminexus アプリケーションサーバ　リファレンス コマンド編』を参照してください。

8.7 CSCIWManagementServer を停止および削除する

次のどちらかの手順を実行して CSCIWManagementServer を停止および削除してください。

前提条件

次に示すアプリケーションが停止および削除されている。

- 業務アプリケーション（削除しないで停止だけでも可）
- 案件運用操作
- REST サービス
- アプリケーション呼び出しサービス

運用管理ポータルを使用する場合

1. 運用管理ポータルにログインする。
2. 運用管理ポータルで［論理サーバのアプリケーション管理］アンカーをクリックする。
3. ツリーペインで次の順にクリックする。
   ［<操作対象の運用管理ドメイン名>］－［論理 J2EE サーバ］－［J2EE サーバ］－［<操作対象の J2EE サーバ名>］－［アプリケーション］
4. 右ペインの［開始／停止］タブをクリックし，［J2EE アプリケーションの開始／停止］画面を表示する。
5. アプリケーション名「CSCIWManagementServer」の［停止］アンカーをクリックする。
6. 内容を確認して，［はい］ボタンをクリックする。
7. 右ペインの［削除］タブをクリックし，［J2EE アプリケーションの削除］画面を表示する。
8. アプリケーション名「CSCIWManagementServer」の［削除］アンカーをクリックする。
9. 内容を確認して，［はい］ボタンをクリックする。

サーバ管理コマンドを使用する場合

1. cjstopapp コマンドを使用して，CSCIWManagementServer を停止する。
   停止方法については，マニュアル『Cosminexus 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「J2EE アプリケーションの停止」の説明を参照してください。
   cjstopapp コマンドの指定形式を次に示します。

   ```
   cjstopapp <サーバ名称> -name CSCIWManagementServer
   ```

   cjstopapp コマンドの詳細については，マニュアル『Cosminexus アプリケーションサーバ　リファレンス コマンド編』を参照してください。
2. cjdeleteapp コマンドを使用して，CSCIWManagementServer を削除する。

削除方法については，マニュアル『Cosminexus 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「J2EE アプリケーションの削除」の説明を参照してください。

cjdeleteapp コマンドの指定形式を次に示します。

```
cjdeleteapp <サーバ名称> -name CSCIWManagementServer
```

cjdeleteapp コマンドの詳細については，マニュアル『Cosminexus アプリケーションサーバ　リファレンス コマンド編』を参照してください。

関連項目

- 8.2　業務アプリケーションを停止および削除する
- 8.3　案件運用操作を停止および削除する
- 8.5　REST サービスを停止および削除する
- 8.6　アプリケーション呼び出しサービスを停止および削除する

8.8　DB Connector を停止および削除する

DB Connector の停止および削除の詳細については，マニュアル『Cosminexus アプリケーションサーバ アプリケーション設定操作ガイド』の「DB Connector の停止」および「DB Connector のアンデプロイ」の説明を参照してください。

8.9　J2EE サーバを停止する

次のどちらかの手順を実行して J2EE サーバを停止してください。

運用管理ポータルを使用する場合

運用管理ポータルで［論理サーバの起動／停止］アンカーをクリックし，対象の J2EE サーバを停止します

詳細は，マニュアル『Cosminexus 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「論理サーバの起動／停止の設定」を参照してください。

サーバ管理コマンドを使用する場合

cjstopsv コマンドを使用して，J2EE サーバを停止します。

cjstopsv コマンドの指定形式を次に示します。

```
cjstopsv 〈J2EEサーバ名〉
```

cjstopsv コマンドの詳細については，マニュアル『Cosminexus アプリケーションサーバ　リファレンス コマンド編』を参照してください。

8.10　環境変数を削除する

「7.9.2　環境変数を取り込む」で設定した環境変数を削除してください。なお，運用管理ポータルで環境変数を設定した場合は，運用管理ポータルで削除します。

8.11　コンテナ拡張ライブラリから JAR ファイルを削除する

コンテナ拡張ライブラリから，「7.9.1　コンテナ拡張ライブラリに JAR ファイルを取り込む」で設定した内容を削除してください。なお，運用管理ポータルでコンテナ拡張ライブラリを設定した場合は，運用管理ポータルで削除します。

8.12　CSCIW の実行環境を削除する

操作手順

1. ciwsetenv コマンドを実行して CSCIW の実行環境を削除する。

   次に示す形式でコマンドを実行すると，CSCIW の実行環境を削除できます。

   ```
   ciwsetenv -sid 〈システムID〉 -del
   ```

   ciwsetenv（環境の構築または削除）コマンドについては，マニュアル『uCosminexus Service Coordinator Interactive Workflow コマンド』を参照してください。

2. 環境変数CSCIW_HOME を削除する。

   なお，マルチインスタンス構成のシステムを構築している場合は，環境変数CSCIW_CONF_DIR も削除してください。

8.13　ワーク管理データベースを削除する（HiRDB の場合）

SQL スクリプトファイルを使用して，ワーク管理データベースのテーブルやインデクスを HiRDB から削除します。

操作手順

1. テキストエディタを使用して，SQL スクリプトファイル中の文字列を編集する。

■テーブルやインデクスの削除時に書き換えが必要な SQL スクリプトファイル

- <CSCIWのインストールディレクトリ>¥sql¥droptable_hirdb.sql
- <CSCIWのインストールディレクトリ>¥sql¥droptableex_hirdb.sql

表 8-1　テーブルやインデクスの削除時に書き換えが必要な SQL スクリプトファイル中の文字列（HiRDB の場合）

項番	書き換えが必要な文字列	書き換える内容
1	<SYSTEMID>	システム ID ワーク管理データベースを作成した際に設定したシステム ID に置換してください。システム ID に指定できる文字は，半角英大文字で始まる半角英大文字および半角数字で，5 文字以内です。

2. HiRDB SQL Executer のpdsql コマンドを使用して，編集した SQL スクリプトファイルを実行する。

HiRDB SQL Executer の使用方法については，HiRDB SQL Executer 付属のヘルプを参照してください。

pdsql コマンドの指定形式を示します。

```
pdsql -u 接続認可識別子/パスワード
      -h HiRDBサーバのホスト名またはIPアドレス
      -n HiRDBサーバのポート番号
       < 編集したSQLスクリプトファイルのパス
```

注

接続認可識別子には，CONNECT 権限のある認可識別子を指定してください。

8.14　ワーク管理データベースを削除する（ORACLE の場合）

SQL スクリプトファイルを使用して，ワーク管理データベースのテーブルやインデクスを ORACLE データベースから削除します。

操作手順

1. テキストエディタを使用して，SQL スクリプトファイル中の文字列を編集する。

■テーブルやインデクスの削除時に書き換えが必要な SQL スクリプトファイル

- <CSCIWのインストールディレクトリ>¥sql¥droptable_oracle.sql
- <CSCIWのインストールディレクトリ>¥sql¥droptableex_oracle.sql

表 8-2　テーブルやインデクスの削除時に書き換えが必要な SQL スクリプトファイル中の文字列（ORACLE の場合）

項番	書き換えが必要な文字列	書き換える内容
1	<SYSTEMID>	システム ID ワーク管理データベースを作成した際に設定したシステム ID に置換してください。システム ID に指定できる文字は，半角英大文字で始まる半角英大文字および半角数字で，5 文字以内です。

2. SQL*Plus を使用して，編集した SQL スクリプトファイルを実行する。

SQL*Plus の使用方法については，ORACLE のマニュアルを参照してください。

SQL*Plus の指定形式を示します。

```
sqlplus　接続ユーザ名/パスワード@Oracle Net接続識別子　@編集したSQLスクリプトファイルの絶対パス
```

注

次の権限を付与したユーザで接続してください。

- CREATE SESSION システム権限
- CREATE TABLE システム権限
- CREATE VIEW システム権限

8.15　ワーク管理データベースを削除する（PostgreSQL の場合）

SQL スクリプトファイルを使用して，ワーク管理データベースのテーブルやインデクスを PostgreSQL データベースから削除します。

操作手順

1. テキストエディタを使用して，SQL スクリプトファイル中の文字列を編集する。

   ■テーブルやインデクスの削除時に書き換えが必要な SQL スクリプトファイル

   - <CSCIWのインストールディレクトリ>/sql/droptable_postgresql.sql
   - <CSCIWのインストールディレクトリ>/sql/droptableex_postgresql.sql

   表 8-3　テーブルやインデクスの削除時に書き換えが必要な SQL スクリプトファイル中の文字列（PostgreSQL の場合）

項番	書き換えが必要な文字列	書き換える内容
1	<SYSTEMID>	システム ID ワーク管理データベースを作成した際に設定したシステム ID に置換してください。
2	<SCHEMANAME>	CSCIW の管理用のスキーマ名 ワーク管理データベースのテーブルやインデクスを作成した際に設定したスキーマ名に置換してください。

2. psql コマンドを使用して，編集した SQL スクリプトファイルを実行する。

   psql コマンドの使用方法については，PostgreSQL のマニュアルを参照してください。

   psql コマンドの指定形式を示します。

```
psql -U 接続ユーザ名 -d 接続データベース名 -f 編集したSQLスクリプトファイルの絶対パス
```

8.16　データベースへのアクセス権限を削除する（HiRDB の場合）

SQL スクリプトファイルを使用して，ワーク管理データベースのテーブルやインデクスへ付与したアクセス権限を HiRDB から削除します。ワーク管理データベースのテーブルを削除済みの場合は，アクセス権限の削除は不要です。

操作手順

1. テキストエディタを使用して，SQL スクリプトファイル中の文字列を編集する。

■アクセス権限の削除時に書き換えが必要な SQL スクリプトファイル

- <CSCIWのインストールディレクトリ>¥sql¥revoketable_hirdb.sql
- <CSCIWのインストールディレクトリ>¥sql¥revoketableex_hirdb.sql

表 8-4　アクセス権限の削除時に書き換えが必要な SQL スクリプトファイル中の文字列（HiRDB の場合）

項番	書き換えが必要な文字列	書き換える内容
1	<SCHEMANAME>	CSCIW の管理用の認可識別子 ワーク管理データベースのテーブルやインデクスを作成した認可識別子に置換してください。
2	<SYSTEMID>	システム ID ワーク管理データベースを作成した際に設定したシステム ID に置換してください。
3	<USERNAME>	業務アプリケーション用の認可識別子 業務アプリケーションが接続するための認可識別子に置換してください。

2. HiRDB SQL Executer のpdsql コマンドを使用して，編集した SQL スクリプトファイルを実行する。

SQL スクリプトファイルの実行方法は，ワーク管理データベースを削除するときと同じです。

関連項目

- 8.13　ワーク管理データベースを削除する（HiRDB の場合）

8.17　データベースへのアクセス権限を削除する（ORACLE の場合）

SQL スクリプトファイルを使用して，ワーク管理データベースのテーブルやインデクスへ付与したアクセス権限を ORACLE データベースから削除します。ワーク管理データベースのテーブルを削除済みの場合は，アクセス権限の削除は不要です。

操作手順

1. テキストエディタを使用して，SQL スクリプトファイル中の文字列を編集する。

■アクセス権限の削除時に書き換えが必要な SQL スクリプトファイル

- <CSCIWのインストールディレクトリ>¥sql¥revoketable_oracle.sql
- <CSCIWのインストールディレクトリ>¥sql¥revoketableex_oracle.sql

表 8-5　アクセス権限の削除時に書き換えが必要な SQL スクリプトファイル中の文字列（ORACLE の場合）

項番	書き換えが必要な文字列	書き換える内容
1	<SCHEMANAME>	CSCIW の管理用のユーザ名 ワーク管理データベースのテーブルやインデクスを作成したユーザ名に置換してください。
2	<SYSTEMID>	システム ID ワーク管理データベースを作成した際に設定したシステム ID に置換してください。
3	<USERNAME>	業務アプリケーション用のユーザ名 業務アプリケーションが接続するためのユーザ名に置換してください。

2. SQL*Plus を使用して，編集した SQL スクリプトファイルを実行する。

SQL スクリプトファイルの実行方法は，ワーク管理データベースを削除するときと同じです。

関連項目

- 8.14　ワーク管理データベースを削除する（ORACLE の場合）

8.18　データベースへのアクセス権限を削除する（PostgreSQL の場合）

SQL スクリプトファイルを使用して，ワーク管理データベースのテーブルやインデクスへ付与したアクセス権限を PostgreSQL データベースから削除します。ワーク管理データベースのテーブルを削除済みの場合は，アクセス権限の削除は不要です。

操作手順

1. テキストエディタを使用して，SQL スクリプトファイル中の文字列を編集する。

   ■アクセス権限の削除時に書き換えが必要な SQL スクリプトファイル

   - <CSCIWのインストールディレクトリ>/sql/revoketable_postgresql.sql
   - <CSCIWのインストールディレクトリ>/sql/revoketableex_postgresql.sql

   表 8-6　アクセス権限の削除時に書き換えが必要な SQL スクリプトファイル中の文字列（PostgreSQL の場合）

項番	書き換えが必要な文字列	書き換える内容
1	<SYSTEMID>	システム ID ワーク管理データベースを作成した際に設定したシステム ID に置換してください。
2	<SCHEMANAME>	CSCIW の管理用のスキーマ名 ワーク管理データベースのテーブルやインデクスを作成した際に設定したスキーマ名に置換してください。
3	<USERNAME>	業務アプリケーション用のユーザ名 業務アプリケーションが接続するためのユーザ名に置換してください。

2. psql コマンドを使用して，編集した SQL スクリプトファイルを実行する。

   SQL スクリプトファイルの実行方法は，ワーク管理データベースを削除するときと同じです。

関連項目

- 8.15　ワーク管理データベースを削除する（PostgreSQL の場合）

9 ワークフローシステムを変更する

ワークフローシステムを変更するために，CSCIW や BPMN 連携機能の設定情報を変更します。

9.1 CSCIW のシステムを変更する

9.1.1 実行環境マシンの IP アドレスの変更

実行環境マシンの IP アドレスを変更する方法については，マニュアル『uCosminexus Service Coordinator Interactive Workflow システム構築・運用ガイド』の「6.1.1　実行環境マシンの IP アドレスの変更」を参照してください。

9.1.2 接続先データベースの変更

接続先データベースを変更する方法については，マニュアル『uCosminexus Service Coordinator Interactive Workflow システム構築・運用ガイド』の「6.1.2　接続先 DBMS の変更」を参照してください。

また，あわせて J2EE サーバの DB Connector の接続先データベースも変更する必要があります。

9.1.3 実行環境の設定情報の変更

実行環境の設定情報を変更する方法については，マニュアル『uCosminexus Service Coordinator Interactive Workflow システム構築・運用ガイド』の「6.2　実行環境の設定情報の変更」を参照してください。

9.1.4 CSCIWManagementServer が使用するデータソース表示名を変更する

CSCIWManagementServer が使用するデータソース表示名には，データベースと接続するための DB Connector のプロパティに設定した DB Connector のリソースアダプタの表示名と同じ名前を指定します。

操作手順

1. cjimportapp コマンドを使用して，CSCIWManagementServer をインポートする。
   CSCIWManagementServer を組み込みます。インポート済みの場合は，この手順は不要です。
2. cjgetappprop コマンドを使用して，XML ファイル（属性ファイル）を取得する。
   cjgetappprop コマンドの実行例

```
cjgetappprop <サーバ名称> -name CSCIWManagementServer -type all -c <属性ファイルパス>
```

cjgetappprop コマンドの詳細については，マニュアル『Cosminexus アプリケーションサーバ　リファレンス コマンド編』を参照してください。

3. 取得した XML ファイル（属性ファイル）を編集する。

取得した XML ファイル（属性ファイル）をテキストエディタで開き，resource-ref 要素の linked-to 要素を「DB_Connector_for_CSCIW」から DB Connector のリソースアダプタの表示名と同じ名前に変更します。

```
<resource-ref>
  <description xml:lang="en"></description>
  <res-ref-name>jdbc/CSCIWDataSource</res-ref-name>
  <res-type>javax.sql.DataSource</res-type>
  <res-auth>Container</res-auth>
  <res-sharing-scope>Shareable</res-sharing-scope>
  <linked-to>DB_Connector_for_CSCIW</linked-to>
</resource-ref>
```

4. cjsetappprop コマンドを使用して，編集した XML ファイル（属性ファイル）の情報をプロパティに設定する。

cjsetappprop コマンドの実行例

```
cjsetappprop <サーバ名称> -name CSCIWManagementServer -type all -c <属性ファイルパス>
```

cjsetappprop コマンドの詳細については，マニュアル『Cosminexus アプリケーションサーバ　リファレンス コマンド編』を参照してください。

5. cjstartapp コマンドを使用して，CSCIWManagementServer を開始する。

関連項目

- 7.9.4　DB Connector を設定する

9.1.5　アプリケーション呼び出しサービスが使用するデータソース表示名を変更する

アプリケーション呼び出しサービスが使用するデータソース表示名には，データベースと接続するための DB Connector のプロパティに設定したリソースアダプタの表示名と同じ名前を指定します。

操作手順

1. cjimportapp コマンドを使用して，アプリケーション呼び出しサービスをインポートする。

アプリケーション呼び出しサービスを組み込みます。インポート済みの場合，この手順は不要です。

2. cjgetappprop コマンドを使用して，XML ファイル（属性ファイル）を取得する。

cjgetappprop コマンドの実行例

```
cjgetappprop <サーバ名称> -name CSCIWBpmnAPService01 -type all -c <属性ファイルパス>
```

cjgetappprop コマンドの詳細については，マニュアル『Cosminexus アプリケーションサーバ　リファレンス コマンド編』を参照してください。

3. 取得した XML ファイル（属性ファイル）を編集する。

取得した XML ファイル（属性ファイル）をテキストエディタで開き，resource-ref 要素の linked-to 要素を「DB_Connector_for_CSCIW」から DB Connector のリソースアダプタの表示名と同じ名前に変更します。

```
 <resource-ref>
    <description xml:lang="en"></description>
    <res-ref-name>jdbc/CSCIWApServiceDataSource</res-ref-name>
    <res-type>javax.sql.DataSource</res-type>
    <res-auth>Container</res-auth>
    <res-sharing-scope>Shareable</res-sharing-scope>
    <injection-target>
        <injection-target-class>jp.co.Hitachi.soft.csciw.bpmn.apservice.APServiceTimer</i
njection-target-class>
        <injection-target-name>mDataSource</injection-target-name>
    </injection-target>
    <linked-to>DB_Connector_for_CSCIW</linked-to>
  </resource-ref>
```

4. cjsetappprop コマンドを使用して，編集した XML ファイル（属性ファイル）の情報をプロパティに設定する。

cjsetappprop コマンドの実行例

```
cjsetappprop <サーバ名称> -name CSCIWBpmnAPService01 -type all -c <属性ファイルパス>
```

cjsetappprop コマンドの詳細については，マニュアル『Cosminexus アプリケーションサーバ　リファレンス コマンド編』を参照してください。

5. cjstartapp コマンドを使用して，アプリケーション呼び出しサービスを開始する。

関連項目

- 7.9.4　DB Connector を設定する

9.1.6　REST サービスが使用するデータソース表示名を変更する

REST サービスが使用するデータソース表示名には，データベースと接続するための DB Connector のプロパティに設定したリソースアダプタの表示名と同じ名前を指定します。

操作手順

1. cjimportapp コマンドを使用して，REST サービスをインポートする。

   REST サービスを組み込みます。インポート済みの場合，この手順は不要です。

2. cjgetappprop コマンドを使用して，XML ファイル（属性ファイル）を取得する。

   cjgetappprop コマンドの実行例

   ```
   cjgetappprop <サーバ名称> -name CSCIWBpmnRESTService -type all -c <属性ファイルパス>
   ```

   cjgetappprop コマンドの詳細については，マニュアル『Cosminexus アプリケーションサーバ　リファレンス コマンド編』を参照してください。

3. 取得した XML ファイル（属性ファイル）を編集する。

   取得した XML ファイル（属性ファイル）をテキストエディタで開き，resource-ref 要素の linked-to 要素を「DB_Connector_for_CSCIW」から DB Connector のリソースアダプタの表示名と同じ名前に変更します。

   ```
   <resource-ref>
     <description xml:lang="en"></description>
     <res-ref-name>jdbc/CSCIWRESTServiceDataSource</res-ref-name>
     <res-type>javax.sql.DataSource</res-type>
     <res-auth>Container</res-auth>
     <res-sharing-scope>Shareable</res-sharing-scope>
     <injection-target>
       <injection-target-class>jp.co.Hitachi.soft.csciw.bpmn.rest.application.RESTServiceBe
   an</injection-target-class>
       <injection-target-name>mDataSource</injection-target-name>
     </injection-target>
     <linked-to>DB_Connector_for_CSCIW</linked-to>
   </resource-ref>
   ```

4. cjsetappprop コマンドを使用して，編集した XML ファイル（属性ファイル）の情報をプロパティに設定する。

   cjsetappprop コマンドの実行例

   ```
   cjsetappprop <サーバ名称> -name CSCIWBpmnRESTService -type all -c <属性ファイルパス>
   ```

   cjsetappprop コマンドの詳細については，マニュアル『Cosminexus アプリケーションサーバ　リファレンス コマンド編』を参照してください。

5. cjstartapp コマンドを使用して，REST 呼び出しサービスを開始する。

関連項目

- 7.9.4　DB Connector を設定する

9.1.7　案件運用操作が使用するデータソース表示名を変更する

案件運用操作が使用するデータソース表示名には，データベースと接続するための DB Connector のプロパティに設定した DB Connector のリソースアダプタの表示名と同じ名前を指定します。

操作手順

1. cjimportapp コマンドを使用して，案件運用操作をインポートする。
   案件運用操作を組み込みます。インポート済みの場合は，この手順は不要です。

2. cjgetappprop コマンドを使用して，XML ファイル（属性ファイル）を取得する。
   cjgetappprop コマンドの実行例

```
cjgetappprop <サーバ名称> -name CSCIWAdminServlet -type all -c <属性ファイルパス>
```

   cjgetappprop コマンドの詳細については，マニュアル『Cosminexus アプリケーションサーバ　リファレンス コマンド編』を参照してください。

3. 取得した XML ファイル（属性ファイル）を編集する。
   取得した XML ファイル（属性ファイル）をテキストエディタで開き，2 か所の resource-ref 要素の linked-to 要素を「DB_Connector_for_CSCIW」から DB Connector のリソースアダプタの表示名と同じ名前に変更します。

```
 <resource-ref>
   <description xml:lang="en"></description>
   <res-ref-name>jdbc/CSCAdminIWDataSource</res-ref-name>
   <res-type>javax.sql.DataSource</res-type>
   <res-auth>Container</res-auth>
   <res-sharing-scope>Shareable</res-sharing-scope>
   <linked-to>DB_Connector_for_CSCIW</linked-to>
 </resource-ref>
…中略…
 <resource-ref>
   <description xml:lang="en"></description>
   <res-ref-name>jdbc/CSCAdminIWDataSource</res-ref-name>
   <res-type>javax.sql.DataSource</res-type>
   <res-auth>Container</res-auth>
   <res-sharing-scope>Shareable</res-sharing-scope>
   <linked-to>DB_Connector_for_CSCIW</linked-to>
 </resource-ref>
```

4. cjsetappprop コマンドを使用して，編集した XML ファイル（属性ファイル）の情報をプロパティに設定する。
   cjsetappprop コマンドの実行例

```
cjsetappprop <サーバ名称> -name CSCIWAdminServlet -type all -c <属性ファイルパス>
```

   cjsetappprop コマンドの詳細については，マニュアル『Cosminexus アプリケーションサーバ　リファレンス コマンド編』を参照してください。

5. cjstartapp コマンドを使用して，案件運用操作を開始する。

関連項目

- 7.9.4　DB Connector を設定する

9.1.8　ビジネスプロセス定義の変更時の注意事項

ビジネスプロセス定義の変更時の注意事項については，マニュアル『uCosminexus Service Coordinator Interactive Workflow システム構築・運用ガイド』の「6.3　ビジネスプロセス定義および振り分けルール定義の変更」を参照してください。

9.2　ファイルに格納した設定情報を変更する

ファイルに格納した BPMN 連携機能に関する設定情報を変更します。

操作手順

1. ファイルの内容を変更する。

   目的に応じて，必要なファイルだけ内容を変更してください。

   - 共通設定ファイル
   - アプリケーション呼び出し情報ファイル
   - REST アプリケーション呼び出し用ヘッダファイル
   - REST アプリケーション呼び出しスキーマ変換用スタイルシート
   - コールアクティビティ情報ファイル

2. 変更したファイルに応じて，必要な作業を実施する。

   - 共通設定ファイルの内容を変更した場合

     変更方法はパラメタのカテゴリごとに異なります。各パラメタがどのカテゴリに属するかについては，「15.2.5　共通設定ファイルに指定する内容」を参照してください。

     カテゴリごとに必要な作業を次に示します。

     - パラメタのカテゴリが「BPMN 連携機能共通」のとき

       J2EE サーバを再起動してください。再起動後に次のどれかの操作をした際に反映されます。

       ・BPMN 連携ライブラリの初期化（CIWBPMNLibAdmin クラスの initializeCIWBPMNLib メソッドの実行）

       ・REST サービスの開始

       ・アプリケーション呼び出しサービスの開始

     - パラメタのカテゴリが「REST API」のとき

       REST サービスを再開始してください。

     - パラメタのカテゴリが「アプリケーション呼び出しサービス」のとき

       アプリケーション呼び出しサービスを再開始してください。

       また，同一の J2EE サーバに複数のアプリケーション呼び出しサービスを配備している場合，すべてのアプリケーション呼び出しサービスを再開始してください。

   - アプリケーション呼び出し情報ファイル，REST アプリケーション呼び出し用ヘッダファイル，または REST アプリケーション呼び出しスキーマ変換用スタイルシートの内容を変更した場合

     アプリケーション呼び出しサービスを再開始してください。

     同一の J2EE サーバに複数のアプリケーション呼び出しサービスを配備している場合，すべてのアプリケーション呼び出しサービスを再開始してください。

   - コールアクティビティ情報ファイルの内容を変更した場合

次のどれかを実行してください。

- アプリケーション呼び出しサービスの再開始
- REST サービスの再開始
- BPMN 連携ライブラリの終了処理（CIWBPMNLibAdmin クラスの finalizeCIWBPMNLib メソッドの実行）および初期化（CIWBPMNLibAdmin クラスの initializeCIWBPMNLib メソッドの実行）

ヒント

各サービスを停止する順序については，「10.2　業務を停止する」を参照してください。

また，各サービスを開始する順序については，「10.1　業務を開始する」を参照してください。

操作結果

変更したファイルの内容が適用されます。

9.3　アプリケーション呼び出しに関する設定情報を変更する

ref 識別子ごとのアプリケーション呼び出し制御情報は，ciwmngap コマンドで変更できます。また，アプリケーション呼び出しグループは，ciwmngapgrp コマンドで変更できます。

各コマンドの詳細については，マニュアル『uCosminexus Service Coordinator Interactive Workflow コマンド』を参照してください。

9.4 WorkManager の最大スレッド数を変更する

WorkManager の最大スレッド数を変更します。

前提条件

重要

WorkManager の最大スレッド数は，スレッド数の見積もりおよびデータベースコネクション数の見積もりに影響があります。

そのため，設定値を変更する場合は，スレッド数やデータベースコネクション数の再見積もりが必要です。

操作手順

1. J2EE サーバ用ユーザプロパティファイル（usrconf.properties）に，WorkManager の最大スレッド数を指定する。

   WorkManager の最大スレッド数の指定例を次に示します。

```
ejbserver.commonj.WorkManager.non_daemon_work_threads=20
```

2. 変更対象の J2EE サーバを再起動する。

操作結果

WorkManager の最大スレッド数に，設定した値が適用されます。

10 ワークフローシステムを運用する

ワークフローシステムの運用について説明します。

10.1　業務を開始する

操作手順

1. CSCIWManagementServer を開始する。
   アプリケーション名：CSCIWManagementServer
2. アプリケーション呼び出しサービスから呼び出される REST アプリケーションを開始する。
3. アプリケーション呼び出しサービスを開始する。
   アプリケーション名：CSCIWBpmnAPService01
4. REST サービスを開始する。
   アプリケーション名：CSCIWBpmnRESTService
5. REST API または Java API を実行する業務アプリケーションを開始する。

運用管理ポータルを使用する場合

1. 運用管理ポータルにログインする。
2. 運用管理ポータルで［論理サーバのアプリケーション管理］アンカーをクリックする。
3. ツリーペインで次の順にクリックする。
   ［論理 J2EE サーバ］－［J2EE サーバ］－［<サーバ名称>］－［アプリケーション］
4. 右ペインの［開始／停止］タブをクリックし，［J2EE アプリケーションの開始／停止］画面を表示する。
5. 対象アプリケーションの［開始］アンカーをクリックする。
6. 内容を確認して，［はい］ボタンをクリックする。

サーバ管理コマンドを使用する場合

1. cjstartapp コマンドを使用して，各アプリケーションを開始する。
   cjstartapp コマンドの指定形式を次に示します。

```
cjstopapp 〈サーバ名称〉 -name 〈アプリケーション名〉
```

   cjstopapp コマンドの詳細については，マニュアル『Cosminexus アプリケーションサーバ　リファレンス コマンド編』を参照してください。

> **メモ**
>
> 同一の J2EE サーバ上にアプリケーション呼び出しサービスと REST アプリケーションを配備している場合は，REST アプリケーションの開始時の順番をアプリケーション呼び出しサービスの開始時の順番（10）より小さくしてください。
>
> 開始時の順番は，アプリケーション属性ファイルのhitachi-application-property/start-orderで設定します。

10.2　業務を停止する

操作手順

1. REST API または Java API を実行する業務アプリケーションを停止する。
2. REST サービスを停止する。
   アプリケーション名：CSCIWBpmnRESTService
3. アプリケーション呼び出しサービスを停止する。
   アプリケーション名：CSCIWBpmnAPService01
4. アプリケーション呼び出しサービスから呼び出される REST アプリケーションを停止する。
5. CSCIWManagementServer を停止する。
   アプリケーション名：CSCIWManagementServer

運用管理ポータルを使用する場合

1. 運用管理ポータルにログインする。
2. 運用管理ポータルで［論理サーバのアプリケーション管理］アンカーをクリックする。
3. ツリーペインで次の順にクリックする。
   ［論理 J2EE サーバ］－［J2EE サーバ］－［<サーバ名称>］－［アプリケーション］
4. 右ペインの［開始／停止］タブをクリックし，［J2EE アプリケーションの開始／停止］画面を表示する。
5. 対象アプリケーションの［停止］アンカーをクリックする。
6. 内容を確認して，［はい］ボタンをクリックする。

サーバ管理コマンドを使用する場合

1. cjstopapp コマンドを使用して，各アプリケーションを停止する。
   cjstopapp コマンドの指定形式を次に示します。

```
cjstopapp 〈サーバ名称〉 -name 〈アプリケーション名〉
```

   cjstopapp コマンドの詳細については，マニュアル『Cosminexus アプリケーションサーバ　リファレンス コマンド編』を参照してください。

重要

強制停止

CSCIW が提供する J2EE アプリケーション（CSCIWManagementServer，アプリケーション呼び出しサービス，REST サービス，およびビジネスプロセスオペレータ）の処理が無応答になるなどの理由で，cjstopapp コマンドで停止できない場合があります。

コマンドで停止できない場合は，J2EE サーバを強制停止する必要があります。J2EE サーバを強制停止した場合の留意点については，マニュアル『uCosminexus Service Coordinator Interactive Workflow システム構築・運用ガイド』の「7.9.3　業務アプリケーションが異常終了した場合や強制終了した（終了処理をしなかった）場合の影響について」を参照してください。

10.3　案件を参照，操作する

ビジネスプロセスオペレータを使用して，案件を参照，操作します。

10.3.1　ビジネスプロセスオペレータを実行する

ビジネスプロセスオペレータを実行するには，次の URL にアクセスします。

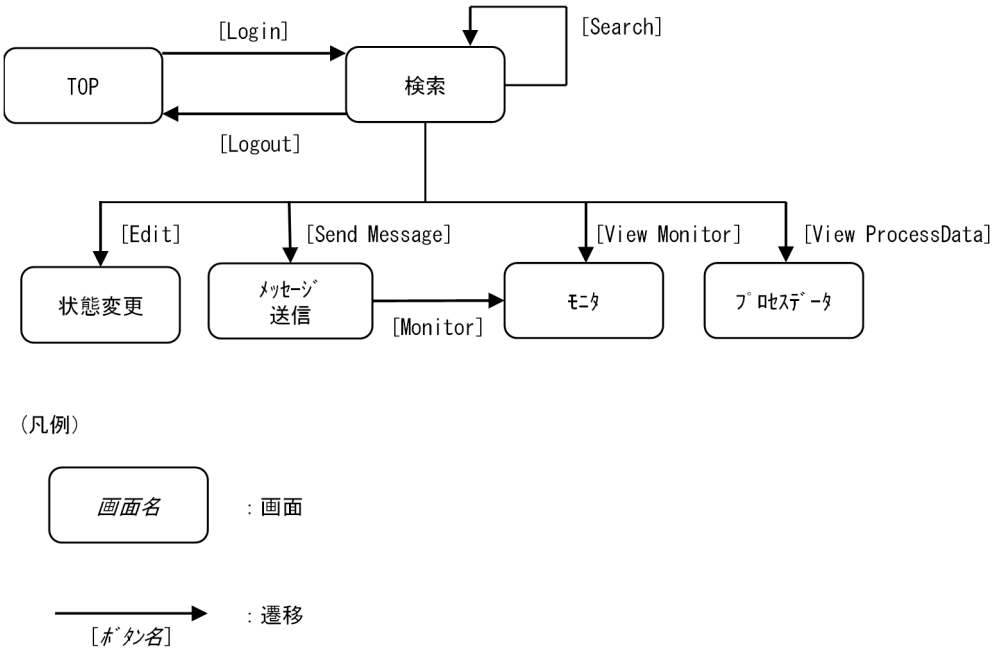
```
http://<ホスト名>:<ポート番号>/WorkItemList/
```

上記の URL にアクセスすると，TOP 画面に遷移します。

10.3.2　ビジネスプロセスオペレータの画面構成

ビジネスプロセスオペレータによって表示される主な画面の種類と，画面の遷移を次の図で示します。

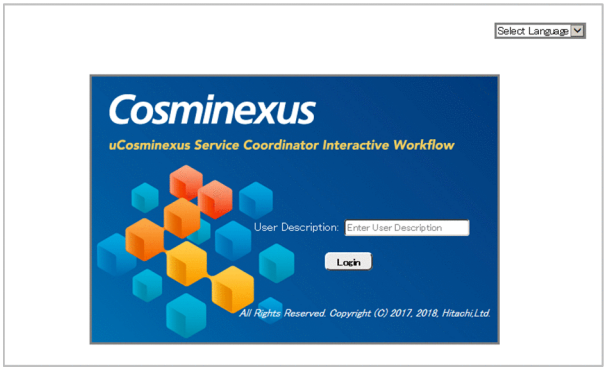
図 10-1　ビジネスプロセスオペレータの画面の遷移



(1)　TOP 画面

TOP 画面の［User Description］テキストボックスに任意の文字列を入力し，［Login］ボタンをクリックすると，検索画面へ遷移します。［User Description］テキストボックスに入力した値は，ビジネスプロセスオペレータが実行する REST API の`UserDescription`パラメタに設定され，トレースファイルに出力されます。

図 10-2　TOP 画面



(2)　検索画面

検索画面上部の検索条件フォームに，検索条件として「表 10-1　検索条件フォームに入力する検索条件」に示す項目を入力します。

図 10-3　検索画面（検索条件フォーム）

表 10-1　検索条件フォームに入力する検索条件

項番	フォーム名	説明
1	案件 ID（作業 ID）※1	案件 ID や作業 ID を指定します。作業 ID の指定は任意です。
2	案件名※1，※2	案件名を指定します。また，検索方法としてドロップダウンリストから「前方一致」または「完全一致」を選択します。
3	作業者※1	作業者 ID を指定します。また，検索方法としてドロップダウンリストから「前方一致」または「完全一致」を選択します。
4	状態※3	状態を選択します。
5	定義名※3	作業定義名を指定します。検索方法は，「部分一致」だけです。
6	発生日時※3	案件の発生日の範囲を指定します。 日付をカレンダーから指定します。次に［時］：［分］：［秒］を指定します。
7	オフセット※3	オフセットを指定します。

注※1

検索条件として，どれかを選択します。

注※2

案件名を指定した検索では，案件名に合致する案件を絞り込んだあとに作業を検索します。検索条件に合致した案件の数が 100 件より多い場合，エラーが表示されます。100 件以下になる条件（案件名）を設定してください。

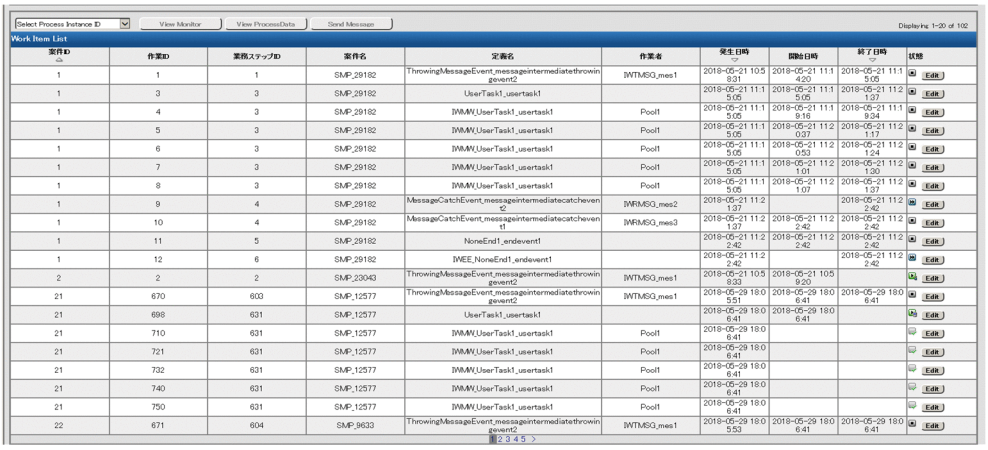
注※3

※1 で指定した検索条件に，※3 の検索条件を AND 条件（論理積）で追加できます。

[Search] ボタンをクリックすると，画面の下部に作業の検索結果が表示されます。作業の検索結果の最大取得数は 100 件で，20 件ずつ表示されます。作業の検索結果の最大取得数は，共通設定ファイルの RestServiceResponseMaxCount パラメタで変更できます。RestServiceResponseMaxCount パラメタについては，「15.2.5　共通設定ファイルに指定する内容」の「(5)　RestServiceResponseMaxCount」を参照してください。

検索結果画面の例を次に示します。

図 10-4　検索結果画面の例



検索結果画面の［Select Process Instance ID］ドロップダウンリストから案件 ID を 1 つ選択します。［View Monitor］ボタン，［View ProcessData］ボタン，または［Send Message］ボタンをクリックすると，選択した案件 ID に対するモニタ画面，プロセスデータ画面，メッセージ送信画面へ遷移します。また，作業の一覧から，状態を変更したい作業の［Edit］ボタンをクリックすると，状態変更画面へ遷移します。

(3)　状態変更画面

状態を変更するには，状態変更画面で［Edit］ボタンをクリックして，［変更後の状態］や［プロセスデータ］を編集し，［Update］ボタンをクリックします。プロセスデータを 4 個以上指定する場合は，［Add］ボタンをクリックすると行を追加できます。

リスト型プロセスデータを登録する場合の指定方法については，「(7)　リスト型プロセスデータの指定形式」を参照してください。

なお，登録済みのプロセスデータの「キー」のテキストボックスは，［Edit］ボタンをクリックしても編集できません。

図 10-5　状態変更画面

(4)　メッセージ送信画面

メッセージを送信するには，メッセージ送信画面で［Edit］ボタンをクリックして，［ref 識別子］や［プロセスデータ］を編集し，［Send Message］ボタンをクリックします。ref 識別子は，［Monitor］ボタンをクリックすると表示されるモニタ画面で確認してください。

リスト型プロセスデータを登録する場合の指定方法については，「(7)　リスト型プロセスデータの指定形式」を参照してください。

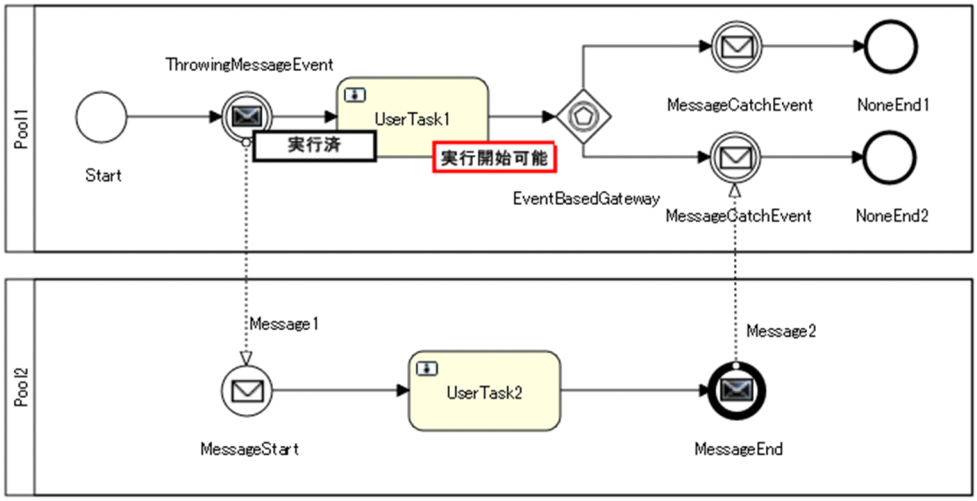
なお，登録済みのプロセスデータの「キー」のテキストボックスは，［Edit］ボタンをクリックしても編集できません。

図 10-6　メッセージ送信画面

(5)　モニタ画面

モニタ画面では，案件の進捗状況を確認できます。モニタ画面の詳細については，「付録 D　ビジネスプロセスモニタとは」を参照してください。

図 10-7　モニタ画面

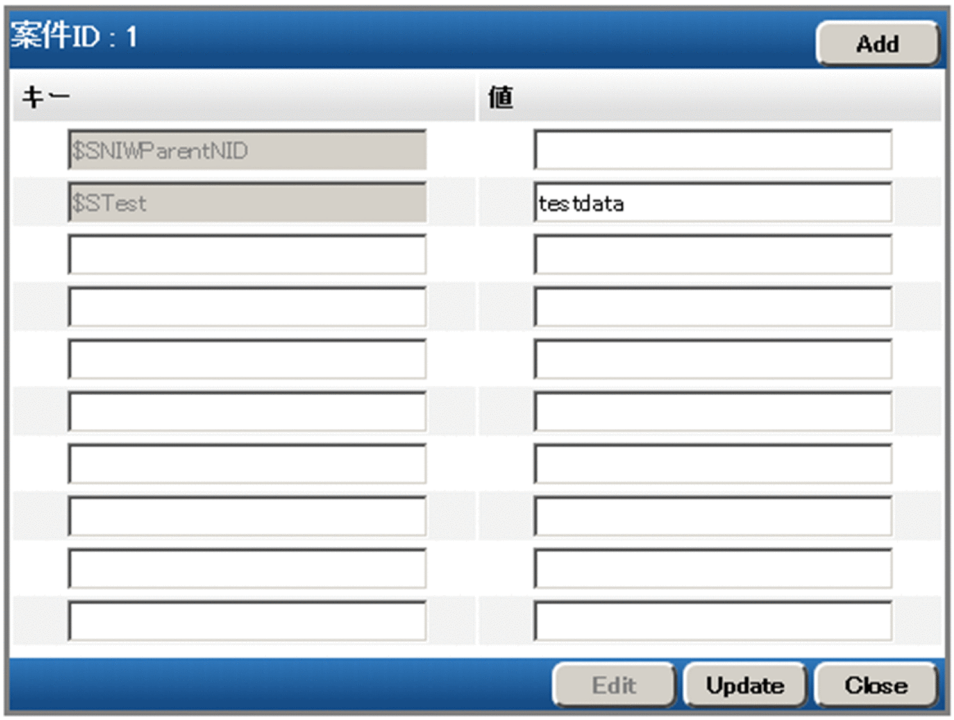


(6)　プロセスデータ画面

プロセスデータ画面では，指定した案件のプロセスデータの一覧を確認できます。また，指定した案件のプロセスデータの値を更新できます。

リスト型プロセスデータを登録する場合の指定方法については，「(7)　リスト型プロセスデータの指定形式」を参照してください。

図 10-8　プロセスデータ画面



「値」列の入力は必須ではありません。省略した場合，値が null のキーが登録されます。なお，空文字列は指定できません。

また，登録済みのプロセスデータの「キー」のテキストボックスは，［Edit］ボタンをクリックしても編集できません。

［Add］ボタンおよび［Update］ボタンは，［Edit］ボタンをクリックすると使用できます。

［Add］ボタンをクリックすると，「キー」および「値」のテキストボックスを増やせます。

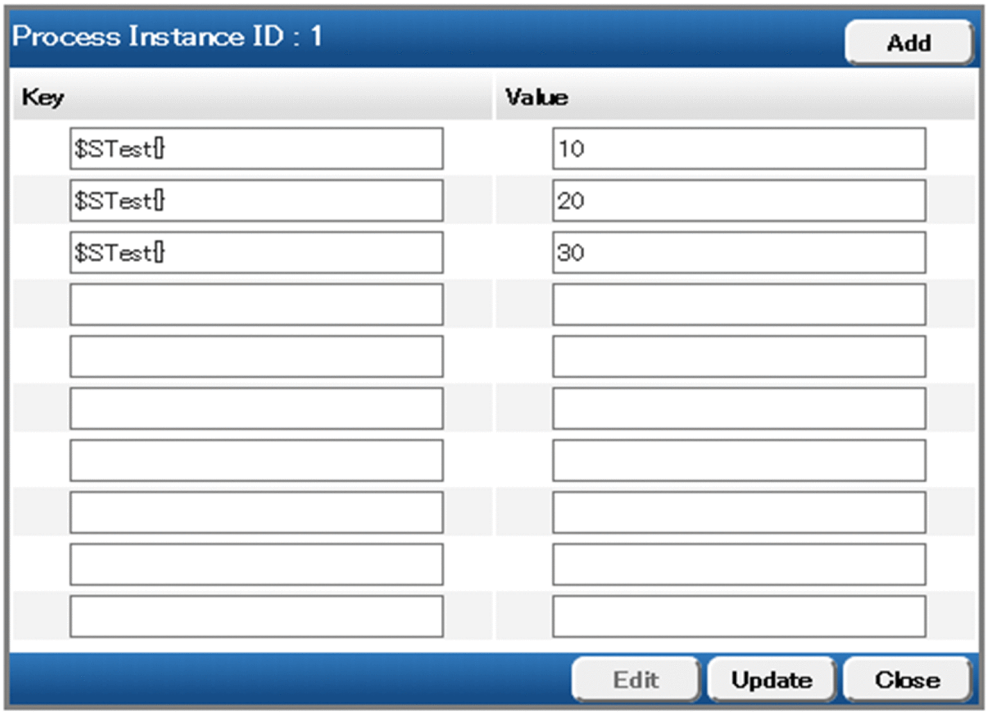
［Update］ボタンをクリックすると，プロセスデータが更新されます。

(7)　リスト型プロセスデータの指定形式

リスト型プロセスデータを登録する場合，プロセスデータキー名の最後に「{}」を指定してください。指定した各値は，上から順にリスト化されます。また，リスト型プロセスデータのリスト内識別子を指定する場合は，プロセスデータキー名の最後に「{<リスト内識別子>}」を指定してください。

プロセスデータ画面でのリスト型プロセスデータの指定例を次に示します。

図 10-9　リスト型プロセスデータの指定例（プロセスデータ画面）



関連項目

- 1.6　プロセスデータとは

10.3.3　ビジネスプロセスオペレータの使用例

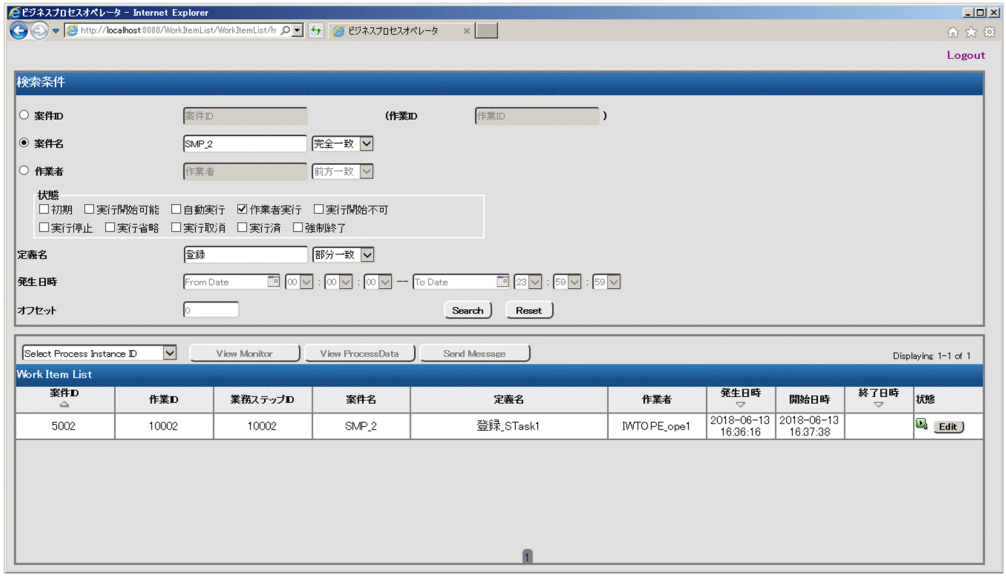
サービスタスクの状態を「作業者実行」※から「実行開始可能」へ変更する場合を想定した，ビジネスプロセスオペレータの使用例を示します。

注※

サービスタスクで業務アプリケーションの呼び出しに失敗した場合，「作業者実行」の状態になります。対策後に「実行開始可能」に変更すると，業務アプリケーションの呼び出しが再度実行されます。

1. 検索画面で，［案件名］，［状態］，［定義名］を指定し，［Search］ボタンをクリックします。
   ［状態］は「作業者実行」にチェックを入れます。また，［定義名］には，サービスタスクの名前の一部を指定します。

図 10-10　検索画面（ビジネスプロセスオペレータの使用例）



検索結果画面が表示されます。

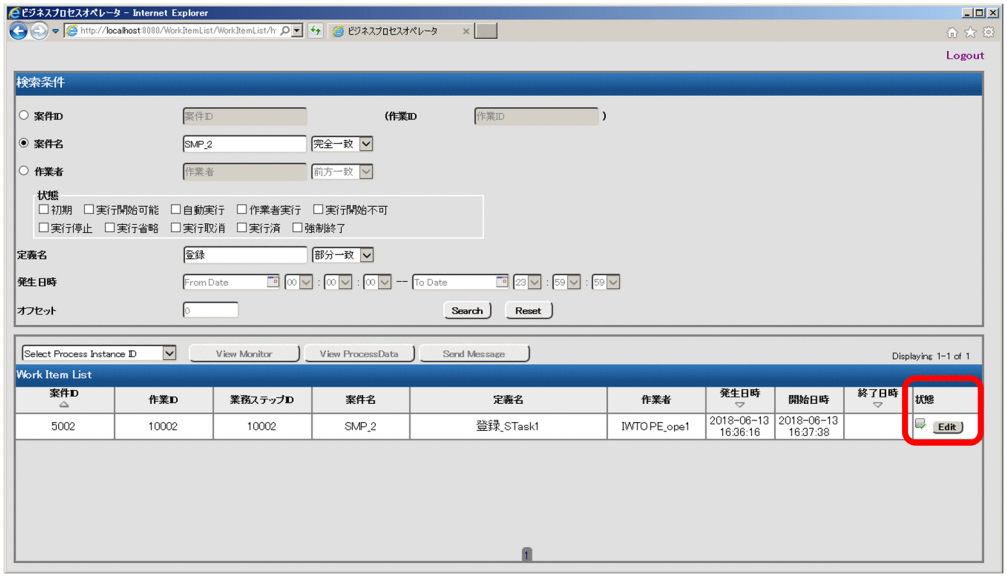
2. 検索結果から，状態を変更する対象の［Edit］ボタンをクリックします。

   状態変更画面が表示されます。

3. 状態変更画面で［Edit］ボタンをクリックし，［変更後の状態］を［実行開始可能］に変更して［Update］ボタンをクリックします。処理が成功すると，サービスタスクの状態が「実行開始可能」状態に変更されます。

図 10-11　状態変更画面（ビジネスプロセスオペレータの使用例）

図 10-12　検索画面の結果（ビジネスプロセスオペレータの使用例）



10.4　案件およびプロセスデータを削除する

案件は，BPMN 連携機能で使用できる API またはciwdelpi コマンドを使用して削除します。案件を削除すると，対応するプロセスデータも削除されます。

ciwdelpi コマンドの詳細については，マニュアル『uCosminexus Service Coordinator Interactive Workflow コマンド』を参照してください。

10.5　リトライの対象から外れた作業に関する運用

障害が発生し，作業が「作業者実行」状態になった場合の対処手順について説明します。

10.5.1　アプリケーション呼び出しを再実行する

「作業者実行」状態の作業を，再びアプリケーション呼び出しの対象に戻すことができます。

操作手順

1. ciwchgapwork コマンドを実行して，「作業者実行」状態の作業の情報を取得する。

   ciwchgapwork コマンドの指定形式を次に示します。

   ```
   ciwchgapwork -sid システムID -list -ope -ref ref識別子 > ファイルパス
   ```

   -ref オプションには対象の ref 識別子を指定します。-ref オプションは省略できます。

   出力結果はファイルにリダイレクトします。

   ciwchgapwork コマンドの詳細については，マニュアル『uCosminexus Service Coordinator Interactive Workflow コマンド』を参照してください。

2. 障害の原因を取り除く。

3. ciwchgapwork コマンドを実行して，作業を「実行開始可能」状態に戻す。

   ciwchgapwork コマンドの指定形式を次に示します。

   ```
   ciwchgapwork -sid システムID -chg -s READY -f ファイルパス
   ```

   -f オプションには，手順 1.でリダイレクトしたファイルを指定してください。

10.5.2　アプリケーション呼び出しをしないで案件を遷移させる

「作業者実行」状態の作業について，アプリケーションの呼び出しをしないで案件を遷移させることができます。

操作手順

1. ciwchgapwork コマンドを実行して，「作業者実行」状態の作業の情報を取得する。

   ciwchgapwork コマンドの指定形式を次に示します。

   ```
   ciwchgapwork -sid システムID -list -ope -ref ref識別子 > ファイルパス
   ```

-ref オプションには対象の ref 識別子を指定します。-ref オプションは省略できます。

出力結果はファイルにリダイレクトします。

ciwchgapwork コマンドの詳細については，マニュアル『uCosminexus Service Coordinator Interactive Workflow コマンド』を参照してください。

2. 障害の原因を取り除く。

3. ciwchgapwork コマンドを用いて作業を完了し，案件を遷移させる。

ciwchgapwork コマンドの指定形式を次に示します。

```
ciwchgapwork -sid システムID -chg -s EXECUTED -f ファイルパス
```

-f オプションには，手順 1.でリダイレクトしたファイルのパスを指定してください。

10.6　バックアップおよびリストアする

BPMN 連携機能に関するテーブルおよびファイルをバックアップおよびリストアします。

背景

障害などの不測の事態に備えて，BPMN 連携機能に関するテーブルおよびファイルのバックアップを取得することを推奨します。

表 10-2　バックアップおよびリストアするテーブル

項番	テーブル名	説明
1	createtable_hirdb.sql で作成したテーブル	「7.2　ワーク管理データベースを作成する（HiRDB の場合）」で作成したワーク管理データベースです。
2	createtableex_hirdb.sql で作成したテーブル	
3	createtable_oracle.sql で作成したテーブル	「7.3　ワーク管理データベースを作成する（ORACLE の場合）」で作成したワーク管理データベースです。
4	createtableex_oracle.sql で作成したテーブル	
5	createtable_postgresql.sql で作成したテーブル	「7.4　ワーク管理データベースを作成する（PostgreSQL の場合）」で作成したワーク管理データベースです。
6	createtableex_postgresql.sql で作成したテーブル	
7	業務テーブル	ワーク管理データベースと同期を取って更新する業務データベースです。

表 10-3　バックアップおよびリストアするファイル

項番	ファイル名	説明
1	システム設定プロパティファイル	ライブラリおよびコマンドを実行するために必要な，ワーク管理データベースの情報や共通情報を管理します。
2	セットアッププロパティファイル	ライブラリおよびコマンドを実行するために必要な情報をシステム ID 単位で管理します。
3	コマンド用環境設定ファイル	コマンドを実行するために必要な環境変数をシステム ID 単位で管理します。
4	共通設定ファイル	BPMN 連携機能全体を実行するために必要な共通情報を管理します。
5	アプリケーション呼び出し情報ファイル	アプリケーション呼び出しサービスが，ほかのアプリケーションを呼び出す際に使用する設定情報を管理します。
6	REST アプリケーション呼び出し用ヘッダファイル	REST アプリケーション呼び出しを行う際のリクエストの HTTP ヘッダに指定する内容を設定します。

項番	ファイル名	説明
7	REST アプリケーション呼び出しスキーマ変換用スタイルシート	リクエストまたはレスポンスのメッセージボディを，異なる XML スキーマに変換する際に使用します。
8	BPMN ビジネスプロセス定義ファイル	BPMN の形式でビジネスプロセスを定義します。
9	コールアクティビティ情報ファイル	コールアクティビティから子案件を投入する際に使用する情報を定義します。
10	uCosminexus Business Process Developer 設定ファイル	BPMN ビジネスプロセス定義ファイルの変換コマンド（ciwtransbpmn）の，ログに関する情報を管理します。

操作手順

1. 対象のテーブルおよびファイルをバックアップする。

   バックアップの詳細については，マニュアル『uCosminexus Service Coordinator Interactive Workflow システム構築・運用ガイド』の「7.6　バックアップとリストア」を参照してください。

2. 対象のテーブルおよびファイルをリストアする。

   リストアの詳細については，マニュアル『uCosminexus Service Coordinator Interactive Workflow システム構築・運用ガイド』の「7.6　バックアップとリストア」を参照してください。

関連項目

- 15.　設定ファイル

10.7　データベースの再編成

ワーク管理データベースへのアクセス性能を維持するためには，定期的にデータベースの再編成を実施する必要があります。再編成の方法については各 DBMS のマニュアルを参照してください。

10.8　運用上の注意事項

ワークフローシステムを運用する上での注意事項について，説明します。

10.8.1　業務アプリケーションが異常終了した場合の影響について

業務アプリケーションが異常終了した場合の影響については，マニュアル『uCosminexus Service Coordinator Interactive Workflow システム構築・運用ガイド』の「7.9.3　業務アプリケーションが異常終了した場合や強制終了した（終了処理をしなかった）場合の影響について」を参照してください。

11 REST API リファレンス

BPMN 連携機能が提供する REST API の詳細について説明します。

11.1　REST API の概要

HTTP リクエスト

REST API のリクエスト URL は，次に示すように，ドメイン名，コンテキストルート，API バージョンおよびリソースアクセスパスから構成されます。

```
http(s)://<ドメイン>/<コンテキストルート>/<APIバージョン>/<リソースアクセスパス>
```

ドメインは Web サーバで設定します。REST サービスのコンテキストルートは csciwws です。API バージョンは，v1，v2 など REST API のバージョンが API ごとに固定値として設定されます。残りの部分がリソースにアクセスするためのパスです。

HTTP メソッド

CSCIW の REST API で使用する HTTP メソッドを次の表で示します。

表 11-1　HTTP メソッド

HTTP メソッド	内容
GET	リソースを取得します。
POST	リソースを新しく追加します。
PUT	指定されたリソースを修正します。
DELETE	指定されたリソースを削除します。

リクエストパラメタ

業務アプリケーションが REST サービスにパラメタを渡す場合，クエリパラメタかリクエストボディを使用します。HTTP メソッドと，使用できるリクエストパラメタの関係を次の表で示します。

表 11-2　リクエストパラメタの送信方法

項番	リクエストパラメタ	GET	POST	PUT	DELETE
1	クエリパラメタ	○	×	×	○
2	リクエストボディ	△	○	○	×

（凡例）

○：利用できる

×：エラーが発生する

△：無視される

リクエストボディにパラメタを指定する場合，XML 形式または JSON 形式で記載します。その際，HTTP ヘッダのコンテンツタイプ（Content-Type）に application/xml か application/json を指定する必要が

あります。コンテンツタイプに application/xml か application/json 以外の値が指定された場合や，未指定の場合はエラーになります。

使用できないリクエストパラメタを送信した場合，ステータスコード 400 のエラーが返されます。

リクエストボディに指定するパラメタをすべて省略する場合は，「リクエストパラメタ省略時のリクエストボディと HTTP ヘッダ」を参照してください。

HTTP レスポンス

REST API が正常終了した場合，HTTP レスポンスには，2xx のステータスコードが返されます。また，レスポンスボディにはリソースなどの情報が，XML または JSON 形式で返されます。コンテンツタイプは application/xml または application/json です。レスポンスで返されるリソースの情報（プロセスデータは除く）については，マニュアル『uCosminexus Service Coordinator Interactive Workflow AP 開発ガイド』の「付録 B.1 指定できる属性一覧」を参照してください。

レスポンスの形式は，HTTP ヘッダの"Accept"に application/xml または application/json を指定すると切り替わります。application/xml または application/json 以外の値が指定された場合はエラーになります。"Accept"が未指定の場合は XML 形式で返されます。

REST API がエラーになった場合，HTTP レスポンスには，3xx，4xx または 5xx のステータスコードが返されます。また，レスポンスボディには次に示すようなエラードキュメントが返されます。エラーを受け取ったクライアントは，返却された HTTP レスポンスと，ステータスコードに従ってエラーに対処できます。HTTP レスポンスのボディに設定しているメッセージにはエラーの詳細情報は付与されません。発生したエラーの詳細情報については，REST API のログファイルを確認してください。

エラー時のレスポンスボディ（XML の場合）

```
<?xml version="1.0" encoding="UTF-8"?>
<error>
  <code>KDIWnnnnn-E</code>
  <message>メッセージ本文</message>
</error>
```

エラー時のレスポンスボディ（JSON の場合）

```
{
  "code" : "KDIWnnnnn-E",
  "message" : "メッセージ本文"
}
```

ユーザ記述子

REST API が内部で Java API を呼び出すとき，REST API から Java API にユーザ記述子を渡します。REST API から渡すユーザ記述子のデフォルト値は csciwws です。デフォルト値とは異なるユーザ記述子を指定する場合は，共通設定ファイルの`RestServiceUserDescription` パラメタを変更するか，各 REST

API のユーザ記述子のパラメタ（userdescription）を指定します。ユーザ記述子の長さは 32 バイト以下で指定してください。

ユーザ記述子については，マニュアル『uCosminexus Service Coordinator Interactive Workflow AP 開発ガイド』の「createCIWServer」を参照してください。

データ型と文字コード

業務アプリケーションと REST サービスのリクエストとレスポンスでやり取りされるデータ型は，数値，文字列，配列，日付です。日付は，yyyy-MM-dd'T'HH:mm:ssZ の形式（ISO 8601 の拡張形式）で表します（例：2016-06-09T10:32:41+09:00）。レスポンスの文字コードは UTF-8 です。

また，無限遠の日付の場合は日付の形式ではなく定数で表します。無限遠の未来の場合は BEYOND となり，無限遠の過去の場合は ORIGIN となります。

オブジェクトの状態および種類

CSCIW のオブジェクトの状態および種類と，それに対応する定数名とコード値の対応を次の表に示します。レスポンスでは，コード値を文字列型で返します。状態の定数名とコード値については，マニュアル『uCosminexus Service Coordinator Interactive Workflow AP 開発ガイド』の「付録 B.2 指定できる属性値」の「(1)State 属性で指定できる値」，および「付録 C 状態遷移モデル」を参照してください。フローノードとフローノード定義の種類に関しては，定数名を返します。定数名の詳細については，次の記述を参照してください。

- 「12.9.2　CIWBPMNFlowNodeInstance.AttributeName（フローノードの属性名の列挙型）」
- 「12.9.3　CIWBPMNFlowNodeDefinition.AttributeName（フローノード定義の属性名の列挙型）」
- 「12.9.4　CIWBPMNFlowNodeDefinition.Type（フローノード定義における BPMN 要素の種類の列挙型）」

表 11-3　オブジェクトの状態および種類

状態および種類	定数名（コード値）
業務ステップの状態	INITIAL(i), READY(j), RUNNING(d), DISABLED(l), INTERMITTED(m), READY_FOR_TRANSITION(s), TRANSITION_COMPLETED(t), NOT_EXECUTED(p), TERMINATED(u), UNDEFINED(z)
業務ステップの種類	NORMAL(0), UNDEFINED(z)
ビジネスプロセス定義の状態	ACTIVE(b), INACTIVE(a), UNDEFINED(z)
作業定義の種類	BUILTIN_CONCURRENT_WORK(b), NORMAL(0), UNDEFINED(z)
案件の状態	NOT_STARTED(h), RUNNING(d), INTERMITTED(m), COMPLETED(o), TERMINATED(u), UNDEFINED(z)
作業の状態	INITIAL(i), READY(j), EXECUTING(e), PERFORMING(f), NOT_EXECUTED(p), CANCELED(q), EXECUTED(r), DISABLED(l), INTERMITTED(m), TERMINATED(u), UNDEFINED(z)

状態および種類	定数名（コード値）
作業の種類	BUILTIN_CONCURRENT_WORK(b), NORMAL(0), UNDEFINED(z)

XML と JSON の相違点

XML の場合

- リクエストボディとレスポンスにルート要素（<Parameter>や<ProcessInstances>など）を含みます。
- タグを省略した場合に null として扱われます。

JSON の場合

- リクエストボディとレスポンスにルート要素を含みません。このため，各 REST API のリクエストボディとレスポンスの構造に記載されているルート要素は無視されます。
- 要素を省略した場合に null として扱われます。
- Cosminexus アプリケーションサーバでは，値に null が指定された場合は空文字として扱われます。
- レスポンスの型が数値，または boolean 型の場合に，レスポンスはダブルクォーテーション「"」で囲われて返されます。

レスポンスの最大取得数

リソースの一覧を取得する REST API の場合，レスポンスで取得できる最大データ件数を指定できます(デフォルト値は 100 件)。最大データ取得件数は，RestServiceResponseMaxCount パラメタにデフォルト値を設定するか，各 REST API の，リクエストパラメタに指定できます。

一覧取得のフィルター条件

リソースの一覧を取得する REST API の場合，情報を絞り込むためのフィルター条件を設定できます。

日付型のデータを条件に指定する場合，1970/01/01 00:00:00 GMT を起点とした通算秒を指定する必要があります。日付が無限遠の過去の場合の通算秒は 0 となり，無限遠の未来の場合は 9223372036854775 となります。

Java での日付型データの変換例を次に示します。

```
SimpleDateFormat format = new SimpleDateFormat("yyyy-MM-dd'T'HH:mm:ssXXX");
Date date = format.parse("2016-01-01T00:00:00+09:00");
// 1970/01/01 00:00:00 GMT を起点とした通算ミリ秒を取得し，通算秒に変換
long second = date.getTime() / 1000L;

String filterCondition = "StartDate<" + second;
```

ワーク管理データベースが PostgreSQL の場合，フィルター条件に PostgreSQL の C 形式エスケープ（E'～'）を使用することはできません。

一覧取得のソート条件

リソースの一覧を取得する REST API の場合，ソート条件を設定できます。

ワーク管理データベースが PostgreSQL の場合，ソート条件に PostgreSQL の C 形式エスケープ（E'～'）を使用することはできません。

REST API でのプロセスデータの扱い

単一型プロセスデータを扱う場合

REST API で単一型プロセスデータを扱う場合，次に示すように ProcessData タグにプロセスデータキー名とプロセスデータ値を指定します。

XML の場合

```
<ProcessDataList>
  <ProcessData>
    <Key>$SProduct</Key>
    <Value>cake</Value>
  </ProcessData>
  <ProcessData>
    <Key>$NStock</Key>
    <Value>30</Value>
  </ProcessData>
</ProcessDataList>
```

JSON の場合

```
"ProcessDataList" : {
  "ProcessData" : [
    {
      "Key" : "$SProduct",
      "Value" : "cake"
    },
    {
      "Key" : "$NStock",
      "Value" : "30"
    }
  ]
}
```

リスト型プロセスデータを扱う場合

REST API でリスト型プロセスデータを扱う場合，次に示すように，key 名の最後に{}を指定します。各データは定義されている順番にリスト化されます。また，リスト型プロセスデータの 1 要素を扱う場合は，key 名の最後に{<リスト内識別子>}を指定します。

XML の場合

```
<ProcessDataList>
  <ProcessData>
    <Key>$SProduct{}</Key>
    <Value>Bread</Value>
  </ProcessData>
  <ProcessData>
```

```
    <Key>$SProduct{}</Key>
    <Value>Butter</Value>
  </ProcessData>
  <ProcessData>
    <Key>$SProduct{}</Key>
    <Value>Strawberry</Value>
  </ProcessData>
  <ProcessData>
    <Key>$SProductCode{1}</Key>
    <Value>01234567895</Value>
  </ProcessData>
</ProcessDataList>
```

JSON の場合

```
"ProcessDataList" : {
  "ProcessData" : [
    {
      "Key" : "$SProduct{}",
      "Value" : "Bread"
    },
    {
      "Key" : "$SProduct{}",
      "Value" : "Butter"
    },
    {
      "Key" : "$SProduct{}",
      "Value" : "Strawberry"
    },
    {
      "Key" : "$SProductCode{1}",
      "Value" : "01234567895"
    }
  ]
}
```

リクエストパラメタ省略時のリクエストボディと HTTP ヘッダ

一部の REST API では，リクエストボディに指定するパラメタをすべて省略できます。パラメタをすべて省略する場合は，リクエストボディを省略できます。リクエストボディを省略するときは，HTTP ヘッダのContent-Type を指定しないでください。リクエストボディを省略してContent-Type を指定した場合の動作はサポートしません。

リクエストボディに指定するパラメタをすべて省略する場合のContent-Type とリクエストボディの指定値を，次の表に示します。

表 11-4　Content-Type とリクエストボディの指定値

項番	Content-Type	リクエストボディ	リクエストボディの例
1	指定しない	指定しない	−
2	application-xml	ルート要素だけ指定する	<Parameter/>
3	application-json	{}だけ指定する	{}

（凡例）

－：該当しない

11.2 REST API 一覧

REST サービスが対応するリソースの操作と，それに対応する URL の一覧を示します。

REST API の一覧

REST API を使用したリソースの操作と URL の対応を次の表に示します。

項番	リソースの操作		URL※
－	案件		
1		案件の一覧を取得する	GET /v1/process-instance
2		指定したプロセスデータを含む案件の一覧を取得する	POST /v1/process-instance/process-data
3		案件数を取得する	GET /v1/process-instance/count
4		案件を取得する	GET /v1/process-instance/(PIID)
5		ビジネスプロセス定義名から案件を取得する	GET /v1/process-instance/queries
6		案件を生成して開始する	POST /v1/process-instance/create-and-start
7		案件（メッセージ）を生成して開始する	POST /v1/process-instance/create-and-start-message
8		案件（タイマー）を生成して開始する	POST /v1/process-instance/create-and-start-timer
9		案件を削除する	DELETE /v1/process-instance/(PIID)
10		案件を強制終了する	PUT /v1/process-instance/(PIID)/terminate
11		親案件を取得する	GET /v1/process-instance/(PIID)/call-activity-parent/process-instance
12		コールアクティビティを取得する	GET /v1/process-instance/(PIID)/call-activity-parent/work-item
－	業務ステップ		
13		業務ステップの一覧を取得する	GET /v1/activity-instance
14		業務ステップ数を取得する	GET /v1/activity-instance/count
15		業務ステップを取得する	GET /v1/activity-instance/(PIID)/(AIID)
16		業務ステップを差し戻すまたは引き戻す	PUT /v1/activity-instance/(PIID)/(AIID)/make-backward-transition
17		業務ステップを強制遷移する	PUT /v1/activity-instance/(PIID)/(AIID)/adhoc-create-and-make-transition
18		業務ステップの状態を変更する	PUT /v1/activity-instance/(PIID)/(AIID)/change-state

項番	リソースの操作		URL※
−	作業		
19		作業の一覧を取得する	GET /v1/work-item
20		作業数を取得する	GET /v1/work-item/count
21		作業を取得する	GET /v1/work-item/(PIID)/(WIID)
22		作業に着手する	PUT /v1/work-item/(PIID)/(WIID)/perform
23		作業を完了する	PUT /v1/work-item/(PIID)/(WIID)/complete
24		作業者を変更する	PUT /v1/work-item/(PIID)/(WIID)/reassign
25		作業者を変更して着手する	PUT /v1/work-item/(PIID)/(WIID)/reassign-and-perform
26		作業を着手して完了する	PUT /v1/work-item/(PIID)/(WIID)/perform-and-complete
27		作業の状態を変更する	PUT /v1/work-item/(PIID)/(WIID)/change-state
28		条件に一致する作業の作業者割り当てと着手をする	PUT /v1/work-item/allocate
29		着手した作業を返却する	PUT /v1/work-item/(PIID)/(WIID)/free
30		タイマーの処理期限を変更する	PUT /v1/work-item/(PIID)/(WIID)/timer/set-deadline
31		子案件を取得する	GET /v1/work-item/(PIID)/(WIID)/call-activity-child/process-instance
−	ビジネスプロセス定義		
32		ビジネスプロセス定義の一覧を取得する	GET /v1/process-definition
33		ビジネスプロセス定義数を取得する	GET /v1/process-definition/count
34		ビジネスプロセス定義を取得する	GET /v1/process-definition/(PDID)
35		ビジネスプロセス定義内の案件の一覧を取得する	GET /v1/process-definition/(PDID)/process-instance
−	作業定義		
36		作業定義の一覧を取得する	GET /v1/work-definition
37		作業定義数を取得する	GET /v1/work-definition/count
38		作業定義を取得する	GET /v1/work-definition/(PDID)/(WDID)
−	BPMN ビジネスプロセス定義ファイル		
39		BPMN ビジネスプロセス定義ファイルを取得する	GET /v1/bpmn/(PIID)
−	イベント		

項番	リソースの操作		URL※
40		イベント（メッセージ）を送信する	POST /v1/event/send-message/(PIID)
－	プロセスデータ		
41		プロセスデータを取得する	POST /v1/process-data
42		プロセスデータを登録する	POST /v1/process-data/set-process-data
43		リスト型プロセスデータのインデクスを取得する	POST /v1/process-data/list-index
－	フローノード		
44		フローノード一覧を取得する	GET /v1/flow-node-instance/queries
45		アドホック・サブプロセスのフローノードを生成する	POST /v1/flow-node-instance/create
－	フローノード定義		
46		フローノード定義一覧を取得する	GET /v1/flow-node-definition/queries
－	アドホック・サブプロセス		
47		アドホック・サブプロセスの状態を変更する	PUT /v1/adhoc-sub-process/change-state
－	業務ステップ定義		
48		業務ステップ定義一覧を取得する	GET /v1/activity-definition
49		業務ステップ定義数を取得する	GET /v1/activity-definition/count
50		業務ステップ定義を取得する	GET /v1/activity-definition/(PDID)/(ADID)

注※　括弧内の文字列は，次の ID を表します。

(PIID)：案件 ID

(AIID)：業務ステップの ID

(WIID)：作業 ID

(PDID)：ビジネスプロセス定義の ID

(WDID)：作業定義の ID

(ADID)：業務ステップ定義の ID

11.3　REST API の記述形式

「各 REST API の詳細」での API の記述形式を示します。ここでは URL スキーム，ドメイン名およびコンテキストルートは省略します。例えば，ドメイン名が restserver の場合，案件一覧を取得する REST API の URL は，http://restserver/csciwws/v1/process-instance のようになります。

メソッドと URL

メソッドと URL の形式を説明しています。

クエリパラメタまたはリクエストボディ

各クエリパラメタまたはリクエストボディに指定する名前，型，省略可否，および内容について説明しています。API によって，クエリパラメタまたはリクエストボディのどちらかを使用します。

> **ヒント**
>
> 一覧を取得する API の場合に，クエリパラメタの`filter`，`sort`，`offset`，および`maxcount` に指定できる内容の詳細については，マニュアル『uCosminexus Service Coordinator Interactive Workflow AP 開発ガイド』の「2.7 一覧情報の取得」および「付録 B.1 指定できる属性一覧」を参照してください。
>
> 案件を生成して開始する API の場合に，リクエストボディの`Name`，`Deadline`，`Priority` に指定できる内容の詳細については，マニュアル『uCosminexus Service Coordinator Interactive Workflow AP 開発ガイド』の「CIWServer（ワーク管理システム全般にかかわる処理を行うための機能を提供するインタフェース)」の「createAndStartProcessInstance」を参照してください。

リクエスト例

リクエスト URL およびリクエストボディの指定例を示しています。

レスポンス

レスポンスの形式を説明しています。レスポンスが XML 形式または JSON 形式の場合，その構造も説明しています。

ステータスコード

ステータスコード，メディアタイプおよび内容を説明しています。

レスポンス例

レスポンス例を示しています。

11.4 XML スキーマファイル

REST API のリクエスト，レスポンスで使用する XML スキーマのファイルの一覧を示します。

XML スキーマファイルの一覧

REST API のリクエストパラメタ，レスポンスデータを XML 形式で定義した，XML スキーマをファイル形式で提供します。

REST API と XML スキーマファイルの対応を次に示します。XML スキーマファイルの格納先は，<CSCIW のインストールディレクトリ>/bpmn/schema です。

項番	REST API		XML スキーマファイル
－	案件		
1		案件の一覧を取得する	getProcessInstanceListResponse.xsd
2		指定したプロセスデータを含む案件の一覧を取得する	getProcessInstanceListWithProcessDataRequest.xsd getProcessInstanceListResponse.xsd
3		案件数を取得する	getResourceCountResponse.xsd
4		案件を取得する	getProcessInstanceResponse.xsd
5		ビジネスプロセス定義名から案件を取得する	getProcessInstanceListResponse.xsd
6		案件を生成して開始する	createAndStartProcessInstanceRequest.xsd getProcessInstanceResponse.xsd
7		案件（メッセージ）を生成して開始する	createAndStartMessageProcessInstanceRequest.xsd getProcessInstanceResponse.xsd
8		案件（タイマー）を生成して開始する	createAndStartTimerProcessInstanceRequest.xsd getProcessInstanceResponse.xsd
9		案件を削除する	－
10		案件を強制終了する	terminateProcessInstanceRequest.xsd getProcessInstanceResponse.xsd
11		親案件を取得する	getProcessInstanceResponse.xsd
12		コールアクティビティを取得する	getWorkItemResponse.xsd
－	業務ステップ		
13		業務ステップの一覧を取得する	getActivityInstanceListResponse.xsd
14		業務ステップ数を取得する	getResourceCountResponse.xsd
15		業務ステップを取得する	getActivityInstanceResponse.xsd

項番	REST API		XML スキーマファイル
16		業務ステップを差し戻すまたは引き戻す	makeBackwardTransitionActivityInstanceRequest.xsd getActivityInstanceResponse.xsd
17		業務ステップを強制遷移する	adhocCreateAndMakeTransitionRequest.xsd getActivityInstanceResponse.xsd
18		業務ステップの状態を変更する	changeActivityInstanceStateRequest.xsd getActivityInstanceResponse.xsd
－	作業		
19		作業の一覧を取得する	getWorkItemListResponse.xsd
20		作業数を取得する	getResourceCountResponse.xsd
21		作業を取得する	getWorkItemResponse.xsd
22		作業に着手する	performWorkItemRequest.xsd getWorkItemResponse.xsd
23		作業を完了する	completeWorkItemRequest.xsd getWorkItemResponse.xsd
24		作業者を変更する	reassignWorkItemRequest.xsd getWorkItemResponse.xsd
25		作業者を変更して着手する	reassignAndPerformWorkItemRequest.xsd getWorkItemResponse.xsd
26		作業を着手して完了する	performAndCompleteWorkItemRequest.xsd getWorkItemResponse.xsd
27		作業の状態を変更する	changeWorkItemStateRequest.xsd getWorkItemResponse.xsd
28		条件に一致する作業の作業者割り当てと着手をする	allocateWorkItemRequest.xsd getWorkItemResponse.xsd
29		着手した作業を返却する	freeWorkItemRequest.xsd getWorkItemResponse.xsd
30		タイマーの処理期限を変更する	setWorkItemDeadlineRequest.xsd getWorkItemResponse.xsd
31		子案件を取得する	getProcessInstanceResponse.xsd
－	ビジネスプロセス定義		
32		ビジネスプロセス定義の一覧を取得する	getProcessDefinitionListResponse.xsd
33		ビジネスプロセス定義数を取得する	getResourceCountResponse.xsd
34		ビジネスプロセス定義を取得する	getProcessDefinitionResponse.xsd

項番	REST API		XML スキーマファイル
35		ビジネスプロセス定義内の案件の一覧を取得する	getProcessInstanceListResponse.xsd
－	作業定義		
36		作業定義の一覧を取得する	getWorkDefinitionListResponse.xsd
37		作業定義数を取得する	getResourceCountResponse.xsd
38		作業定義を取得する	getWorkDefinitionResponse.xsd
－	BPMN ビジネスプロセス定義ファイル		
39		BPMN ビジネスプロセス定義ファイルを取得する	－
－	イベント		
40		イベント（メッセージ）を送信する	sendMessageRequest.xsd
－	プロセスデータ		
41		プロセスデータを取得する	getProcessDataRequest.xsd getProcessDataResponse.xsd
42		プロセスデータを登録する	setProcessDataRequest.xsd
43		リスト型プロセスデータのインデクスを取得する	getListProcessDataIndexRequest.xsd getListProcessDataIndexResponse.xsd
－	フローノード		
44		フローノード一覧を取得する	getFlowNodeInstanceListResponse.xsd
45		アドホック・サブプロセスのフローノードを生成する	createFlowNodeInstanceRequest.xsd getFlowNodeInstanceListResponse.xsd
－	フローノード定義		
46		フローノード定義一覧を取得する	getFlowNodeDefinitionListResponse.xsd
－	アドホック・サブプロセス		
47		アドホック・サブプロセスの状態を変更する	changeAdHocSubProcessStateRequest.xsd
－	業務ステップ定義		
48		業務ステップ定義の一覧を取得する	getActivityDefinitionListResponse.xsd
49		業務ステップ定義数を取得する	getResourceCountResponse.xsd
50		業務ステップ定義を取得する	getActivityDefinitionResponse.xsd

(凡例)

－：該当なし

11.5　各 REST API の詳細

各 REST API の詳細を示します。

11.5.1　案件の一覧取得

指定したフィルター条件を満たす案件の一覧を取得します。この API を実行すると，CSCIW Java API のCIWServer.getProcessInstancesList インタフェースが呼び出されます。

メソッドと URL

```
GET /v1/process-instance
```

クエリパラメタ

項番	名前	型	指定要否	内容
1	userdescription	文字列	省略可	ユーザ記述子を指定します。省略した場合はデフォルト値が指定されます。空文字列は指定できません。
2	filter	文字列	省略可	案件一覧を取得する場合のフィルター条件を指定します。省略した場合はフィルター条件は指定されません。空文字列は指定できません。
3	sort	文字列	省略可	取得した案件一覧をソートする場合の条件を指定します。省略した場合はソート条件は指定されません。空文字列は指定できません。
4	offset	数値	省略可	案件一覧を取得する場合のオフセットを指定します。先頭は 0 になります。省略した場合は 0 が指定されます。0 未満の値は指定できません。
5	maxcount	数値	省略可	案件一覧を取得する場合の最大取得数を指定します。省略した場合デフォルト値が指定されます。すべてを取得する場合は，-1 を指定します。-1 未満の値は指定できません。

リクエスト例

リクエスト URL

```
GET http://restserver/csciwws/v1/process-instance?filter=NAME%3D%27PI0001%27
```

レスポンス

案件一覧を返します。個々の案件のプロパティを次に示します。

項番	名前	型	内容
1	ClosedDate	日付	案件の終了日時
2	Creator	文字列	案件の投入者
3	Deadline	日付	案件の処理期限の絶対日時
4	ID	数値	案件の ID
5	MovedDate	日付	案件の乗せ替え日時
6	Name	文字列	案件の名称（案件キー）
7	Priority	数値	案件の優先度
8	ProcessDefinitionID	数値	案件が所属するビジネスプロセス定義の ID
9	ProcessDefinitionName	文字列	案件が所属するビジネスプロセス定義の名称
10	StartDate	日付	案件の開始日時
11	StateCode	文字列	案件の状態

レスポンスの構造を次に示します。

項番	名前			出現回数
1	ProcessInstances			1
2		ProcessInstance		0 または 1 以上
3			ClosedDate	1
4			Creator	1
5			Deadline	1
6			ID	1
7			MovedDate	1
8			Name	1
9			Priority	1
10			ProcessDefinitionID	1
11			ProcessDefinitionName	1
12			StartDate	1
13			StateCode	1

ステータスコード

項番	ステータスコード	内容
1	200	成功（データが 0 件の場合も成功）

項番	ステータスコード	内容
2	400	リクエストパラメタの不正
3	500	内部処理エラー

レスポンス例

レスポンス（XML の場合）

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ProcessInstances>
  <ProcessInstance>
    <ClosedDate></ClosedDate>
    <Creator>csciwws</Creator>
    <Deadline></Deadline>
    <ID>5001</ID>
    <MovedDate></MovedDate>
    <Name>PI0001</Name>
    <Priority>0</Priority>
    <ProcessDefinitionID>3001</ProcessDefinitionID>
    <ProcessDefinitionName>販売業務</ProcessDefinitionName>
    <StartDate>2016-12-06T15:31:12+09:00</StartDate>
    <StateCode>d</StateCode>
  </ProcessInstance>
</ProcessInstances>
```

レスポンス（JSON の場合）

```
{
  "ProcessInstance" : [
    {
      "ClosedDate" : "",
      "Creator" : "csciwws",
      "Deadline" : "",
      "ID" : "5001",
      "MovedDate" : "",
      "Name" : "PI0001",
      "Priority" : "0",
      "ProcessDefinitionID" : "3001",
      "ProcessDefinitionName" : "販売業務",
      "StartDate" : "2016-12-06T15:31:12+09:00",
      "StateCode" : "d"
    }
  ]
}
```

11.5.2　指定したプロセスデータを含む案件の一覧取得

指定したプロセスデータを含む案件の一覧を取得します。この API を実行すると，BPMN 連携ライブラリ Java API の`CIWBPMNLib.getPIIDListByProcessData` インタフェースが呼び出されます。

メソッドと URL

```
POST /v1/process-instance/process-data
```

リクエストボディ

項番	名前	型	指定要否	内容
1	UserDescription	文字列	省略可	ユーザ記述子を指定します。省略した場合はデフォルト値が指定されます。空文字列は指定できません。
2	Sort	文字列	省略可	取得した案件一覧をソートする場合の条件を指定します。昇順の場合は"ASC"，降順の場合は"DESC を指定します。省略した場合は，昇順"ASC"でソートされます。空文字列は指定できません。
3	Offset	数値	省略可	案件一覧を取得する場合のオフセットを指定します。先頭には 0 を指定します。省略した場合は 0 が指定されます。0 未満の値は指定できません。
4	MaxCount	数値	省略可	案件一覧を取得する場合の最大取得数を指定します。省略した場合デフォルト値が指定されます。すべてを取得する場合は，-1 を指定します。-1 未満の値は指定できません。
5	ProcessDataList	配列	必須	プロセスデータ一覧を指定します。
6	ProcessData	オブジェクト	必須	プロセスデータオブジェクトを指定します。次の 2 つの 要素（Key および Value）から構成されています。
7	Key	文字列	必須	プロセスデータキー名を指定します。空文字列は指定で きません。
8	Value	文字列	省略可	検索するプロセスデータ値を指定します。省略した場合，値が null のキーを検索します。空文字列は指定できません。

リクエストボディの構造を次に示します。

項番	名前		出現回数
1	Parameter		1
2		UserDescription	0 または 1
3		Sort	0 または 1
4		Offset	0 または 1
5		MaxCount	0 または 1
6		ProcessDataList	1

項番	名前				出現回数
7			ProcessData		1 以上
8				Key	1
9				Value	0 または 1

リクエスト例

リクエスト URL

```
POST http://restserver/csciwws/v1/process-instance/process-data
```

リクエストボディ（XML の場合）

```
<?xml version="1.0" encoding="UTF-8" ?>
<Parameter>
  <UserDescription>csciwuser</UserDescription>
  <Sort>ASC</Sort>
  <Offset>30</Offset>
  <MaxCount>50</MaxCount>
  <ProcessDataList>
    <ProcessData>
      <Key>$SdateStr</Key>
      <Value>April</Value>
    </ProcessData>
    <ProcessData>
      <Key>$Sproduct{}</Key>
      <Value>product1</Value>
    </ProcessData>
    <ProcessData>
      <Key>$Sproduct{}</Key>
      <Value>product2</Value>
    </ProcessData>
    <ProcessData>
      <Key>$Sdate{1}</Key>
      <Value>January</Value>
    </ProcessData>
  </ProcessDataList>
</Parameter>
```

リクエストボディ（JSON の場合）

```
{
  "UserDescription" : "csciwuser",
  "Sort" : "ASC",
  "Offset" : "30",
  "MaxCount" : "50",
  "ProcessDataList" : {
    "ProcessData" : [
      {
        "Key" : "$SdateStr",
        "Value" : "April"
```

```
      },
      {
        "Key" : "$Sproduct{}",
        "Value" : "product1"
      },
      {
        "Key" : "$Sproduct{}",
        "Value" : "product2"
      },
      {
        "Key" : "$Sdate{1}",
        "Value" : "January"
      }
    ]
  }
}
```

レスポンス

案件一覧を返します。個々の案件のプロパティを次に示します。

項番	名前	型	内容
1	ClosedDate	日付	案件の終了日時
2	Creator	文字列	案件の投入者
3	Deadline	日付	案件の処理期限の絶対日時
4	ID	数値	案件の ID
5	MovedDate	日付	案件の乗せ替え日時
6	Name	文字列	案件の名称（案件キー）
7	Priority	数値	案件の優先度
8	ProcessDefinitionID	数値	案件が所属するビジネスプロセス定義の ID
9	ProcessDefinitionName	文字列	案件が所属するビジネスプロセス定義の名称
10	StartDate	日付	案件の開始日時
11	StateCode	文字列	案件の状態

レスポンスの構造を次に示します。

項番	名前			出現回数
1	ProcessInstances			1
2		ProcessInstance		0 または 1 以上
3			ClosedDate	1
4			Creator	1
5			Deadline	1

項番	名前			出現回数
6			ID	1
7			MovedDate	1
8			Name	1
9			Priority	1
10			ProcessDefinitionID	1
11			ProcessDefinitionName	1
12			StartDate	1
13			StateCode	1

ステータスコード

項番	ステータスコード	内容
1	200	成功（データが 0 件の場合も成功）
2	400	リクエストパラメタの不正
3	500	内部処理エラー

レスポンス例

レスポンス（XML の場合）

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ProcessInstances>
  <ProcessInstance>
    <ClosedDate></ClosedDate>
    <Creator>csciwws</Creator>
    <Deadline></Deadline>
    <ID>1</ID>
    <MovedDate></MovedDate>
    <Name>案件A</Name>
    <Priority>0</Priority>
    <ProcessDefinitionID>3001</ProcessDefinitionID>
    <ProcessDefinitionName>販売業務</ProcessDefinitionName>
    <StartDate>2016-12-06T15:31:12+09:00</StartDate>
    <StateCode>d</StateCode>
  </ProcessInstance>
  <ProcessInstance>
    <ClosedDate></ClosedDate>
    <Creator>csciwws</Creator>
    <Deadline></Deadline>
    <ID>2</ID>
    <MovedDate></MovedDate>
    <Name>案件B</Name>
    <Priority>0</Priority>
    <ProcessDefinitionID>3001</ProcessDefinitionID>
    <ProcessDefinitionName>販売業務</ProcessDefinitionName>
```

```
    <StartDate>2016-12-06T15:31:23+09:00</StartDate>
    <StateCode>d</StateCode>
  </ProcessInstance>
</ProcessInstances>
```

レスポンス（JSON の場合）

```
{
  "ProcessInstance" : [
    {
      "ClosedDate" : "",
      "Creator" : "csciwws",
      "Deadline" : "",
      "ID" : "1",
      "MovedDate" : "",
      "Name" : "案件A",
      "Priority" : "0",
      "ProcessDefinitionID" : "3001",
      "ProcessDefinitionName" : "販売業務",
      "StartDate" : "2016-12-06T15:31:12+09:00",
      "StateCode" : "d"
    },
    {
      "ClosedDate" : "",
      "Creator" : "csciwws",
      "Deadline" : "",
      "ID" : "2",
      "MovedDate" : "",
      "Name" : "案件B",
      "Priority" : "0",
      "ProcessDefinitionID" : "3001",
      "ProcessDefinitionName" : "販売業務",
      "StartDate" : "2016-12-06T15:31:23+09:00",
      "StateCode" : "d"
    }
  ]
}
```

11.5.3　案件数の取得

指定した条件を満たす案件の数を取得します。この API を実行すると，CSCIW Java API の CIWServer.getListCount インタフェースが呼び出されます。

メソッドと URL

```
GET /v1/process-instance/count
```

クエリパラメタ

項番	名前	型	指定要否	内容
1	userdescription	文字列	省略可	ユーザ記述子を指定します。省略した場合はデフォルト値が指定されます。空文字列は指定できません。
2	filter	文字列	省略可	案件数を取得する場合のフィルター条件を指定します。省略した場合はフィルター条件は指定されません。空文字列は指定できません。

リクエスト例

リクエスト URL

```
GET http://restserver/csciwws/v1/process-instance/count?filter=ID%3E5000
```

レスポンス

案件数を返します。レスポンスとして返されるプロパティを次に示します。

項番	名前	型	内容
1	Count	数値	案件の数

レスポンスの構造を次に示します。

項番	名前	出現回数
1	Resources	1
2	Count	1

ステータスコード

項番	ステータスコード	内容
1	200	成功（データが 0 件の場合も成功）
2	400	リクエストパラメタの不正
3	500	内部処理エラー

レスポンス例

レスポンス（XML の場合）

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Resources>
  <Count>25</Count>
</Resources>
```

レスポンス（JSON の場合）

```
{
  ”Count” : ”25”
}
```

11.5.4　案件の取得

指定した ID の案件を取得します。この API を実行すると，CSCIW Java API の CIWServer.getProcessInstance インタフェースが呼び出されます。

メソッドと URL

```
GET /v1/process-instance/<案件ID>
```

<案件 ID>：取得する案件の ID（必須）

クエリパラメタ

項番	名前	型	指定要否	内容
1	userdescription	文字列	省略可	ユーザ記述子を指定します。省略した場合はデフォルト値が指定されます。空文字列は指定できません。

リクエスト例

リクエスト URL

```
GET http://restserver/csciwws/v1/process-instance/5005
```

レスポンス

案件を返します。案件のプロパティを次に示します。

項番	名前	型	内容
1	ClosedDate	日付	案件の終了日時
2	Creator	文字列	案件の投入者
3	Deadline	日付	案件の処理期限の絶対日時
4	ID	数値	案件の ID
5	MovedDate	日付	案件の乗せ替え日時
6	Name	文字列	案件の名称（案件キー）

項番	名前	型	内容
7	Priority	数値	案件の優先度
8	ProcessDefinitionID	数値	案件が所属するビジネスプロセス定義の ID
9	ProcessDefinitionName	文字列	案件が所属するビジネスプロセス定義の名称
10	StartDate	日付	案件の開始日時
11	StateCode	文字列	案件の状態

レスポンスの構造を次に示します。

項番	名前		出現回数
1	ProcessInstance		1
2		ClosedDate	1
3		Creator	1
4		Deadline	1
5		ID	1
6		MovedDate	1
7		Name	1
8		Priority	1
9		ProcessDefinitionID	1
10		ProcessDefinitionName	1
11		StartDate	1
12		StateCode	1

ステータスコード

項番	ステータスコード	内容
1	200	成功
2	400	リクエストパラメタの不正
3	404	データが見つからなかった
4	500	内部処理エラー

レスポンス例

レスポンス（XML の場合）

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ProcessInstance>
  <ClosedDate></ClosedDate>
  <Creator>csciwws</Creator>
  <Deadline></Deadline>
  <ID>5005</ID>
  <MovedDate></MovedDate>
  <Name>案件20160609103302250</Name>
  <Priority>0</Priority>
  <ProcessDefinitionID>3001</ProcessDefinitionID>
  <ProcessDefinitionName>販売業務</ProcessDefinitionName>
  <StartDate>2016-12-06T15:31:34+09:00</StartDate>
  <StateCode>d</StateCode>
</ProcessInstance>
```

レスポンス（JSON の場合）

```
{
  "ClosedDate" : "",
  "Creator" : "csciwws",
  "Deadline" : "",
  "ID" : "5005",
  "MovedDate" : "",
  "Name" : "案件20160609103302250",
  "Priority" : "0",
  "ProcessDefinitionID" : "3001",
  "ProcessDefinitionName" : "販売業務",
  "StartDate" : "2016-12-06T15:31:34+09:00",
  "StateCode" : "d"
}
```

11.5.5　ビジネスプロセス定義名からの案件の取得

指定したビジネスプロセス定義名，案件名，および案件の状態に該当する案件の一覧を取得します。この API を実行すると，BPMN 連携ライブラリ Java API のCIWBPMNLib.getProcessInstancesListByPDName インタフェース，またはCIWBPMNLib.getProcessInstancesListByPIName インタフェースが呼び出されます。

メソッドと URL

```
GET /v1/process-instance/queries
```

クエリパラメタ

項番	名前	型	指定要否	内容
1	userdescription	文字列	省略可	ユーザ記述子を指定します。省略した場合はデフォルト値が指定されます。空文字列は指定できません。

項番	名前	型	指定要否	内容
2	pdname	文字列	必須	ビジネスプロセス定義名を指定します。バージョンが複数登録されている場合，すべてのバージョンを検索します。空文字列は指定できません。
3	piname	文字列	省略可	取得する案件の案件名を指定します。省略した場合，すべての案件名を検索します。空文字は指定できません。また，piname:isnull と同時には指定できません。
4	piname:isnull	文字列	省略可	案件名が未設定の案件を取得する場合に true を指定します。true 以外は指定できません。また，piname と同時には指定できません。
5	statecode	文字列	省略可（クエリパラメタの piname および piname:isnull を省略した場合は必須）	取得する案件の状態を指定します。空文字列は指定できません。複数の状態を指定する場合は，State のコード値を同時に指定します。例えば，「実行中」と「完了」の状態を取得する場合は"do"と指定します。

リクエスト例

リクエスト URL

```
GET http://restserver/csciwws/v1/process-instance/queries?pdname=販売業務&piname=案件1&statecode=do
GET http://restserver/csciwws/v1/process-instance/queries?pdname=販売業務&piname:isnull=true&statecode=do
```

レスポンス

案件一覧を返します。個々の案件のプロパティを次に示します。

項番	名前	型	内容
1	ClosedDate	日付	案件の終了日時
2	Creator	文字列	案件の投入者
3	Deadline	日付	案件の処理期限の絶対日時
4	ID	数値	案件の ID
5	MovedDate	日付	案件の乗せ替え日時
6	Name	文字列	案件の名称（案件キー）
7	Priority	数値	案件の優先度
8	ProcessDefinitionID	数値	案件が所属するビジネスプロセス定義の ID
9	ProcessDefinitionName	文字列	案件が所属するビジネスプロセス定義の名称

項番	名前	型	内容
10	StartDate	日付	案件の開始日時
11	StateCode	文字列	案件の状態

レスポンスの構造を次に示します。

項番	名前			出現回数
1	ProcessInstances			1
2		ProcessInstance		0 または 1 以上
3			ClosedDate	1
4			Creator	1
5			Deadline	1
6			ID	1
7			MovedDate	1
8			Name	1
9			Priority	1
10			ProcessDefinitionID	1
11			ProcessDefinitionName	1
12			StartDate	1
13			StateCode	1

ステータスコード

項番	ステータスコード	内容
1	200	成功（データが 0 件の場合も成功）
2	400	リクエストパラメタの不正
3	500	内部処理エラー

レスポンス例

レスポンス（XML の場合）

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ProcessInstances>
  <ProcessInstance>
    <ClosedDate></ClosedDate>
    <Creator>csciwws</Creator>
    <Deadline></Deadline>
    <ID>5001</ID>
```

```
    <MovedDate></MovedDate>
    <Name>案件1</Name>
    <Priority>0</Priority>
    <ProcessDefinitionID>3001</ProcessDefinitionID>
    <ProcessDefinitionName>販売業務</ProcessDefinitionName>
    <StartDate>2016-12-06T15:31:12+09:00</StartDate>
    <StateCode>d</StateCode>
  </ProcessInstance>
</ProcessInstances>
```

レスポンス（JSON の場合）

```
{
  "ProcessInstance" : [
    {
      "ClosedDate" : "",
      "Creator" : "csciwws",
      "Deadline" : "",
      "ID" : "5001",
      "MovedDate" : "",
      "Name" : "案件1",
      "Priority" : "0",
      "ProcessDefinitionID" : "3001",
      "ProcessDefinitionName" : "販売業務",
      "StartDate" : "2016-12-06T15:31:12+09:00",
      "StateCode" : "d"
    }
  ]
}
```

11.5.6　案件を生成して開始

案件を生成して開始します。この API を実行すると，BPMN 連携ライブラリ Java API の CIWBPMNLib.createAndStartPI インタフェースが呼び出されます。

メソッドと URL

```
POST /v1/process-instance/create-and-start
```

リクエストボディ

項番	名前	型	指定要否	内容
1	UserDescription	文字列	省略可	ユーザ記述子を指定します。省略した場合はデフォルト値が指定されます。空文字列は指定できません。
2	Name	文字列	省略可	案件名を指定します。省略した場合は案件名が設定されません。空文字列は指定できません。

項番	名前	型	指定要否	内容
3	Deadline	日付	省略可	処理期限を指定します。省略した場合は処理期限が設定されません。
4	Priority	数値	省略可	優先度を指定します。省略した場合は優先度が設定されません。
5	ProcessDataList	配列	省略可	プロセスデータ一覧を指定します。省略した場合はプロセスデータが追加されません。
6	ProcessData	オブジェクト	必須	プロセスデータオブジェクトを指定します。次の 2 つの要素（Key および Value）から構成されています。
7	Key	文字列	必須	プロセスデータキー名を指定します。空文字列は指定できません。
8	Value	文字列	省略可	プロセスデータ値を指定します。省略した場合，Key に対応する値は設定されません。空文字列は指定できません。
9	DefinitionName	文字列	必須	ビジネスプロセス定義の名称を指定します。空文字列は指定できません。
10	DefinitionVersion	数値	省略可	ビジネスプロセス定義のバージョンを指定します。省略した場合は指定したビジネスプロセス定義の中で投入できる（状態が活性かつ案件投入期間内である）最新バージョンとなります。

リクエストボディの構造を次に示します。

項番	名前				出現回数
1	Parameter				1
2		UserDescription			0 または 1
3		Name			0 または 1
4		Deadline			0 または 1
5		Priority			0 または 1
6		ProcessDataList			0 または 1
7			ProcessData		1 以上
8				Key	1
9				Value	0 または 1
10		DefinitionName			1

項番	名前		出現回数
11		DefinitionVersion	0 または 1

リクエスト例

リクエスト URL

```
POST http://restserver/csciwws/v1/process-instance/create-and-start
```

リクエストボディ（XML の場合）

```
<?xml version="1.0" encoding="UTF-8" ?>
<Parameter>
  <UserDescription>csciwuser</UserDescription>
  <Name>案件A</Name>
  <Deadline>2017-06-09T10:32:42+09:00</Deadline>
  <Priority>10</Priority>
  <DefinitionName>販売業務</DefinitionName>
  <DefinitionVersion>5</DefinitionVersion>
</Parameter>
```

リクエストボディ（JSON の場合）

```
{
  "UserDescription" : "csciwuser",
  "Name" : "案件A",
  "Deadline" : "2017-06-09T10:32:42+09:00",
  "Priority" : "10",
  "DefinitionName" : "販売業務",
  "DefinitionVersion" : "5"
}
```

レスポンス

作成した案件を返します。案件のプロパティを次に示します。

項番	名前	型	内容
1	ClosedDate	日付	案件の終了日時
2	Creator	文字列	案件の投入者
3	Deadline	日付	案件の処理期限の絶対日時
4	ID	数値	案件の ID
5	MovedDate	日付	案件の乗せ替え日時
6	Name	文字列	案件の名称（案件キー）
7	Priority	数値	案件の優先度

項番	名前	型	内容
8	ProcessDefinitionID	数値	案件が所属するビジネスプロセス定義の ID
9	ProcessDefinitionName	文字列	案件が所属するビジネスプロセス定義の名称
10	StartDate	日付	案件の開始日時
11	StateCode	文字列	案件の状態

レスポンスの構造を次に示します。

項番	名前		出現回数
1	ProcessInstance		1
2		ClosedDate	1
3		Creator	1
4		Deadline	1
5		ID	1
6		MovedDate	1
7		Name	1
8		Priority	1
9		ProcessDefinitionID	1
10		ProcessDefinitionName	1
11		StartDate	1
12		StateCode	1

ステータスコード

項番	ステータスコード	内容
1	201	成功
2	400	リクエストパラメタの不正
3	404	データが見つからなかった
4	500	内部処理エラー

レスポンス例

レスポンス（XML の場合）

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ProcessInstance>
  <ClosedDate></ClosedDate>
  <Creator>csciwuser</Creator>
  <Deadline>2017-06-09T10:32:42+09:00</Deadline>
  <ID>2000</ID>
  <MovedDate></MovedDate>
  <Name>案件A</Name>
  <Priority>10</Priority>
  <ProcessDefinitionID>3001</ProcessDefinitionID>
  <ProcessDefinitionName>販売業務</ProcessDefinitionName>
  <StartDate>2016-12-15T11:50:51+09:00</StartDate>
  <StateCode>d</StateCode>
</ProcessInstance>
```

レスポンス（JSON の場合）

```
{
  "ClosedDate" : "",
  "Creator" : "csciwuser",
  "Deadline" : "2017-06-09T10:32:42+09:00",
  "ID" : "2000",
  "MovedDate" : "",
  "Name" : "案件A",
  "Priority" : "10",
  "ProcessDefinitionID" : "3001",
  "ProcessDefinitionName" : "販売業務",
  "StartDate" : "2016-12-15T11:50:51+09:00",
  "StateCode" : "d"
}
```

11.5.7　案件（メッセージ）を生成して開始

案件（メッセージ）を生成して開始します。この API を実行すると，BPMN 連携ライブラリ Java API の CIWBPMNLib.startMessage インタフェースが呼び出されます。

メソッドと URL

```
POST /v1/process-instance/create-and-start-message
```

リクエストボディ

項番	名前	型	指定要否	内容
1	UserDescription	文字列	省略可	ユーザ記述子を指定します。省略した場合はデフォルト値が指定されます。空文字列は指定できません。

項番	名前	型	指定要否	内容
2	Name	文字列	省略可	案件名を指定します。省略した場合は案件名が設定されません。空文字列は指定できません。
3	Deadline	日付	省略可	処理期限を指定します。省略した場合は処理期限が設定されません。
4	Priority	数値	省略可	優先度を指定します。省略した場合は優先度が設定されません。
5	ProcessDataList	配列	省略可	プロセスデータ一覧を指定します。省略した場合はプロセスデータが設定されません。
6	ProcessData	オブジェクト	必須	プロセスデータオブジェクトを指定します。次の 2 つの要素（Key および Value）から構成されています。
7	Key	文字列	必須	プロセスデータキー名を指定します。空文字列は指定できません。
8	Value	文字列	省略可	プロセスデータ値を指定します。省略した場合はキーに対応する値は未設定となります。空文字列は指定できません。
9	MessageRef	文字列	必須	案件投入する開始（メッセージ）の MessageRef を指定します。空文字列は指定できません。
10	DefinitionName	文字列	必須	ビジネスプロセス定義の名称を指定します。空文字列は指定できません。
11	DefinitionVersion	数値	省略可	ビジネスプロセス定義のバージョンを指定します。省略した場合は指定したビジネスプロセス定義の中で 投入できる（状態が活性かつ

項番	名前	型	指定要否	内容
11	DefinitionVersion	数値	省略可	案件投入期間内である）最新バージョンとなります。

リクエストボディの構造を次に示します。

項番	名前			出現回数
1	Parameter			1
2		UserDescription		0 または 1
3		Name		0 または 1
4		Deadline		0 または 1
5		Priority		0 または 1
6		ProcessDataList		0 または 1
7			ProcessData	1 以上
8			Key	1
9			Value	0 または 1
10		MessageRef		1
11		DefinitionName		1
12		DefinitionVersion		0 または 1

リクエスト例

リクエスト URL

```
POST http://restserver/csciwws/v1/process-instance/create-and-start-message
```

リクエストボディ（XML の場合）

```
<?xml version="1.0" encoding="UTF-8" ?>
<Parameter>
  <UserDescription>csciwuser</UserDescription>
  <Deadline>2017-06-09T10:32:42+09:00</Deadline>
  <Priority>5</Priority>
  <ProcessDataList>
```

```
    <ProcessData>
      <Key>$Sdata1</Key>
      <Value>stringvalue1</Value>
    </ProcessData>
    <ProcessData>
      <Key>$Ndata2</Key>
      <Value>100</Value>
    </ProcessData>
  </ProcessDataList>
  <MessageRef>message1</MessageRef>
  <DefinitionName>販売業務</DefinitionName>
  <DefinitionVersion>3</DefinitionVersion>
</Parameter>
```

リクエストボディ（JSON の場合）

```
{
  "UserDescription" : "csciwuser",
  "Deadline" : "2017-06-09T10:32:42+09:00",
  "Priority" : "5",
  "ProcessDataList" : {
    "ProcessData" : [
      {
        "Key" : "$Sdata1",
        "Value" : "stringvalue1"
      },
      {
        "Key" : "$Ndata2",
        "Value" : "100"
      }
    ]
  },
  "MessageRef" : "message1",
  "DefinitionName" : "販売業務",
  "DefinitionVersion" : "3"
}
```

レスポンス

作成した案件を返します。案件のプロパティを次に示します。

項番	名前	型	内容
1	ClosedDate	日付	案件の終了日時
2	Creator	文字列	案件の投入者
3	Deadline	日付	案件の処理期限の絶対日時
4	ID	数値	案件の ID
5	MovedDate	日付	案件の乗せ替え日時
6	Name	文字列	案件の名称（案件キー）
7	Priority	数値	案件の優先度

項番	名前	型	内容
8	ProcessDefinitionID	数値	案件が所属するビジネスプロセス定義の ID
9	ProcessDefinitionName	文字列	案件が所属するビジネスプロセス定義の名称
10	StartDate	日付	案件の開始日時
11	StateCode	文字列	案件の状態

レスポンスの構造を次に示します。

項番	名前		出現回数
1	ProcessInstance		1
2		ClosedDate	1
3		Creator	1
4		Deadline	1
5		ID	1
6		MovedDate	1
7		Name	1
8		Priority	1
9		ProcessDefinitionID	1
10		ProcessDefinitionName	1
11		StartDate	1
12		StateCode	1

ステータスコード

項番	ステータスコード	内容
1	201	成功
2	400	リクエストパラメタの不正
3	404	データが見つからなかった
4	500	内部処理エラー

レスポンス例

レスポンス（XML の場合）

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ProcessInstance>
  <ClosedDate></ClosedDate>
  <Creator>csciwuser</Creator>
  <Deadline>2017-06-09T10:32:42+09:00</Deadline>
  <ID>2001</ID>
  <MovedDate></MovedDate>
  <Name>案件A</Name>
  <Priority>5</Priority>
  <ProcessDefinitionID>3001</ProcessDefinitionID>
  <ProcessDefinitionName>販売業務</ProcessDefinitionName>
  <StartDate>2016-12-15T12:11:45+09:00</StartDate>
  <StateCode>d</StateCode>
</ProcessInstance>
```

レスポンス（JSON の場合）

```
{
  "ClosedDate" : "",
  "Creator" : "csciwuser",
  "Deadline" : "2017-06-09T10:32:42+09:00",
  "ID" : "2001",
  "MovedDate" : "",
  "Name" : "案件A",
  "Priority" : "5",
  "ProcessDefinitionID" : "3001",
  "ProcessDefinitionName" : "販売業務",
  "StartDate" : "2016-12-15T12:11:45+09:00",
  "StateCode" : "d"
}
```

11.5.8　案件（タイマー）を生成して開始

指定された作業定義名の開始（タイマー）から，案件を生成して開始します。この API を実行すると，BPMN 連携ライブラリ Java API の CIWBPMNLib.createAndStartPIForTimer インタフェースが呼び出されます。

メソッドと URL

```
POST /v1/process-instance/create-and-start-timer
```

リクエストボディ

項番	名前	型	指定要否	内容
1	UserDescription	文字列	省略可	ユーザ記述子を指定します。省略した場合はデフォルト値が指定されます。空文字列は指定できません。

項番	名前	型	指定要否	内容
2	ProcessDefinitionName	文字列	必須	ビジネスプロセス定義の名称を指定します。空文字列は指定できません。
3	ProcessDefinitionVersion	数値	省略可	ビジネスプロセス定義のバージョンを指定します。省略した場合は指定したビジネスプロセス定義の中で投入可能（状態が活性かつ案件投入期間内）な最新バージョンになります。
4	WorkDefinitionName	文字列	必須	開始（タイマー）の作業定義名を指定します。空文字列は指定できません。

リクエストボディの構造を次に示します。

項番	名前		出現回数
1	Parameter		1
2		UserDescription	0 または 1
3		ProcessDefinitionName	1
4		ProcessDefinitionVersion	0 または 1
5		WorkDefinitionName	1

リクエスト例

リクエスト URL

```
POST http://restserver/csciwws/v1/process-instance/create-and-start-timer
```

リクエストボディ（XML の場合）

```
<?xml version="1.0" encoding="UTF-8" ?>
<Parameter>
  <UserDescription>csciwuser</UserDescription>
  <ProcessDefinitionName>販売業務</ProcessDefinitionName>
  <ProcessDefinitionVersion>5</ProcessDefinitionVersion>
  <WorkDefinitionName>Timer_Timer1</WorkDefinitionName>
</Parameter>
```

リクエストボディ（JSON の場合）

```
{
  "UserDescription" : "csciwuser",
  "ProcessDefinitionName" : "販売業務",
  "ProcessDefinitionVersion" : "5",
  "WorkDefinitionName" : "Timer_Timer1"
}
```

レスポンス

作成した案件を返します。案件のプロパティを次に示します。

項番	名前	型	内容
1	ClosedDate	日付	案件の終了日時
2	Creator	文字列	案件の投入者
3	Deadline	日付	案件の処理期限の絶対日時
4	ID	数値	案件の ID
5	MovedDate	日付	案件の乗せ替え日時
6	Name	文字列	案件の名称（案件キー）
7	Priority	数値	案件の優先度
8	ProcessDefinitionID	数値	案件が所属するビジネスプロセス定義の ID
9	ProcessDefinitionName	文字列	案件が所属するビジネスプロセス定義の名称
10	StartDate	日付	案件の開始日時
11	StateCode	文字列	案件の状態

レスポンスの構造を次に示します。

項番	名前		出現回数
1	ProcessInstance		1
2		ClosedDate	1
3		Creator	1
4		Deadline	1
5		ID	1
6		MovedDate	1
7		Name	1
8		Priority	1
9		ProcessDefinitionID	1
10		ProcessDefinitionName	1
11		StartDate	1
12		StateCode	1

ステータスコード

項番	ステータスコード	内容
1	201	成功
2	400	リクエストパラメタの不正
3	404	データが見つからなかった
4	500	内部処理エラー

レスポンス例

レスポンス（XML の場合）

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ProcessInstance>
  <ClosedDate></ClosedDate>
  <Creator>csciwuser</Creator>
  <Deadline>2017-06-09T10:32:42+09:00</Deadline>
  <ID>2000</ID>
  <MovedDate></MovedDate>
  <Name>OrderProcess_0001_20170601120000</Name>
  <Priority>10</Priority>
  <ProcessDefinitionID>3001</ProcessDefinitionID>
  <ProcessDefinitionName>販売業務</ProcessDefinitionName>
  <StartDate>2016-12-15T11:50:51+09:00</StartDate>
  <StateCode>d</StateCode>
</ProcessInstance>
```

レスポンス（JSON の場合）

```
{
  "ClosedDate" : "",
  "Creator" : "csciwuser",
  "Deadline" : "2017-06-09T10:32:42+09:00",
  "ID" : "2000",
  "MovedDate" : "",
  "Name" : "OrderProcess_0001_20170601120000",
  "Priority" : "10",
  "ProcessDefinitionID" : "3001",
  "ProcessDefinitionName" : "販売業務",
  "StartDate" : "2016-12-15T11:50:51+09:00",
  "StateCode" : "d"
}
```

11.5.9　案件の削除

指定した ID の案件を削除します。この API を実行すると，BPMN 連携ライブラリ Java API の CIWBPMNLib.deletePI インタフェースが呼び出されます。案件が削除済みだった（削除対象の案件が存在

しなかった）場合は，リクエストは成功します（何もしないでステータスコード 204 を返します）。詳細については，「5.3　Java API 利用時の注意事項」の「べき等性について」の説明を参照してください。

メソッドと URL

```
DELETE /v1/process-instance/<案件ID>
```

<案件 ID>：削除する案件の ID（必須）

クエリパラメタ

項番	名前	型	指定要否	内容
1	userdescription	文字列	省略可	ユーザ記述子を指定します。省略した場合はデフォルト値が指定されます。空文字列は指定できません。

リクエスト例

リクエスト URL

```
DELETE http://restserver/csciwws/v1/process-instance/5005
```

レスポンス

レスポンスボディは空になります。

ステータスコード

項番	ステータスコード	内容
1	204	成功
2	400	リクエストパラメタの不正
3	500	内部処理エラー

レスポンス例

レスポンスボディは空になります。

11.5.10　案件の強制終了

指定した ID の案件を強制停止します。この API を実行すると，BPMN 連携ライブラリ Java API の CIWBPMNLib.terminatePI インタフェースが呼び出されます。案件が強制終了済みだった場合は，リクエス

トは成功します（何もしないでステータスコード 200 を返します）。詳細については，「5.3　Java API 利用時の注意事項」の「べき等性について」の説明を参照してください。

メソッドと URL

```
PUT /v1/process-instance/<案件ID>/terminate
```

<案件 ID>：強制停止する案件の ID（必須）

リクエストボディ

項番	名前	型	指定要否	内容
1	UserDescription	文字列	省略可	ユーザ記述子を指定します。省略した場合はデフォルト値が指定されます。空文字列は指定できません。

リクエストボディの構造を次に示します。UserDescription を省略する場合は，リクエストボディを省略できます。リクエストボディを省略するときは，HTTP ヘッダの`Content-Type` を指定しないでください。

項番	名前		出現回数
1	Parameter		1
2		UserDescription	0 または 1

リクエスト例

リクエスト URL

```
PUT http://restserver/csciwws/v1/process-instance/2034/terminate
```

レスポンス

強制停止した案件を返します。案件のプロパティを次に示します。

項番	名前	型	内容
1	ClosedDate	日付	案件の終了日時
2	Creator	文字列	案件の投入者
3	Deadline	日付	案件の処理期限の絶対日時
4	ID	数値	案件の ID
5	MovedDate	日付	案件の乗せ替え日時
6	Name	文字列	案件の名称（案件キー）

項番	名前	型	内容
7	Priority	数値	案件の優先度
8	ProcessDefinitionID	数値	案件が所属するビジネスプロセス定義の ID
9	ProcessDefinitionName	文字列	案件が所属するビジネスプロセス定義の名称
10	StartDate	日付	案件の開始日時
11	StateCode	文字列	案件の状態

レスポンスの構造を次に示します。

項番	名前		出現回数
1	ProcessInstance		1
2		ClosedDate	1
3		Creator	1
4		Deadline	1
5		ID	1
6		MovedDate	1
7		Name	1
8		Priority	1
9		ProcessDefinitionID	1
10		ProcessDefinitionName	1
11		StartDate	1
12		StateCode	1

ステータスコード

項番	ステータスコード	内容
1	200	成功
2	400	リクエストパラメタの不正
3	404	データが見つからなかった
4	500	内部処理エラー

レスポンス例

レスポンス（XML の場合）

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ProcessInstance>
  <ClosedDate>2016-12-15T12:16:19+09:00</ClosedDate>
  <Creator>csciwuser</Creator>
  <Deadline>2017-01-01T00:00:00+09:00</Deadline>
  <ID>2034</ID>
  <MovedDate></MovedDate>
  <Name>案件A</Name>
  <Priority>10</Priority>
  <ProcessDefinitionID>3002</ProcessDefinitionID>
  <ProcessDefinitionName>販売業務</ProcessDefinitionName>
  <StartDate>2016-12-15T11:50:51+09:00</StartDate>
  <StateCode>u</StateCode>
</ProcessInstance>
```

レスポンス（JSON の場合）

```
{
  "ClosedDate" : "2016-12-15T12:16:19+09:00",
  "Creator" : "csciwuser",
  "Deadline" : "2017-01-01T00:00:00+09:00",
  "ID" : "2034",
  "MovedDate" : "",
  "Name" : "案件A",
  "Priority" : "10",
  "ProcessDefinitionID" : "3002",
  "ProcessDefinitionName" : "販売業務",
  "StartDate" : "2016-12-15T11:50:51+09:00",
  "StateCode" : "u"
}
```

11.5.11　親案件の取得

指定した案件の親案件を取得します。具体的には，指定した案件を投入したコールアクティビティの案件を取得します。この API を実行すると，BPMN 連携ライブラリ Java API の CIWBPMNLib.getCallActivityParentPI インタフェースが呼び出されます。

メソッドと URL

```
GET /v1/process-instance/<案件ID>/call-activity-parent/process-instance
```

<案件 ID>：親案件を取得する案件の ID（必須）

クエリパラメタ

項番	名前	型	指定要否	内容
1	userdescription	文字列	省略可	ユーザ記述子を指定します。省略した場合はデフォルト値が指定されます。空文字列は指定できません。

リクエスト例

リクエスト URL

```
GET http://restserver/csciwws/v1/process-instance/5005/call-activity-parent/process-instance
```

レスポンス

親案件を返します。親案件のプロパティを次に示します。

項番	名前	型	内容
1	ClosedDate	日付	親案件の終了日時
2	Creator	文字列	親案件の投入者
3	Deadline	日付	親案件の処理期限の絶対日時
4	ID	数値	親案件の ID
5	MovedDate	日付	親案件の乗せ替え日時
6	Name	文字列	親案件の名称（案件キー）
7	Priority	数値	親案件の優先度
8	ProcessDefinitionID	数値	親案件が所属するビジネスプロセス定義の ID
9	ProcessDefinitionName	文字列	親案件が所属するビジネスプロセス定義の名称
10	StartDate	日付	親案件の開始日時
11	StateCode	文字列	親案件の状態

レスポンスの構造を次に示します。

項番	名前		出現回数
1	ProcessInstance		1
2		ClosedDate	1
3		Creator	1
4		Deadline	1
5		ID	1
6		MovedDate	1
7		Name	1
8		Priority	1
9		ProcessDefinitionID	1
10		ProcessDefinitionName	1

項番	名前		出現回数
11		StartDate	1
12		StateCode	1

ステータスコード

項番	ステータスコード	内容
1	200	成功
2	400	リクエストパラメタの不正
3	404	データが見つからなかった
4	500	内部処理エラー

レスポンス例

レスポンス（XML の場合）

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ProcessInstance>
  <ClosedDate></ClosedDate>
  <Creator>csciwws</Creator>
  <Deadline></Deadline>
  <ID>5</ID>
  <MovedDate></MovedDate>
  <Name></Name>
  <Priority>0</Priority>
  <ProcessDefinitionID>3001</ProcessDefinitionID>
  <ProcessDefinitionName>販売業務</ProcessDefinitionName>
  <StartDate>2016-12-06T15:31:34+09:00</StartDate>
  <StateCode>d</StateCode>
</ProcessInstance>
```

レスポンス（JSON の場合）

```
{
  "ClosedDate" : "",
  "Creator" : "csciwws",
  "Deadline" : "",
  "ID" : "5",
  "MovedDate" : "",
  "Name" : "",
  "Priority" : "0",
  "ProcessDefinitionID" : "3001",
  "ProcessDefinitionName" : "販売業務",
  "StartDate" : "2016-12-06T15:31:34+09:00",
  "StateCode" : "d"
}
```

11.5.12　コールアクティビティの取得

案件を投入した作業を取得します。具体的には，指定した案件を投入したコールアクティビティを取得します。この API を実行すると，BPMN 連携ライブラリ Java API のCIWBPMNLib.getCallActivityParentWIインタフェースが呼び出されます。

メソッドと URL

```
GET /v1/process-instance/<案件ID>/call-activity-parent/work-item
```

<案件 ID>：コールアクティビティが投入した案件 ID（必須）

クエリパラメタ

項番	名前	型	指定要否	内容
1	userdescription	文字列	省略可	ユーザ記述子を指定します。省略した場合はデフォルト値が指定されます。空文字列は指定できません。

リクエスト例

リクエスト URL

```
GET http://restserver/csciwws/v1/process-instance/5004/call-activity-parent/work-item
```

レスポンス

コールアクティビティの作業を返します。レスポンスとして返される作業のプロパティを次に示します。

項番	名前	型	内容
1	ActivityInstanceID	数値	作業が所属する業務ステップの ID
2	ClosedDate	日付	作業の終了日時
3	CreationDate	日付	作業の発生日時
4	Deadline	日付	作業の処理期限の絶対日時
5	ID	数値	作業の ID
6	Name	文字列	作業の名称（作業キー）
7	Participant	文字列	作業の作業者 ID
8	Priority	数値	作業の優先度
9	ProcessDefinitionID	数値	作業が所属するビジネスプロセス定義の ID
10	ProcessInstanceID	数値	作業が所属する案件の ID

項番	名前	型	内容
11	ProcessInstanceName	文字列	作業の案件名（案件キー）
12	StartDate	日付	作業の開始日時
13	StateCode	文字列	作業の状態
14	WorkDefinitionID	数値	作業定義の ID
15	WorkDefinitionName	文字列	作業定義の名称
16	WorkTypeCode	文字列	作業の種類

レスポンスの構造を次に示します。

項番	名前		出現回数
1	WorkItem		1
2		ActivityInstanceID	1
3		ClosedDate	1
4		CreationDate	1
5		Deadline	1
6		ID	1
7		Name	1
8		Participant	1
9		Priority	1
10		ProcessDefinitionID	1
11		ProcessInstanceID	1
12		ProcessInstanceName	1
13		StartDate	1
14		StateCode	1
15		WorkDefinitionID	1
16		WorkDefinitionName	1
17		WorkTypeCode	1

ステータスコード

項番	ステータスコード	内容
1	200	成功
2	400	リクエストパラメタの不正

項番	ステータスコード	内容
3	404	データが見つからなかった
4	500	内部処理エラー

レスポンス例

レスポンス（XML の場合）

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<WorkItem>
  <ActivityInstanceID>10005</ActivityInstanceID>
  <ClosedDate></ClosedDate>
  <CreationDate>2016-12-09T15:00:40+09:00</CreationDate>
  <Deadline></Deadline>
  <ID>10005</ID>
  <Name></Name>
  <Participant>IWCALL_GLOBALBP</Participant>
  <Priority></Priority>
  <ProcessDefinitionID>3001</ProcessDefinitionID>
  <ProcessInstanceID>5003</ProcessInstanceID>
  <ProcessInstanceName></ProcessInstanceName>
  <StartDate></StartDate>
  <StateCode>j</StateCode>
  <WorkDefinitionID>3003</WorkDefinitionID>
  <WorkDefinitionName>CheckCredit_callactivity1</WorkDefinitionName>
  <WorkTypeCode>0</WorkTypeCode>
</WorkItem>
```

レスポンス（JSON の場合）

```
{
  "ActivityInstanceID" : "10005",
  "ClosedDate" : "",
  "CreationDate" : "2016-12-09T15:00:40+09:00",
  "Deadline" : "",
  "ID" : "10005",
  "Name" : "",
  "Participant" : "IWCALL_GLOBALBP",
  "Priority" : "",
  "ProcessDefinitionID" : "3001",
  "ProcessInstanceID" : "5003",
  "ProcessInstanceName" : "",
  "StartDate" : "",
  "StateCode" : "j",
  "WorkDefinitionID" : "3003",
  "WorkDefinitionName" : "CheckCredit_callactivity1",
  "WorkTypeCode" : "0"
}
```

11.5.13　業務ステップの一覧取得

指定したフィルター条件を満たす，業務ステップの一覧を取得します。この API を実行すると，CSCIW Java API の`CIWServer.getActivityInstancesList` インタフェースを呼び出します。

メソッドと URL

```
GET /v1/activity-instance
```

クエリパラメタ

項番	名前	型	指定要否	内容
1	userdescription	文字列	省略可	ユーザ記述子を指定します。省略した場合はデフォルト値が指定されます。空文字列は指定できません。
2	filter	文字列	省略可	業務ステップ一覧を取得する場合のフィルター条件を指定します。省略した場合はフィルター条件を指定しません。空文字列は指定できません。
3	sort	文字列	省略可	取得した業務ステップ一覧をソートする場合の条件を指定します。省略した場合はソート条件を指定しません。空文字列は指定できません。
4	offset	数値	省略可	業務ステップ一覧を取得する場合のオフセットを指定します。先頭は 0 になります。省略した場合は 0 が指定されます。0 未満の値は指定できません。
5	maxcount	数値	省略可	業務ステップ一覧を取得する場合の最大取得数を指定します。省略した場合はデフォルト値が指定されます。すべてを取得する場合は，-1 を指定します。-1 未満の値は指定できません。

リクエスト例

リクエスト URL

```
GET http://restserver/csciwws/v1/activity-instance?filter=ProcessInstanceID%3D5002
```

レスポンス

業務ステップ一覧を返します。個々の業務ステップのプロパティを次に示します。

項番	名前	型	内容
1	ActivityDefinitionID	数値	業務ステップ定義の ID
2	ActivityDefinitionName	文字列	業務ステップ定義の名称
3	ActivityTypeCode	文字列	業務ステップの種類
4	ClosedDate	日付	業務ステップの終了日時
5	Deadline	日付	業務ステップの処理期限の絶対日時
6	ID	数値	業務ステップの ID
7	Name	文字列	業務ステップの名称（業務ステップキー）
8	Priority	数値	業務ステップの優先度
9	ProcessDefinitionID	数値	業務ステップが所属するビジネスプロセス定義の ID
10	ProcessInstanceID	数値	業務ステップが所属する案件の ID
11	ProcessInstanceName	文字列	業務ステップの案件名（案件キー）
12	StartDate	日付	業務ステップの開始日時
13	StateCode	文字列	業務ステップの状態

レスポンスの構造を次に示します。

項番	名前			出現回数
1	ActivityInstances			1
2		ActivityInstance		0 または 1 以上
3			ActivityDefinitionID	1
4			ActivityDefinitionName	1
5			ActivityTypeCode	1

項番	名前			出現回数
6			ClosedDate	1
7			Deadline	1
8			ID	1
9			Name	1
10			Priority	1
11			ProcessDefinitionID	1
12			ProcessInstanceID	1
13			ProcessInstanceName	1
14			StartDate	1
15			StateCode	1

ステータスコード

項番	ステータスコード	内容
1	200	成功（データが 0 件の場合も成功）
2	400	リクエストパラメタの不正
3	500	内部処理エラー

レスポンス例

レスポンス（XML の場合）

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ActivityInstances>
  <ActivityInstance>
    <ActivityDefinitionID>3003</ActivityDefinitionID>
    <ActivityDefinitionName>QuotationHandling_UTask1</ActivityDefinitionName>
    <ActivityTypeCode>0</ActivityTypeCode>
    <ClosedDate>2016-12-09T15:00:42+09:00</ClosedDate>
    <Deadline></Deadline>
    <ID>10001</ID>
    <Name></Name>
    <Priority></Priority>
    <ProcessDefinitionID>3001</ProcessDefinitionID>
    <ProcessInstanceID>5002</ProcessInstanceID>
    <ProcessInstanceName>案件20160609103241625</ProcessInstanceName>
    <StartDate>2016-12-09T15:00:41+09:00</StartDate>
    <StateCode>t</StateCode>
  </ActivityInstance>
  <ActivityInstance>
    <ActivityDefinitionID>3001</ActivityDefinitionID>
    <ActivityDefinitionName>VerifyCreditHistory_UTask1</ActivityDefinitionName>
```

```
    <ActivityTypeCode>0</ActivityTypeCode>
    <ClosedDate></ClosedDate>
    <Deadline></Deadline>
    <ID>10002</ID>
    <Name></Name>
    <Priority></Priority>
    <ProcessDefinitionID>3001</ProcessDefinitionID>
    <ProcessInstanceID>5002</ProcessInstanceID>
    <ProcessInstanceName>案件20160609103241625</ProcessInstanceName>
    <StartDate>2016-12-09T15:00:42+09:00</StartDate>
    <StateCode>d</StateCode>
  </ActivityInstance>
</ActivityInstances>
```

レスポンス（JSON の場合）

```
{
  "ActivityInstance" : [
    {
      "ActivityDefinitionID" : "3003",
      "ActivityDefinitionName" : "QuotationHandling_UTask1",
      "ActivityTypeCode" : "0",
      "ClosedDate" : "2016-12-09T15:00:42+09:00",
      "Deadline" : "",
      "ID" : "10001",
      "Name" : "",
      "Priority" : "",
      "ProcessDefinitionID" : "3001",
      "ProcessInstanceID" : "5002",
      "ProcessInstanceName" : "案件20160609103241625",
      "StartDate" : "2016-12-09T15:00:41+09:00",
      "StateCode" : "t"
    },
    {
      "ActivityDefinitionID" : "3001",
      "ActivityDefinitionName" : "VerifyCreditHistory_UTask1",
      "ActivityTypeCode" : "0",
      "ClosedDate" : "",
      "Deadline" : "",
      "ID" : "10002",
      "Name" : "",
      "Priority" : "",
      "ProcessDefinitionID" : "3001",
      "ProcessInstanceID" : "5002",
      "ProcessInstanceName" : "案件20160609103241625",
      "StartDate" : "2016-12-09T15:00:42+09:00",
      "StateCode" : "d"
    }
  ]
}
```

11.5.14　業務ステップ数の取得

指定した条件を満たす業務ステップの数を取得します。この API を実行すると，CSCIW Java API の CIWServer.getListCount インタフェースが呼び出されます。

メソッドと URL

```
GET /v1/activity-instance/count
```

クエリパラメタ

項番	名前	型	指定要否	内容
1	userdescription	文字列	省略可	ユーザ記述子を指定します。省略した場合はデフォルト値が指定されます。空文字列は指定できません。
2	filter	文字列	省略可	業務ステップ数を取得する場合のフィルター条件を指定します。省略した場合はフィルター条件は指定されません。空文字列は指定できません。

リクエスト例

リクエスト URL

```
GET http://restserver/csciwws/v1/activity-instance/count?filter=ProcessInstanceID%3D100
```

レスポンス

業務ステップ数を返します。レスポンスとして返されるプロパティを次に示します。

項番	名前	型	内容
1	Count	数値	業務ステップの数

レスポンスの構造を次に示します。

項番	名前	出現回数
1	Resources	1
2	Count	1

ステータスコード

項番	ステータスコード	内容
1	200	成功（データが 0 件の場合も成功）
2	400	リクエストパラメタの不正

項番	ステータスコード	内容
3	500	内部処理エラー

レスポンス例

レスポンス（XML の場合）

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Resources>
  <Count>23</Count>
</Resources>
```

レスポンス（JSON の場合）

```
{
  "Count" : "23"
}
```

11.5.15　業務ステップの取得

指定した ID の業務ステップを取得します。この API を実行すると，CSCIW Java API の CIWServer.getActivityInstance インタフェースが呼び出されます。

メソッドと URL

```
GET /v1/activity-instance/<案件ID>/<業務ステップID>
```

<案件 ID>：取得する業務ステップの案件 ID（必須）

<業務ステップ ID>：取得する業務ステップの ID（必須）

クエリパラメタ

項番	名前	型	指定要否	内容
1	userdescription	文字列	省略可	ユーザ記述子を指定します。省略した場合はデフォルト値が指定されます。空文字列は指定できません。

リクエスト例

リクエスト URL

```
GET http://restserver/csciwws/v1/activity-instance/5002/10002
```

レスポンス

業務ステップを返します。業務ステップのプロパティを次に示します。

項番	名前	型	内容
1	ActivityDefinitionID	数値	業務ステップ定義の ID
2	ActivityDefinitionName	文字列	業務ステップ定義の名称
3	ActivityTypeCode	文字列	業務ステップの種類
4	ClosedDate	日付	業務ステップの終了日時
5	Deadline	日付	業務ステップの処理期限の絶対日時
6	ID	数値	業務ステップの ID
7	Name	文字列	業務ステップの名称（業務ステップキー）
8	Priority	数値	業務ステップの優先度
9	ProcessDefinitionID	数値	業務ステップが所属するビジネスプロセス定義の ID
10	ProcessInstanceID	数値	業務ステップが所属する案件の ID
11	ProcessInstanceName	文字列	業務ステップの案件名（案件キー）
12	StartDate	日付	業務ステップの開始日時
13	StateCode	文字列	業務ステップの状態

レスポンスの構造を次に示します。

項番	名前		出現回数
1	ActivityInstance		1
2		ActivityDefinitionID	1
3		ActivityDefinitionName	1
4		ActivityTypeCode	1
5		ClosedDate	1
6		Deadline	1
7		ID	1
8		Name	1

項番	名前		出現回数
9		Priority	1
10		ProcessDefinitionID	1
11		ProcessInstanceID	1
12		ProcessInstanceName	1
13		StartDate	1
14		StateCode	1

ステータスコード

項番	ステータスコード	内容
1	200	成功
2	400	リクエストパラメタの不正
3	404	データが見つからなかった
4	500	内部処理エラー

レスポンス例

レスポンス（XML の場合）

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ActivityInstance>
  <ActivityDefinitionID>3001</ActivityDefinitionID>
  <ActivityDefinitionName>QuotationHandling_UTask1</ActivityDefinitionName>
  <ActivityTypeCode>0</ActivityTypeCode>
  <ClosedDate></ClosedDate>
  <Deadline></Deadline>
  <ID>10002</ID>
  <Name></Name>
  <Priority></Priority>
  <ProcessDefinitionID>3001</ProcessDefinitionID>
  <ProcessInstanceID>5002</ProcessInstanceID>
  <ProcessInstanceName>案件20160609103241625</ProcessInstanceName>
  <StartDate>2016-12-09T15:00:42+09:00</StartDate>
  <StateCode>d</StateCode>
<ActivityInstance>
```

レスポンス（JSON の場合）

```
{
  "ActivityDefinitionID" : "3001",
  "ActivityDefinitionName" : "QuotationHandling_UTask1",
  "ActivityTypeCode" : "0",
  "ClosedDate" : "",
  "Deadline" : "",
```

```
  "ID" : "10002",
  "Name" : "",
  "Priority" : "",
  "ProcessDefinitionID" : "3001",
  "ProcessInstanceID" : "5002",
  "ProcessInstanceName" : "案件20160609103241625",
  "StartDate" : "2016-12-09T15:00:42+09:00",
  "StateCode" : "d"
}
```

11.5.16　業務ステップの差し戻しまたは引き戻し

指定した ID の業務ステップを差し戻します（引き戻します）。この API を実行すると，BPMN 連携ライブラリ Java API の`CIWBPMNLib.makeBackwardTransitionAI` インタフェースが呼び出されます。業務ステップが差し戻し（引き戻し）済みだった場合は，リクエストは成功します（何もしないでステータスコード 200 を返します）。詳細については，「5.3　Java API 利用時の注意事項」の「べき等性について」の説明を参照してください。

メソッドと URL

```
PUT /v1/activity-instance/<案件ID>/<業務ステップID>/make-backward-transition
```

<案件 ID>：差し戻す業務ステップの案件 ID（必須）

<業務ステップ ID>：差し戻す業務ステップの ID（必須）

リクエストボディ

項番	名前	型	指定要否	内容
1	UserDescription	文字列	省略可	ユーザ記述子を指定します。省略した場合はデフォルト値が指定されます。空文字列は指定できません。
2	ProcessDataList	配列	省略可	プロセスデータ一覧を指定します。省略した場合はプロセスデータを更新しません。
3	ProcessData	オブジェクト	必須	プロセスデータオブジェクトを指定します。次の 2 つの要素（Key および Value）から構成されています。

項番	名前	型	指定要否	内容
4	Key	文字列	必須	プロセスデータキー名を指定します。空文字列は定できません。
5	Value	文字列	省略可	プロセスデータ値を指定します。省略した場合，Key に対応する値は未設定となります。空文字列は指定できません。
6	ActivityDefinitionName	文字列	必須	差し戻し先または引き戻し先の業務 ステップ定義名を指定します。空文字列は指定できません。

リクエストボディの構造を次に示します。

項番	名前				出現回数
1	Parameter				1
2		UserDescription			0 または 1
3		ProcessDataList			0 または 1
4			ProcessData		1 以上
5				Key	1
6				Value	0 または 1
7		ActivityDefinitionName			1

リクエスト例

リクエスト URL

```
PUT http://restserver/csciwws/v1/activity-instance/5002/10002/make-backward-transition
```

リクエストボディ（XML の場合）

```
<?xml version="1.0" encoding="UTF-8" ?>
<Parameter>
  <ProcessDataList>
    <ProcessData>
      <Key>$Sdata1</Key>
      <Value>stringvalue1</Value>
    </ProcessData>
    <ProcessData>
      <Key>$Ndata2</Key>
```

```
      <Value>100</Value>
    </ProcessData>
  </ProcessDataList>
  <ActivityDefinitionName>QuotationHandling_UTask1</ActivityDefinitionName>
</Parameter>
```

リクエストボディ（JSON の場合）

```
{
  "ProcessDataList" : {
    "ProcessData" : [
      {
        "Key" : "$SData1",
        "Value" : "stringvalue1"
      },
      {
        "Key" : "$Ndata2",
        "Value" : "100"
      }
    ]
  },
  "ActivityDefinitionName" : "QuotationHandling_UTask1"
}
```

レスポンス

差し戻した業務ステップを返します。業務ステップのプロパティを次に示します。

項番	名前	型	内容
1	ActivityDefinitionID	数値	業務ステップ定義の ID
2	ActivityDefinitionName	文字列	業務ステップ定義の名称
3	ActivityTypeCode	文字列	業務ステップの種類
4	ClosedDate	日付	業務ステップの終了日時
5	Deadline	日付	業務ステップの処理期限の絶対日時
6	ID	数値	業務ステップの ID
7	Name	文字列	業務ステップの名称（業務ステップキー）
8	Priority	数値	業務ステップの優先度
9	ProcessDefinitionID	数値	業務ステップが所属するビジネスプロセス定義の ID
10	ProcessInstanceID	数値	業務ステップが所属する案件の ID
11	ProcessInstanceName	文字列	業務ステップの案件名（案件キー）

項番	名前	型	内容
12	StartDate	日付	業務ステップの開始日時
13	StateCode	文字列	業務ステップの状態

レスポンスの構造を次に示します。

項番	名前		出現回数
1	ActivityInstance		1
2		ActivityDefinitionID	1
3		ActivityDefinitionName	1
4		ActivityTypeCode	1
5		ClosedDate	1
6		Deadline	1
7		ID	1
8		Name	1
9		Priority	1
10		ProcessDefinitionID	1
11		ProcessInstanceID	1
12		ProcessInstanceName	1
13		StartDate	1
14		StateCode	1

ステータスコード

項番	ステータスコード	内容
1	200	成功
2	400	リクエストパラメタの不正
3	404	データが見つからなかった
4	500	内部処理エラー

レスポンス例

レスポンス（XML の場合）

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ActivityInstance>
  <ActivityDefinitionID>3001</ActivityDefinitionID>
```

```
  <ActivityDefinitionName>QuotationHandling_UTask1</ActivityDefinitionName>
  <ActivityTypeCode>0</ActivityTypeCode>
  <ClosedDate>2016-12-09T16:00:42+09:00</ClosedDate>
  <Deadline></Deadline>
  <ID>10002</ID>
  <Name></Name>
  <Priority></Priority>
  <ProcessDefinitionID>3002</ProcessDefinitionID>
  <ProcessInstanceID>5002</ProcessInstanceID>
  <ProcessInstanceName></ProcessInstanceName>
  <StartDate>2016-12-09T15:00:42+09:00</StartDate>
  <StateCode>u</StateCode>
</ActivityInstance>
```

レスポンス（JSON の場合）

```
{
  "ActivityDefinitionID" : "3001",
  "ActivityDefinitionName" : "QuotationHandling_UTask1",
  "ActivityTypeCode" : "0",
  "ClosedDate" : "2016-12-09T16:00:42+09:00",
  "Deadline" : "",
  "ID" : "10002",
  "Name" : "",
  "Priority" : "",
  "ProcessDefinitionID" : "3002",
  "ProcessInstanceID" : "5002",
  "ProcessInstanceName" : "",
  "StartDate" : "2016-12-09T15:00:42+09:00",
  "StateCode" : "u"
}
```

11.5.17　業務ステップの強制遷移

指定した業務ステップを任意の業務ステップ定義に強制遷移します。この API を実行すると，BPMN 連携ライブラリ Java API の**CIWBPMNLib.adhocCreateAndMakeTransition** インタフェースが呼び出されます。業務ステップが遷移済みだった場合は，リクエストは成功します（何もしないでステータスコード 200 を返します）。詳細については，「5.3　Java API 利用時の注意事項」の「べき等性について」の説明を参照してください。

メソッドと URL

```
PUT /v1/activity-instance/<案件ID>/<業務ステップID>/adhoc-create-and-make-transition
```

<案件 ID>：遷移する業務ステップの案件 ID（必須）

<業務ステップ ID>：遷移する業務ステップの ID（必須）

リクエストボディ

項番	名前	型	指定要否	内容
1	UserDescription	文字列	省略可	ユーザ記述子を指定します。省略した場合はデフォルト値が指定されます。空文字列は指定できません。
2	ProcessDataList	配列	省略可	プロセスデータ一覧を指定します。省略した場合はプロセスデータは更新されません。
3	ProcessData	オブジェクト	必須	プロセスデータオブジェクト。次の 2 つの 要素（Key および Value）から構成されています。
4	Key	文字列	必須	プロセスデータキー名を指定します。空文字列は指定できません。
5	Value	文字列	省略可	プロセスデータ値を指定します。省略した場合，Key に対応する値は設定されません。空文字は指定できません。
6	ActivityDefinitionName	文字列	必須	遷移先の業務 ステップ定義名を指定します。空文字列は指定できません。

リクエストボディの構造を次に示します。

項番	名前				出現回数
1	Parameter				1
2		UserDescription			0 または 1
3		ProcessDataList			0 または 1
4			ProcessData		1 以上
5				Key	1
6				Value	0 または 1
7		ActivityDefinitionName			1

リクエスト例

リクエスト URL

```
PUT http://restserver/csciwws/v1/activity-instance/5002/10002/adhoc-create-and-make-transition
```

リクエストボディ（XML の場合）

```
<?xml version="1.0" encoding="UTF-8" ?>
<Parameter>
  <ActivityDefinitionName>QuotationHandling_UTask1</ActivityDefinitionName>
</Parameter>
```

リクエストボディ（JSON の場合）

```
{
  "ActivityDefinitionName" : "QuotationHandling_UTask1"
}
```

レスポンス

遷移した業務ステップを返します。業務ステップのプロパティを次に示します。

項番	名前	型	内容
1	ActivityDefinitionID	数値	業務ステップ定義の ID
2	ActivityDefinitionName	文字列	業務ステップ定義の名称
3	ActivityTypeCode	文字列	業務ステップの種類
4	ClosedDate	日付	業務ステップの終了日時
5	Deadline	日付	業務ステップの処理期限の絶対日時
6	ID	数値	業務ステップの ID
7	Name	文字列	業務ステップの名称（業務ステップキー）
8	Priority	数値	業務ステップの優先度
9	ProcessDefinitionID	数値	業務ステップが所属するビジネスプロセス定義の ID
10	ProcessInstanceID	数値	業務ステップが所属する案件の ID
11	ProcessInstanceName	文字列	業務ステップの案件名（案件キー）
12	StartDate	日付	業務ステップの開始日時
13	StateCode	文字列	業務ステップの状態

レスポンスの構造を次に示します。

項番	名前		出現回数
1	ActivityInstance		1
2		ActivityDefinitionID	1
3		ActivityDefinitionName	1
4		ActivityTypeCode	1
5		ClosedDate	1

項番	名前		出現回数
6		Deadline	1
7		ID	1
8		Name	1
9		Priority	1
10		ProcessDefinitionID	1
11		ProcessInstanceID	1
12		ProcessInstanceName	1
13		StartDate	1
14		StateCode	1

ステータスコード

項番	ステータスコード	内容
1	200	成功
2	400	リクエストパラメタの不正
3	404	データが見つからなかった
4	500	内部処理エラー

レスポンス例

レスポンス（XML の場合）

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ActivityInstance>
  <ActivityDefinitionID>3001</ActivityDefinitionID>
  <ActivityDefinitionName>QuotationHandling_UTask1</ActivityDefinitionName>
  <ActivityTypeCode>0</ActivityTypeCode>
  <ClosedDate>2016-12-09T16:00:42+09:0</ClosedDate>
  <Deadline></Deadline>
  <ID>10002</ID>
  <Name></Name>
  <Priority></Priority>
  <ProcessDefinitionID>3002</ProcessDefinitionID>
  <ProcessInstanceID>5002</ProcessInstanceID>
  <ProcessInstanceName></ProcessInstanceName>
  <StartDate>2016-12-09T15:00:42+09:00</StartDate>
  <StateCode>u</StateCode>
</ActivityInstance>
```

レスポンス（JSON の場合）

```
{
  "ActivityDefinitionID" : "3001",
  "ActivityDefinitionName" : "QuotationHandling_UTask1",
  "ActivityTypeCode" : "0",
  "ClosedDate" : "2016-12-09T16:00:42+09:00",
  "Deadline" : "",
  "ID" : "10002",
  "Name" : "",
  "Priority" : "",
  "ProcessDefinitionID" : "3002",
  "ProcessInstanceID" : "5002",
  "ProcessInstanceName" : "",
  "StartDate" : "2016-12-09T15:00:42+09:00",
  "StateCode" : "u"
}
```

11.5.18　業務ステップの状態の変更

指定した業務ステップの状態を変更します。この API を実行すると，BPMN 連携ライブラリ Java API の CIWBPMNLib.changeStateAI インタフェースが呼び出されます。業務ステップの状態が変更済みの場合は，リクエストは成功します（何もしないでステータスコード 200 を返します）。詳細については，「5.3　Java API 利用時の注意事項」の「べき等性について」の説明を参照してください。

メソッドと URL

```
PUT /v1/activity-instance/<案件ID>/<業務ステップID>/change-state
```

<案件 ID>：業務ステップの案件 ID（必須）

<業務ステップ ID>：状態変更する業務ステップの ID（必須）

リクエストボディ

項番	名前	型	指定要否	内容
1	UserDescription	文字列	省略可	ユーザ記述子を指定します。省略した場合はデフォルト値が指定されます。空文字列は指定できません。
2	ProcessDataList	配列	省略可	プロセスデータ一覧を指定します。省略した場合はプロセスデータが更新されません。
3	ProcessData	オブジェクト	必須	プロセスデータオブジェクトを指定します。次の 2 つの要素（Key および Value）から構成されています。
4	Key	文字列	必須	プロセスデータキー名を指定します。空文字列は指定できません。

項番	名前	型	指定要否	内容
5	Value	文字列	省略可	プロセスデータ値を指定します。省略した場合，Key に対応する値は設定されません。空文字列は指定できません。
6	StateCode	文字列	必須	変更する業務ステップの状態をコード値で指定します。空文字列は指定できません。

リクエストボディの構造を次に示します。

項番	名前				出現回数
1	Parameter				1
2		UserDescription			0 または 1
3		ProcessDataList			0 または 1
4			ProcessData		1 以上
5				Key	1
6				Value	0 または 1
7		StateCode			1

リクエスト例

リクエスト URL

```
PUT http://restserver/csciwws/v1/activity-instance/5002/10002/change-state
```

リクエストボディ（XML の場合）

```
<?xml version="1.0" encoding="UTF-8" ?>
<Parameter>
  <StateCode>u</StateCode>
</Parameter>
```

リクエストボディ（JSON の場合）

```
{
  "StateCode" : "u"
}
```

レスポンス

状態を変更した業務ステップを返します。業務ステップのプロパティを次に示します。

項番	名前	型	内容
1	ActivityDefinitionID	数値	業務ステップ定義の ID
2	ActivityDefinitionName	文字列	業務ステップ定義の名称
3	ActivityTypeCode	文字列	業務ステップの種類
4	ClosedDate	日付	業務ステップの終了日時
5	Deadline	日付	業務ステップの処理期限の絶対日時
6	ID	数値	業務ステップの ID
7	Name	文字列	業務ステップの名称（業務ステップキー）
8	Priority	数値	業務ステップの優先度
9	ProcessDefinitionID	数値	業務ステップが所属するビジネスプロセス定義の ID
10	ProcessInstanceID	数値	業務ステップが所属する案件の ID
11	ProcessInstanceName	文字列	業務ステップの案件名（案件キー）
12	StartDate	日付	業務ステップの開始日時
13	StateCode	文字列	業務ステップの状態

レスポンスの構造を次に示します。

項番	名前		出現回数
1	ActivityInstance		1
2		ActivityDefinitionID	1
3		ActivityDefinitionName	1
4		ActivityTypeCode	1
5		ClosedDate	1
6		Deadline	1
7		ID	1
8		Name	1
9		Priority	1
10		ProcessDefinitionID	1
11		ProcessInstanceID	1
12		ProcessInstanceName	1
13		StartDate	1
14		StateCode	1

ステータスコード

項番	ステータスコード	内容
1	200	成功
2	400	リクエストパラメタの不正
3	404	データが見つからなかった
4	500	内部処理エラー

レスポンス例

レスポンス（XML の場合）

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ActivityInstance>
  <ActivityDefinitionID>3001</ActivityDefinitionID>
  <ActivityDefinitionName>QuotationHandling_UTask1</ActivityDefinitionName>
  <ActivityTypeCode>0</ActivityTypeCode>
  <ClosedDate>2016-12-09T16:00:42+09:00</ClosedDate>
  <Deadline></Deadline>
  <ID>10002</ID>
  <Name></Name>
  <Priority></Priority>
  <ProcessDefinitionID>3002</ProcessDefinitionID>
  <ProcessInstanceID>5002</ProcessInstanceID>
  <ProcessInstanceName></ProcessInstanceName>
  <StartDate>2016-12-09T15:00:42+09:00</StartDate>
  <StateCode>u</StateCode>
</ActivityInstance>
```

レスポンス（JSON の場合）

```
{
  "ActivityDefinitionID" : "3001",
  "ActivityDefinitionName" : "QuotationHandling_UTask1",
  "ActivityTypeCode" : "0",
  "ClosedDate" : "2016-12-09T16:00:42+09:00",
  "Deadline" : "",
  "ID" : "10002",
  "Name" : "",
  "Priority" : "",
  "ProcessDefinitionID" : "3002",
  "ProcessInstanceID" : "5002",
  "ProcessInstanceName" : "",
  "StartDate" : "2016-12-09T15:00:42+09:00",
  "StateCode" : "u"
}
```

11.5.19　作業の一覧取得

指定したフィルター条件を満たす，作業の一覧を取得します。この API を実行すると，CSCIW Java API のCIWServer.getWorkItemsList インタフェースが呼び出されます。

メソッドと URL

```
GET /v1/work-item
```

クエリパラメタ

項番	名前	型	指定要否	内容
1	userdescription	文字列	省略可	ユーザ記述子を指定します。省略した場合はデフォルト値が指定されます。空文字列は指定できません。
2	filter	文字列	省略可	作業一覧を取得する場合のフィルター条件を指定します。省略した場合はフィルター条件を指定しません。空文字列は指定できません。
3	sort	文字列	省略可	取得した作業一覧をソートする場合の条件を指定します。省略した場合はソート条件を指定しません。空文字列は指定できません。
4	offset	数値	省略可	作業一覧を取得する場合のオフセットを指定します。先頭は 0 になります。省略した場合は 0 が指定されます。0 未満の値は指定できません。
5	maxcount	数値	省略可	作業一覧を取得する場合の最大取得数を指定します。省略した場合はデフォルト値が指定されます。すべてを取得する場合は，-1 を指定します。-1 未満の値は指定できません。

リクエスト例

リクエスト URL

```
GET http://restserver/csciwws/v1/work-item?filter=ProcessInstanceID%3D5002
```

レスポンス

作業一覧を返します。個々の作業のプロパティを次に示します。

項番	名前	型	内容
1	ActivityInstanceID	数値	作業が所属する業務ステップの ID
2	ClosedDate	日付	作業の終了日時
3	CreationDate	日付	作業の発生日時
4	Deadline	日付	作業の処理期限の絶対日時

項番	名前	型	内容
5	ID	数値	作業の ID
6	Name	文字列	作業の名称（作業キー）
7	Participant	文字列	作業の作業者 ID
8	Priority	数値	作業の優先度
9	ProcessDefinitionID	数値	作業が所属するビジネスプロセス定義の ID
10	ProcessInstanceID	数値	作業が所属する案件の ID
11	ProcessInstanceName	文字列	作業の案件名（案件キー）
12	StartDate	日付	作業の開始日時
13	StateCode	文字列	作業の状態
14	WorkDefinitionID	数値	作業定義の ID
15	WorkDefinitionName	文字列	作業定義の名称
16	WorkTypeCode	文字列	作業の種類

レスポンスの構造を次に示します。

項番	名前			出現回数
1	WorkItems			1
2		WorkItem		0 または 1 以上
3			ActivityInstanceID	1
4			ClosedDate	1
5			CreationDate	1
6			Deadline	1
7			ID	1
8			Name	1
9			Participant	1
10			Priority	1
11			ProcessDefinitionID	1
12			ProcessInstanceID	1
13			ProcessInstanceName	1
14			StartDate	1
15			StateCode	1
16			WorkDefinitionID	1

項番	名前			出現回数
17			WorkDefinitionName	1
18			WorkTypeCode	1

ステータスコード

項番	ステータスコード	内容
1	200	成功（データが 0 件の場合も含む）
2	400	リクエストパラメタの不正
3	500	内部処理エラー

レスポンス例

レスポンス（XML の場合）

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<WorkItems>
  <WorkItem>
    <ActivityInstanceID>10001</ActivityInstanceID>
    <ClosedDate>2016-12-09T15:00:41+09:00</ClosedDate>
    <CreationDate>2016-12-09T15:00:40+09:00</CreationDate>
    <Deadline></Deadline>
    <ID>10001</ID>
    <Name></Name>
    <Participant>User001</Participant>
    <Priority>0</Priority>
    <ProcessDefinitionID>3001</ProcessDefinitionID>
    <ProcessInstanceID>5002</ProcessInstanceID>
    <ProcessInstanceName>案件20160609103241625</ProcessInstanceName>
    <StartDate>2016-12-09T15:00:41+09:00</StartDate>
    <StateCode>r</StateCode>
    <WorkDefinitionID>3003</WorkDefinitionID>
    <WorkDefinitionName>QuotationHandling_UTask1</WorkDefinitionName>
    <WorkTypeCode>0</WorkTypeCode>
  </WorkItem>
  <WorkItem>
    <ActivityInstanceID>10002</ActivityInstanceID>
    <ClosedDate>2016-12-09T15:25:22+09:00</ClosedDate>
    <CreationDate>2016-12-09T15:00:41+09:00</CreationDate>
    <Deadline></Deadline>
    <ID>10002</ID>
    <Name></Name>
    <Participant>User001</Participant>
    <Priority>0</Priority>
    <ProcessDefinitionID>3001</ProcessDefinitionID>
    <ProcessInstanceID>5002</ProcessInstanceID>
    <ProcessInstanceName>案件20160609103241625</ProcessInstanceName>
    <StartDate>2016-12-09T15:25:22+09:00</StartDate>
    <StateCode>r</StateCode>
    <WorkDefinitionID>3001</WorkDefinitionID>
    <WorkDefinitionName>VerifyCreditHistory_UTask1</WorkDefinitionName>
```

```
    <WorkTypeCode>0</WorkTypeCode>
  </WorkItem>
</WorkItems>
```

レスポンス（JSON の場合）

```
{
  "WorkItem" : [
    {
      "ActivityInstanceID" : "10001",
      "ClosedDate" : "2016-12-09T15:00:41+09:00",
      "CreationDate" : "2016-12-09T15:00:40+09:00",
      "Deadline" : "",
      "ID" : "10001",
      "Name" : "",
      "Participant" : "User001",
      "Priority" : "0",
      "ProcessDefinitionID" : "3001",
      "ProcessInstanceID" : "5002",
      "ProcessInstanceName" : "案件20160609103241625",
      "StartDate" : "2016-12-09T15:00:41+09:00",
      "StateCode" : "r",
      "WorkDefinitionID" : "3003",
      "WorkDefinitionName" : "QuotationHandling_UTask1",
      "WorkTypeCode" : "0"
    },
    {
      "ActivityInstanceID" : "10002",
      "ClosedDate" : "2016-12-09T15:25:22+09:00",
      "CreationDate" : "2016-12-09T15:00:41+09:00",
      "Deadline" : "",
      "ID" : "10002",
      "Name" : "",
      "Participant" : "User001",
      "Priority" : "0",
      "ProcessDefinitionID" : "3001",
      "ProcessInstanceID" : "5002",
      "ProcessInstanceName" : "案件20160609103241625",
      "StartDate" : "2016-12-09T15:25:22+09:00",
      "StateCode" : "r",
      "WorkDefinitionID" : "3001",
      "WorkDefinitionName" : "VerifyCreditHistory_UTask1",
      "WorkTypeCode" : "0"
    }
  ]
}
```

11.5.20　作業数の取得

指定した条件を満たす作業の数を取得します。この API を実行すると，CSCIW Java API の CIWServer.getListCount インタフェースが呼び出されます。

メソッドと URL

```
GET /v1/work-item/count
```

クエリパラメタ

項番	名前	型	指定要否	内容
1	userdescription	文字列	省略可	ユーザ記述子を指定します。省略した場合はデフォルト値が指定されます。空文字列は指定できません。
2	filter	文字列	省略可	作業数を取得する場合のフィルター条件を指定します。省略した場合はフィルター条件は指定されません。空文字列は指定できません。

リクエスト例

リクエスト URL

```
GET http://restserver/csciwws/v1/work-item/count?filter=ProcessInstanceID%3D100
```

レスポンス

作業数を返します。レスポンスとして返されるプロパティを次に示します。

項番	名前	型	内容
1	Count	数値	作業の数

レスポンスの構造を次に示します。

項番	名前	出現回数
1	Resources	1
2	Count	1

ステータスコード

項番	ステータスコード	内容
1	200	成功（データが 0 件の場合も成功）
2	400	リクエストパラメタの不正
3	500	内部処理エラー

レスポンス例

レスポンス（XML の場合）

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Resources>
  <Count>25</Count>
</Resources>
```

レスポンス（JSON の場合）

```
{
  "Count" : "25"
}
```

11.5.21　作業の取得

指定した ID の作業を取得します。この API を実行すると，CSCIW Java API の**CIWServer.getWorkItem** インタフェースが呼び出されます。

メソッドと URL

```
GET /v1/work-item/<案件ID>/<作業ID>
```

<案件 ID>：取得する作業の案件 ID（必須）

<作業 ID>：取得する作業の ID（必須）

クエリパラメタ

項番	名前	型	指定要否	内容
1	userdescription	文字列	省略可	ユーザ記述子を指定します。省略した場合はデフォルト値が指定されます。空文字列は指定できません。

レスポンス

作業を返します。作業のプロパティを次に示します。

項番	名前	型	内容
1	ActivityInstanceID	数値	作業が所属する業務ステップの ID
2	ClosedDate	日付	作業の終了日時
3	CreationDate	日付	作業の発生日時
4	Deadline	日付	作業の処理期限の絶対日時
5	ID	数値	作業の ID
6	Name	文字列	作業の名称（作業キー）

項番	名前	型	内容
7	Participant	文字列	作業の作業者 ID
8	Priority	数値	作業の優先度
9	ProcessDefinitionID	数値	作業が所属するビジネスプロセス定義の ID
10	ProcessInstanceID	数値	作業が所属する案件の ID
11	ProcessInstanceName	文字列	作業の案件名（案件キー）
12	StartDate	日付	作業の開始日時
13	StateCode	文字列	作業の状態
14	WorkDefinitionID	数値	作業定義の ID
15	WorkDefinitionName	文字列	作業定義の名称
16	WorkTypeCode	文字列	作業の種類

レスポンスの構造を次に示します。

項番	名前		出現回数
1	WorkItem		1
2		ActivityInstanceID	1
3		ClosedDate	1
4		CreationDate	1
5		Deadline	1
6		ID	1
7		Name	1
8		Participant	1
9		Priority	1
10		ProcessDefinitionID	1
11		ProcessInstanceID	1
12		ProcessInstanceName	1
13		StartDate	1
14		StateCode	1
15		WorkDefinitionID	1
16		WorkDefinitionName	1
17		WorkTypeCode	1

リクエスト例

リクエスト URL

```
GET http://restserver/csciwws/v1/work-item/5004/10005
```

ステータスコード

項番	ステータスコード	内容
1	200	成功
2	400	リクエストパラメタの不正
3	404	データが見つからなかった
4	500	内部処理エラー

レスポンス例

レスポンス（XML の場合）

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<WorkItem>
  <ActivityInstanceID>10005</ActivityInstanceID>
  <ClosedDate>2016-12-09T15:00:41+09:00</ClosedDate>
  <CreationDate>2016-12-09T15:00:40+09:00</CreationDate>
  <Deadline></Deadline>
  <ID>10005</ID>
  <Name></Name>
  <Participant>User001</Participant>
  <Priority></Priority>
  <ProcessDefinitionID>3001</ProcessDefinitionID>
  <ProcessInstanceID>5004</ProcessInstanceID>
  <ProcessInstanceName>案件20160609103256259</ProcessInstanceName>
  <StartDate>2016-12-09T15:00:41+09:00</StartDate>
  <StateCode>r</StateCode>
  <WorkDefinitionID>3003</WorkDefinitionID>
  <WorkDefinitionName>QuotationHandling_UTask1</WorkDefinitionName>
  <WorkTypeCode>0</WorkTypeCode>
</WorkItem>
```

レスポンス（JSON の場合）

```
{
  "ActivityInstanceID" : "10005",
  "ClosedDate" : "2016-12-09T15:00:41+09:00",
  "CreationDate" : "2016-12-09T15:00:40+09:00",
  "Deadline" : "",
  "ID" : "10005",
  "Name" : "",
  "Participant" : "User001",
  "Priority" : "",
  "ProcessDefinitionID" : "3001",
```

```
  "ProcessInstanceID" : "5004",
  "ProcessInstanceName" : "案件20160609103256259",
  "StartDate" : "2016-12-09T15:00:41+09:00",
  "StateCode" : "r",
  "WorkDefinitionID" : "3003",
  "WorkDefinitionName" : "QuotationHandling_UTask1",
  "WorkTypeCode" : "0"
}
```

11.5.22　作業の着手

指定した作業に着手します。この API を実行すると，BPMN 連携ライブラリ Java API の CIWBPMNLib.performWI インタフェースが呼び出されます。作業が着手済みの場合は，リクエストは成功します（何もしないでステータスコード 200 を返します）。詳細については，「5.3　Java API 利用時の注意事項」の「べき等性について」の説明を参照してください。

メソッドと URL

```
PUT /v1/work-item/<案件ID>/<作業ID>/perform
```

<案件 ID>：着手する作業の案件 ID（必須）

<作業 ID>：着手する作業の ID（必須）

リクエストボディ

項番	名前	型	指定要否	内容
1	UserDescription	文字列	省略可	ユーザ記述子を指定します。省略した場合はデフォルト値が指定されます。空文字列は指定できません。
2	ProcessDataList	配列	省略可	プロセスデータ一覧を指定します。省略した場合はプロセスデータが更新されません。
3	ProcessData	オブジェクト	必須	プロセスデータオブジェクトを指定します。次の 2 つの要素（Key および Value）から構成されています。
4	Key	文字列	必須	プロセスデータキー名を指定します。空文字列は指定できません。
5	Value	文字列	省略可	プロセスデータ値を指定します。省略した場合，Key に対応する値は設定されません。空文字列は指定できません。

リクエストボディの構造を次に示します。UserDescription および ProcessDataList を省略する場合は，リクエストボディを省略できます。リクエストボディを省略するときは，HTTP ヘッダのContent-Type を指定しないでください。

項番	名前				出現回数
1	Parameter				1
2		UserDescription			0 または 1
3		ProcessDataList			0 または 1
4			ProcessData		1 以上
5				Key	1
6				Value	0 または 1

リクエスト例

リクエスト URL

```
PUT http://restserver/csciwws/v1/work-item/5004/10005/perform
```

リクエストボディ（XML の場合）

```
<?xml version="1.0" encoding="UTF-8" ?>
<Parameter>
  <UserDescription>csciwuser</UserDescription>
  <ProcessDataList>
    <ProcessData>
      <Key>$Sdata1</Key>
      <Value>stringvalue1</Value>
    </ProcessData>
    <ProcessData>
      <Key>$Ndata2</Key>
      <Value>100</Value>
    </ProcessData>
  </ProcessDataList>
</Parameter>
```

リクエストボディ（JSON の場合）

```
{
  "UserDescription" : "csciwuser",
  "ProcessDataList" : {
    "ProcessData" : [
      {
        "Key" : "$Sdata1",
        "Value" : "stringvalue1"
      },
      {
        "Key" : "$Ndata2",
        "Value" : "100"
```

```
      }
    ]
  }
}
```

レスポンス

着手した作業を返します。作業のプロパティを次に示します。

項番	名前	型	内容
1	ActivityInstanceID	数値	作業が所属する業務ステップの ID
2	ClosedDate	日付	作業の終了日時
3	CreationDate	日付	作業の発生日時
4	Deadline	日付	作業の処理期限の絶対日時
5	ID	数値	作業の ID
6	Name	文字列	作業の名称（作業キー）
7	Participant	文字列	作業の作業者 ID
8	Priority	数値	作業の優先度
9	ProcessDefinitionID	数値	作業が所属するビジネスプロセス定義の ID
10	ProcessInstanceID	数値	作業が所属する案件の ID
11	ProcessInstanceName	文字列	作業の案件名（案件キー）
12	StartDate	日付	作業の開始日時
13	StateCode	文字列	作業の状態
14	WorkDefinitionID	数値	作業定義の ID
15	WorkDefinitionName	文字列	作業定義の名称
16	WorkTypeCode	文字列	作業の種類

レスポンスの構造を次に示します。

項番	名前		出現回数
1	WorkItem		1
2		ActivityInstanceID	1
3		ClosedDate	1
4		CreationDate	1
5		Deadline	1
6		ID	1

項番	名前		出現回数
7		Name	1
8		Participant	1
9		Priority	1
10		ProcessDefinitionID	1
11		ProcessInstanceID	1
12		ProcessInstanceName	1
13		StartDate	1
14		StateCode	1
15		WorkDefinitionID	1
16		WorkDefinitionName	1
17		WorkTypeCode	1

ステータスコード

項番	ステータスコード	内容
1	200	成功
2	400	リクエストパラメタの不正
3	404	データが見つからなかった
4	500	内部処理エラー

レスポンス例

レスポンス（XML の場合）

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<WorkItem>
  <ActivityInstanceID>10005</ActivityInstanceID>
  <ClosedDate>2016-12-09T15:58:52+09:00</ClosedDate>
  <CreationDate>2016-12-09T15:54:55+09:00</CreationDate>
  <Deadline></Deadline>
  <ID>10005</ID>
  <Name></Name>
  <Participant>User001</Participant>
  <Priority>0</Priority>
  <ProcessDefinitionID>3001</ProcessDefinitionID>
  <ProcessInstanceID>5004</ProcessInstanceID>
  <ProcessInstanceName>案件20160609103256259</ProcessInstanceName>
  <StartDate>2016-12-09T15:58:52+09:00</StartDate>
  <StateCode>r</StateCode>
  <WorkDefinitionID>3001</WorkDefinitionID>
  <WorkDefinitionName>QuotationHandling_UTask1</WorkDefinitionName>
```

```
  <WorkTypeCode>0</WorkTypeCode>
</WorkItem>
```

レスポンス（JSON の場合）

```
{
  "ActivityInstanceID" : "10005",
  "ClosedDate" : "2016-12-09T15:58:52+09:00",
  "CreationDate" : "2016-12-09T15:54:55+09:00",
  "Deadline" : "",
  "ID" : "10005",
  "Name" : "",
  "Participant" : "User001",
  "Priority" : "0",
  "ProcessDefinitionID" : "3001",
  "ProcessInstanceID" : "5004",
  "ProcessInstanceName" : "案件20160609103256259",
  "StartDate" : "2016-12-09T15:58:52+09:00",
  "StateCode" : "r",
  "WorkDefinitionID" : "3001",
  "WorkDefinitionName" : "QuotationHandling_UTask1",
  "WorkTypeCode" : "0"
}
```

11.5.23　作業の完了

指定した作業を完了します。この API を実行すると，BPMN 連携ライブラリ Java API の CIWBPMNLib.completeWI インタフェースが呼び出されます。作業が完了済みの場合は，リクエストは成功します（何もしないでステータスコード 200 を返します）。詳細については，「5.3　Java API 利用時の注意事項」の「べき等性について」の説明を参照してください。

メソッドと URL

```
PUT /v1/work-item/<案件ID>/<作業ID>/complete
```

<案件 ID>：完了する作業の案件 ID（必須）

<作業 ID>：完了する作業の ID（必須）

リクエストボディ

項番	名前	型	指定要否	内容
1	UserDescription	文字列	省略可	ユーザ記述子を指定します。省略した場合はデフォルト値が指定されます。空文字列は指定できません。

項番	名前	型	指定要否	内容
2	ProcessDataList	配列	省略可	プロセスデータ一覧を指定します。省略した場合はプロセスデータが更新されません。
3	ProcessData	オブジェクト	必須	プロセスデータオブジェクトを指定します。次の 2 つの要素（Key および Value）から構成されています。
4	Key	文字列	必須	プロセスデータキー名を指定します。空文字列は指定できません。
5	Value	文字列	省略可	プロセスデータ値を指定します。省略した場合，Key に対応する値は設定されません。空文字列は指定できません。

リクエストボディの構造を次に示します。UserDescription および ProcessDataList を省略する場合は，リクエストボディを省略できます。リクエストボディを省略するときは，HTTP ヘッダの`Content-Type` を指定しないでください。

項番	名前				出現回数
1	Parameter				1
2		UserDescription			0 または 1
3		ProcessDataList			0 または 1
4			ProcessData		1 以上
5				Key	1
6				Value	0 または 1

リクエスト例

リクエスト URL

```
PUT http://restserver/csciwws/v1/work-item/5004/10005/complete
```

リクエストボディ（XML の場合）

```
<?xml version="1.0" encoding="UTF-8" ?>
<Parameter>
  <UserDescription>csciwuser</UserDescription>
  <ProcessDataList>
    <ProcessData>
      <Key>$Sdata1</Key>
      <Value>stringvalue1</Value>
    </ProcessData>
    <ProcessData>
      <Key>$Ndata2</Key>
```

```
      <Value>100</Value>
    </ProcessData>
  </ProcessDataList>
</Parameter>
```

リクエストボディ（JSON の場合）

```
{
  "UserDescription" : "csciwuser",
  "ProcessDataList" : {
    "ProcessData" : [
      {
        "Key" : "$Sdata1",
        "Value" : "stringvalue1"
      },
      {
        "Key" : "$Ndata2",
        "Value" : "100"
      }
    ]
  }
}
```

レスポンス

完了した作業を返します。作業のプロパティを次に示します。

項番	名前	型	内容
1	ActivityInstanceID	数値	作業が所属する業務ステップの ID
2	ClosedDate	日付	作業の終了日時
3	CreationDate	日付	作業の発生日時
4	Deadline	日付	作業の処理期限の絶対日時
5	ID	数値	作業の ID
6	Name	文字列	作業の名称（作業キー）
7	Participant	文字列	作業の作業者 ID
8	Priority	数値	作業の優先度
9	ProcessDefinitionID	数値	作業が所属するビジネスプロセス定義の ID
10	ProcessInstanceID	数値	作業が所属する案件の ID
11	ProcessInstanceName	文字列	作業の案件名（案件キー）
12	StartDate	日付	作業の開始日時
13	StateCode	文字列	作業の状態
14	WorkDefinitionID	数値	作業定義の ID

項番	名前	型	内容
15	WorkDefinitionName	文字列	作業定義の名称
16	WorkTypeCode	文字列	作業の種類

レスポンスの構造を次に示します。

項番	名前		出現回数
1	WorkItem		1
2		ActivityInstanceID	1
3		ClosedDate	1
4		CreationDate	1
5		Deadline	1
6		ID	1
7		Name	1
8		Participant	1
9		Priority	1
10		ProcessDefinitionID	1
11		ProcessInstanceID	1
12		ProcessInstanceName	1
13		StartDate	1
14		StateCode	1
15		WorkDefinitionID	1
16		WorkDefinitionName	1
17		WorkTypeCode	1

ステータスコード

項番	ステータスコード	内容
1	200	成功
2	400	リクエストパラメタの不正
3	404	データが見つからなかった
4	500	内部処理エラー

レスポンス例

レスポンス（XML の場合）

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<WorkItem>
  <ActivityInstanceID>10005</ActivityInstanceID>
  <ClosedDate>2016-12-09T15:58:52+09:00</ClosedDate>
  <CreationDate>2016-12-09T15:54:55+09:00</CreationDate>
  <Deadline></Deadline>
  <ID>10005</ID>
  <Name></Name>
  <Participant>User001</Participant>
  <Priority>0</Priority>
  <ProcessDefinitionID>3001</ProcessDefinitionID>
  <ProcessInstanceID>5004</ProcessInstanceID>
  <ProcessInstanceName>案件20160609103256259</ProcessInstanceName>
  <StartDate>2016-12-09T15:58:52+09:00</StartDate>
  <StateCode>r</StateCode>
  <WorkDefinitionID>3001</WorkDefinitionID>
  <WorkDefinitionName>QuotationHandling_UTask1</WorkDefinitionName>
  <WorkTypeCode>0</WorkTypeCode>
</WorkItem>
```

レスポンス（JSON の場合）

```
{
  "ActivityInstanceID" : "10005",
  "ClosedDate" : "2016-12-09T15:58:52+09:00",
  "CreationDate" : "2016-12-09T15:54:55+09:00",
  "Deadline" : "",
  "ID" : "10005",
  "Name" : "",
  "Participant" : "User001",
  "Priority" : "0",
  "ProcessDefinitionID" : "3001",
  "ProcessInstanceID" : "5004",
  "ProcessInstanceName" : "案件20160609103256259",
  "StartDate" : "2016-12-09T15:58:52+09:00",
  "StateCode" : "r",
  "WorkDefinitionID" : "3001",
  "WorkDefinitionName" : "QuotationHandling_UTask1",
  "WorkTypeCode" : "0"
}
```

11.5.24　作業者の変更

指定した作業の作業者を変更します。この API を実行すると，BPMN 連携ライブラリ Java API の CIWBPMNLib.reassignWI インタフェースが呼び出されます。指定した作業者に変更済みだった場合は，リクエストは成功します（何もしないでステータスコード 200 を返します）。詳細については，「5.3　Java API 利用時の注意事項」の「べき等性について」の説明を参照してください。

メソッドと URL

```
PUT /v1/work-item/<案件ID>/<作業ID>/reassign
```

<案件 ID>：作業者を変更する作業の案件 ID（必須）

<作業 ID>：作業者を変更する作業の ID（必須）

リクエストボディ

項番	名前	型	指定要否	内容
1	UserDescription	文字列	省略可	ユーザ記述子を指定します。省略した場合はデフォルト値が指定されます。空文字列は指定できません。
2	Source	文字列	省略可	変更する前の作業者 ID を指定します。変更する前の作業者 ID が未設定の場合は省略します。空文字列は指定できません。
3	Target	文字列	省略可	新しい作業者 ID を指定します。省略した場合，作業者 ID は未設定となります。空文字列は指定できません。

リクエストボディの構造を次に示します。UserDescription，Source および Target を省略する場合は，リクエストボディを省略できます。リクエストボディを省略するときは，HTTP ヘッダの`Content-Type` を指定しないでください。

項番	名前		出現回数
1	Parameter		1
2		UserDescription	0 または 1
3		Source	0 または 1
4		Target	0 または 1

リクエスト例

リクエスト URL

```
PUT http://restserver/csciwws/v1/work-item/5004/10005/reassign
```

リクエストボディ（XML の場合）

```
<?xml version="1.0" encoding="UTF-8" ?>
<Parameter>
  <UserDescription>csciwuser</UserDescription>
  <Source>User001</Source>
```

```
  <Target>User002</Target>
</Parameter>
```

リクエストボディ（JSON の場合）

```
{
  "UserDescription" : "csciwuser",
  "Source" : "User001",
  "Target" : "User002"
}
```

レスポンス

作業者を変更した作業を返します。作業のプロパティを次に示します。

項番	名前	型	内容
1	ActivityInstanceID	数値	作業が所属する業務ステップの ID
2	ClosedDate	日付	作業の終了日時
3	CreationDate	日付	作業の発生日時
4	Deadline	日付	作業の処理期限の絶対日時
5	ID	数値	作業の ID
6	Name	文字列	作業の名称（作業キー）
7	Participant	文字列	作業の作業者 ID
8	Priority	数値	作業の優先度
9	ProcessDefinitionID	数値	作業が所属するビジネスプロセス定義の ID
10	ProcessInstanceID	数値	作業が所属する案件の ID
11	ProcessInstanceName	文字列	作業の案件名（案件キー）
12	StartDate	日付	作業の開始日時
13	StateCode	文字列	作業の状態
14	WorkDefinitionID	数値	作業定義の ID
15	WorkDefinitionName	文字列	作業定義の名称
16	WorkTypeCode	文字列	作業の種類

レスポンスの構造を次に示します。

項番	名前		出現回数
1	WorkItem		1
2		ActivityInstanceID	1

項番	名前		出現回数
3		ClosedDate	1
4		CreationDate	1
5		Deadline	1
6		ID	1
7		Name	1
8		Participant	1
9		Priority	1
10		ProcessDefinitionID	1
11		ProcessInstanceID	1
12		ProcessInstanceName	1
13		StartDate	1
14		StateCode	1
15		WorkDefinitionID	1
16		WorkDefinitionName	1
17		WorkTypeCode	1

ステータスコード

項番	ステータスコード	内容
1	200	成功
2	400	リクエストパラメタの不正
3	404	データが見つからなかった
4	500	内部処理エラー

レスポンス例

レスポンス（XML の場合）

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<WorkItem>
  <ActivityInstanceID>10005</ActivityInstanceID>
  <ClosedDate></ClosedDate>
  <CreationDate>2016-12-09T15:36:04+09:00</CreationDate>
  <Deadline></Deadline>
  <ID>10005</ID>
  <Name></Name>
  <Participant>User002</Participant>
  <Priority></Priority>
```

```
  <ProcessDefinitionID>3001</ProcessDefinitionID>
  <ProcessInstanceID>5004</ProcessInstanceID>
  <ProcessInstanceName>案件20160609103256259</ProcessInstanceName>
  <StartDate></StartDate>
  <StateCode>j</StateCode>
  <WorkDefinitionID>3001</WorkDefinitionID>
  <WorkDefinitionName>QuotationHandling_UTask1</WorkDefinitionName>
  <WorkTypeCode>0</WorkTypeCode>
</WorkItem>
```

レスポンス（JSON の場合）

```
{
  "ActivityInstanceID" : "10005",
  "ClosedDate" : "",
  "CreationDate" : "2016-12-09T15:36:04+09:00",
  "Deadline" : "",
  "ID" : "10005",
  "Name" : "",
  "Participant" : "User002",
  "Priority" : "",
  "ProcessDefinitionID" : "3001",
  "ProcessInstanceID" : "5004",
  "ProcessInstanceName" : "案件20160609103256259",
  "StartDate" : "",
  "StateCode" : "j",
  "WorkDefinitionID" : "3001",
  "WorkDefinitionName" : "QuotationHandling_UTask1",
  "WorkTypeCode" : "0"
}
```

11.5.25　作業者を変更して着手

指定した作業を，作業者を変更してから着手します。この API を実行すると，BPMN 連携ライブラリ Java API の CIWBPMNLib.reassignAndPerformWI インタフェースが呼び出されます。作業者を変更して作業が着手済みだった場合，リクエストは成功します（何もしないでステータスコード 200 を返します）。詳細については，「5.3　Java API 利用時の注意事項」の「べき等性について」の説明を参照してください。

メソッドと URL

```
PUT /v1/work-item/<案件ID>/<作業ID>/reassign-and-perform
```

<案件 ID>：作業者を変更して着手する作業の案件 ID（必須）

<作業 ID>：作業者を変更して着手する作業の ID（必須）

リクエストボディ

項番	名前	型	指定要否	内容
1	UserDescription	文字列	省略可	ユーザ記述子を指定します。省略した場合はデフォルト値が指定されます。空文字列は指定できません。
2	Source	文字列	省略可	変更する前の作業者 ID を指定します。変更する前の作業者 ID が未設定の場合は省略できます。空文字列は指定できません。
3	Target	文字列	省略可	新しい作業者 ID を指定します。省略した場合，作業者 ID は設定されません。空文字列は指定できません。
4	ProcessDataList	配列	省略可	プロセスデータ一覧を指定します。省略した場合はプロセスデータが更新されません。
5	ProcessData	オブジェクト	必須	プロセスデータオブジェクトを指定します。次の 2 つの要素（Key および Value）から構成されています。
6	Key	文字列	必須	プロセスデータキー名を指定します。空文字列は指定できません。
7	Value	文字列	省略可	プロセスデータ値を指定します。省略した場合，Key に対応する値は設定されません。空文字列は指定できません。

リクエストボディの構造を次に示します。UserDescription，Source，Target および ProcessDataList を省略する場合は，リクエストボディを省略できます。リクエストボディを省略するときは，HTTP ヘッダの`Content-Type` を指定しないでください。

項番	名前				出現回数
1	Parameter				1
2		UserDescription			0 または 1
3		Source			0 または 1
4		Target			0 または 1
5		ProcessDataList			0 または 1
6			ProcessData		1 以上
7				Key	1
8				Value	0 または 1

リクエスト例

リクエスト URL

```
PUT http://restserver/csciwws/v1/work-item/5004/10005/reassign-and-perform
```

リクエストボディ（XML の場合）

```
<?xml version="1.0" encoding="UTF-8" ?>
<Parameter>
  <UserDescription>csciwuser</UserDescription>
  <Source>guest</Source>
  <Target>planner</Target>
  <ProcessDataList>
    <ProcessData>
      <Key>$Sdata1</Key>
      <Value>stringvalue1</Value>
    </ProcessData>
    <ProcessData>
      <Key>$Ndata2</Key>
      <Value>100</Value>
    </ProcessData>
  </ProcessDataList>
</Parameter>
```

リクエストボディ（JSON の場合）

```
{
  "UserDescription" : "csciwuser",
  "Source" : "guest",
  "Target" : "planner",
  "ProcessDataList" : {
    "ProcessData" : [
      {
        "Key" : "$Sdata1",
        "Value" : "stringvalue1"
      },
      {
        "Key" : "$Ndata2",
        "Value" : "100"
      }
    ]
  }
}
```

レスポンス

着手した作業を返します。作業のプロパティを次に示します。

項番	名前	型	内容
1	ActivityInstanceID	数値	作業が所属する業務ステップの ID
2	ClosedDate	日付	作業の終了日時
3	CreationDate	日付	作業の発生日時
4	Deadline	日付	作業の処理期限の絶対日時

項番	名前	型	内容
5	ID	数値	作業の ID
6	Name	文字列	作業の名称（作業キー）
7	Participant	文字列	作業の作業者 ID
8	Priority	数値	作業の優先度
9	ProcessDefinitionID	数値	作業が所属するビジネスプロセス定義の ID
10	ProcessInstanceID	数値	作業が所属する案件の ID
11	ProcessInstanceName	文字列	作業の案件名（案件キー）
12	StartDate	日付	作業の開始日時
13	StateCode	文字列	作業の状態
14	WorkDefinitionID	数値	作業定義の ID
15	WorkDefinitionName	文字列	作業定義の名称
16	WorkTypeCode	文字列	作業の種類

レスポンスの構造を次に示します。

項番	名前		出現回数
1	WorkItem		1
2		ActivityInstanceID	1
3		ClosedDate	1
4		CreationDate	1
5		Deadline	1
6		ID	1
7		Name	1
8		Participant	1
9		Priority	1
10		ProcessDefinitionID	1
11		ProcessInstanceID	1
12		ProcessInstanceName	1
13		StartDate	1
14		StateCode	1
15		WorkDefinitionID	1

項番	名前		出現回数
16		WorkDefinitionName	1
17		WorkTypeCode	1

ステータスコード

項番	ステータスコード	内容
1	200	成功
2	400	リクエストパラメタの不正
3	404	データが見つからなかった
4	500	内部処理エラー

レスポンス例

レスポンス（XML の場合）

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<WorkItem>
  <ActivityInstanceID>10005</ActivityInstanceID>
  <ClosedDate>2016-12-09T15:58:52+09:00</ClosedDate>
  <CreationDate>2016-12-09T15:54:55+09:00</CreationDate>
  <Deadline></Deadline>
  <ID>10005</ID>
  <Name></Name>
  <Participant>planner</Participant>
  <Priority>0</Priority>
  <ProcessDefinitionID>3001</ProcessDefinitionID>
  <ProcessInstanceID>5004</ProcessInstanceID>
  <ProcessInstanceName>案件20160609103256259</ProcessInstanceName>
  <StartDate>2016-12-09T15:58:52+09:00</StartDate>
  <StateCode>f</StateCode>
  <WorkDefinitionID>3001</WorkDefinitionID>
  <WorkDefinitionName>QuotationHandling_UTask1</WorkDefinitionName>
  <WorkTypeCode>0</WorkTypeCode>
</WorkItem>
```

レスポンス（JSON の場合）

```
{
  "ActivityInstanceID" : "10005",
  "ClosedDate" : "2016-12-09T15:58:52+09:00",
  "CreationDate" : "2016-12-09T15:54:55+09:00",
  "Deadline" : "",
  "ID" : "10005",
  "Name" : "",
  "Participant" : "planner",
  "Priority" : "0",
  "ProcessDefinitionID" : "3001",
  "ProcessInstanceID" : "5004",
```

```
  "ProcessInstanceName" : "案件20160609103256259",
  "StartDate" : "2016-12-09T15:58:52+09:00",
  "StateCode" : "f",
  "WorkDefinitionID" : "3001",
  "WorkDefinitionName" : "QuotationHandling_UTask1",
  "WorkTypeCode" : "0"
}
```

11.5.26　作業を着手して完了

指定した ID の作業を完了します。この API を実行すると，BPMN 連携ライブラリ Java API の CIWBPMNLib.performAndCompleteWI インタフェースが呼び出されます。指定された作業を着手して完了済みだった場合は，リクエストは成功します（何もしないでステータスコード 200 を返します）。詳細については，「5.3　Java API 利用時の注意事項」の「べき等性について」の説明を参照してください。

メソッドと URL

```
PUT /v1/work-item/<案件ID>/<作業ID>/perform-and-complete
```

<案件 ID>：完了する作業の案件 ID（必須）

<作業 ID>：完了する作業の ID（必須）

リクエストボディ

項番	名前	型	指定要否	内容
1	UserDescription	文字列	省略可	ユーザ記述子を指定します。省略した場合はデフォルト値が指定されます。空文字列は指定できません。
2	ProcessDataList	配列	省略可	プロセスデータ一覧を指定します。省略した場合はプロセスデータを更新しません。
3	ProcessData	オブジェクト	必須	プロセスデータオブジェクトを指定します。次の 2 つの要素（Key および Value）から構成されています。
4	Key	文字列	必須	プロセスデータキー名を指定します。空文字列は指定できません。
5	Value	文字列	省略可	プロセスデータ値を指定します。省略した場合，Key に対応する値は未設定となります。空文字列は指定できません。

リクエストボディの構造を次に示します。UserDescription および ProcessDataList を省略する場合は，リクエストボディを省略できます。リクエストボディを省略するときは，HTTP ヘッダの Content-Type を指定しないでください。

項番	名前				出現回数
1	Parameter				1
2		UserDescription			0 または 1
3		ProcessDataList			0 または 1
4			ProcessData		1 以上
5				Key	1
6				Value	0 または 1

リクエスト例

リクエスト URL

```
PUT http://restserver/csciwws/v1/work-item/5004/10005/perform-and-complete
```

リクエストボディ（XML の場合）

```
<?xml version="1.0" encoding="UTF-8" ?>
<Parameter>
  <UserDescription>csciwuser</UserDescription>
  <ProcessDataList>
    <ProcessData>
      <Key>$Sdata1</Key>
      <Value>stringvalue1</Value>
    </ProcessData>
    <ProcessData>
      <Key>$Ndata2</Key>
      <Value>100</Value>
    </ProcessData>
  </ProcessDataList>
</Parameter>
```

リクエストボディ（JSON の場合）

```
{
  "UserDescription" : "csciwuser",
  "ProcessDataList" : {
    "ProcessData" : [
      {
        "Key" : "$Sdata1",
        "Value" : "stringvalue1"
      },
      {
        "Key" : "$Ndata2",
        "Value" : "100"
      }
    ]
  }
}
```

レスポンス

着手して完了した作業を返します。作業のプロパティを次に示します。

項番	名前	型	内容
1	ActivityInstanceID	数値	作業が所属する業務ステップの ID
2	ClosedDate	日付	作業の終了日時
3	CreationDate	日付	作業の発生日時
4	Deadline	日付	作業の処理期限の絶対日時
5	ID	数値	作業の ID
6	Name	文字列	作業の名称（作業キー）
7	Participant	文字列	作業の作業者 ID
8	Priority	数値	作業の優先度
9	ProcessDefinitionID	数値	作業が所属するビジネスプロセス定義の ID
10	ProcessInstanceID	数値	作業が所属する案件の ID
11	ProcessInstanceName	文字列	作業の案件名（案件キー）
12	StartDate	日付	作業の開始日時
13	StateCode	文字列	作業の状態
14	WorkDefinitionID	数値	作業定義の ID
15	WorkDefinitionName	文字列	作業定義の名称
16	WorkTypeCode	文字列	作業の種類

レスポンスの構造を次に示します。

項番	名前		出現回数
1	WorkItem		1
2		ActivityInstanceID	1
3		ClosedDate	1
4		CreationDate	1
5		Deadline	1
6		ID	1
7		Name	1
8		Participant	1
9		Priority	1

項番	名前		出現回数
10		ProcessDefinitionID	1
11		ProcessInstanceID	1
12		ProcessInstanceName	1
13		StartDate	1
14		StateCode	1
15		WorkDefinitionID	1
16		WorkDefinitionName	1
17		WorkTypeCode	1

ステータスコード

項番	ステータスコード	内容
1	200	成功
2	400	リクエストパラメタの不正
3	404	データが見つからなかった
4	500	内部処理エラー

レスポンス例

レスポンス（XML の場合）

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<WorkItem>
  <ActivityInstanceID>10005</ActivityInstanceID>
  <ClosedDate>2016-12-09T15:58:52+09:00</ClosedDate>
  <CreationDate>2016-12-09T15:54:55+09:00</CreationDate>
  <Deadline></Deadline>
  <ID>10005</ID>
  <Name></Name>
  <Participant>User001</Participant>
  <Priority>0</Priority>
  <ProcessDefinitionID>3001</ProcessDefinitionID>
  <ProcessInstanceID>5004</ProcessInstanceID>
  <ProcessInstanceName>案件20160609103256259</ProcessInstanceName>
  <StartDate>2016-12-09T15:58:52+09:00</StartDate>
  <StateCode>r</StateCode>
  <WorkDefinitionID>3001</WorkDefinitionID>
  <WorkDefinitionName>QuotationHandling_UTask1</WorkDefinitionName>
  <WorkTypeCode>0</WorkTypeCode>
</WorkItem>
```

レスポンス（JSON の場合）

```
{
  ”ActivityInstanceID” : ”10005”,
  ”ClosedDate” : ”2016-12-09T15:58:52+09:00”,
  ”CreationDate” : ”2016-12-09T15:54:55+09:00”,
  ”Deadline” : ””,
  ”ID” : ”10005”,
  ”Name” : ””,
  ”Participant” : ”User001”,
  ”Priority” : ”0”,
  ”ProcessDefinitionID” : ”3001”,
  ”ProcessInstanceID” : ”5004”,
  ”ProcessInstanceName” : ”案件20160609103256259”,
  ”StartDate” : ”2016-12-09T15:58:52+09:00”,
  ”StateCode” : ”r”,
  ”WorkDefinitionID” : ”3001”,
  ”WorkDefinitionName” : ”QuotationHandling_UTask1”,
  ”WorkTypeCode” : ”0”
}
```

11.5.27　作業の状態の変更

指定した ID の作業の状態を変更します。この API を実行すると，BPMN 連携ライブラリ Java API の CIWBPMNLib.changeStateWI インタフェースが呼び出されます。作業の状態が変更済みの場合，リクエストは成功します（何もしないでステータスコード 200 を返します）。詳細については，「5.3　Java API 利用時の注意事項」の「べき等性について」の説明を参照してください。

メソッドと URL

```
PUT /v1/work-item/<案件ID>/<作業ID>/change-state
```

<案件 ID>：状態を変更する作業の案件 ID（必須）

<作業 ID>：状態を変更する作業の ID（必須）

リクエストボディ

項番	名前	型	指定要否	内容
1	UserDescription	文字列	省略可	ユーザ記述子を指定します。省略した場合はデフォルト値が指定されます。空文字列は指定できません。
2	ProcessDataList	配列	省略可	プロセスデータ一覧を指定します。省略した場合はプロセスデータを更新しません。
3	ProcessData	オブジェクト	必須	プロセスデータオブジェクトを指定します。次の 2 つの要素（Key および Value）から構成されています。
4	Key	文字列	必須	プロセスデータキー名を指定します。空文字列は指定できません。

項番	名前	型	指定要否	内容
5	Value	文字列	省略可	プロセスデータ値を指定します。省略した場合，Key に対応する値は設定されません。空文字列は指定できません。
6	StateCode	文字列	必須	変更する作業の状態をコード値で指定します。

リクエストボディの構造を次に示します。

項番	名前				出現回数
1	Parameter				1
2		UserDescription			0 または 1
3		ProcessDataList			0 または 1
4			ProcessData		1 以上
5				Key	1
6				Value	0 または 1
7		StateCode			1

リクエスト例

リクエスト URL

```
PUT http://restserver/csciwws/v1/work-item/5004/10005/change-state
```

リクエストボディ（XML の場合）

```
<?xml version="1.0" encoding="UTF-8" ?>
<Parameter>
  <StateCode>u</StateCode>
</Parameter>
```

リクエストボディ（JSON の場合）

```
{
  "StateCode" : "u"
}
```

レスポンス

状態を変更した作業を返します。作業のプロパティを次に示します。

項番	名前	型	内容
1	ActivityInstanceID	数値	作業が所属する業務ステップの ID

項番	名前	型	内容
2	ClosedDate	日付	作業の終了日時
3	CreationDate	日付	作業の発生日時
4	Deadline	日付	作業の処理期限の絶対日時
5	ID	数値	作業の ID
6	Name	文字列	作業の名称（作業キー）
7	Participant	文字列	作業の作業者 ID
8	Priority	数値	作業の優先度
9	ProcessDefinitionID	数値	作業が所属するビジネスプロセス定義の ID
10	ProcessInstanceID	数値	作業が所属する案件の ID
11	ProcessInstanceName	文字列	作業の案件名（案件キー）
12	StartDate	日付	作業の開始日時
13	StateCode	文字列	作業の状態
14	WorkDefinitionID	数値	作業定義の ID
15	WorkDefinitionName	文字列	作業定義の名称
16	WorkTypeCode	文字列	作業の種類

レスポンスの構造を次に示します。

項番	名前		出現回数
1	WorkItem		1
2		ActivityInstanceID	1
3		ClosedDate	1
4		CreationDate	1
5		Deadline	1
6		ID	1
7		Name	1
8		Participant	1
9		Priority	1
10		ProcessDefinitionID	1
11		ProcessInstanceID	1
12		ProcessInstanceName	1

項番	名前		出現回数
13		StartDate	1
14		StateCode	1
15		WorkDefinitionID	1
16		WorkDefinitionName	1
17		WorkTypeCode	1

ステータスコード

項番	ステータスコード	内容
1	200	成功
2	400	リクエストパラメタの不正
3	404	データが見つからなかった
4	500	内部処理エラー

レスポンス例

レスポンス（XML の場合）

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<WorkItem>
  <ActivityInstanceID>10005</ActivityInstanceID>
  <ClosedDate>2016-12-09T15:58:52+09:00</ClosedDate>
  <CreationDate>2016-12-09T15:54:55+09:00</CreationDate>
  <Deadline></Deadline>
  <ID>10005</ID>
  <Name></Name>
  <Participant>User001</Participant>
  <Priority>0</Priority>
  <ProcessDefinitionID>3001</ProcessDefinitionID>
  <ProcessInstanceID>5004</ProcessInstanceID>
  <ProcessInstanceName>案件20160609103256259</ProcessInstanceName>
  <StartDate>2016-12-09T15:58:52+09:00</StartDate>
  <StateCode>u</StateCode>
  <WorkDefinitionID>3001</WorkDefinitionID>
  <WorkDefinitionName>QuotationHandling_UTask1</WorkDefinitionName>
  <WorkTypeCode>0</WorkTypeCode>
</WorkItem>
```

レスポンス（JSON の場合）

```
{
  "ActivityInstanceID" : "10005",
  "ClosedDate" : "2016-12-09T15:58:52+09:00",
  "CreationDate" : "2016-12-09T15:54:55+09:00",
  "Deadline" : "",
```

```
  "ID" : "10005",
  "Name" : "",
  "Participant" : "User001",
  "Priority" : "0",
  "ProcessDefinitionID" : "3001",
  "ProcessInstanceID" : "5004",
  "ProcessInstanceName" : "案件20160609103256259",
  "StartDate" : "2016-12-09T15:58:52+09:00",
  "StateCode" : "u",
  "WorkDefinitionID" : "3001",
  "WorkDefinitionName" : "QuotationHandling_UTask1",
  "WorkTypeCode" : "0"
}
```

11.5.28　条件に一致する作業の作業者割り当てと着手

指定したフィルター条件とソート条件を満たす作業群に対して，作業者を割り当てます。そして，最初に割り当てに成功した作業を，「実行開始可能」状態から「作業者実行」状態に変更して返します。この API を実行すると，BPMN 連携ライブラリ Java API の`CIWBPMNLib.allocateWIEx` インタフェースが呼び出されます。

メソッドと URL

```
PUT /v1/work-item/allocate
```

リクエストボディ

項番	名前	型	指定要否	内容
1	UserDescription	文字列	省略可	ユーザ記述子を指定します。省略した場合はデフォルト値が指定されます。空文字列は指定できません。
2	Participant	文字列	必須	割り当てる作業者 ID を指定します。
3	Filter	文字列	省略可	作業を検索する場合のフィルター条件を指定します。フィルター条件には，次に示すようにレーン名を含めた条件を指定してください。 • Participant ＝ 'レーン名' • Participant like 'レーン名の前方の文字列%' 作業群がすべてユーザタスクの場合のような特殊なケースに限って，フィルター条件を省略しても正常に動作します。省略した場合はフィルター条件は指定されません。空文字列は指定できません。 このフィルター条件には，作業の状態が「実行開始可能」状態と一致する条件が自動的に追加されます。

項番	名前	型	指定要否	内容
4	Sort	文字列	省略可	作業をソートする場合の条件を指定します。省略した場合はソート条件は指定されません。空文字列は指定できません。
5	FetchCount	数値	必須	一度に取得する件数を指定します。すべてを取得する場合は，-1 を指定します。-1 未満の値は指定できません。

リクエストボディの構造を次に示します。

項番	名前		出現回数
1	Parameter		1
2		UserDescription	0 または 1
3		Participant	1
4		Filter	0 または 1
5		Sort	0 または 1
6		FetchCount	1

リクエスト例

リクエスト URL

```
PUT http://restserver/csciwws/v1/work-item/allocate
```

リクエストボディ（XML の場合）

```
<?xml version="1.0" encoding="UTF-8" ?>
<Parameter>
  <Participant>User1</Participant>
  <Filter>Participant=&amp;apos;MyLane1&amp;apos;</Filter>   <!-- Participant='MyLane1' -->
  <Sort>ID</Sort>
  <FetchCount>100</FetchCount>
</Parameter>
```

リクエストボディ（JSON の場合）

```
{
  "Participant" : "User1",
  "Filter" : "Participant='MyLane1'",
  "Sort" : "ID",
  "FetchCount" : "100"
}
```

レスポンス

着手した作業を返します。作業のプロパティを次に示します。

項番	名前	型	内容
1	ActivityInstanceID	数値	作業が所属する業務ステップの ID
2	ClosedDate	日付	作業の終了日時
3	CreationDate	日付	作業の発生日時
4	Deadline	日付	作業の処理期限の絶対日時
5	ID	数値	作業の ID
6	Name	文字列	作業の名称（作業キー）
7	Participant	文字列	作業の作業者 ID
8	Priority	数値	作業の優先度
9	ProcessDefinitionID	数値	作業が所属するビジネスプロセス定義の ID
10	ProcessInstanceID	数値	作業が所属する案件の ID
11	ProcessInstanceName	文字列	作業の案件名（案件キー）
12	StartDate	日付	作業の開始日時
13	StateCode	文字列	作業の状態
14	WorkDefinitionID	数値	作業定義の ID
15	WorkDefinitionName	文字列	作業定義の名称
16	WorkTypeCode	文字列	作業の種類

レスポンスの構造を次に示します。

項番	名前		出現回数
1	WorkItem		1
2		ActivityInstanceID	1
3		ClosedDate	1
4		CreationDate	1
5		Deadline	1
6		ID	1
7		Name	1
8		Participant	1
9		Priority	1

項番	名前		出現回数
10		ProcessDefinitionID	1
11		ProcessInstanceID	1
12		ProcessInstanceName	1
13		StartDate	1
14		StateCode	1
15		WorkDefinitionID	1
16		WorkDefinitionName	1
17		WorkTypeCode	1

ステータスコード

項番	ステータスコード	内容
1	200	成功
2	400	リクエストパラメタの不正
3	404	データが見つからなかった
4	500	内部処理エラー

レスポンス例

レスポンス（XML の場合）

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<WorkItem>
  <ActivityInstanceID>10005</ActivityInstanceID>
  <ClosedDate></ClosedDate>
  <CreationDate>2016-12-09T15:54:55+09:00</CreationDate>
  <Deadline></Deadline>
  <ID>10005</ID>
  <Name></Name>
  <Participant>User1</Participant>
  <Priority>0</Priority>
  <ProcessDefinitionID>3001</ProcessDefinitionID>
  <ProcessInstanceID>5004</ProcessInstanceID>
  <ProcessInstanceName>案件20160609103256259</ProcessInstanceName>
  <StartDate>2016-12-09T15:58:52+09:00</StartDate>
  <StateCode>f</StateCode>
  <WorkDefinitionID>3001</WorkDefinitionID>
  <WorkDefinitionName>QuotationHandling_UTask1</WorkDefinitionName>
  <WorkTypeCode>0</WorkTypeCode>
</WorkItem>
```

レスポンス（JSON の場合）

```
{
  ”ActivityInstanceID” : ”10005”,
  ”ClosedDate” : ””,
  ”CreationDate” : ”2016-12-09T15:54:55+09:00”,
  ”Deadline” : ””,
  ”ID” : ”10005”,
  ”Name” : ””,
  ”Participant” : ”User1”,
  ”Priority” : ”0”,
  ”ProcessDefinitionID” : ”3001”,
  ”ProcessInstanceID” : ”5004”,
  ”ProcessInstanceName” : ”案件20160609103256259”,
  ”StartDate” : ”2016-12-09T15:58:52+09:00”,
  ”StateCode” : ”f”,
  ”WorkDefinitionID” : ”3001”,
  ”WorkDefinitionName” : ”QuotationHandling_UTask1”,
  ”WorkTypeCode” : ”0”
}
```

11.5.29　作業の返却

作業者を割り当てて着手した作業の状態を，「作業者実行」状態から「実行開始可能」状態に変更し，作業者をレーン名に戻します。この API を実行すると，BPMN 連携ライブラリ Java API のCIWBPMNLib.freeWIインタフェースが呼び出されます。作業が返却済みだった場合，リクエストは成功します（何もしないでステータスコード 200 が返されます）。詳細については，「5.3　Java API 利用時の注意事項」の「べき等性について」の説明を参照してください。

メソッドと URL

```
PUT /v1/work-item/<案件ID>/<作業ID>/free
```

<案件 ID>：返却する作業の案件 ID（必須）

<作業 ID>：返却する作業の ID（必須）

リクエストボディ

項番	名前	型	指定要否	内容
1	UserDescription	文字列	省略可	ユーザ記述子を指定します。省略した場合はデフォルト値が指定されます。空文字列は指定できません。

リクエストボディの構造を次に示します。UserDescription を省略する場合は，リクエストボディを省略できます。リクエストボディを省略するときは，HTTP ヘッダのContent-Type を指定しないでください。

項番	名前	出現回数
1	Parameter	1

項番	名前		出現回数
2		UserDescription	0 または 1

リクエスト例

リクエスト URL

```
PUT http://restserver/csciwws/v1/work-item/5004/10005/free
```

レスポンス

返却した作業を返します。作業のプロパティを次に示します。

項番	名前	型	内容
1	ActivityInstanceID	数値	作業が所属する業務ステップの ID
2	ClosedDate	日付	作業の終了日時
3	CreationDate	日付	作業の発生日時
4	Deadline	日付	作業の処理期限の絶対日時
5	ID	数値	作業の ID
6	Name	文字列	作業の名称（作業キー）
7	Participant	文字列	作業の作業者 ID
8	Priority	数値	作業の優先度
9	ProcessDefinitionID	数値	作業が所属するビジネスプロセス定義の ID
10	ProcessInstanceID	数値	作業が所属する案件の ID
11	ProcessInstanceName	文字列	作業の案件名（案件キー）
12	StartDate	日付	作業の開始日時
13	StateCode	文字列	作業の状態
14	WorkDefinitionID	数値	作業定義の ID
15	WorkDefinitionName	文字列	作業定義の名称
16	WorkTypeCode	文字列	作業の種類

レスポンスの構造を次に示します。

項番	名前		出現回数
1	WorkItem		1
2		ActivityInstanceID	1
3		ClosedDate	1

項番	名前		出現回数
4		CreationDate	1
5		Deadline	1
6		ID	1
7		Name	1
8		Participant	1
9		Priority	1
10		ProcessDefinitionID	1
11		ProcessInstanceID	1
12		ProcessInstanceName	1
13		StartDate	1
14		StateCode	1
15		WorkDefinitionID	1
16		WorkDefinitionName	1
17		WorkTypeCode	1

ステータスコード

項番	ステータスコード	内容
1	200	成功
2	400	リクエストパラメタの不正
3	404	データが見つからなかった
4	500	内部処理エラー

レスポンス例

レスポンス（XML の場合）

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<WorkItem>
  <ActivityInstanceID>10005</ActivityInstanceID>
  <ClosedDate></ClosedDate>
  <CreationDate>2016-12-09T15:54:55+09:00</CreationDate>
  <Deadline></Deadline>
  <ID>10005</ID>
  <Name></Name>
  <Participant>MyLane1</Participant>
  <Priority>0</Priority>
  <ProcessDefinitionID>3001</ProcessDefinitionID>
  <ProcessInstanceID>5004</ProcessInstanceID>
```

```
  <ProcessInstanceName>案件20160609103256259</ProcessInstanceName>
  <StartDate>2016-12-09T15:58:52+09:00</StartDate>
  <StateCode>j</StateCode>
  <WorkDefinitionID>3001</WorkDefinitionID>
  <WorkDefinitionName>QuotationHandling_UTask1</WorkDefinitionName>
  <WorkTypeCode>0</WorkTypeCode>
</WorkItem>
```

レスポンス（JSON の場合）

```
{
  "ActivityInstanceID" : "10005",
  "ClosedDate" : "",
  "CreationDate" : "2016-12-09T15:54:55+09:00",
  "Deadline" : "",
  "ID" : "10005",
  "Name" : "",
  "Participant" : " MyLane1",
  "Priority" : "0",
  "ProcessDefinitionID" : "3001",
  "ProcessInstanceID" : "5004",
  "ProcessInstanceName" : "案件20160609103256259",
  "StartDate" : "2016-12-09T15:58:52+09:00",
  "StateCode" : "j",
  "WorkDefinitionID" : "3001",
  "WorkDefinitionName" : "QuotationHandling_UTask1",
  "WorkTypeCode" : "0"
}
```

11.5.30　タイマーの処理期限の変更

タイマーイベントから変換された作業について，指定した作業の処理期限を変更します。この API を実行すると，BPMN 連携ライブラリ Java API の`CIWBPMNLib.setDeadlineForTimer` インタフェースが呼び出されます。

メソッドと URL

```
PUT /v1/work-item/<案件ID>/<作業ID>/timer/set-deadline
```

<案件 ID>：処理期限を変更するタイマーの作業の案件 ID（必須）

<作業 ID>：処理期限を変更するタイマーの作業の ID（必須）

リクエストボディ

項番	名前	型	指定要否	内容
1	UserDescription	文字列	省略可	ユーザ記述子を指定します。省略した場合はデフォルト値が指定されます。空文字列は指定できません。

項番	名前	型	指定要否	内容
2	Deadline	文字列	必須	作業の処理期限の絶対日時を指定します。空文字は指定できません。

リクエストボディの構造を次に示します。

項番	名前		出現回数
1	Parameter		1
2		UserDescription	0 または 1
3		Deadline	1

リクエスト例

リクエスト URL

```
PUT http://restserver/csciwws/v1/work-item/5004/10005/timer/set-deadline
```

リクエストボディ（XML の場合）

```
<?xml version="1.0" encoding="UTF-8" ?>
<Parameter>
  <Deadline>2018-06-01T10:30:00+09:00</Deadline>
</Parameter>
```

リクエストボディ（JSON の場合）

```
{
  "Deadline" : "2018-06-01T10:30:00+09:00"
}
```

レスポンス

状態を変更した作業を返します。作業のプロパティを次に示します。

項番	名前	型	内容
1	ActivityInstanceID	数値	作業が所属する業務ステップの ID
2	ClosedDate	日付	作業の終了日時
3	CreationDate	日付	作業の発生日時
4	Deadline	日付	作業の処理期限の絶対日時
5	ID	数値	作業の ID
6	Name	文字列	作業の名称（作業キー）
7	Participant	文字列	作業の作業者 ID

項番	名前	型	内容
8	Priority	数値	作業の優先度
9	ProcessDefinitionID	数値	作業が所属するビジネスプロセス定義の ID
10	ProcessInstanceID	数値	作業が所属する案件の ID
11	ProcessInstanceName	文字列	作業の案件名（案件キー）
12	StartDate	日付	作業の開始日時
13	StateCode	文字列	作業の状態
14	WorkDefinitionID	数値	作業定義の ID
15	WorkDefinitionName	文字列	作業定義の名称
16	WorkTypeCode	文字列	作業の種類

レスポンスの構造を次に示します。

項番	名前		出現回数
1	WorkItem		1
2		ActivityInstanceID	1
3		ClosedDate	1
4		CreationDate	1
5		Deadline	1
6		ID	1
7		Name	1
8		Participant	1
9		Priority	1
10		ProcessDefinitionID	1
11		ProcessInstanceID	1
12		ProcessInstanceName	1
13		StartDate	1
14		StateCode	1
15		WorkDefinitionID	1
16		WorkDefinitionName	1
17		WorkTypeCode	1

ステータスコード

項番	ステータスコード	内容
1	200	成功
2	400	リクエストパラメタの不正
3	404	データが見つからなかった
4	500	内部処理エラー

レスポンス例

レスポンス（XML の場合）

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<WorkItem>
  <ActivityInstanceID>10005</ActivityInstanceID>
  <ClosedDate></ClosedDate>
  <CreationDate>2016-12-09T15:54:55+09:00</CreationDate>
  <Deadline>2018-06-01T10:30:00+09:00</Deadline>
  <ID>10005</ID>
  <Name></Name>
  <Participant>User001</Participant>
  <Priority>0</Priority>
  <ProcessDefinitionID>3001</ProcessDefinitionID>
  <ProcessInstanceID>5004</ProcessInstanceID>
  <ProcessInstanceName>案件20160609103256259</ProcessInstanceName>
  <StartDate>2016-12-09T15:58:52+09:00</StartDate>
  <StateCode>j</StateCode>
  <WorkDefinitionID>3001</WorkDefinitionID>
  <WorkDefinitionName>QuotationHandling_UTask1</WorkDefinitionName>
  <WorkTypeCode>0</WorkTypeCode>
</WorkItem>
```

レスポンス（JSON の場合）

```
{
  "ActivityInstanceID" : "10005",
  "ClosedDate" : "",
  "CreationDate" : "2016-12-09T15:54:55+09:00",
  "Deadline" : "2018-06-01T10:30:00+09:00",
  "ID" : "10005",
  "Name" : "",
  "Participant" : "User001",
  "Priority" : "0",
  "ProcessDefinitionID" : "3001",
  "ProcessInstanceID" : "5004",
  "ProcessInstanceName" : "案件20160609103256259",
  "StartDate" : "2016-12-09T15:58:52+09:00",
  "StateCode" : "j",
  "WorkDefinitionID" : "3001",
  "WorkDefinitionName" : "QuotationHandling_UTask1",
```

```
  "WorkTypeCode" : "0"
}
```

11.5.31　子案件の取得

子案件を取得します。具体的には，指定したコールアクティビティが投入した案件を取得します。この API を実行すると，BPMN 連携ライブラリ Java API の CIWBPMNLib.getCallActivityChildPI インタフェースが呼び出されます。

メソッドと URL

```
GET /v1/work-item/<案件ID>/<作業ID>/call-activity-child/process-instance
```

<案件 ID>：コールアクティビティの案件の ID（必須）

<作業 ID>：コールアクティビティの作業の ID（必須）

クエリパラメタ

項番	名前	型	指定要否	内容
1	userdescription	文字列	省略可	ユーザ記述子を指定します。省略した場合はデフォルト値が指定されます。空文字列は指定できません。

リクエスト例

リクエスト URL

```
GET http://restserver/csciwws/v1/work-item/5005/32001/call-activity-child/process-instance
```

レスポンス

子案件を返します。子案件のプロパティを次に示します。

項番	名前	型	内容
1	ClosedDate	日付	子案件の終了日時
2	Creator	文字列	子案件の投入者
3	Deadline	日付	子案件の処理期限の絶対日時
4	ID	数値	子案件の ID
5	MovedDate	日付	子案件の乗せ替え日時
6	Name	文字列	子案件の名称（案件キー）

項番	名前	型	内容
7	Priority	数値	子案件の優先度
8	ProcessDefinitionID	数値	子案件が所属するビジネスプロセス定義の ID
9	ProcessDefinitionName	文字列	子案件が所属するビジネスプロセス定義の名称
10	StartDate	日付	子案件の開始日時
11	StateCode	文字列	子案件の状態

レスポンスの構造を次に示します。

項番	名前		出現回数
1	ProcessInstance		1
2		ClosedDate	1
3		Creator	1
4		Deadline	1
5		ID	1
6		MovedDate	1
7		Name	1
8		Priority	1
9		ProcessDefinitionID	1
10		ProcessDefinitionName	1
11		StartDate	1
12		StateCode	1

ステータスコード

項番	ステータスコード	内容
1	200	成功
2	400	リクエストパラメタの不正
3	404	データが見つからなかった
4	500	内部処理エラー

レスポンス例

レスポンス（XML の場合）

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ProcessInstance>
```

```
  <ClosedDate></ClosedDate>
  <Creator>csciwws</Creator>
  <Deadline></Deadline>
  <ID>105</ID>
  <MovedDate></MovedDate>
  <Name></Name>
  <Priority>0</Priority>
  <ProcessDefinitionID>3001</ProcessDefinitionID>
  <ProcessDefinitionName>販売業務</ProcessDefinitionName>
  <StartDate>2016-12-06T15:31:34+09:00</StartDate>
  <StateCode>d</StateCode>
</ProcessInstance>
```

レスポンス（JSON の場合）

```
{
  "ClosedDate" : "",
  "Creator" : "csciwws",
  "Deadline" : "",
  "ID" : "105",
  "MovedDate" : "",
  "Name" : "",
  "Priority" : "0",
  "ProcessDefinitionID" : "3001",
  "ProcessDefinitionName" : "販売業務",
  "StartDate" : "2016-12-06T15:31:34+09:00",
  "StateCode" : "d"
}
```

11.5.32　ビジネスプロセス定義の一覧取得

指定したフィルター条件を満たす，ビジネスプロセス定義の一覧を取得します。この API を実行すると，CSCIW Java API の`CIWServer.getProcessDefinitionsList` インタフェースが呼び出されます。

メソッドと URL

```
GET /v1/process-definition
```

クエリパラメタ

項番	名前	型	指定要否	内容
1	userdescription	文字列	省略可	ユーザ記述子を指定します。省略した場合はデフォルト値が指定されます。空文字列は指定できません。
2	filter	文字列	省略可	ビジネスプロセス定義一覧を取得する場合のフィルター条件を指定します。省略した場合はフィルター条件を指定しません。空文字列は指定できません。

項番	名前	型	指定要否	内容
3	sort	文字列	省略可	取得したビジネスプロセス定義一覧をソートする場合の条件を指定します。省略した場合はソート条件を指定しません。空文字列は指定できません。
4	offset	数値	省略可	ビジネスプロセス定義一覧を取得する場合のオフセットを指定します。先頭は 0 になります。省略した場合は 0 が指定されます。0 未満の値は指定できません。
5	maxcount	数値	省略可	ビジネスプロセス定義一覧を取得する場合の最大取得数を指定します。省略した場合はデフォルト値が指定されます。すべてを取得する場合は，-1 を指定します。-1 未満の値は指定できません。

リクエスト例

リクエスト URL

```
GET http://restserver/csciwws/v1/process-definition?filter=ID%3D1%20OR%20ID%3D3
```

レスポンス

ビジネスプロセス定義一覧を返します。個々のビジネスプロセス定義のプロパティを次に示します。

項番	名前	型	内容
1	Author	文字列	ビジネスプロセス定義の作成者
2	CreationDate	日付	ビジネスプロセス定義の作成日
3	Description	文字列	ビジネスプロセス定義の説明
4	ID	数値	ビジネスプロセス定義の ID
5	Name	文字列	ビジネスプロセス定義の名称
6	Responsible	文字列	ビジネスプロセス定義の管理者 ID
7	StateCode	文字列	ビジネスプロセス定義の状態
8	ValidFromDate	日付	ビジネスプロセス定義の有効となる日
9	ValidToDate	日付	ビジネスプロセス定義の無効となる日
10	Version	数値	ビジネスプロセス定義のバージョン

レスポンスの構造を次に示します。

項番	名前		出現回数
1	ProcessDefinitions		1
2		ProcessDefinition	0 または 1 以上

項番	名前			出現回数
3			Author	1
4			CreationDate	1
5			Description	1
6			ID	1
7			Name	1
8			Responsible	1
9			StateCode	1
10			ValidFromDate	1
11			ValidToDate	1
12			Version	1

ステータスコード

項番	ステータスコード	内容
1	200	成功（データが 0 件の場合も成功）
2	400	リクエストパラメタの不正
3	500	内部処理エラー

レスポンス例

レスポンス（XML の場合）

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ProcessDefinitions>
  <ProcessDefinition>
    <Author>user1</Author>
    <CreationDate>2016-12-06T15:18:34+09:00</CreationDate>
    <Description></Description>
    <ID>1</ID>
    <Name>販売業務</Name>
    <Responsible>user1</Responsible>
    <StateCode>b</StateCode>
    <ValidFromDate>ORIGIN</ValidFromDate>
    <ValidToDate>BEYOND</ValidToDate>
    <Version>1</Version>
  </ProcessDefinition>
  <ProcessDefinition>
    <Author>user1</Author>
    <CreationDate>2016-12-06T15:18:35+09:00</CreationDate>
    <Description></Description>
    <ID>3</ID>
    <Name>販売業務</Name>
    <Responsible>user1</Responsible>
```

```
    <StateCode>b</StateCode>
    <ValidFromDate>ORIGIN</ValidFromDate>
    <ValidToDate>BEYOND</ValidToDate>
    <Version>2</Version>
  </ProcessDefinition>
</ProcessDefinitions>
```

レスポンス（JSON の場合）

```
{
  "ProcessDefinition" : [
    {
      "Author" : "user1",
      "CreationDate" : "2016-12-06T15:18:34+09:00",
      "Description" : "",
      "ID" : "1",
      "Name" : "販売業務",
      "Responsible" : "user1",
      "StateCode" : "b",
      "ValidFromDate" : "ORIGIN",
      "ValidToDate" : "BEYOND",
      "Version" : "1"
    },
    {
      "Author" : "user1",
      "CreationDate" : "2016-12-06T15:18:35+09:00",
      "Description" : "",
      "ID" : "3",
      "Name" : "販売業務",
      "Responsible" : "user1",
      "StateCode" : "b",
      "ValidFromDate" : "ORIGIN",
      "ValidToDate" : "BEYOND",
      "Version" : "2"
    }
  ]
}
```

11.5.33　ビジネスプロセス定義数の取得

指定した条件を満たすビジネスプロセス定義の数を取得します。この API を実行すると，CSCIW Java API の CIWServer.getListCount インタフェースが呼び出されます。

メソッドと URL

```
GET /v1/process-definition/count
```

クエリパラメタ

項番	名前	型	指定要否	内容
1	userdescription	文字列	省略可	ユーザ記述子を指定します。省略した場合はデフォルト値が指定されます。空文字列は指定できません。
2	filter	文字列	省略可	ビジネスプロセス定義数を取得する場合のフィルター条件を指定します。省略した場合はフィルター条件は指定されません。空文字列は指定できません。

リクエスト例

リクエスト URL

```
GET http://restserver/csciwws/v1/process-definition/count
```

レスポンス

ビジネスプロセス定義数を返します。レスポンスとして返されるプロパティを次に示します。

項番	名前	型	内容
1	Count	数値	ビジネスプロセス定義の数

レスポンスの構造を次に示します。

項番	名前	出現回数
1	Resources	1
2	Count	1

ステータスコード

項番	ステータスコード	内容
1	200	成功（データが 0 件の場合も成功）
2	400	リクエストパラメタの不正
3	500	内部処理エラー

レスポンス例

レスポンス（XML の場合）

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Resources>
  <Count>15</Count>
</Resources>
```

レスポンス（JSON の場合）

```
{
  "Count" : "15"
}
```

11.5.34　ビジネスプロセス定義の取得

指定した ID のビジネスプロセス定義を取得します。この API を実行すると，CSCIW Java API の CIWServer.getProcessDefinition インタフェースが呼び出されます。

メソッドと URL

```
GET /v1/process-definition/<ビジネスプロセス定義のID>
```

<ビジネスプロセス定義の ID>：取得するビジネスプロセス定義の ID（必須）

クエリパラメタ

項番	名前	型	指定要否	内容
1	userdescription	文字列	省略可	ユーザ記述子を指定します。省略した場合はデフォルト値が指定されます。空文字列は指定できません。

リクエスト例

リクエスト URL

```
GET http://restserver/csciwws/v1/process-definition/3
```

レスポンス

ビジネスプロセス定義を返します。レスポンスとして返されるプロパティを次に示します。

項番	名前	型	内容
1	Author	文字列	ビジネスプロセス定義の作成者
2	CreationDate	日付	ビジネスプロセス定義の作成日
3	Description	文字列	ビジネスプロセス定義の説明
4	ID	数値	ビジネスプロセス定義の ID
5	Name	文字列	ビジネスプロセス定義の名称
6	Responsible	文字列	ビジネスプロセス定義の管理者 ID

項番	名前	型	内容
7	StateCode	文字列	ビジネスプロセス定義の状態
8	ValidFromDate	日付	ビジネスプロセス定義の有効となる日
9	ValidToDate	日付	ビジネスプロセス定義の無効となる日
10	Version	数値	ビジネスプロセス定義のバージョン

レスポンスの構造を次に示します。

項番	名前		出現回数
1	ProcessDefinition		1
2		Author	1
3		CreationDate	1
4		Description	1
5		ID	1
6		Name	1
7		Responsible	1
8		StateCode	1
9		ValidFromDate	1
10		ValidToDate	1
11		Version	1

ステータスコード

項番	ステータスコード	内容
1	200	成功
2	400	リクエストパラメタの不正
3	404	データが見つからなかった
4	500	内部処理エラー

レスポンス例

レスポンス（XML の場合）

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ProcessDefinition>
  <Author>user1</Author>
  <CreationDate>2016-12-06T15:18:35+09:00</CreationDate>
  <Description></Description>
```

```
  <ID>3</ID>
  <Name>販売業務</Name>
  <Responsible>user1</Responsible>
  <StateCode>b</StateCode>
  <ValidFromDate>ORIGIN</ValidFromDate>
  <ValidToDate>BEYOND</ValidToDate>
  <Version>2</Version>
</ProcessDefinition>
```

レスポンス（JSON の場合）

```
{
  "Author" : "user1",
  "CreationDate" : "2016-12-06T15:18:35+09:00",
  "Description" : "",
  "ID" : "3",
  "Name" : "販売業務",
  "Responsible" : "user1",
  "StateCode" : "b",
  "ValidFromDate" : "ORIGIN",
  "ValidToDate" : "BEYOND",
  "Version" : "2"
}
```

11.5.35　ビジネスプロセス定義内の案件の一覧取得

指定したフィルター条件を満たす，ビジネスプロセス定義内の案件一覧を取得します。この API を実行すると，CSCIW Java API の`CIWProcessDefinition.getProcessInstancesList` インタフェースが呼び出されます。

メソッドと URL

```
GET /v1/process-definition/<ビジネスプロセス定義のID>/process-instance
```

<ビジネスプロセス定義の ID>：取得する案件一覧のビジネスプロセス定義の ID（必須）

クエリパラメタ

項番	名前	型	指定要否	内容
1	userdescription	文字列	省略可	ユーザ記述子を指定します。省略した場合はデフォルト値が指定されます。空文字列は指定できません。
2	filter	文字列	省略可	案件一覧を取得する場合のフィルター条件を指定します。省略した場合はフィルター条件を指定しません。空文字列は指定できません。
3	sort	文字列	省略可	取得した案件一覧をソートする場合の条件を指定します。省略した場合はソート条件を指定しません。空文字列は指定できません。

項番	名前	型	指定要否	内容
4	offset	数値	省略可	案件一覧を取得する場合のオフセットを指定します。先頭は 0 になります。省略した場合は 0 が指定されます。0 未満の値は指定できません。
5	maxcount	数値	省略可	案件一覧を取得する場合の最大取得数を指定します。省略した場合はデフォルト値が指定されます。すべてを取得する場合は，-1 を指定します。-1 未満の値は指定できません。

リクエスト例

リクエスト URL

```
GET http://restserver/csciwws/v1/process-definition/3/process-instance?filter=NAME%3D%27PI0001%27
```

レスポンス

案件一覧を返します。個々の案件のプロパティを次に示します。

項番	名前	型	内容
1	ClosedDate	日付	案件の終了日時
2	Creator	文字列	案件の投入者
3	Deadline	日付	案件の処理期限の絶対日時
4	ID	数値	案件の ID
5	MovedDate	日付	案件の乗せ替え日時
6	Name	文字列	案件の名称（案件キー）
7	Priority	数値	案件の優先度
8	ProcessDefinitionID	数値	案件が所属するビジネスプロセス定義の ID
9	ProcessDefinitionName	文字列	案件が所属するビジネスプロセス定義の名称
10	StartDate	日付	案件の開始日時
11	StateCode	文字列	案件の状態

レスポンスの構造を次に示します。

項番	名前			出現回数
1	ProcessInstances			1
2		ProcessInstance		0 または 1 以上
3			ClosedDate	1

項番	名前			出現回数
4			Creator	1
5			Deadline	1
6			ID	1
7			MovedDate	1
8			Name	1
9			Priority	1
10			ProcessDefinitionID	1
11			ProcessDefinitionName	1
12			StartDate	1
13			StateCode	1

ステータスコード

項番	ステータスコード	内容
1	200	成功（データが 0 件の場合も成功）
2	400	リクエストパラメタの不正
3	404	データが見つからなかった
4	500	内部処理エラー

レスポンス例

レスポンス（XML の場合）

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ProcessInstances>
  <ProcessInstance>
    <ClosedDate></ClosedDate>
    <Creator>csciwws</Creator>
    <Deadline></Deadline>
    <ID>5001</ID>
    <MovedDate></MovedDate>
    <Name>PI0001</Name>
    <Priority>0</Priority>
    <ProcessDefinitionID>3</ProcessDefinitionID>
    <ProcessDefinitionName>販売業務</ProcessDefinitionName>
    <StartDate>2016-12-09T15:00:41+09:00</StartDate>
    <StateCode>d</StateCode>
  </ProcessInstance>
</ProcessInstances>
```

レスポンス（JSON の場合）

```
{
  "ProcessInstance" : [
    {
      "ClosedDate" : "",
      "Creator" : "csciwws",
      "Deadline" : "",
      "ID" : "5001",
      "MovedDate" : "",
      "Name" : "PI0001",
      "Priority" : "0",
      "ProcessDefinitionID" : "3",
      "ProcessDefinitionName" : "販売業務",
      "StartDate" : "2016-12-09T15:00:41+09:00",
      "StateCode" : "d"
    }
  ]
}
```

11.5.36　作業定義の一覧取得

指定したフィルター条件を満たす，作業定義の一覧を取得します。この API を実行すると，CSCIW Java API のCIWServer.getWorkDefinitionsList インタフェースが呼び出されます。

メソッドと URL

```
GET /v1/work-definition
```

クエリパラメタ

項番	名前	型	指定要否	内容
1	userdescription	文字列	省略可	ユーザ記述子を指定します。省略した場合はデフォルト値が指定されます。空文字列は指定できません。
2	filter	文字列	省略可	作業定義一覧を取得する場合のフィルター条件を指定します。省略した場合はフィルター条件を指定しません。空文字列は指定できません。
3	sort	文字列	省略可	取得した作業定義一覧をソートする場合の条件を指定します。省略した場合はソート条件を指定しません。空文字列は指定できません。
4	offset	数値	省略可	作業定義一覧を取得する場合のオフセットを指定します。

項番	名前	型	指定要否	内容
4	offset	数値	省略可	先頭は 0 になります。省略した場合は 0 が指定されます。0 未満の値は指定できません。
5	maxcount	数値	省略可	作業定義一覧を取得する場合の最大取得数を指定します。省略した場合はデフォルト値が指定されます。すべてを取得する場合は，-1 を指定します。-1 未満の値は指定できません。

リクエスト例

リクエスト URL

```
GET http://restserver/csciwws/v1/work-definition?filter=ProcessDefinitionID%3D1
```

レスポンス

作業定義一覧を返します。個々の作業定義のプロパティを次に示します。

項番	名前	型	内容
1	ActivityDefinitionID	数値	作業定義が所属する業務ステップ定義の ID
2	CastingRuleName	文字列	作業定義の振り分けルール名
3	Description	文字列	作業の説明
4	ID	数値	作業定義の ID
5	Name	文字列	作業定義の名称
6	ProcessDefinitionID	数値	作業定義が所属するビジネスプロセス定義の ID
7	WorkTypeCode	文字列	作業の種類

レスポンスの構造を次に示します。

項番	名前			出現回数
1	WorkDefinitions			1
2		WorkDefinition		0 または 1 以上
3			ActivityDefinitionID	1
4			CastingRuleName	1
5			Description	1

項番	名前			出現回数
6			ID	1
7			Name	1
8			ProcessDefinitionID	1
9			WorkTypeCode	1

ステータスコード

項番	ステータスコード	内容
1	200	成功（データが 0 件の場合も成功
2	400	リクエストパラメタの不正
3	500	内部処理エラー

レスポンス例

レスポンス（XML の場合）

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<WorkDefinitions>
  <WorkDefinition>
    <ActivityDefinitionID>1</ActivityDefinitionID>
    <CastingRuleName></CastingRuleName>
    <Description>USER_TASK</Description>
    <ID>1</ID>
    <Name>QuotationHandling_UTask1</Name>
    <ProcessDefinitionID>1</ProcessDefinitionID>
    <WorkTypeCode>0</WorkTypeCode>
  </WorkDefinition>
  <WorkDefinition>
    <ActivityDefinitionID>2</ActivityDefinitionID>
    <CastingRuleName></CastingRuleName>
    <Description>USER_TASK</Description>
    <ID>2</ID>
    <Name>ApproveOrder_UTask2</Name>
    <ProcessDefinitionID>1</ProcessDefinitionID>
    <WorkTypeCode>0</WorkTypeCode>
  </WorkDefinition>
</WorkDefinitions>
```

レスポンス（JSON の場合）

```
{
  "WorkDefinition" : [
    {
      "ActivityDefinitionID" : "1",
      "CastingRuleName" : "",
      "Description" : "USER_TASK",
      "ID" : "1",
```

```
      "Name" : "QuotationHandling_UTask1",
      "ProcessDefinitionID" : "1",
      "WorkTypeCode" : "0"
    },
    {
      "ActivityDefinitionID" : "2",
      "CastingRuleName" : "",
      "Description" : "USER_TASK",
      "ID" : "2",
      "Name" : "ApproveOrder_UTask2",
      "ProcessDefinitionID" : "1",
      "WorkTypeCode" : "0"
    }
  ]
}
```

11.5.37　作業定義数の取得

指定した条件を満たす作業定義の数を取得します。この API を実行すると，CSCIW Java API の CIWServer.getListCount インタフェースが呼び出されます。

メソッドと URL

```
GET /v1/work-definition/count
```

クエリパラメタ

項番	名前	型	指定要否	内容
1	userdescription	文字列	省略可	ユーザ記述子を指定します。省略した場合はデフォルト値が指定されます。空文字列は指定できません。
2	filter	文字列	省略可	作業定義数を取得する場合のフィルター条件を指定します。省略した場合はフィルター条件は指定されません。空文字列は指定できません。

リクエスト例

リクエスト URL

```
GET http://restserver/csciwws/v1/work-definition/count
```

レスポンス

作業定義数を返します。レスポンスとして返されるプロパティを次に示します。

項番	名前	型	内容
1	Count	数値	作業定義の数

レスポンスの構造を次に示します。

項番	名前	出現回数
1	Resources	1
2	Count	1

ステータスコード

項番	ステータスコード	内容
1	200	成功（データが 0 件の場合も成功）
2	400	リクエストパラメタの不正
3	500	内部処理エラー

レスポンス例

レスポンス（XML の場合）

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Resources>
  <Count>15</Count>
</Resources>
```

レスポンス（JSON の場合）

```
{
  "Count" : "15"
}
```

11.5.38　作業定義の取得

指定した ID の作業定義を取得します。この API を実行すると，CSCIW Java API の CIWServer.getWorkDefinition インタフェースが呼び出されます。

メソッドと URL

```
GET /v1/work-definition/<ビジネスプロセス定義のID>/<作業定義のID>
```

<ビジネスプロセス定義の ID>：取得する作業定義のビジネスプロセス定義の ID（必須）

<作業定義の ID>：取得する作業定義の ID（必須）

クエリパラメタ

項番	名前	型	指定要否	内容
1	userdescription	文字列	省略可	ユーザ記述子を指定します。省略した場合はデフォルト値が指定されます。空文字列は指定できません。

リクエスト例

リクエスト URL

```
GET http://restserver/csciwws/v1/work-definition/1/5
```

レスポンス

作業定義を返します。作業定義のプロパティを次に示します。

項番	名前	型	内容
1	ActivityDefinitionID	数値	作業定義が所属する業務ステップ定義の ID
2	CastingRuleName	文字列	作業定義の振り分けルール名
3	Description	文字列	作業の説明
4	ID	数値	作業定義の ID
5	Name	文字列	作業定義の名称
6	ProcessDefinitionID	数値	作業定義が所属するビジネスプロセス定義の ID
7	WorkTypeCode	文字列	作業の種類

レスポンスの構造を次に示します。

項番	名前		出現回数
1	WorkDefinition		1
2		ActivityDefinitionID	1
3		CastingRuleName	1
4		Description	1
5		ID	1
6		Name	1
7		ProcessDefinitionID	1
8		WorkTypeCode	1

ステータスコード

項番	ステータスコード	内容
1	200	成功
2	400	リクエストパラメタの不正
3	404	データが見つからなかった
4	500	内部処理エラー

レスポンス例

レスポンス（XML の場合）

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<WorkDefinition>
  <ActivityDefinitionID>3</ActivityDefinitionID>
  <CastingRuleName></CastingRuleName>
  <Description>USER_TASK</Description>
  <ID>5</ID>
  <Name>QuotationHandling_UTask1</Name>
  <ProcessDefinitionID>1</ProcessDefinitionID>
  <WorkTypeCode>0</WorkTypeCode>
</WorkDefinition>
```

レスポンス（JSON の場合）

```
{
  "ActivityDefinitionID" : "3",
  "CastingRuleName" : "",
  "Description" : "USER_TASK",
  "ID" : "5",
  "Name" : "QuotationHandling_UTask1",
  "ProcessDefinitionID" : "1",
  "WorkTypeCode" : "0"
}
```

11.5.39　BPMN ビジネスプロセス定義ファイルの取得

指定した案件の BPMN ビジネスプロセス定義ファイルを取得します。案件 ID に関連するビジネスプロセス名とビジネスプロセス定義バージョンから，対象となる BPMN ビジネスプロセス定義ファイルを取得します。指定した案件の BPMN ビジネスプロセス定義ファイルの内容がワーク管理データベースに登録済みの場合は，ワーク管理データベースから BPMN ビジネスプロセス定義ファイルの内容を取得します。未登録の場合は，**<環境変数CSCIW_HOME>**/bpmn/bpmnxml に配置された BPMN ビジネスプロセス定義ファイルを取得します。

メモ

- この API で取得する BPMN ビジネスプロセス定義ファイルは，BPMN ビジネスプロセス定義の変換コマンド（ciwtransbpmn）の成果物である BPMN ビジネスプロセス定義ファイルです。
- この API で取得できる BPMN ビジネスプロセス定義ファイルの文字コードは UTF-8 だけです。
- HTTP ヘッダの"Accept"には，application/xml だけが指定できます（Accept を省略した場合，application/xml が適用されます）。

ヒント

BPMN ビジネスプロセス定義ファイルの格納先ディレクトリは，共通設定ファイルの「BpmnDefinitionFileDir」で変更できます。BpmnDefinitionFileDir については，「15.2.5　共通設定ファイルに指定する内容」の「(2)　BpmnDefinitionFileDir」を参照してください。

メソッドと URL

```
GET /v1/bpmn/<案件ID>
```

<案件 ID>：取得する BPMN ビジネスプロセス定義ファイルの案件 ID（必須）

クエリパラメタ

項番	名前	型	指定要否	内容
1	userdescription	文字列	省略可	ユーザ記述子を指定します。省略した場合はデフォルト値が指定されます。空文字列は指定できません。

リクエスト例

リクエスト URL

```
GET http://restserver/csciwws/v1/bpmn/5005
```

レスポンス

BPMN ビジネスプロセス定義ファイルの内容を返します。

ステータスコード

項番	ステータスコード	内容
1	200	成功
2	400	リクエストパラメタの不正
3	404	データが見つからなかった
4	500	内部処理エラー

レスポンス例

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions xmlns="http://www.omg.org/spec/BPMN/20100524/MODEL" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:activiti="http://activiti.org/bpmn" xmlns:bpmndi="http://www.omg.org/spec/BPMN/20100524/DI" xmlns:omgdc="http://www.omg.org/spec/DD/20100524/DC" xmlns:omgdi="http://www.omg.org/spec/DD/20100524/DI" xmlns:csciw="http://www.hitachi.co.jp/soft/xml/csciw/bpmn" typeLanguage="http://www.w3.org/2001/XMLSchema" expressionLanguage="http://www.w3.org/1999/XPath" targetNamespace="http://www.activiti.org/test" exporter="uCosminexus Business Process Developer" exporterVersion="02-20" csciw:userCheck="4">
  <collaboration id="Collaboration">
    <participant id="pool1" name="RestAP" processRef="process_pool1"></participant>
  </collaboration>
  <process id="process_pool1" name="RestAP" isExecutable="true">
    <laneSet id="laneSet_process_pool1">
      <lane id="lane1">
        <flowNodeRef>Start1</flowNodeRef>
        <flowNodeRef>STask1</flowNodeRef>
        <flowNodeRef>End1</flowNodeRef>
      </lane>
    </laneSet>
    <startEvent id="Start1" name="TopLevelStart"></startEvent>
    <serviceTask id="STask1" name="ServiceTaskA" operationRef="RestCall"></serviceTask>
    <endEvent id="End1" name="NoneEnd"></endEvent>
    <sequenceFlow id="flow1" sourceRef="Start1" targetRef="STask1"></sequenceFlow>
    <sequenceFlow id="flow2" sourceRef="STask1" targetRef="End1"></sequenceFlow>
  </process>
  <bpmndi:BPMNDiagram id="BPMNDiagram_Collaboration">
    <bpmndi:BPMNPlane bpmnElement="Collaboration" id="BPMNPlane_Collaboration">
      <bpmndi:BPMNShape bpmnElement="pool1" id="BPMNShape_pool1">
        <omgdc:Bounds height="281.0" width="501.0" x="0.0" y="80.0"></omgdc:Bounds>
      </bpmndi:BPMNShape>
      <bpmndi:BPMNShape bpmnElement="lane1" id="BPMNShape_lane1">
        <omgdc:Bounds height="281.0" width="481.0" x="20.0" y="80.0"></omgdc:Bounds>
      </bpmndi:BPMNShape>
      <bpmndi:BPMNShape bpmnElement="Start1" id="BPMNShape_Start1">
        <omgdc:Bounds height="35.0" width="35.0" x="50.0" y="193.0"></omgdc:Bounds>
        <bpmndi:BPMNLabel>
          <omgdc:Bounds height="18.0" width="71.0" x="32.0" y="230.0"></omgdc:Bounds>
        </bpmndi:BPMNLabel>
      </bpmndi:BPMNShape>
      <bpmndi:BPMNShape bpmnElement="STask1" id="BPMNShape_STask1">
        <omgdc:Bounds height="55.0" width="105.0" x="180.0" y="183.0"></omgdc:Bounds>
      </bpmndi:BPMNShape>
```

```
      <bpmndi:BPMNShape bpmnElement="End1" id="BPMNShape_End1">
        <omgdc:Bounds height="35.0" width="35.0" x="400.0" y="193.0"></omgdc:Bounds>
        <bpmndi:BPMNLabel>
          <omgdc:Bounds height="18.0" width="47.0" x="394.0" y="230.0"></omgdc:Bounds>
        </bpmndi:BPMNLabel>
      </bpmndi:BPMNShape>
      <bpmndi:BPMNEdge bpmnElement="flow1" id="BPMNEdge_flow1">
        <omgdi:waypoint x="85.0" y="210.0"></omgdi:waypoint>
        <omgdi:waypoint x="180.0" y="210.0"></omgdi:waypoint>
      </bpmndi:BPMNEdge>
      <bpmndi:BPMNEdge bpmnElement="flow2" id="BPMNEdge_flow2">
        <omgdi:waypoint x="285.0" y="210.0"></omgdi:waypoint>
        <omgdi:waypoint x="400.0" y="210.0"></omgdi:waypoint>
      </bpmndi:BPMNEdge>
    </bpmndi:BPMNPlane>
  </bpmndi:BPMNDiagram>
</definitions>
```

11.5.40　イベント（メッセージ）の送信処理

メッセージイベントの送信処理をします。この API を実行すると，BPMN 連携ライブラリの Java API の CIWBPMNLib.sendMessage インタフェースが呼び出されます。

メソッドと URL

```
POST /v1/event/send-message/<案件ID>
```

<案件 ID>：イベント（メッセージ）の送信処理をする案件 ID（必須）

リクエストボディ

項番	名前	型	指定要否	内容
1	UserDescription	文字列	省略可	ユーザ記述子を指定します。省略した場合はデフォルト値が指定されます。空文字列は指定できません。
2	ProcessDataList	配列	省略可	プロセスデータ一覧を指定します。省略した場合はプロセスデータを更新しません。
3	ProcessData	オブジェクト	必須	プロセスデータオブジェクトを指定します。次の 2 つの要素（Key および Value）から構成されています。
4	Key	文字列	必須	プロセスデータキー名を指定します。空文字列は指定できません。

項番	名前	型	指定要否	内容
5	Value	文字列	省略可	プロセスデータ値を指定します。省略した場合，Key に対応する値は未設定となります。空文字列は指定できません。
6	MessageRef	文字列	必須	メッセージを受信するイベントの MessageRef を指定します。空文字列は指定できません。

リクエストボディの構造を次に示します。

項番	名前				出現回数
1	Parameter				1
2		UserDescription			0 または 1
3		ProcessDataList			0 または 1
4			ProcessData		1 以上
5				Key	1
6				Value	0 または 1
7		MessageRef			1

リクエスト例

リクエスト URL

```
POST http://restserver/csciwws/v1/event/send-message/5001
```

リクエストボディ（XML の場合）

```
<?xml version="1.0" encoding="UTF-8" ?>
<Parameter>
  <UserDescription>csciwuser</UserDescription>
  <ProcessDataList>
    <ProcessData>
      <Key>$Sdata1</Key>
      <Value>stringvalue1</Value>
    </ProcessData>
    <ProcessData>
      <Key>$Ndata2</Key>
      <Value>100</Value>
    </ProcessData>
  </ProcessDataList>
  <MessageRef>message1</MessageRef>
</Parameter>
```

リクエストボディ（JSON の場合）

```
{
  "UserDescription" : "csciwuser",
  "ProcessDataList" : {
    "ProcessData" : [
      {
        "Key" : "$Sdata1",
        "Value" : "stringvalue1"
      },
      {
        "Key" : "$Ndata2",
        "Value" : "100"
      }
    ]
  },
  "MessageRef" : "message1"
}
```

レスポンス

レスポンスボディは空になります。

ステータスコード

項番	ステータスコード	内容
1	204	成功
2	400	リクエストパラメタの不正
3	404	データが見つからなかった
4	500	内部処理エラー

レスポンス例

レスポンスボディは空になります。

11.5.41　プロセスデータの取得

指定した案件のプロセスデータを取得します。この API を実行すると，BPMN 連携ライブラリ Java API のCIWBPMNLib.getProcessDataMapByMultiplePIID インタフェースが呼び出されます。

メソッドと URL

```
POST /v1/process-data
```

リクエストボディ

項番	名前	型	指定要否	内容
1	UserDescription	文字列	省略可	ユーザ記述子を指定します。省略した場合はデフォルト値が指定されます。空文字列は指定できません。
2	ProcessInstanceList	配列	必須	プロセスデータを取得する案件一覧を指定します。
3	ID	数値	必須	プロセスデータを取得する案件の ID を指定します。
4	ProcessDataList	配列	省略可	プロセスデータ一覧を指定します。省略した場合は指定した案件の ID に合致したプロセスデータがすべて返されます。
5	Key	文字列	必須	取得するプロセスデータキー名を指定します。空文字列は指定できません。

リクエストボディの構造を次に示します。

項番	名前			出現回数
1	Parameter			1
2		UserDescription		0 または 1
3		ProcessInstanceList		1
4			ID	1 以上
5		ProcessDataList		0 または 1
6			Key	1 以上

リクエスト例

リクエスト URL

```
POST http://restserver/csciwws/v1/process-data
```

リクエストボディ（XML の場合）

```
<?xml version="1.0" encoding="UTF-8" ?>
<Parameter>
  <UserDescription>csciwuser</UserDescription>
  <ProcessInstanceList>
    <ID>12100</ID>
    <ID>52331</ID>
  </ProcessInstanceList>
  <ProcessDataList>
```

```
    <Key>$SGYOUMU1STATE</Key>
    <Key>$SGYOUMU1DATE</Key>
  </ProcessDataList>
</Parameter>
```

リクエストボディ（JSON の場合）

```
{
  "UserDescription" : "csciwuser",
  "ProcessInstanceList" : {
    "ID" : [
      "12100",
      "52331"
    ]
  },
  "ProcessDataList" : {
    "Key" : [
      "$SGYOUMU1STATE",
      "$SGYOUMU1DATE"
    ]
  }
}
```

レスポンス

案件ごとのプロセスデータを返します。案件ごとのプロセスデータのプロパティを次に示します。

項番	名前	型	内容
1	ID	数値	案件の ID
2	ProcessDataList	配列	プロセスデーター覧
3	ProcessData	オブジェクト	プロセスデータオブジェクト 次の 2 つの要素（Key および Value）から構成されています。
4	Key	文字列	プロセスデータキー名
5	Value	文字列	プロセスデータ値

レスポンスの構造を次に示します。

項番	名前				出現回数
1	ProcessInstances				1
2		ProcessInstance			0 または 1 以上
3			ID		1
4			ProcessDataList		0 または 1
5				ProcessData	1 以上

項番	名前					出現回数
6					Key	1
7					Value	0 または 1

ステータスコード

項番	ステータスコード	内容
1	200	成功（データが 0 件の場合も成功）
2	400	リクエストパラメタの不正
3	500	内部処理エラー

レスポンス例

レスポンス（XML の場合）

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ProcessInstances>
  <ProcessInstance>
    <ID>12100</ID>
    <ProcessDataList>
      <ProcessData>
        <Key>$SGYOUMU1STATE</Key>
        <Value>d</Value>
      </ProcessData>
      <ProcessData>
        <Key>$SGYOUMU1DATE</Key>
        <Value>20120124</Value>
      </ProcessData>
    </ProcessDataList>
  </ProcessInstance>
  <ProcessInstance>
    <ID>52331</ID>
    <ProcessDataList>
      <ProcessData>
        <Key>$SGYOUMU1STATE</Key>
        <Value>j</Value>
      </ProcessData>
      <ProcessData>
        <Key>$SGYOUMU1DATE</Key>
        <Value>20110221</Value>
      </ProcessData>
    </ProcessDataList>
  </ProcessInstance>
</ProcessInstances>
```

レスポンス（JSON の場合）

```
{
  "ProcessInstance" : [
    {
```

```
      "ID" : "12100",
      "ProcessDataList" : {
        "ProcessData" : [
          {
            "Key" : "$SGYOUMU1STATE",
            "Value" : "d"
          },
          {
            "Key" : "$SGYOUMU1DATE",
            "Value" : "20120124"
          }
        ]
      }
    },
    {
      "ID" : "52331",
      "ProcessDataList" : {
        "ProcessData" : [
          {
            "Key" : "$SGYOUMU1STATE ",
            "Value" : "j"
          },
          {
            "Key" : "$SGYOUMU1DATE ",
            "Value" : "20110221"
          }
        ]
      }
    }
  ]
}
```

11.5.42　プロセスデータの登録

指定したプロセスデータを登録します。この API を実行すると，BPMN 連携ライブラリ Java API の CIWBPMNLib.setProcessData インタフェースが呼び出されます。

メソッドと URL

```
POST /v1/process-data/set-process-data
```

リクエストボディ

項番	名前	型	指定要否	内容
1	UserDescription	文字列	省略可	ユーザ記述子を指定します。省略した場合はデフォルト値が指定されます。空文字列は指定できません。
2	ID	数値	必須	プロセスデータを取得する案件の ID を指定します。

項番	名前	型	指定要否	内容
3	ProcessDataList	配列	必須	登録するプロセスデータ一覧を指定します。
4	ProcessData	オブジェクト	必須	プロセスデータオブジェクトを指定します。次の 2 つの要素（Key および Value)から構成されています。
5	Key	文字列	必須	登録するプロセスデータのキー名を指定します。空文字列は指定できません。
6	Value	文字列	省略可	登録するプロセスデータの値を指定します。省略した場合，値が null のプロセスデータを登録します。空文字列は指定できません。

リクエストボディの構造を次に示します。

項番	名前				出現回数
1	Parameter				1
2		UserDescription			0 または 1
3		ID			1
4		ProcessDataList			1
5			ProcessData		1 以上
6				Key	1
7				Value	0 または 1

リクエスト例

リクエスト URL

```
POST http://restserver/csciwws/v1/process-data/set-process-data
```

リクエストボディ（XML の場合）

```
<?xml version="1.0" encoding="UTF-8" ?>
<Parameter>
  <UserDescription>csciwuser</UserDescription>
  <ID>12100</ID>
  <ProcessDataList>
    <ProcessData>
      <Key>$SSTATE</Key>
      <Value>ACTIVE</Value>
    </ProcessData>
  </ProcessDataList>
</Parameter>
```

リクエストボディ（JSON の場合）

```
{
  ”UserDescription” : ”csciwuser”,
  ”ID” : ”12100”,
  ”ProcessDataList” : {
    ”ProcessData” : [
      {
        ”Key” : ”$SSTATE”,
        ”Value” : ”ACTIVE”
      }
    ]
  }
}
```

レスポンス

レスポンスボディは空になります。

ステータスコード

項番	ステータスコード	内容
1	204	成功
2	400	リクエストパラメタの不正
3	404	データが見つからなかった
4	500	内部処理エラー

11.5.43　リスト型プロセスデータのインデクス取得

指定したリスト型プロセスデータの要素が，リストの何番目の要素かのインデクスを取得します。この API を実行すると，BPMN 連携ライブラリ Java API の`CIWBPMNLib.getListProcessDataIndex` インタフェースが呼び出されます。

メソッドと URL

```
POST /v1/process-data/list-index
```

リクエストボディ

項番	名前	型	指定要否	内容
1	UserDescription	文字列	省略可	ユーザ記述子を指定します。省略した場合はデフォルト値が指定されます。空文字列は指定できません。
2	ID	数値	必須	プロセスデータのインデクスを取得する案件の ID を指定します。

項番	名前	型	指定要否	内容
3	ProcessData	オブジェクト	必須	プロセスデータオブジェクトを指定します。次の2つの要素（Key および Value)から構成されています。
4	Key	文字列	必須	インデクスを取得するリスト型プロセスデータの全要素を示すキー名を指定します。空文字列は指定できません。
5	Value	文字列	省略可	インデクスを取得するリスト型プロセスデータの値を指定します。省略した場合，値が null のプロセスデータを検索します。空文字列は指定できません。

リクエストボディの構造を次に示します。

項番	名前			出現回数
1	Parameter			1
2		UserDescription		0 または 1
3		ID		1
4		ProcessData		1
5			Key	1
6			Value	0 または 1

リクエスト例

リクエスト URL

```
POST http://restserver/csciwws/v1/process-data/list-index
```

リクエストボディ（XML の場合）

```
<?xml version="1.0" encoding="UTF-8" ?>
<Parameter>
  <UserDescription>csciwuser</UserDescription>
  <ID>12100</ID>
  <ProcessData>
    <Key>$SSTATE{}</Key>
    <Value>ACTIVE</Value>
  </ProcessData>
</Parameter>
```

リクエストボディ（JOIN の場合）

```
{
  "UserDescription" : "csciwuser",
```

```
  "ID" : "12100",
  "ProcessData" : {
    "Key" : "$SSTATE{}",
    "Value" : "ACTIVE"
  }
}
```

レスポンス

リスト型プロセスデータのインデクスを返します。

項番	名前	型	内容
1	Index	文字列	リスト型プロセスデータのインデクスを返します。案件が見つからなかった場合は，ステータスコード 404 を返します。プロセスデータが見つからなかった場合は，ステータスコード 200 を返し，Index の値は-1 になります。

レスポンスの構造を次に示します。

項番	名前		出現回数
1	Resources		1
2		Index	1

ステータスコード

項番	ステータスコード	内容
1	200	成功
2	400	リクエストパラメタの不正
3	404	データが見つからなかった
4	500	内部処理エラー

レスポンス例

レスポンス（XML の場合）

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Resources>
  <Index>3</Index>
</Resources>
```

レスポンス（JSON の場合）

```
{
  "Index" : "3"
}
```

11.5.44　フローノードの一覧取得

指定された条件を満たすフローノードの一覧を取得します。この API を実行すると，BPMN 連携ライブラリ Java API のCIWBPMNLib.getFlowNodeInstancesListByPDName インタフェース，または CIWBPMNLib.getFlowNodeInstancesListByPIID インタフェースが呼び出されます。

メソッドと URL

```
GET /v1/flow-node-instance/queries
```

クエリパラメタ

項番	名前	型	指定要否	内容
1	userdescription	文字列	省略可	ユーザ記述子を指定します。省略した場合はデフォルト値が指定されます。空文字列は指定できません。
2	pdname	文字列	次のどちらかが必須 • pdname • piid	取得するフローノードのビジネスプロセス定義名を指定します。バージョンが複数登録されている場合，すべてのバージョンを検索します。piid と同時には指定できません。
3	piid	数値	次のどちらかが必須 • pdname • piid	取得するフローノードの案件の ID を指定します。pdname と同時には指定できません。
4	flownodeid	文字列	省略可	取得するフローノード ID を指定します。省略した場合，指定された案件のすべてのフローノード ID を検索します。空文字列は指定できません。
5	flownodename	文字列	省略可	取得するフローノード名を指定します。省略した場合，指定された案件のすべてのフローノード名を検索します。
6	statecode	文字列	• pdname 指定時：必須 • piid 指定時：省略可	取得するフローノードに対応する CSCIW の作業の状態を指定します。空文字列は指定できません。複数の状態を指定する場合は，状態コードを同時に指定します。例えば，「実行開始可能」と「実行済」の状態を取得する場合は"jr"と指定します。
7	miindex	数値	省略可	取得するフローノードのマルチインスタンスインデクスを指定します。省略した場合，マルチインスタンスインデクスを指定しません。

リクエスト例

リクエスト URL

```
GET http://restserver/csciwws/v1/flow-node-instance/queries?pdname=販売業務&flownodeid=UTask1&flownodename=UserTask&statecode=j
GET http://restserver/csciwws/v1/flow-node-instance/queries?piid=2215&flownodeid=UTask1&flownodename=UserTask&statecode=j
```

レスポンス

フローノード一覧を返します。個々のフローノードのプロパティを次に示します。

項番	名前	型	内容
1	ActivityInstanceID	数値	フローノードに対応する CSCIW の作業が所属する業務ステップの ID
2	FlowNodeID	数値	フローノードでの BPMN 要素の id 属性値
3	FlowNodeMIIndex	数値	フローノードでのマルチインスタンスインデクス
4	FlowNodeName	文字列	フローノードでの BPMN 要素の name 属性値
5	FlowNodeType	文字列	フローノードでの BPMN 要素の種類
6	IsMultiInstance	文字列	フローノードがマルチインスタンスかどうか
7	ProcessDefinitionID	数値	フローノードに対応する CSCIW の作業が所属するビジネスプロセス定義の ID
8	ProcessDefinitionName	文字列	フローノードに対応する CSCIW の作業が所属するビジネスプロセス定義の名称
9	ProcessInstanceID	数値	フローノードに対応する CSCIW の作業が所属する案件の ID
10	ProcessInstanceName	文字列	フローノードに対応する CSCIW の作業の案件名(案件キー)
11	WorkItemClosedDate	日付	フローノードに対応する CSCIW の作業の終了日時
12	WorkItemCreationDate	日付	フローノードに対応する CSCIW の作業の発生日時
13	WorkItemDeadline	日付	フローノードに対応する CSCIW の作業の処理期限の絶対日時
14	WorkItemID	数値	フローノードに対応する CSCIW の作業の ID
15	WorkItemParticipant	文字列	フローノードに対応する CSCIW の作業の作業者 ID
16	WorkItemStartDate	日付	フローノードに対応する CSCIW の作業の開始日時
17	WorkItemState	文字列	フローノードに対応する CSCIW の作業の状態

レスポンスの構造を次に示します。

項番	名前			出現回数
1	FlowNodeInstances			1
2		FlowNodeInstance		0 または 1 以上
3			ActivityInstanceID	1
4			FlowNodeID	1
5			FlowNodeMIIndex	1
6			FlowNodeName	1
7			FlowNodeType	1
8			IsMultiInstance	1
9			ProcessDefinitionID	1
10			ProcessDefinitionName	1
11			ProcessInstanceID	1
12			ProcessInstanceName	1
13			WorkItemClosedDate	1
14			WorkItemCreationDate	1
15			WorkItemDeadline	1
16			WorkItemID	1
17			WorkItemParticipant	1
18			WorkItemStartDate	1
19			WorkItemState	1

ステータスコード

項番	ステータスコード	内容
1	200	成功（データが 0 件の場合も成功）
2	400	リクエストパラメタの不正
3	500	内部処理エラー

レスポンス例

レスポンス（XML の場合）

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<FlowNodeInstances>
  <FlowNodeInstance>
    <ActivityInstanceID>13006</ActivityInstanceID>
    <FlowNodeID>UTask1</FlowNodeID>
```

```
    <FlowNodeMIIndex></FlowNodeMIIndex>
    <FlowNodeName>UserTask</FlowNodeName>
    <FlowNodeType>USER_TASK</FlowNodeType>
    <IsMultiInstance>false</IsMultiInstance>
    <ProcessDefinitionID>3</ProcessDefinitionID>
    <ProcessDefinitionName>販売業務</ProcessDefinitionName>
    <ProcessInstanceID>2215</ProcessInstanceID>
    <ProcessInstanceName>案件43466</ProcessInstanceName>
    <WorkItemClosedDate></WorkItemClosedDate>
    <WorkItemCreationDate>2016-12-03T15:31:12+09:00</WorkItemCreationDate>
    <WorkItemDeadline></WorkItemDeadline>
    <WorkItemID>15220</WorkItemID>
    <WorkItemParticipant></WorkItemParticipant>
    <WorkItemStartDate>2016-12-03T15:33:00+09:00</WorkItemStartDate>
    <WorkItemState>j</WorkItemState>
  </FlowNodeInstance>
</FlowNodeInstances>
```

レスポンス（JSON の場合）

```
{
  "FlowNodeInstance" : [
    {
      "ActivityInstanceID" : "13006",
      "FlowNodeID" : "UTask1",
      "FlowNodeMIIndex" : "",
      "FlowNodeName" : "UserTask",
      "FlowNodeType" : "USER_TASK",
      "IsMultiInstance" : "false",
      "ProcessDefinitionID" : "3",
      "ProcessDefinitionName" : "販売業務",
      "ProcessInstanceID" : "2215",
      "ProcessInstanceName" : "案件43466",
      "WorkItemClosedDate" : "",
      "WorkItemCreationDate" : "2016-12-03T15:31:12+09:00",
      "WorkItemDeadline" : "",
      "WorkItemID" : "15220",
      "WorkItemParticipant" : "",
      "WorkItemStartDate" : "2016-12-03T15:33:00+09:00",
      "WorkItemState" : "j"
    }
  ]
}
```

11.5.45　アドホック・サブプロセスのフローノードを生成

指定されたアドホック・サブプロセス内のフローノードを生成します。この API を実行すると，BPMN 連携ライブラリ Java API の`CIWBPMNLib.createFlowNodeInstanceForAdHocSubProcess` インタフェースが呼び出されます。指定したアドホック・サブプロセスのフローノードが生成済みの場合は，リクエストは成功します（何もしないでステータスコード 201 を返します）。詳細については，「5.3　Java API 利用時の注意事項」の「べき等性について」の説明を参照してください。

メソッドと URL

```
POST /v1/flow-node-instance/create
```

リクエストボディ

項番	名前	型	指定要否	内容
1	UserDescription	文字列	省略可	ユーザ記述子を指定します。省略した場合はデフォルト値が指定されます。空文字列は指定できません。
2	ProcessInstanceID	数値	必須	対象であるアドホック・サブプロセスが所属する案件 ID を指定します。
3	FlowNodeID	文字列	省略可（FlowNodeName を省略した場合は必須）	生成するフローノードのフローノード ID を指定します。省略した場合，フローノード ID は指定されません。空文字列は指定できません。
4	FlowNodeName	文字列	省略可（FlowNodeID を省略した場合は必須）	生成するフローノードのフローノード名を指定します。省略した場合，フローノード名は指定されません。
5	MIIndex	数値	省略可	生成するフローノードがあるアドホック・サブプロセスのマルチインスタンスインデクスを指定します。省略した場合，マルチインスタンスインデクスは指定されません。
6	ProcessDataList	配列	省略可	プロセスデータ一覧を指定します。省略した場合，プロセスデータは更新されません。
7	ProcessData	オブジェクト	必須	プロセスデータオブジェクトを指定します。次の 2 つの要素（Key および Value）から構成されています。
8	Key	文字列	必須	プロセスデータのキー名を指定します。空文字列は指定できません。
9	Value	文字列	省略可	プロセスデータ値を指定します。省略した場合，キーに対応する値は設定されません。空文字列は指定できません。

リクエストボディの構造を次に示します。

項番	名前		出現回数
1	Parameter		1
2		UserDescription	0 または 1
3		ProcessInstanceID	1
4		FlowNodeID	0 または 1

項番	名前				出現回数
5		FlowNodeName			0 または 1
6		MIIndex			0 または 1
7		ProcessDataList			0 または 1
8			ProcessData		1 以上
9				Key	1
10				Value	0 または 1

リクエスト例

リクエスト URL

```
POST http://restserver/csciwws/v1/flow-node-instance/create
```

リクエストボディ（XML の場合）

```
<?xml version="1.0" encoding="UTF-8" ?>
<Parameter>
  <UserDescription>csciwuser</UserDescription>
  <ProcessInstanceID>3315</ProcessInstanceID>
  <FlowNodeID>UTask1</FlowNodeID>
  <FlowNodeName>UserTask</FlowNodeName>
</Parameter>
```

リクエストボディ（JSON の場合）

```
{
  "UserDescription" : "csciwuser",
  "ProcessInstanceID" : "3315",
  "FlowNodeID" : "UTask1",
  "FlowNodeName" : "UserTask"
}
```

レスポンス

作成したフローノードの一覧を返します。個々のフローノードのプロパティを次に示します。

項番	名前	型	内容
1	ActivityInstanceID	数値	フローノードに対応する CSCIW の作業が所属する業務ステップの ID
2	FlowNodeID	数値	フローノードでの BPMN 要素の id 属性値
3	FlowNodeMIIndex	数値	フローノードでのマルチインスタンスインデクス
4	FlowNodeName	文字列	フローノードでの BPMN 要素の name 属性値

項番	名前	型	内容
5	FlowNodeType	文字列	フローノードでの BPMN 要素の種類
6	IsMultiInstance	文字列	フローノードがマルチインスタンスかどうか
7	ProcessDefinitionID	数値	フローノードに対応する CSCIW の作業が所属するビジネスプロセス定義の ID
8	ProcessDefinitionName	文字列	フローノードに対応する CSCIW の作業が所属するビジネスプロセス定義の名称
9	ProcessInstanceID	数値	フローノードに対応する CSCIW の作業が所属する案件の ID
10	ProcessInstanceName	文字列	フローノードに対応する CSCIW の作業の案件名(案件キー)
11	WorkItemClosedDate	日付	フローノードに対応する CSCIW の作業の終了日時
12	WorkItemCreationDate	日付	フローノードに対応する CSCIW の作業の発生日時
13	WorkItemDeadline	日付	フローノードに対応する CSCIW の作業の処理期限の絶対日時
14	WorkItemID	数値	フローノードに対応する CSCIW の作業の ID
15	WorkItemParticipant	文字列	フローノードに対応する CSCIW の作業の作業者 ID
16	WorkItemStartDate	日付	フローノードに対応する CSCIW の作業の開始日時
17	WorkItemState	文字列	フローノードに対応する CSCIW の作業の状態

レスポンスの構造を次に示します。

項番	名前			出現回数
1	FlowNodeInstances			1
2		FlowNodeInstance		0 または 1 以上
3			ActivityInstanceID	1
4			FlowNodeID	1
5			FlowNodeMIIndex	1
6			FlowNodeName	1
7			FlowNodeType	1
8			IsMultiInstance	1
9			ProcessDefinitionID	1

項番	名前			出現回数
10			ProcessDefinitionName	1
11			ProcessInstanceID	1
12			ProcessInstanceName	1
13			WorkItemClosedDate	1
14			WorkItemCreationDate	1
15			WorkItemDeadline	1
16			WorkItemID	1
17			WorkItemParticipant	1
18			WorkItemStartDate	1
19			WorkItemState	1

ステータスコード

項番	ステータスコード	内容
1	201	成功
2	400	リクエストパラメタの不正
3	404	データが見つからなかった
4	500	内部処理エラー

レスポンス例

レスポンス（XML の場合）

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<FlowNodeInstances>
  <FlowNodeInstance>
    <ActivityInstanceID>13508</ActivityInstanceID>
    <FlowNodeID>UTask1</FlowNodeID>
    <FlowNodeMIIndex></FlowNodeMIIndex>
    <FlowNodeName>UserTask</FlowNodeName>
    <FlowNodeType>USER_TASK</FlowNodeType>
    <IsMultiInstance>false</IsMultiInstance>
    <ProcessDefinitionID>4</ProcessDefinitionID>
    <ProcessDefinitionName>販売業務</ProcessDefinitionName>
    <ProcessInstanceID>3315</ProcessInstanceID>
    <ProcessInstanceName>案件43496</ProcessInstanceName>
    <WorkItemClosedDate></WorkItemClosedDate>
    <WorkItemCreationDate>2016-12-03T17:31:12+09:00</WorkItemCreationDate>
    <WorkItemDeadline></WorkItemDeadline>
    <WorkItemID>16422</WorkItemID>
    <WorkItemParticipant></WorkItemParticipant>
    <WorkItemStartDate>2016-12-03T17:33:00+09:00</WorkItemStartDate>
```

```
    <WorkItemState>d</WorkItemState>
  </FlowNodeInstance>
</FlowNodeInstances>
```

レスポンス（JSON の場合）

```
{
  "FlowNodeInstance" : [
    {
      "ActivityInstanceID" : "13508",
      "FlowNodeID" : "UTask1",
      "FlowNodeMIIndex" : "",
      "FlowNodeName" : "UserTask",
      "FlowNodeType" : "USER_TASK",
      "IsMultiInstance" : "false",
      "ProcessDefinitionID" : "4",
      "ProcessDefinitionName" : "販売業務",
      "ProcessInstanceID" : "3315",
      "ProcessInstanceName" : "案件43496",
      "WorkItemClosedDate" : "",
      "WorkItemCreationDate" : "2016-12-03T17:31:12+09:00",
      "WorkItemDeadline" : "",
      "WorkItemID" : "16422",
      "WorkItemParticipant" : "",
      "WorkItemStartDate" : "2016-12-03T17:33:00+09:00",
      "WorkItemState" : "d"
    }
  ]
}
```

11.5.46　フローノード定義の一覧取得

指定した条件を満たすフローノード定義の一覧を取得します。この API を実行すると，BPMN 連携ライブラリ Java API の CIWBPMNLib.getFlowNodeDefinitionsList インタフェースが呼び出されます。

メソッドと URL

```
GET /v1/flow-node-definition/queries
```

クエリパラメタ

項番	名前	型	指定要否	内容
1	userdescription	文字列	省略可	ユーザ記述子を指定します。省略した場合はデフォルト値が指定されます。空文字列は指定できません。
2	pdname	文字列	必須	取得するフローノード定義のビジネスプロセス定義名を指定します。空文字列は指定できません。

項番	名前	型	指定要否	内容
2	pdname	文字列	必須	バージョンが複数登録されている場合は，すべてのバージョンを検索します。
3	flownodeid	文字列	省略可	取得するフローノード定義での BPMN 要素の id 属性値を指定します。省略した場合，指定されたビジネスプロセス定義のすべてのフローノード ID を検索します。空文字列は指定できません。
4	flownodename	文字列	省略可	取得するフローノード定義での BPMN 要素の name 属性値を指定します。省略した場合，指定されたビジネスプロセス定義のすべてのフローノード名を検索します。

リクエスト例

リクエスト URL

```
GET http://restserver/csciwws/v1/flow-node-definition/queries?pdname=OrderProcess&flownodeid=UTask1&flownodename=UserTask
```

レスポンス

フローノード定義一覧を返します。個々のフローノード定義のプロパティを次に示します。

項番	名前	型	内容
1	FlowNodeCalledElement	文字列	フローノード定義での BPMN 要素の calledElement 属性値
2	FlowNodeID	文字列	フローノード定義での BPMN 要素の id 属性値
3	FlowNodeName	文字列	フローノード定義での BPMN 要素の name 属性値
4	FlowNodeRefID	文字列	フローノード定義での BPMN 要素の ref 識別子属性値
5	FlowNodeType	文字列	フローノード定義での BPMN 要素の種類
6	IsMultiInstance	文字列	フローノード定義がマルチインスタンスかどうか
7	ProcessDefinitionID	数値	フローノード定義に対応する CSCIW の作業定義が所属するビジネスプロセス定義の ID
8	ProcessDefinitionName	文字列	フローノード定義に対応する CSCIW の作業定義が所属するビジネスプロセス定義の名称
9	WorkDefinitionID	数値	フローノード定義に対応する CSCIW の作業定義の ID
10	WorkDefinitionName	文字列	フローノード定義に対応する CSCIW の作業定義の名称

レスポンスの構造を次に示します。

項番	名前			出現回数
1	FlowNodeDefinitions			1
2		FlowNodeDefinition		0 または 1 以上
3			FlowNodeCalledElement	1
4			FlowNodeID	1
5			FlowNodeName	1
6			FlowNodeRefID	1
7			FlowNodeType	1
8			IsMultiInstance	1
9			ProcessDefinitionID	1
10			ProcessDefinitionName	1
11			WorkDefinitionID	1
12			WorkDefinitionName	1

ステータスコード

項番	ステータスコード	内容
1	200	成功（データが 0 件の場合も成功）
2	400	リクエストパラメタの不正
3	500	内部処理エラー

レスポンス例

レスポンス（XML の場合）

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<FlowNodeDefinitions>
  <FlowNodeDefinition>
    <FlowNodeCalledElement></FlowNodeCalledElement>
    <FlowNodeID>UTask1</FlowNodeID>
    <FlowNodeName>UserTask</FlowNodeName>
    <FlowNodeRefID></FlowNodeRefID>
    <FlowNodeType>USER_TASK</FlowNodeType>
    <IsMultiInstance>false</IsMultiInstance>
    <ProcessDefinitionID>5</ProcessDefinitionID>
    <ProcessDefinitionName>OrderProcess</ProcessDefinitionName>
    <WorkDefinitionID>178</WorkDefinitionID>
    <WorkDefinitionName>UserTask_UTask1</WorkDefinitionName>
  </FlowNodeDefinition>
</FlowNodeDefinitions>
```

レスポンス（JSON の場合）

```
{
  ”FlowNodeDefinition” : [
    {
      ”FlowNodeCalledElement” : ””,
      ”FlowNodeID” : ”UTask1”,
      ”FlowNodeName” : ”UserTask”,
      ”FlowNodeRefID” : ””,
      ”FlowNodeType” : ”USER_TASK”,
      ”IsMultiInstance” : ”false”,
      ”ProcessDefinitionID” : ”5”,
      ”ProcessDefinitionName” : ”OrderProcess”,
      ”WorkDefinitionID” : ”178”,
      ”WorkDefinitionName” : ”UserTask_UTask1”
    }
  ]
}
```

11.5.47　アドホック・サブプロセスの状態の変更

指定されたアドホック・サブプロセスの状態を変更します。この API を実行すると，BPMN 連携ライブラリ Java API の CIWBPMNLib.changeStateAdHocSubProcess インタフェースが呼び出されます。アドホック・サブプロセスの状態が変更済みの場合は，リクエストは成功します（何もしないでステータスコード 204 を返します）。詳細については，「5.3　Java API 利用時の注意事項」の「べき等性について」の説明を参照してください。

メソッドと URL

```
PUT /v1/adhoc-sub-process/change-state
```

リクエストボディ

項番	名前	型	指定要否	内容
1	UserDescription	文字列	省略可	ユーザ記述子を指定します。省略した場合はデフォルト値が指定されます。空文字列は指定できません。
2	ProcessInstanceID	数値	必須	対象であるアドホック・サブプロセスが所属する案件 ID を指定します。
3	AdHocSubProcessID	文字列	省略可（AdHocSubProcessName を省略した場合は必須）	状態変更するアドホック・サブプロセスのアドホック・サブプロセス ID を指定します。省略した場合，アドホック・サブプロセス ID は指定されません。空文字列は指定できません。

項番	名前	型	指定要否	内容
4	AdHocSubProcessName	文字列	省略可（AdHocSubProcessID を省略した場合は必須）	状態変更するアドホック・サブプロセスのアドホック・サブプロセス名を指定します。省略した場合，アドホック・サブプロセス名は指定されません。
5	MIIndex	数値	省略可	状態変更するアドホック・サブプロセスのマルチインスタンスインデクスを指定します。省略した場合，マルチインスタンスインデクスは指定されません。
6	ProcessDataList	配列	省略可	プロセスデータ一覧を指定します。省略した場合，プロセスデータは更新されません。
7	ProcessData	オブジェクト	必須	プロセスデータオブジェクトを指定します。次の 2 つの要素（Key および Value）から構成されています。
8	Key	文字列	必須	プロセスデータのキー名を指定します。空文字列は指定できません。
9	Value	文字列	省略可	プロセスデータ値を指定します。省略した場合，キーに対応する値は設定されません。空文字列は指定できません。
10	StateCode	文字列	必須	変更するアドホック・サブプロセスの状態に対応する作業の状態をコード値で指定します。アドホック・サブプロセスの状態の詳細については，「1.5.6　アドホック・サブプロセスの状態遷移モデル」を参照してください。

リクエストボディの構造を次に示します。

項番	名前				出現回数
1	Parameter				1
2		UserDescription			0 または 1
3		ProcessInstanceID			1
4		AdHocSubProcessID			0 または 1
5		AdHocSubProcessName			0 または 1
6		MIIndex			0 または 1
7		ProcessDataList			0 または 1
8			ProcessData		1 以上
9				Key	1
10				Value	0 または 1

項番	名前		出現回数
11		StateCode	1

リクエスト例

リクエスト URL

```
PUT http://restserver/csciwws/v1/adhoc-sub-process/change-state
```

リクエストボディ（XML の場合）

```
<?xml version="1.0" encoding="UTF-8" ?>
<Parameter>
  <UserDescription>csciwuser</UserDescription>
  <ProcessInstanceID>3315</ProcessInstanceID>
  <AdHocSubProcessID>AdHocSub1</AdHocSubProcessID>
  <AdHocSubProcessName>AdHocSubProcess</AdHocSubProcessName>
  <StateCode>r</StateCode>
</Parameter>
```

リクエストボディ（JSON の場合）

```
{
  "UserDescription" : "csciwuser",
  "ProcessInstanceID" : "3315",
  "AdHocSubProcessID" : "AdHocSub1",
  "AdHocSubProcessName" : "AdHocSubProces"
  "StateCode" : "r"
}
```

レスポンス

レスポンスボディは空になります。

ステータスコード

項番	ステータスコード	内容
1	204	成功
2	400	リクエストパラメタの不正
3	404	データが見つからなかった
4	500	内部処理エラー

11.5.48　業務ステップ定義の一覧取得

指定されたフィルター条件を満たす，業務ステップ定義の一覧を取得します。この API を実行すると，CSCIW Java API の`CIWServer.getActivityDefinitionsList` インタフェースが呼び出されます。

メソッドと URL

```
GET /v1/activity-definition
```

クエリパラメタ

項番	名前	型	指定要否	内容
1	userdescription	文字列	省略可	ユーザ記述子を指定します。省略した場合，デフォルト値が指定されます。空文字列は指定できません。
2	filter	文字列	省略可	業務ステップ定義一覧を取得する場合のフィルター条件を指定します。省略した場合，フィルター条件は指定されません。空文字列は指定できません。
3	sort	文字列	省略可	取得した業務ステップ定義一覧をソートする場合の条件を指定します。省略した場合，ソート条件は指定されません。空文字列は指定できません。
4	offset	数値	省略可	業務ステップ定義一覧を取得する場合のオフセットを指定します。先頭は 0 になります。省略した場合は 0 が指定されます。0 未満の値は指定できません。
5	maxcount	数値	省略可	業務ステップ定義一覧を取得する場合の最大取得数を指定します。省略した場合，デフォルト値が指定されます。すべてを取得する場合は，-1 を指定します。-1 未満の値は指定できません。

リクエスト例

リクエスト URL

```
GET http://restserver/csciwws/v1/activity-definition?filter=ProcessDefinitionID%3D1001
```

レスポンス

業務ステップ定義一覧を返します。個々の業務ステップ定義のプロパティを次に示します。

項番	名前	型	内容
1	ID	数値	業務ステップ定義の ID
2	Name	文字列	業務ステップ定義の名称
3	ProcessDefinitionID	数値	業務ステップ定義が所属するビジネスプロセス定義の ID
4	Description	文字列	業務ステップ定義の説明

レスポンスの構造を次に示します。

項番	名前			出現回数
1	ActivityDefinitions			1
2		ActivityDefinition		0 または 1 以上
3			ID	1
4			Name	1
5			ProcessDefinitionID	1
6			Description	1

ステータスコード

項番	ステータスコード	内容
1	200	成功（データが 0 件の場合も成功）
2	400	リクエストパラメタの不正
3	500	内部処理エラー

レスポンス例

レスポンス（XML の場合）

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ActivityDefinitions>
  <ActivityDefinition>
    <ID>1</ID>
    <Name>USER_TASK1</Name>
    <ProcessDefinitionID>1001</ProcessDefinitionID>
    <Description></Description>
  </ActivityDefinition>
  <ActivityDefinition>
    <ID>2</ID>
    <Name>USER_TASK2</Name>
    <ProcessDefinitionID>1001</ProcessDefinitionID>
    <Description>ABCDEFG</Description>
  </ActivityDefinition>
</ActivityDefinitions>
```

レスポンス（JSON の場合）

```
{
  ”ActivityDefinition” : [
    {
      ”ID” : ”1”,
      ”Name” : ”USER_TASK1”,
      ”ProcessDefinitionID” : ”1001”,
      ”Description” : ””
    },
    {
      ”ID” : ”2”,
      ”Name” : ”USER_TASK2”,
      ”ProcessDefinitionID” : ”1001”,
      ”Description” : ”ABCDEFG”
    }
  ]
}
```

11.5.49　業務ステップ定義数の取得

指定された条件を満たす業務ステップ定義の数を取得します。この API を実行すると，CSCIW Java API のCIWServer.getListCount インタフェースが呼び出されます。

メソッドと URL

```
GET /v1/activity-definition/count
```

クエリパラメタ

項番	名前	型	指定要否	内容
1	userdescription	文字列	省略可	ユーザ記述子を指定します。省略した場合，デフォルト値が指定されます。空文字列は指定できません。
2	filter	文字列	省略可	業務ステップ定義数を取得する場合のフィルター条件を指定します。省略した場合，フィルター条件は指定されません。空文字列は指定できません。

リクエスト例

リクエスト URL

```
GET http://restserver/csciwws/v1/activity-definition/count?filter=ProcessDefinitionID%3D1001
```

レスポンス

業務ステップ定義数を返します。レスポンスとして返されるプロパティを次に示します。

項番	名前	型	内容
1	Count	数値	業務ステップ定義の数

レスポンスの構造を次に示します。

項番	名前	出現回数
1	Resources	1
2	Count	1

ステータスコード

項番	ステータスコード	内容
1	200	成功（データが 0 件の場合も成功）
2	400	リクエストパラメタの不正
3	500	内部処理エラー

レスポンス例

レスポンス（XML の場合）

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Resources>
  <Count>3</Count>
</Resources>
```

レスポンス（JSON の場合）

```
{
  "Count" : "3"
}
```

11.5.50　業務ステップ定義の取得

指定された ID の業務ステップ定義を取得します。この API を実行すると，CSCIW Java API の CIWServer.getActivityDefinition インタフェースが呼び出されます。

メソッドと URL

```
GET /v1/activity-definition/<ビジネスプロセス定義のID>/<業務ステップ定義のID>
```

<ビジネスプロセス定義の ID>：取得する業務ステップ定義のビジネスプロセス定義の ID（必須）

<業務ステップ定義の ID>：取得する業務ステップ定義の ID（必須）

クエリパラメタ

項番	名前	型	指定要否	内容
1	userdescription	文字列	省略可	ユーザ記述子を指定します。省略した場合，デフォルト値が指定されます。空文字列は指定できません。

リクエスト例

リクエスト URL

```
GET http://restserver/csciwws/v1/activity-definition/1/5
```

レスポンス

業務ステップ定義を返します。業務ステップ定義のプロパティを次に示します。

項番	名前	型	内容
1	ID	数値	業務ステップ定義の ID
2	Name	文字列	業務ステップ定義の名称
3	ProcessDefinitionID	数値	業務ステップ定義が所属するビジネスプロセス定義の ID
4	Description	文字列	業務ステップ定義の説明

レスポンスの構造を次に示します。

項番	名前		出現回数
1	ActivityDefinition		1
2		ID	1
3		Name	1
4		ProcessDefinitionID	1
5		Description	1

ステータスコード

項番	ステータスコード	内容
1	200	成功
2	400	リクエストパラメタの不正
3	404	データが見つからなかった
4	500	内部処理エラー

レスポンス例

レスポンス（XML の場合）

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<ActivityDefinition>
  <ID>1</ID>
  <Name>USER_TASK1</Name>
  <ProcessDefinitionID>1001</ProcessDefinitionID>
  <Description></Description>
</ActivityDefinition>
```

レスポンス（JSON の場合）

```
{
  "ID" : "1",
  "Name" : "USER_TASK1",
  "ProcessDefinitionID" : "1001",
  "Description" : ""
}
```

12 Java API リファレンス

BPMN 連携機能が提供する Java API の詳細について説明します。

12.1　BPMN 連携ライブラリで提供するクラスの一覧

BPMN 連携ライブラリが提供するクラスおよびインタフェースを次に示します。

パッケージ：jp.co.Hitachi.soft.csciw.bpmn のクラスおよびインタフェース一覧

項番	クラス名・インタフェース名	説明
1	CIWBPMNLibAdmin	BPMN 連携ライブラリの初期化および終了処理を行うクラス
2	CIWBPMNLibFactory	CIWBPMNLib オブジェクトのファクトリクラス
3	CIWBPMNLib	BPMN 連携ライブラリの機能を提供するインタフェース
4	CIWBPMNFlowNodeInstance	フローノードオブジェクトの属性を取得する機能を提供するインタフェース
5	CIWBPMNFlowNodeDefinition	フローノード定義オブジェクトの属性を取得する機能を提供するインタフェース

パッケージ：jp.co.Hitachi.soft.csciw.bpmn.processdata のクラスおよびインタフェース一覧

項番	クラス名・インタフェース名	説明
1	CIWBPMNProcessData	プロセスデータのインタフェース
2	CIWBPMNProcessDataFactory	プロセスデータオブジェクトのファクトリクラス

12.2 CIWBPMNLibAdmin（BPMN 連携ライブラリの初期化および終了処理をするクラス）

BPMN 連携ライブラリの初期化および終了処理をするクラスです。

クラス定義

```
public final class CIWBPMNLibAdmin extends java.lang.Object
```

メソッド

CIWBPMNLibAdmin クラスのメソッドを次の表に示します。

表 12-1 CIWBPMNLibAdmin クラスのメソッド

項番	メソッド名	説明
1	initializeCIWBPMNLib	BPMN 連携ライブラリの初期化
2	finalizeCIWBPMNLib	BPMN 連携ライブラリの終了処理

12.2.1 initializeCIWBPMNLib

構文

```
public static synchronized void initializeCIWBPMNLib(
     java.sql.Connection aDBConnection
)
     throws CIWFatalException
```

機能

BPMN 連携ライブラリを初期化します。

引数

initializeCIWBPMNLib の引数を次の表に示します。

表 12-2 initializeCIWBPMNLib の引数

項番	仮引数名	名称	I/O	説明
1	aDBConnection	JDBC コネクション	in	null は指定できません。

戻り値

なし

例外

initializeCIWBPMNLib で発生する例外を次の表に示します。

表 12-3　initializeCIWBPMNLib の例外

項番	発生する例外	説明
1	CIWFatalException	処理を続行できない障害が発生した場合

注意事項

- JDBC コネクションにnull を指定した場合は，例外（java.lang.IllegalArgumentException）が発生します。
- J2EE アプリケーションで使用する場合，CSCIWManagementServer を開始する前に，このメソッドを実行すると例外（CIWFatalException）が発生します。
- Java アプリケーションで使用する場合，CIWAdmin クラスのinitializeCIWFactory メソッドを実行する前に，このメソッドを実行すると例外（CIWFatalException）が発生します。
- セットアッププロパティファイルでBpmnMode にtrue が設定されていない場合は例外（CIWFatalException）が発生します。

12.2.2　finalizeCIWBPMNLib

構文

```
public static synchronized void finalizeCIWBPMNLib()
```

機能

BPMN 連携ライブラリの終了処理をします。

引数

なし

戻り値

なし

例外

なし

注意事項

なし

12.3　CIWBPMNLibFactory（BPMNLib オブジェクトの生成クラス）

CIWBPMNLibFactory はBPMNLib のオブジェクトの生成クラスです。

クラス定義

```
public final class CIWBPMNLibFactory extends java.lang.Object
```

メソッド

CIWBPMNLibFactory クラスのメソッドを次の表に示します。

表 12-4　CIWBPMNLibFactory クラスのメソッド

項番	メソッド名	説明
1	createCIWBPMNLib	CIWBPMNLib オブジェクトの生成

12.3.1　createCIWBPMNLib

構文

```
public static CIWBPMNLib createCIWBPMNLib()
     throws CIWFatalException
```

機能

CIWBPMNLib オブジェクトを生成します。

引数

なし

戻り値

CIWBPMNLib オブジェクト

例外

createCIWBPMNLib で発生する例外を次の表に示します。

表 12-5　createCIWBPMNLib の例外

項番	発生する例外	説明
1	CIWFatalException	処理を続行できない障害が発生した場合

注意事項

- CIWBPMNLibAdmin クラスのinitializeCIWBPMNLib メソッドを実行する前に，このメソッドを実行した場合，例外（CIWFatalException）が発生します。

12.4 CIWBPMNLib（BPMN 連携ライブラリの機能を提供するインタフェース）

BPMN 連携ライブラリの機能を提供するインタフェースです。

クラス定義

```
public interface CIWBPMNLib
```

メソッド

表 12-6 CIWBPMNLib インタフェースのメソッド

項番	メソッド名	説明	API の種類
1	createAndStartPI	案件投入（イベントなし）	更新系 API（案件）
2	deletePI	案件の削除	更新系 API（案件）
3	terminatePI	案件の強制終了	更新系 API（案件）
4	adhocCreateAndMakeTransitionAI	強制的に任意の業務ステップに遷移させるアドホック処理	更新系 API（業務ステップ）
5	changeStateAI	業務ステップの状態変更	更新系 API（業務ステップ）
6	makeBackwardTransitionAI	差し戻しまたは引き戻し	更新系 API（業務ステップ）
7	allocateWIEx	指定した条件に一致する作業の着手	更新系 API（作業）
8	changeStateWI	作業の状態変更	更新系 API（作業）
9	completeWI	作業の完了	更新系 API（作業）
10	freeWI	作業の返却	更新系 API（作業）
11	performAndCompleteWI	作業の着手と完了	更新系 API（作業）
12	performWI	作業の着手	更新系 API（作業）
13	reassignAndPerformWI	作業の作業者変更および作業の着手	更新系 API（作業）
14	reassignWI	作業の作業者変更	更新系 API（作業）
15	setProcessData	プロセスデータの登録	更新系 API（プロセスデータ）
16	sendMessage	メッセージイベントの送信	更新系 API（メッセージ）
17	startMessage	案件投入（メッセージ）	更新系 API（メッセージ）
18	createAndStartPIForTimer	案件投入（タイマー）	更新系 API（タイマー）
19	setDeadlineForTimer	タイマーの処理期限を変更	更新系 API（タイマー）

項番	メソッド名	説明	API の種類
20	createFlowNodeInstanceForAdHocSubProcess	アドホック・サブプロセスのフローノードを生成	更新系 API（アドホック・サブプロセス）
21	changeStateAdHocSubProcess	アドホック・サブプロセスの状態変更	更新系 API（アドホック・サブプロセス）
22	getListProcessDataIndex	リスト型プロセスデータのインデクスを取得	検索系 API（プロセスデータ）
23	getPIIDListByProcessData	プロセスデータから案件 ID のリストを取得	検索系 API（プロセスデータ）
24	getProcessDataMapByMultiplePIID	複数の案件 ID からプロセスデータのマップを取得	検索系 API（プロセスデータ）
25	getFlowNodeDefinitionsList	フローノード定義のリストを取得	検索系 API（BPMN）
26	getFlowNodeInstancesListByPDName	ビジネスプロセス定義名を指定してフローノードのリストを取得	検索系 API（BPMN）
27	getFlowNodeInstancesListByPIID	案件 ID を指定してフローノードのリストを取得	検索系 API（BPMN）
28	getFlowNodeInstancesListWithChildPIByPIID	案件 ID を指定して子案件を含めたフローノードのリストを取得	検索系 API（BPMN）
29	getProcessInstancesListByPDName	ビジネスプロセス定義名を指定して案件のリストを取得	検索系 API（BPMN）
30	getProcessInstancesListByPIName	ビジネスプロセス定義名と案件名を指定して案件のリストを取得	検索系 API（BPMN）
31	getCallActivityChildPI	コールアクティビティから投入された子案件を取得	検索系 API（コールアクティビティ）
32	getCallActivityParentPI	対象案件の親案件の取得	検索系 API（コールアクティビティ）
33	getCallActivityParentWI	対象案件の呼び元の作業の取得	検索系 API（コールアクティビティ）

12.4.1 createAndStartPI（案件投入）

構文

```
CIWProcessInstance createAndStartPI(
     java.sql.Connection aDBConnection,
     CIWServer aCIWServer,
     java.lang.String aProcessDefinitionName,
     java.lang.Short aProcessDefinitionVersion,
```

```
    java.util.Map<CIWProcessInstance.AttributeName,java.lang.Object> aAttributes,
    java.util.Collection<CIWBPMNProcessData<?>> aProcessDataCollection
)
    throws CIWFatalException,
    CIWTransitionFailedException,
    CIWTransientException,
    CIWEntityNotExistException,
    CIWStateException
```

機能

指定されたビジネスプロセス定義名とビジネスプロセス定義バージョンのビジネスプロセス定義で案件を生成し開始します。指定されたプロセスデータは，生成した案件の案件 ID を指定して登録します。

開始（タイプなし）が定義されているビジネスプロセス定義だけ，案件投入できます。

引数

createAndStartPI の引数を次の表に示します。

表 12-7　createAndStartPI の引数

項番	仮引数名	名称	I/O	説明
1	aDBConnection	JDBC コネクション	in	CIWServer オブジェクトに関連づけられた JDBC コネクションを指定します。 null は指定できません。
2	aCIWServer	CIWServer オブジェクト	in	CIWServer オブジェクトを指定します。 null は指定できません。
3	aProcessDefinitionName	ビジネスプロセス定義名	in	ビジネスプロセス定義名を指定します。 空文字列およびnull は指定できません。
4	aProcessDefinitionVersion	ビジネスプロセス定義バージョン	in	ビジネスプロセス定義バージョンを指定します。null を指定した場合，指定したビジネスプロセス定義の中で投入可能（状態が活性かつ案件投入期間内）な最新バージョンとなります。
5	aAttributes	案件属性のマップ	in	案件属性のマップを指定します。詳細は CIWServer インタフェースの createAndStartProcessInstance メソッドを参照してください。
6	aProcessDataCollection	プロセスデータのコレクション	in	プロセスデータを追加しない場合は null を指定します。

戻り値

開始した案件オブジェクトを返します。

例外

createAndStartPI で発生する例外を次の表に示します。

表 12-8　createAndStartPI の例外

項番	発生する例外	説明
1	CIWFatalException	処理を続行できない障害が発生した場合
2	CIWTransitionFailedException	案件処理中にエラーが発生した場合
3	CIWTransientException	一時的なエラーが発生した場合
4	CIWEntityNotExistException	処理しようとしたオブジェクトが存在しない場合
5	CIWStateException	状態や属性の変更に失敗した場合

注意事項

- JDBC コネクションおよび CIWServer オブジェクトにnull を指定した場合は，例外（java.lang.IllegalArgumentException）が発生します。
- JDBC コネクションと CIWServer オブジェクトが関連づけられていない場合の動作は保証しません。
- 指定されたビジネスプロセス定義が存在しない場合や案件を投入できない場合（状態が非活性または案件投入期間外の場合），例外（CIWEntityNotExistException）が発生します。
- ビジネスプロセス定義バージョンにnull を指定した場合で，指定したビジネスプロセス定義に案件投入可能（状態が活性かつ案件投入期間内）なバージョンが存在しないときは，例外（CIWEntityNotExistException）が発生します。
- 開始（タイプなし）（省略されている場合も含む）が定義されていないビジネスプロセス定義を指定した場合，例外（CIWFatalException)(メッセージKDIW01834-E）が発生します。
- 案件名の値に空文字列を指定した場合は，例外（java.lang.IllegalArgumentException）が発生します。
- 指定された属性名と属性値の型が一致しない場合は，例外（java.lang.IllegalArgumentException）が発生します。
- ビジネスプロセス定義名に空文字列またはnull を指定した場合は，例外（java.lang.IllegalArgumentException）が発生します。

12.4.2　deletePI（案件の削除）

構文

```
boolean deletePI(
     java.sql.Connection aDBConnection,
     CIWServer aCIWServer,
     java.lang.Integer aProcessInstanceID
)
     throws CIWFatalException,
     CIWTransientException,
     CIWStateException
```

機能

案件 ID で指定された案件を削除します。削除する案件のプロセスデータも削除します。

ルート案件だけを指定できます。子案件がある場合は，子案件（子案件の子案件も含む）もすべて削除します。

引数

deletePI の引数を次の表に示します。

表 12-9　deletePI の引数

項番	仮引数名	名称	I/O	説明
1	aDBConnection	JDBC コネクション	in	CIWServer オブジェクトに関連づけられた JDBC コネクションを指定します。 null は指定できません。
2	aCIWServer	CIWServer オブジェクト	in	CIWServer オブジェクトを指定します。 null は指定できません。
3	aProcessInstanceID	案件 ID	in	削除する案件の ID を指定します。 null は指定できません。

戻り値

案件の削除を実行したかどうかを返します。

true：実行しました

false：案件が削除済みだったので実行しませんでした

例外

deletePI で発生する例外を次の表に示します。

表 12-10　deletePI の例外

項番	発生する例外	説明
1	CIWFatalException	処理を続行できない障害が発生した場合
2	CIWTransientException	一時的なエラーが発生した場合
3	CIWStateException	状態や属性の変更に失敗した場合

注意事項

- JDBC コネクションおよび CIWServer オブジェクトにnull を指定した場合は，例外（java.lang.IllegalArgumentException）が発生します。
- JDBC コネクションと CIWServer オブジェクトが関連づけられていない場合の動作は保証しません。
- 案件 ID にnull を指定した場合は，例外（java.lang.IllegalArgumentException）が発生します。
- 指定した条件を満たす案件オブジェクトが存在しない場合は，false を返します。
- 案件が「終了」状態の場合だけ案件を削除できます。そのほかの状態の場合は，例外（CIWStateException）が発生します。
- ルート案件だけを指定できます。ルート案件以外を指定した場合は，例外（CIWStateException）が発生します。

12.4.3　terminatePI（案件の強制終了）

構文

```
boolean terminatePI(
     java.sql.Connection aDBConnection,
     CIWServer aCIWServer,
     java.lang.Integer aProcessInstanceID
)
     throws CIWFatalException,
     CIWTransitionFailedException,
     CIWTransientException,
     CIWEntityNotExistException,
     CIWStateException
```

機能

案件 ID で指定された案件を強制終了します。

指定された案件が親案件だった場合は，子案件も強制終了します。

指定された案件が子案件だった場合は，親案件のコールアクティビティも強制終了します。

引数

terminatePI の引数を次の表に示します。

表 12-11　terminatePI の引数

項番	仮引数名	名称	I/O	説明
1	aDBConnection	JDBC コネクション	in	CIWServer オブジェクトに関連づけられた JDBC コネクションを指定します。 null は指定できません。
2	aCIWServer	CIWServer オブジェクト	in	CIWServer オブジェクトを指定します。 null は指定できません。
3	aProcessInstanceID	案件 ID	in	強制終了する案件の ID を指定します。 null は指定できません。

戻り値

案件の強制終了を実行したかどうかを返します。

true：実行しました

false：すでに案件の状態が強制終了だったので実行しませんでした

例外

terminatePI で発生する例外を次の表に示します。

表 12-12　terminatePI の例外

項番	発生する例外	説明
1	CIWFatalException	処理を続行できない障害が発生した場合
2	CIWTransitionFailedException	案件処理中にエラーが発生した場合
3	CIWTransientException	一時的なエラーが発生した場合
4	CIWEntityNotExistException	処理しようとしたオブジェクトが存在しない場合
5	CIWStateException	状態や属性の変更に失敗した場合

注意事項

- JDBC コネクションおよび CIWServer オブジェクトにnull を指定した場合は，例外（java.lang.IllegalArgumentException）が発生します。

- JDBC コネクションと CIWServer オブジェクトが関連づけられていない場合の動作は保証しません。
- 案件 ID にnull を指定した場合は，例外（java.lang.IllegalArgumentException）が発生します。
- 指定した条件を満たす案件オブジェクトが存在しない場合は，例外（CIWEntityNotExistException）が発生します。
- 案件が「未終了」状態の場合に強制終了できます。「強制終了」状態の場合はfalse が返されます。そのほかの状態の場合は，例外（CIWStateException）が発生します。

12.4.4 adhocCreateAndMakeTransitionAI（強制的に任意の業務ステップに遷移させるアドホック処理）

構文

```
boolean adhocCreateAndMakeTransitionAI(
        java.sql.Connection aDBConnection,
        CIWServer aCIWServer,
        java.lang.Integer aProcessInstanceID,
        java.lang.Integer aActivityInstanceID,
        java.lang.String aTargetActivityDefName,
        java.util.Collection<CIWBPMNProcessData<?>> aProcessDataCollection
)
     throws CIWFatalException,
     CIWTransientException,
     CIWTransitionFailedException,
     CIWStateException,
     CIWEntityNotExistException
```

機能

案件 ID と業務ステップ ID で指定された業務ステップを強制終了し，指定された業務ステップ定義にフローとは関係なく強制遷移します。また，指定されたプロセスデータを更新します。

変換前のタスク，イベントの種別に関係なく，どの業務ステップも遷移元と遷移先の対象とします。

引数

adhocCreateAndMakeTransitionAI の引数を次の表に示します。

表 12-13　adhocCreateAndMakeTransitionAI の引数

項番	仮引数名	名称	I/O	説明
1	aDBConnection	JDBC コネクション	in	CIWServer オブジェクトに関連づけられた JDBC コネクションを指定します。
2	aCIWServer	CIWServer オブジェクト	in	CIWServer オブジェクトを指定します。

項番	仮引数名	名称	I/O	説明
3	aProcessInstanceID	案件 ID	in	対象である業務ステップの案件 ID を指定します。 null は指定できません。
4	aActivityInstanceID	業務ステップ ID	in	対象である遷移元の業務ステップの業務ステップ ID を指定します。 null は指定できません。
5	aTargetActivityDefName	業務ステップ定義名	in	強制遷移先である業務ステップの業務ステップ定義名を指定します。 空文字列およびnull は指定できません。 業務ステップ定義名は「<名前>_id」形式※で指定します。
6	aProcessDataCollection	プロセスデータのコレクション	in	プロセスデータを更新しない場合は null を指定します。

注※

BPMN 要素のname 属性値とid 属性値

戻り値

業務ステップの強制遷移を実行したかどうかを返します。

true：実行しました

false：業務ステップが強制遷移済みだったので実行しませんでした

強制遷移済みの条件は次の両方が成立した場合です。

- 指定された遷移元の業務ステップの状態が強制終了
- 指定された遷移先に実行開始不可，実行中，実行停止，遷移済み，または強制終了のどれかの状態の業務ステップが存在する

例外

adhocCreateAndMakeTransitionAI で発生する例外を次の表に示します。

表 12-14　adhocCreateAndMakeTransitionAI の例外

項番	発生する例外	説明
1	CIWFatalException	処理を続行できない障害が発生した場合
2	CIWTransitionFailedException	案件処理中にエラーが発生した場合
3	CIWTransientException	一時的なエラーが発生した場合

項番	発生する例外	説明
4	CIWEntityNotExistException	処理しようとしたオブジェクトが存在しない場合
5	CIWStateException	状態や属性の変更に失敗した場合

注意事項

- JDBC コネクションおよび CIWServer オブジェクトにnull を指定した場合は，例外（java.lang.IllegalArgumentException）が発生します。
- JDBC コネクションと CIWServer オブジェクトが関連づけられていない場合の動作は保証しません。
- 案件 ID，業務ステップ ID，または業務ステップ定義名にnull を指定した場合は，例外（java.lang.IllegalArgumentException）が発生します。
- 指定された案件 ID と業務ステップ ID の業務ステップが存在しない場合は，例外（CIWEntityNotExistException）が発生します。
- 強制遷移先に指定した業務ステップ定義が存在しない場合は，例外（CIWEntityNotExistException）が発生します。
- 強制遷移元の業務ステップの状態が「実行中」以外の場合は，例外（CIWStateException）が発生します。ただし，強制遷移元の業務ステップが「強制終了」状態でも，強制遷移先に指定した業務ステップ定義に「実行開始不可」「実行中」「実行停止」「遷移済」または「強制終了」のどれかの状態の業務ステップが存在する場合は，例外は発生しないでfalse が返されます。
- 境界中断（タイマー），境界非中断（タイマー），イベント・サブプロセス中断開始（タイマー）およびイベント・サブプロセス非中断開始（タイマー）が定義されたサブプロセスの中に遷移した場合は，タイマーは受信待ちになりません。
- 境界中断（タイマー），境界非中断（タイマー），イベント・サブプロセス中断開始（タイマー）およびイベント・サブプロセス非中断開始（タイマー）が定義されたサブプロセスの外に遷移した場合，次のようになります。
  - サブプロセス内に，ほかに実行中の業務ステップがあるとき，タイマーは受信待ちのままになります。
  - サブプロセス内に，ほかに実行中の業務ステップがないとき，タイマーは強制終了します。
- サブプロセスに対する境界中断（タイマー）および境界非中断（タイマー），イベント・サブプロセス中断開始（タイマー）およびイベント・サブプロセス非中断開始（タイマー）の受信用の業務ステップを，遷移元または遷移先に指定できません。指定した場合は，例外（CIWStateException）が発生します。
- サブプロセス（マルチインスタンス）の中の業務ステップを，遷移元または遷移先に指定できません。サブプロセス（マルチインスタンス）の中の業務ステップを指定した場合は，例外（CIWStateException）が発生します。
- アドホック・サブプロセスの中の業務ステップを，遷移元または遷移先に指定できません。アドホック・サブプロセスの中の業務ステップを指定した場合は，例外（CIWStateException）が発生します。

12.4.5 changeStateAI（業務ステップの状態変更）

構文

```
boolean changeStateAI(
        java.sql.Connection aDBConnection,
        CIWServer aCIWServer,
        java.lang.Integer aProcessInstanceID,
        java.lang.Integer aActivityInstanceID,
        CIWActivityInstance.State aNewState,
        java.util.Collection<CIWBPMNProcessData<?>> aProcessDataCollection
)
     throws CIWFatalException,
     CIWTransientException,
     CIWTransitionFailedException,
     CIWStateException,
     CIWEntityNotExistException
```

機能

案件 ID と業務ステップ ID で指定された業務ステップの状態を変更します。また，指定されたプロセスデータを更新します。

変換前のタスク，イベントの種別に関係なく，どの業務ステップの状態も変更できます。

引数

changeStateAI の引数を次の表に示します。

表 12-15　changeStateAI の引数

項番	仮引数名	名称	I/O	説明
1	aDBConnection	JDBC コネクション	in	CIWServer オブジェクトに関連づけられた JDBC コネクションを指定します。
2	aCIWServer	CIWServer オブジェクト	in	CIWServer オブジェクトを指定します。
3	aProcessInstanceID	案件 ID	in	対象にする業務ステップの案件 ID を指定します。 null は指定できません。
4	aActivityInstanceID	業務ステップ ID	in	対象にする業務ステップの業務ステップ ID を指定します。 null は指定できません。
5	aNewState	業務ステップの状態	in	変更する業務ステップの状態を指定します。

項番	仮引数名	名称	I/O	説明
5	aNewState	業務ステップの状態	in	次の表に示す，CIWActivityInstance.State 列挙型の定数を指定できます。 null は指定できません。
6	aProcessDataCollection	プロセスデータのコレクション	in	プロセスデータを更新しない場合は null を指定します。

aNewState に指定できる業務ステップの状態を次の表に示します。

現在の状態		指定できる状態		状態遷移の意味
定数	意味	定数	意味	
－	未終了	TERMINATED	強制終了	強制終了
RUNNING	実行中	READY_FOR_TRANSITION	遷移可	業務ステップ終了

（凡例）

定数：CIWActivityInstance.State 列挙型の定数を表します。

－：定数がないことを表します。

戻り値

業務ステップの状態変更を実行したかどうかを返します。

true：実行しました

false：業務ステップの状態変更が実行済み（指定された業務ステップの状態が変更後の業務ステップの状態）だったので実行しませんでした

例外

changeStateAI で発生する例外を次の表に示します。

表 12-16　changeStateAI の例外

項番	発生する例外	説明
1	CIWFatalException	処理を続行できない障害が発生した場合
2	CIWTransitionFailedException	案件処理中にエラーが発生した場合
3	CIWTransientException	一時的なエラーが発生した場合
4	CIWEntityNotExistException	処理しようとしたオブジェクトが存在しない場合
5	CIWStateException	状態や属性の変更に失敗した場合

注意事項

- JDBC コネクションおよび CIWServer オブジェクトにnull を指定した場合は，例外（java.lang.IllegalArgumentException）が発生します。
- JDBC コネクションと CIWServer オブジェクトが関連づけられていない場合の動作は保証しません。
- 案件 ID，業務ステップ ID，または業務ステップ定義名にnull を指定した場合は，例外（java.lang.IllegalArgumentException）が発生します。
- 変更する業務ステップの状態にnull を指定した場合は，例外（java.lang.IllegalArgumentException）が発生します。
- 対象となる業務ステップが存在しない場合は，例外（CIWEntityNotExistException）が発生します。
- 対象となる業務ステップの状態と，変更する業務ステップの状態の組み合わせが不正な場合は，例外（CIWStateException）が発生します。ただし，対象となる業務ステップが変更する業務ステップの状態の場合は，例外は発生しないでfalse が返されます。
- 業務ステップを強制終了した場合は，次の業務ステップへは遷移しません。そのため，ほかに「実行中」状態の業務ステップが存在しない場合は，案件の強制終了以外の操作ができなくなります。また，実行中の業務ステップが存在した場合でも，待合ノードに続く業務ステップを強制終了すると，待合ノードから遷移しなくなります。
- 対象となる業務ステップが排他イベントゲートウェイから変換された業務ステップの場合は，業務ステップを終了すると，例外（CIWFatalException）が発生します。
- サブプロセスに対する境界中断（タイマー），境界非中断（タイマー），イベント・サブプロセス中断開始（タイマー）およびイベント・サブプロセス非中断開始（タイマー）の受信用の業務ステップは，「遷移可」の状態に変更できません。「遷移可」の状態に変更しようとした場合は，例外（CIWStateException）が発生します。
- 境界中断（タイマー），境界非中断（タイマー），イベント・サブプロセス中断開始（タイマー）およびイベント・サブプロセス非中断開始（タイマー）が定義されたプロセス／サブプロセスの作業を強制終了した場合は，次のようになります。
  - プロセスまたはサブプロセス内に，ほかに実行中の業務ステップがあるとき，タイマーは受信待ちのままになります。
  - プロセスまたはサブプロセス内に，ほかに実行中の業務ステップがないとき，タイマーは強制終了します。
- アドホック・サブプロセスの状態表示用の業務ステップ（業務ステップ定義名「＜アドホック・サブプロセスの BPMN 要素名＞_＜アドホック・サブプロセスの BPMN 要素 ID ＞」）は，状態変更できません。状態変更しようとした場合は，例外（CIWStateException）が発生します。

12.4.6 makeBackwardTransitionAI（業務ステップの差し戻しまたは引き戻し）

構文

```
boolean makeBackwardTransitionAI(
     java.sql.Connection aDBConnection,
     CIWServer aCIWServer,
     java.lang.Integer aProcessInstanceID,
     java.lang.Integer aSourceActivityInstID,
     java.lang.String aTargetActivityDefName
     java.util.Collection<CIWBPMNProcessData<?>> aProcessDataCollection
)
     throws CIWFatalException,
     CIWTransitionFailedException,
     CIWTransientException,
     CIWEntityNotExistException,
     CIWStateException
```

機能

案件 ID と業務ステップ ID で指定された業務ステップを，指定した業務ステップ定義に差し戻しまたは引き戻します。指定されたプロセスデータを更新します。

差し戻しまたは引き戻し元となる業務ステップはユーザタスクから変換された業務ステップだけを対象とします。差し戻しまたは引き戻し先の業務ステップ定義はユーザタスク以外から変換された業務ステップ定義も対象とします。

引数

makeBackwardTransitionAI の引数を次の表に示します。

表 12-17　makeBackwardTransitionAI の引数

項番	仮引数名	名称	I/O	説明
1	aDBConnection	JDBC コネクション	in	CIWServer オブジェクトに関連づけられた JDBC コネクションを指定します。 null は指定できません。
2	aCIWServer	CIWServer オブジェクト	in	CIWServer オブジェクトを指定します。 null は指定できません。
3	aProcessInstanceID	案件 ID	in	差し戻しまたは引き戻し対象の業務ステップの案件 ID を指定します。 null は指定できません。

項番	仮引数名	名称	I/O	説明
4	aSourceActivityInstID	業務ステップ ID	in	差し戻しまたは引き戻し対象の業務ステップの業務ステップ ID を指定します。 null は指定できません。
5	aTargetActivityDefName	業務ステップ定義名	in	差し戻しまたは引き戻し先の業務ステップ定義の業務ステップ定義名を指定します。 空文字列およびnull は指定できません。 業務ステップ定義名は「<名前>_id」※形式で指定します。
6	aProcessDataCollection	プロセスデータのコレクション	in	プロセスデータを更新しない場合は null を指定します。

注※

BPMN 要素のname 属性値とid 属性値

戻り値

差し戻しまたは引き戻しを実行したかどうかを返します。

true：実行しました

false：差し戻しまたは引き戻し済みだったので実行しませんでした

差し戻しまたは引き戻し済みの条件は次の両方が成立した場合です。

- 指定された遷移元の業務ステップの状態が強制終了
- 指定された遷移先に実行開始不可，実行中，実行停止，遷移済み，または強制終了のどれかの状態の業務ステップが存在する

例外

makeBackwardTransitionAI で発生する例外を次の表に示します。

表 12-18　makeBackwardTransitionAI の例外

項番	発生する例外	説明
1	CIWFatalException	処理を続行できない障害が発生した場合
2	CIWTransitionFailedException	案件処理中にエラーが発生した場合
3	CIWTransientException	一時的なエラーが発生した場合
4	CIWEntityNotExistException	処理しようとしたオブジェクトが存在しない場合
5	CIWStateException	状態や属性の変更に失敗した場合

注意事項

- JDBC コネクションおよび CIWServer オブジェクトにnull を指定した場合は，例外（java.lang.IllegalArgumentException）が発生します。
- JDBC コネクションと CIWServer オブジェクトが関連づけられていない場合の動作は保証しません。
- 案件 ID，業務ステップ ID または業務ステップ定義名にnull を指定した場合は，例外（java.lang.IllegalArgumentException）が発生します。
- 差し戻しまたは引き戻し元として指定できる業務ステップはユーザタスクから変換された業務ステップである必要があります。そのほかの業務ステップを指定した場合は，例外（CIWStateException）が発生します。
- 指定した条件を満たす業務ステップオブジェクトが存在しない場合は，例外（CIWEntityNotExistException）が発生します。
- 差し戻し先または引き戻し先に指定した業務ステップ定義が存在しない場合，例外（CIWEntityNotExistException）が発生します。
- 差し戻し先または引き戻し先に指定した業務ステップ定義に業務ステップが存在しない場合，例外（CIWStateException）が発生します。
- 差し戻し先または引き戻し先に指定した業務ステップ定義に「初期」または「終了」のどちらでもない状態の業務ステップが存在する場合，例外（CIWStateException）が発生します。差し戻し先または引き戻し先に指定した業務ステップ定義に「実行中」状態の業務ステップが存在しても，対象となる業務ステップが「強制終了」状態の場合は，例外は発生しないでfalse が返ります。
- 差し戻し元または引き戻し元の業務ステップの状態が「初期」または「終了」のどちらかの場合は，例外（CIWStateException）が発生します。差し戻し元または引き戻し元の業務ステップが「強制終了」状態でも，差し戻し先または引き戻し先に指定した業務ステップ定義に「実行開始不可/実行中/実行停止/遷移済/強制終了」状態の業務ステップが存在する場合は，例外は発生しないでfalse が返されます。
- 境界中断（タイマー），境界非中断（タイマー），イベント・サブプロセス中断開始（タイマー）およびイベント・サブプロセス非中断開始（タイマー）が定義されたサブプロセスの中に差し戻した場合は，タイマーは受信待ちになりません。
- 境界中断（タイマー），境界非中断（タイマー），イベント・サブプロセス中断開始（タイマー）およびイベント・サブプロセス非中断開始（タイマー）が定義されたサブプロセスの外に差し戻した場合は，次のようになります
  - サブプロセス内に，ほかに実行中の業務ステップがあるとき，タイマーは受信待ちのままになります。
  - サブプロセス内に，ほかに実行中の業務ステップがないとき，タイマーは強制終了します。
- サブプロセスに対する境界中断（タイマー），境界非中断（タイマー），イベント・サブプロセス中断開始（タイマー），およびイベント・サブプロセス非中断開始（タイマー）の受信用の業務ステップは，差し戻し先または引き戻し先として先に指定できません。指定した場合は，例外（CIWStateException）が発生します。

- サブプロセス（マルチインスタンス）の中の業務ステップを差し戻し元，引き戻し元，差し戻し先，または引き戻し先に指定できません。サブプロセス（マルチインスタンス）の中の業務ステップを指定した場合は，例外（CIWStateException）が発生します。
- アドホック・サブプロセスの中の業務ステップを，差し戻し元，引き戻し元，差し戻し先，または引き戻し先に指定できません。アドホック・サブプロセスの中の業務ステップを指定した場合は，例外（CIWStateException）が発生します。

12.4.7 changeStateWI（作業の状態変更）

構文

```
boolean changeStateWI(
        java.sql.Connection aDBConnection,
        CIWServer aCIWServer,
        java.lang.Integer aProcessInstanceID,
        java.lang.Integer aWorkItemID,
        CIWWorkItem.State aNewState,
        java.util.Collection<CIWBPMNProcessData<?>> aProcessDataCollection
)
     throws CIWFatalException,
     CIWTransientException,
     CIWTransitionFailedException,
     CIWStateException,
     CIWEntityNotExistException
```

機能

案件 ID と作業 ID で指定された作業の状態を変更します。また，指定されたプロセスデータを更新します。

変換前のタスク，イベントの種別に関係なく，どの作業の状態も変更できます。

引数

changeStateWI の引数を次の表に示します。

表 12-19　changeStateWI の引数

項番	仮引数名	名称	I/O	説明
1	aDBConnection	JDBC コネクション	in	CIWServer オブジェクトに関連づけられた JDBC コネクションを指定します。
2	aCIWServer	CIWServer オブジェクト	in	CIWServer オブジェクトを指定します。
3	aProcessInstanceID	案件 ID	in	対象にする作業の案件 ID を指定します。 null は指定できません。

項番	仮引数名	名称	I/O	説明
4	aWorkItemID	作業 ID	in	対象にする作業の作業 ID を指定します。 null は指定できません。
5	aNewState	作業の状態	in	変更する作業の状態を指定します。 次の表に示す，CIWWorkItem.State 列挙型の定数を指定できます。 null は指定できません。
6	aProcessDataCollection	プロセスデータのコレクション	in	プロセスデータを更新しない場合は null を指定します。

aNewState に指定できる作業の状態を次の表に示します。

現在の状態		指定できる状態		状態遷移の意味
定数	意味	定数	意味	
READY	実行開始可能	PERFORMING	作業者実行	着手
PERFORMING	作業者実行	READY	実行開始可能	返却
PERFORMING	作業者実行	EXECUTED	実行済	完了通知
RUNNING	実行中	INTERMITTED	実行停止	中断
INTERMITTED	実行停止	RUNNING	実行中	再開
－	未終了	TERMINATED	強制終了	強制終了

（凡例）

定数：CIWWorkItem.State 列挙型の定数を表します。

－：定数がないことを表します。

戻り値

作業の状態変更を実行したかどうかを返します。

true：実行しました

false：作業の状態変更が実行済み（指定された作業の状態が変更後の作業の状態）だったので実行しませんでした

例外

changeStateWI で発生する例外を次の表に示します。

表 12-20　changeStateWI の例外

項番	発生する例外	説明
1	CIWFatalException	処理を続行できない障害が発生した場合
2	CIWTransitionFailedException	案件処理中にエラーが発生した場合
3	CIWTransientException	一時的なエラーが発生した場合
4	CIWEntityNotExistException	処理しようとしたオブジェクトが存在しない場合
5	CIWStateException	状態や属性の変更に失敗した場合

注意事項

- JDBC コネクションおよび CIWServer オブジェクトにnull を指定した場合は，例外（java.lang.IllegalArgumentException）が発生します。
- JDBC コネクションと CIWServer オブジェクトが関連づけられていない場合の動作は保証しません。
- 案件 ID または作業 ID にnull を指定した場合は，例外（java.lang.IllegalArgumentException）が発生します。
- 指定した条件を満たす作業オブジェクトが存在しない場合は，例外（CIWEntityNotExistException）が発生します。
- 対象となる作業の状態と，変更する作業の状態の組み合わせが不正な場合は，例外（CIWStateException）が発生します。ただし，対象となる作業が変更する作業の状態の場合は，例外は発生しないでfalse が返されます。
- 作業を強制終了した場合，対象となる作業が所属する業務ステップも強制終了します。また，ユーザタスク（マルチインスタンス），サービスタスク（マルチインスタンス），ビジネスルールタスク（マルチインスタンス）およびコールアクティビティ（マルチインスタンス）の一般作業を強制終了した場合，対象となる一般作業が所属する業務ステップも強制終了します。
- 作業を完了した場合は，対象となる作業が所属する業務ステップも完了します。
- 対象となる作業が，排他イベントゲートウェイの遷移先のキャッチから変換された作業の場合は，作業を完了すると，例外（CIWFatalException（KDIW01834-E メッセージ））が発生します。
- 対象となる作業が境界から変換された受信用の作業の場合は，作業を完了すると，対象となる作業が所属する業務ステップも完了し，通常フローに遷移します。
- 組み込み作業の場合は，例外（CIWStateException）が発生します。
- サブプロセスに対する境界中断（タイマー），境界非中断（タイマー），イベント・サブプロセス中断開始（タイマー），およびイベント・サブプロセス非中断開始（タイマー）の受信用の作業は「実行済」状態に変更できません。「実行済」に状態を変更しようとした場合は，例外（CIWStateException）が発生します。
- 境界中断（タイマー），境界非中断（タイマー），イベント・サブプロセス中断開始（タイマー）およびイベント・サブプロセス非中断開始（タイマー）が定義されたプロセスまたはサブプロセスの作業を強制終了した場合は，次のようになります。

- プロセスまたはサブプロセス内に，ほかに実行中の業務ステップがあるとき，タイマーは受信待ちのままになります。
- プロセスまたはサブプロセス内に，ほかに実行中の業務ステップがないとき，タイマーは強制終了します。

- アドホック・サブプロセスの状態表示用の作業（作業定義名「＜アドホック・サブプロセスの BPMN 要素名＞_＜アドホック・サブプロセスの BPMN 要素 ID ＞」）は，状態変更できません。状態変更しようとした場合は，例外（CIWStateException）が発生します。

12.4.8 completeWI（作業の完了）

構文

```
boolean completeWI(
        java.sql.Connection aDBConnection,
        CIWServer aCIWServer,
        java.lang.Integer aProcessInstanceID,
        java.lang.Integer aWorkItemID,
        java.util.Collection<CIWBPMNProcessData<?>> aProcessDataCollection
)
     throws CIWFatalException,
     CIWTransientException,
     CIWTransitionFailedException,
     CIWStateException,
     CIWEntityNotExistException
```

機能

案件 ID と作業 ID で指定された作業を完了します。また，指定されたプロセスデータを更新します。

対象となる作業は，ユーザタスクから変換された作業です。

引数

completeWI の引数を次の表に示します。

表 12-21 completeWI の引数

項番	仮引数名	名称	I/O	説明
1	aDBConnection	JDBC コネクション	in	CIWServer オブジェクトに関連づけられた JDBC コネクションを指定します。
2	aCIWServer	CIWServer オブジェクト	in	CIWServer オブジェクトを指定します。
3	aProcessInstanceID	案件 ID	in	対象にする作業の案件 ID を指定します。

項番	仮引数名	名称	I/O	説明
3	aProcessInstanceID	案件 ID	in	null は指定できません。
4	aWorkItemID	作業 ID	in	対象にする作業の作業 ID を指定します。 null は指定できません。
5	aProcessDataCollection	プロセスデータのコレクション	in	プロセスデータを更新しない場合は null を指定します。

戻り値

作業の完了を実行したかどうかを返します。

true：実行しました

false：作業の完了が実行済みだったので実行しませんでした

例外

completeWI で発生する例外を次の表に示します。

表 12-22　completeWI の例外

項番	発生する例外	説明
1	CIWFatalException	処理を続行できない障害が発生した場合
2	CIWTransitionFailedException	案件処理中にエラーが発生した場合
3	CIWTransientException	一時的なエラーが発生した場合
4	CIWEntityNotExistException	処理しようとしたオブジェクトが存在しない場合
5	CIWStateException	状態の遷移に失敗した場合

注意事項

- JDBC コネクションおよび CIWServer オブジェクトにnull を指定した場合は，例外（java.lang.IllegalArgumentException）が発生します。
- JDBC コネクションと CIWServer オブジェクトが関連づけられていない場合の動作は保証しません。
- 案件 ID または作業 ID にnull を指定した場合は，例外（java.lang.IllegalArgumentException）が発生します。
- 対象となる作業はユーザタスクから変換された作業である必要があります。ユーザタスク以外から変換された作業を指定した場合は，例外（CIWStateException）が発生します。
- 指定した条件を満たす業務ステップオブジェクトが存在しない場合は，例外（CIWEntityNotExistException）が発生します。

- 対象となる作業は，「実行中」状態である必要があります。そのほかの場合は，例外（CIWStateException）が発生します。ただし，対象となる作業が「実行済」状態の場合は，例外は発生しないでfalse が返されます。

12.4.9 performAndCompleteWI（作業の着手と完了）

構文

```
boolean performAndCompleteWI (
     java.sql.Connection aDBConnection,
     CIWServer aCIWServer,
     java.lang.Integer aProcessInstanceID,
     java.lang.Integer aWorkItemID,
     java.util.Collection<CIWBPMNProcessData<?>> aProcessDataCollection
)
     throws CIWFatalException,
     CIWTransitionFailedException,
     CIWTransientException,
     CIWEntityNotExistException,
     CIWStateException
```

機能

案件 ID と作業 ID で指定された作業を着手して完了します。指定されたプロセスデータを更新します。

対象となるのは，ユーザタスクから変換された作業です。

引数

performAndCompleteWI の引数を次の表に示します。

表 12-23 performAndCompleteWI の引数

項番	仮引数名	名称	I/O	説明
1	aDBConnection	JDBC コネクション	in	CIWServer オブジェクトに関連づけられた JDBC コネクションを指定します。 null は指定できません。
2	aCIWServer	CIWServer オブジェクト	in	CIWServer オブジェクトを指定します。 null は指定できません。
3	aProcessInstanceID	案件 ID	in	対象となる作業の案件 ID を指定します。 null は指定できません。
4	aWorkItemID	作業 ID	in	対象となる作業の作業 ID を指定します。

項番	仮引数名	名称	I/O	説明
4	aWorkItemID	作業 ID	in	null は指定できません。
5	aProcessDataCollection	プロセスデータのコレクション	in	プロセスデータを更新しない場合は null を指定します。

戻り値

作業の着手と完了を実行したかどうかを返します。

true：実行しました

false：作業の状態が実行済みだったので実行しませんでした

例外

performAndCompleteWI で発生する例外を次の表に示します。

表 12-24　performAndCompleteWI の例外

項番	発生する例外	説明
1	CIWFatalException	処理を続行できない障害が発生した場合
2	CIWTransitionFailedException	案件処理中にエラーが発生した場合
3	CIWTransientException	一時的なエラーが発生した場合
4	CIWEntityNotExistException	処理しようとしたオブジェクトが存在しない場合
5	CIWStateException	状態や属性の変更に失敗した場合

注意事項

- JDBC コネクションおよび CIWServer オブジェクトにnull を指定した場合は，例外（java.lang.IllegalArgumentException）が発生します。
- JDBC コネクションと CIWServer オブジェクトが関連づけられていない場合の動作は保証しません。
- 案件 ID および作業 ID にnull を指定した場合は，例外（java.lang.IllegalArgumentException）が発生します。
- 対象となる作業はユーザタスクから変換された作業である必要があります。ユーザタスク以外から変換された作業を指定した場合は，例外（CIWStateException）が発生します。
- 指定した条件を満たす作業オブジェクトが存在しない場合は，例外（CIWEntityNotExistException）が発生します。

- 対象となる作業は，「実行開始可能」状態である必要があります。そのほかの状態では，例外（CIWStateException）が発生します。対象となる作業が「実行済」状態の場合は，例外は発生しないで false が返ります。

12.4.10　performWI（作業の着手）

構文

```
boolean performWI(
        java.sql.Connection aDBConnection,
        CIWServer aCIWServer,
        java.lang.Integer aProcessInstanceID,
        java.lang.Integer aWorkItemID,
        java.util.Collection<CIWBPMNProcessData<?>> aProcessDataCollection
)
     throws CIWFatalException,
     CIWTransientException,
     CIWStateException,
     CIWEntityNotExistException
```

機能

案件 ID と作業 ID で指定された作業に着手します。また，指定されたプロセスデータを更新します。

対象となる作業は，ユーザタスクから変換された作業です。

引数

performWI の引数を次の表に示します。

表 12-25　performWI の引数

項番	仮引数名	名称	I/O	説明
1	aDBConnection	JDBC コネクション	in	CIWServer オブジェクトに関連づけられた JDBC コネクションを指定します。
2	aCIWServer	CIWServer オブジェクト	in	CIWServer オブジェクトを指定します。
3	aProcessInstanceID	案件 ID	in	対象にする作業の案件 ID を指定します。 null は指定できません。
4	aWorkItemID	作業 ID	in	対象にする作業の作業 ID を指定します。 null は指定できません。

項番	仮引数名	名称	I/O	説明
5	aProcessDataCollection	プロセスデータのコレクション	in	プロセスデータを更新しない場合は null を指定します。

戻り値

作業の着手を実行したかどうかを返します。

true：実行しました

false：作業の着手が実行済みだったので実行しませんでした

例外

performWI で発生する例外を次の表に示します。

表 12-26　performWI の例外

項番	発生する例外	説明
1	CIWFatalException	処理を続行できない障害が発生した場合
2	CIWTransientException	一時的なエラーが発生した場合
3	CIWEntityNotExistException	処理しようとしたオブジェクトが存在しない場合
4	CIWStateException	状態の遷移に失敗した場合

注意事項

- JDBC コネクションおよび CIWServer オブジェクトにnull を指定した場合は，例外（java.lang.IllegalArgumentException）が発生します。
- JDBC コネクションと CIWServer オブジェクトが関連づけられていない場合の動作は保証しません。
- 案件 ID または作業 ID にnull を指定した場合は，例外（java.lang.IllegalArgumentException）が発生します。
- 対象となる作業は，ユーザタスクから変換された作業である必要があります。ユーザタスク以外から変換された作業を指定した場合は，例外（CIWStateException）が発生します。
- 指定した条件を満たす作業オブジェクトが存在しない場合は，例外（CIWEntityNotExistException）が発生します。
- 対象となる作業は，「実行開始可能」状態である必要があります。そのほかの場合は，例外（CIWStateException）が発生します。ただし，対象となる作業が「作業者実行」状態の場合は，例外は発生しないでfalse が返されます。

12.4.11　reassignAndPerformWI（作業の作業者変更および作業の着手）

構文

```
boolean reassignAndPerformWI(
        java.sql.Connection aDBConnection,
        CIWServer aCIWServer,
        java.lang.Integer aProcessInstanceID,
        java.lang.Integer aWorkItemID,
        java.lang. String aSource,
        java.lang. String aTarget,
        java.util.Collection<CIWBPMNProcessData<?>> aProcessDataCollection
)
     throws CIWFatalException,
     CIWTransientException,
     CIWStateException,
     CIWEntityNotExistException
```

機能

案件 ID と作業 ID で指定された作業の作業者を，指定された別の作業者に変更してから，その作業に着手します。また，指定されたプロセスデータを更新します。

対象となる作業は，ユーザタスクから変換された作業です。

引数

reassignAndPerformWI の引数を次の表に示します。

表 12-27　reassignAndPerformWI の引数

項番	仮引数名	名称	I/O	説明
1	aDBConnection	JDBC コネクション	in	CIWServer オブジェクトに関連づけられた JDBC コネクションを指定します。
2	aCIWServer	CIWServer オブジェクト	in	CIWServer オブジェクトを指定します。
3	aProcessInstanceID	案件 ID	in	対象にする作業の案件 ID を指定します。 null は指定できません。
4	aWorkItemID	作業 ID	in	対象にする作業の作業 ID を指定します。 null は指定できません。
5	aSource	変更する前の作業者 ID	in	変更する前の作業者 ID を指定します。 未設定の場合はnull を指定します。空文字列は指定できません。

項番	仮引数名	名称	I/O	説明
6	aTarget	新しい作業者 ID	in	新しい作業者 ID を <SYSTEMID>_WORK_ITEM テーブルの Participant カラムのバイト数以下で指定します。 null を指定した場合，新しい作業者 ID は設定されません。空文字列は指定できません。
7	aProcessDataCollection	プロセスデータのコレクション	in	プロセスデータを更新しない場合は null を指定します。

戻り値

作業の作業者の変更および作業の着手を実行したかどうかを返します。

true：実行しました

false：作業者の変更および作業の着手が実行済みだったので実行しませんでした

作業者の変更および作業の着手が実行済みの条件は次の両方が成立した場合です。

- 指定された作業の作業者が変更後の作業者と同じ
- 指定された作業の状態が作業者実行である

例外

reassignAndPerformWI で発生する例外を次の表に示します。

表 12-28　reassignAndPerformWI の例外

項番	発生する例外	説明
1	CIWFatalException	処理を続行できない障害が発生した場合
2	CIWTransientException	一時的なエラーが発生した場合
3	CIWEntityNotExistException	処理しようとしたオブジェクトが存在しない場合
4	CIWStateException	状態の遷移に失敗した場合

注意事項

- JDBC コネクションおよび CIWServer オブジェクトにnull を指定した場合は，例外（java.lang.IllegalArgumentException）が発生します。
- JDBC コネクションと CIWServer オブジェクトが関連づけられていない場合の動作は保証しません。
- 案件 ID または作業 ID にnull を指定した場合は，例外（java.lang.IllegalArgumentException）が発生します。

- 対象となる作業は，ユーザタスクから変換された作業である必要があります。ユーザタスク以外から変換された作業を指定した場合は，例外（CIWStateException)が発生します。
- 指定した条件を満たす作業オブジェクトが存在しない場合は，例外（CIWEntityNotExistException）が発生します。
- 対象となる作業は，「実行開始可能」状態である必要があります。そのほかの場合は，例外（CIWStateException）が発生します。
- 引数aSource に指定した作業者 ID と，対象となる作業の作業者 ID が異なる場合は，例外（CIWStateException）が発生します。ただし，対象となる作業の作業者 ID が引数aTarget に指定した作業者 ID と同じで，状態が「作業者実行」状態の場合，例外は発生しないでfalse が返されます。
- 作業者 ID に空文字列を指定した場合は，例外（java.lang.IllegalArgumentException）が発生します。
- 作業者 ID に<SYSTEMID>_WORK_ITEM テーブルのParticipant カラムのバイト数を超える文字列を指定した場合は，例外（java.lang.IllegalArgumentException）が発生します。

12.4.12 reassignWI（作業の作業者変更）

構文

```
boolean reassignWI(
     java.sql.Connection aDBConnection,
     CIWServer aCIWServer,
     java.lang.Integer aProcessInstanceID,
     java.lang.Integer aWorkItemID,
     java.lang.String aSource,
     java.lang.String aTarget
)
     throws CIWFatalException,
     CIWTransientException,
     CIWEntityNotExistException,
     CIWStateException
```

機能

案件 ID と作業 ID で指定された作業の作業者を，指定した別の作業者に再割り当てします。

対象となるのは，ユーザタスクから変換された作業です。

引数

reassignWI の引数を次の表に示します。

表 12-29　reassignWI の引数

項番	仮引数名	名称	I/O	説明
1	aDBConnection	JDBC コネクション	in	CIWServer オブジェクトに関連づけられた JDBC コネクションを指定します。 null は指定できません。
2	aCIWServer	CIWServer オブジェクト	in	CIWServer オブジェクトを指定します。 null は指定できません。
3	aProcessInstanceID	案件 ID	in	対象となる作業の案件 ID を指定します。 null は指定できません。
4	aWorkItemID	作業 ID	in	対象となる作業の作業 ID を指定します。 null は指定できません。
5	aSource	変更する前の作業者 ID	in	変更する前の作業者 ID を指定します。 未設定の場合は，null を指定します。空文字列は指定できません。
6	aTarget	新しい作業者 ID	in	新しい作業者 ID を <SYSTEMID>_WORK_ITEM テーブルの Participant カラムのバイト数以下で指定します。 null を指定した場合，新しい作業者 ID は未設定となります。空文字列は指定できません。

戻り値

作業の作業者の再割り当てを実行したかどうかを返します。

true：実行しました

false：すでに作業の作業者が指定した新しい作業者 ID に割り当てられていたので実行しませんでした

例外

reassignWI で発生する例外を次の表に示します。

表 12-30　reassignWI の例外

項番	発生する例外	説明
1	CIWFatalException	処理を続行できない障害が発生した場合
2	CIWTransientException	一時的なエラーが発生した場合

項番	発生する例外	説明
3	CIWEntityNotExistException	処理しようとしたオブジェクトが存在しない場合
4	CIWStateException	状態や属性の変更に失敗した場合

注意事項

- JDBC コネクションおよび CIWServer オブジェクトにnull を指定した場合は，例外（java.lang.IllegalArgumentException）が発生します。
- JDBC コネクションと CIWServer オブジェクトが関連づけられていない場合の動作は保証しません。
- 案件 ID および作業 ID にnull を指定した場合は，例外（java.lang.IllegalArgumentException）が発生します。
- 対象となる作業はユーザタスクから変換された作業である必要があります。ユーザタスク以外から変換された作業を指定した場合は，例外（CIWStateException）が発生します。
- 指定した条件を満たす作業オブジェクトが存在しない場合は，例外（CIWEntityNotExistException）が発生します。
- 作業が「未終了」状態の場合だけ変更できます。「未終了」状態以外の場合は，例外（CIWStateException）が発生します。
- 引数aSource に指定した作業者 ID と対象となる作業の作業者 ID が異なる場合は，例外（CIWStateException）が発生します。対象となる作業の作業者 ID が引数aTarget に指定した作業者 ID と同じ場合は例外は発生しないでfalse が返ります。
- 作業者 ID に空文字列を指定した場合は，例外（java.lang.IllegalArgumentException）が発生します。
- 作業者 ID に<SYSTEMID>_WORK_ITEM テーブルのParticipant カラムのバイト数を超える文字列を指定した場合は，例外（java.lang.IllegalArgumentException）が発生します。

12.4.13　allocateWIEx（指定した条件に一致する作業の着手）

構文

```
CIWWorkItem allocateWIEx (
     java.sql.Connection aDBConnection,
     CIWServer aCIWServer,
     java.lang.String aParticipant,
     java.lang.String aFilter,
     java.lang.String aSort,
     java.util.Set<CIW WorkItemAttributeName> aAttributeNames,
     int aFetchCount
)
     throws CIWFatalException,
     CIWTransientException
```

機能

フィルター条件に指定した作業者 ID を持つユーザタスクの作業を，割り当て先作業者が着手した状態に変更します。作業者 ID は，フィルター条件に次のような条件を含めて指定します。

- Participant = '作業者 ID'
- Participant like '作業者 ID の前方の文字列%'

フィルター条件を指定しなくてもエラーにはなりませんが，ユーザタスク以外に実行中の作業がある場合，作業が取得できないことがあります。

動作の詳細は，次のとおりです。

指定したフィルター条件を満たし，指定した条件でソートした作業群に対して，指定した割り当て先の作業者 ID への作業の割り当てを実行します。最初に割り当てに成功した作業オブジェクトを返します。割り当てを実行する時，作業オブジェクトを「実行開始可能」状態から「作業者実行」状態に変更します。指定したフィルター条件を満たす作業がない場合や作業者を再割り当てできる作業がない場合はnull を返します。

フィルター条件には，「StateCode が"j"（実行開始可能）と一致すること」が自動的に追加されます。

引数

allocateWIEx の引数を次の表に示します。

表 12-31　allocateWIEx の引数

項番	仮引数名	名称	I/O	説明
1	aDBConnection	JDBC コネクション	in	CIWServer オブジェクトに関連づけられた JDBC コネクションを指定します。 null は指定できません。
2	aCIWServer	CIWServer オブジェクト	in	CIWServer オブジェクトを指定します。 null は指定できません。
3	aParticipant	割り当て先の作業者	in	割り当て先作業者を指定します。 空文字列およびnull は指定できません。
4	aFilter	フィルター条件	in	作業を検索する際のフィルター条件を指定します。ユーザタスクだけが対象になるParticipant を含んだフィルター条件を指定します。
5	aSort	ソート条件	in	作業を検索する際のソート条件を指定します。ソート条件を指定しない場合は，null を指定します。
6	aAttributeNames	属性名のセット	in	取得したい作業の属性名のセットを指定します。取得属性名を指定しない場

項番	仮引数名	名称	I/O	説明
6	aAttributeNames	属性名のセット	in	合は，サイズ 0 の属性名のセットまたはnull を指定します。取得属性名に null は指定できません。取得属性名を指定していない場合でも，作業の ID，作業が所属する案件の ID，作業者，作業の状態，作業の開始日時，および作業の作業定義 ID は必ず取得します。取得できる属性は，CIWWorkItem.AttributeName 列挙型で指定できる属性です。
7	aFetchCount	フェッチ件数	in	一度に取得する件数を指定します。

戻り値

着手した作業オブジェクトを返します。

例外

allocateWIEx で発生する例外を次の表に示します。

表 12-32　allocateWIEx の例外

項番	発生する例外	説明
1	CIWFatalException	処理を続行できない障害が発生した場合
2	CIWTransientException	一時的なエラーが発生した場合

注意事項

- JDBC コネクションおよび CIWServer オブジェクトにnull を指定した場合は，例外（java.lang.IllegalArgumentException）が発生します。
- JDBC コネクションおよび CIWServer オブジェクトが関連づけられていない場合の動作は保証しません。
- 不正な引数を指定した場合は，例外（java.lang.IllegalArgumentException）が発生します。
- このほかの注意事項は，CIWServer クラスのallocateWorkItemEx メソッドと共通です。詳細は，マニュアル『uCosminexus Service Coordinator Interactive Workflow AP 開発ガイド』の「CIWServer（ワーク管理システム全般にかかわる処理を行うための機能を提供するインタフェース)」のallocateWorkItemEx メソッドの説明を参照してください。

12.4.14　freeWI（作業の返却）

構文

```
boolean freeWI(
        java.sql.Connection aDBConnection,
        CIWServer aCIWServer,
        java.lang.Integer aProcessInstanceID,
        java.lang.Integer aWorkItemID
)
     throws CIWFatalException,
     CIWTransientException,
     CIWStateException,
     CIWEntityNotExistException,
     CIWTransitionFailedException
```

機能

案件 ID と作業 ID で指定された作業の状態を「作業者実行」から「実行開始可能」に変更し，作業者をレーン名に戻します。

対象となる作業は，ユーザタスクから変換された作業です。

引数

freeWI の引数を次の表に示します。

表 12-33　freeWI の引数

項番	仮引数名	名称	I/O	説明
1	aDBConnection	JDBC コネクション	in	CIWServer オブジェクトに関連づけられた JDBC コネクションを指定します。
2	aCIWServer	CIWServer オブジェクト	in	CIWServer オブジェクトを指定します。
3	aProcessInstanceID	案件 ID	in	対象である作業の案件 ID を指定します。 null は指定できません。
4	aWorkItemID	作業 ID	in	対象である作業の作業 ID を指定します。 null は指定できません。

戻り値

作業の返却を実行したかどうかを返します。

true：実行しました

false：作業の返却が実行済みだったので実行しませんでした

作業の返却が実行済みだったかどうかの判定については，「5.3　Java API 利用時の注意事項」の「べき等性について」の説明を参照してください。

例外

freeWI で発生する例外を次の表に示します。

表 12-34　freeWI の例外

項番	発生する例外	説明
1	CIWFatalException	処理を続行できない障害が発生した場合
2	CIWTransientException	一時的なエラーが発生した場合
3	CIWTransitionFailedException	案件処理中にエラーが発生した場合
4	CIWStateException	状態の遷移に失敗した場合
5	CIWEntityNotExistException	処理しようとしたオブジェクトが存在しない場合

注意事項

- JDBC コネクションおよび CIWServer オブジェクトにnull を指定した場合は，例外（java.lang.IllegalArgumentException）が発生します。
- JDBC コネクションおよび CIWServer オブジェクトが関連づけられていない場合の動作は保証しません。
- 案件 ID および作業 ID にnull を指定した場合は，例外（java.lang.IllegalArgumentException）が発生します。
- 対象となる作業はユーザタスクから変換された作業だけです。ユーザタスク以外から変換された作業を指定した場合は，例外（CIWStateException）が発生します。
- 指定した条件を満たす作業オブジェクトが存在しない場合は，例外（CIWEntityNotExistException）が発生します。
- 対象となる作業は，「作業者実行」状態である必要があります。そのほかの場合では，例外（CIWStateException）が発生します。対象となる作業が「実行開始可能」状態かつ作業者がレーン名の場合は，例外は発生しないでfalse が返ります。

12.4.15　setProcessData（プロセスデータの登録）

構文

```
void setProcessData(
        java.sql.Connection aDBConnection,
```

```
        java.lang.Integer aProcessInstanceID,
        java.util.Collection<CIWBPMNProcessData<?>> aProcessDataCollection
)
     throws CIWFatalException,
     CIWTransientException,
     CIWEntityNotExistException
```

機能

案件 ID で指定された案件に，プロセスデータを登録します。プロセスデータキー名がすでに存在している場合は，指定されたプロセスデータ値で更新します。なお，プロセスデータテーブルのProcessDataNameカラム値は次の形式で登録します。

- プロセスデータキー名の先頭の$N または$S が削除される
- リスト型プロセスデータを登録する場合，プロセスデータキー名の末尾の"{}"内に，先頭のデータから順番にリスト内識別子（番号）が付与される

引数

setProcessData の引数を次の表に示します。

表 12-35　setProcessData の引数

項番	仮引数名	名称	I/O	説明
1	aDBConnection	JDBC コネクション	in	JDBC コネクションを指定します。 null は指定できません。
2	aProcessInstanceID	案件 ID	in	プロセスデータを登録する案件の案件 ID を指定します。 null は指定できません。
3	aProcessDataCollection	プロセスデータのコレクション	in	複数件のプロセスデータを持つプロセスデータのコレクションを指定します。

戻り値

なし

例外

setProcessData で発生する例外を次の表に示します。

表 12-36　setProcessData の例外

項番	発生する例外	説明
1	CIWFatalException	処理を続行できない障害が発生した場合
2	CIWTransientException	一時的なエラーが発生した場合

項番	発生する例外	説明
3	CIWEntityNotExistException	処理しようとしたオブジェクトが存在しない場合

注意事項

- JDBC コネクションにnull を指定した場合は，例外（java.lang.IllegalArgumentException）が発生します。
- 案件 ID にnull を指定した場合は，例外（java.lang.IllegalArgumentException）が発生します。
- プロセスデータのコレクションにnull または空のコレクションを指定した場合は，プロセスデータを登録しません。
- プロセスデータのコレクションに重複したプロセスデータキー名のプロセスデータを指定した場合，例外（java.lang.IllegalArgumentException）が発生します。
- 指定した案件が存在する場合，案件の状態に関係なく，プロセスデータを登録または更新します。
- 指定した案件が存在しない場合は，例外（CIWEntityNotExistException）が発生します。

12.4.16　sendMessage（メッセージイベント送信）

構文

```
boolean sendMessage (
     java.sql.Connection aDBConnection,
     CIWServer aCIWServer,
     java.lang.Integer aProcessInstanceID,
     java.util.Collection<CIWBPMNProcessData<?>> aProcessDataCollection,
     String aMessageRef
)
     throws CIWFatalException,
     CIWTransitionFailedException,
     CIWTransientException,
     CIWEntityNotExistException,
     CIWStateException
```

機能

案件 ID で指定された案件に対して指定されたmessageRef のキャッチ処理を行います。指定されたmessageRef の受信待ちのイベントが複数ある場合はすべて処理します。指定されたプロセスデータを更新します。

引数

sendMessage の引数を次の表に示します。

表 12-37　sendMessage の引数

項番	仮引数名	名称	I/O	説明
1	aDBConnection	JDBC コネクション	in	CIWServer オブジェクトに関連づけられた JDBC コネクションを指定します。 null は指定できません。
2	aCIWServer	CIWServer オブジェクト	in	CIWServer オブジェクトを指定します。 null は指定できません。
3	aProcessInstanceID	案件 ID	in	メッセージを受信する案件の ID を指定します。 null は指定できません。
4	aProcessDataCollection	プロセスデータのコレクション	in	プロセスデータを更新しない場合は null を指定します。
5	aMessageRef	messageRef	in	メッセージを受信するイベントの messageRef を指定します。 空文字列および null は指定できません。

戻り値

キャッチを実行したかどうかを返します。

true：実行しました

false：指定した案件にキャッチ待ちが存在しなかったので実行しませんでした

例外

sendMessage で発生する例外を次の表に示します。

表 12-38　sendMessage の例外

項番	発生する例外	説明
1	CIWFatalException	処理を続行できない障害が発生した場合
2	CIWTransitionFailedException	案件処理中にエラーが発生した場合
3	CIWTransientException	一時的なエラーが発生した場合
4	CIWEntityNotExistException	処理しようとしたオブジェクトが存在しない場合
5	CIWStateException	状態や属性の変更に失敗した場合

注意事項

- JDBC コネクションおよび CIWServer オブジェクトにnull を指定した場合は，例外（java.lang.IllegalArgumentException）が発生します。
- JDBC コネクションと CIWServer オブジェクトが関連づけられていない場合の動作は保証しません。
- messageRef に空文字列またはnull を指定した場合は，例外（java.lang.IllegalArgumentException）が発生します。
- 案件 ID にnull を指定した場合は，例外（java.lang.IllegalArgumentException）が発生します。
- 指定した条件を満たす案件オブジェクトが存在しない場合は，例外（CIWEntityNotExistException）が発生します。
- 指定された案件が「実行中」状態ではない場合は，例外（CIWStateException）が発生します。

12.4.17　startMessage（案件投入（メッセージ））

構文

```
CIWProcessInstance startMessage(
     java.sql.Connection aDBConnection,
     CIWServer aCIWServer,
     java.lang.String aProcessDefinitionName,
     java.lang.Short aProcessDefinitionVersion,
     java.util.Map<CIWProcessInstance.AttributeName,java.lang.Object> aAttributes,
     java.util.Collection<CIWBPMNProcessData<?>> aProcessDataCollection,
     java.lang.String aMessageRef
)
     throws CIWFatalException,
     CIWTransitionFailedException,
     CIWTransientException,
     CIWEntityNotExistException,
     CIWStateException
```

機能

指定されたビジネスプロセス定義名とビジネスプロセス定義バージョンに対応するビジネスプロセス定義で，messageRef を指定して案件を生成し開始します。プロセスデータに親案件の案件 ID を登録します。指定されたプロセスデータは，生成した案件の案件 ID を指定して登録します。

指定されたmessageRef の開始(メッセージ)が定義されているビジネスプロセス定義しか案件投入できません。

引数

startMessage の引数を次の表に示します。

表 12-39　startMessage の引数

項番	仮引数名	名称	I/O	説明
1	aDBConnection	JDBC コネクション	in	CIWServer オブジェクトに関連づけられた JDBC コネクションを指定します。 null は指定できません。
2	aCIWServer	CIWServer オブジェクト	in	CIWServer オブジェクトを指定します。 null は指定できません。
3	aProcessDefinitionName	ビジネスプロセス定義名	in	ビジネスプロセス定義名を指定します。 空文字列およびnull は指定できません。
4	aProcessDefinitionVersion	ビジネスプロセス定義バージョン	in	ビジネスプロセス定義バージョンを指定します。null を指定した場合，指定したビジネスプロセス定義の中で投入可能（状態が活性かつ案件投入期間内）な最新バージョンになります。
5	aAttributes	案件属性のマップ	in	案件属性のマップを指定します。詳細は CIWServer インタフェースの createAndStartProcessInstance メソッドを参照。
6	aProcessDataCollection	プロセスデータのコレクション	in	プロセスデータを追加しない場合は null を指定します。
7	aMessageRef	messageRef	in	案件投入する開始(メッセージ)の messageRef を指定します。 空文字列およびnull は指定できません。

戻り値

開始した案件オブジェクトを返します。

例外

startMessage で発生する例外を次の表に示します。

表 12-40　startMessage の例外

項番	発生する例外	説明
1	CIWFatalException	処理を続行できない障害が発生した場合
2	CIWTransitionFailedException	案件処理中にエラーが発生した場合

項番	発生する例外	説明
3	CIWTransientException	一時的なエラーが発生した場合
4	CIWEntityNotExistException	処理しようとしたオブジェクトが存在しない場合
5	CIWStateException	状態や属性の変更に失敗した場合

注意事項

- JDBC コネクションおよび CIWServer オブジェクトにnull を指定した場合は，例外（java.lang.IllegalArgumentException）が発生します。
- JDBC コネクションと CIWServer オブジェクトが関連づけられていない場合の動作は保証しません。
- 指定されたビジネスプロセス定義が存在しない場合や案件投入できない場合（状態が非活性または案件投入期間外の場合），例外（CIWEntityNotExistException）が発生します。
- ビジネスプロセス定義バージョンにnull が指定された場合，指定されたビジネスプロセス定義の案件投入可能（状態が活性かつ案件投入期間内）な最新バージョンのビジネスプロセス定義で案件投入します。案件投入するビジネスプロセス定義に，指定されたmessageRef の開始(メッセージ)が定義されていない場合，例外（CIWEntityNotExistException）が発生します。
- messageRef に空文字列またはnull を指定した場合は，例外（java.lang.IllegalArgumentException）が発生します。
- 案件名の値に空文字列を指定した場合は，例外（java.lang.IllegalArgumentException）が発生します。
- 指定された属性名と属性値の型が一致しない場合は，例外（java.lang.IllegalArgumentException）が発生します。
- ビジネスプロセス定義の名称に空文字列またはnull を指定した場合は，例外（java.lang.IllegalArgumentException）が発生します。

12.4.18　createAndStartPIForTimer（案件投入（タイマー））

構文

```
CIWProcessInstance createAndStartPIForTimer (
     java.sql.Connection aDBConnection,
     CIWServer aCIWServer,
     java.lang.String aProcessDefinitionName,
     java.lang.Short aProcessDefinitionVersion,
     java.lang.String aWorkDefName
)
     throws CIWFatalException,
     CIWTransitionFailedException,
     CIWTransientException,
     CIWEntityNotExistException,
     CIWStateException
```

機能

ビジネスプロセス定義に定義された作業定義名を指定して，開始（タイマー）から案件を生成して開始します。対象となるビジネスプロセス定義は，ビジネスプロセス定義名，およびビジネスプロセス定義バージョンで指定します。

案件投入できるのは，指定された作業定義名の開始（タイマー）が定義されているビジネスプロセス定義だけです。

この API で案件を投入しても，次回の案件投入時刻や案件投入回数には影響ありません。

引数

createAndStartPIForTimer の引数を次の表に示します。

表 12-41　createAndStartPIForTimer の引数

項番	仮引数名	名称	I/O	説明
1	aDBConnection	JDBC コネクション	in	CIWServer オブジェクトに関連づけられた JDBC コネクションを指定します。 null は指定できません。
2	aCIWServer	CIWServer オブジェクト	in	CIWServer オブジェクトを指定します。 null は指定できません。
3	aProcessDefinitionName	ビジネスプロセス定義名	in	ビジネスプロセス定義名を指定します。 空文字列およびnull は指定できません。
4	aProcessDefinitionVersion	ビジネスプロセス定義バージョン	in	ビジネスプロセス定義バージョンを指定します。null を指定した場合，指定したビジネスプロセス定義の中で投入可能（状態が活性かつ案件投入期間内）な最新バージョンになります。
5	aWorkDefName	作業定義名	in	案件投入する開始（タイマー）の作業定義名を指定します。 空文字列およびnull は指定できません。

戻り値

開始した案件オブジェクトを返します。

例外

createAndStartPIForTimer で発生する例外を次の表に示します。

表 12-42　createAndStartPIForTimer の例外

項番	発生する例外	説明
1	CIWFatalException	処理を続行できない障害が発生した場合
2	CIWTransitionFailedException	案件処理中にエラーが発生した場合
3	CIWTransientException	一時的なエラーが発生した場合
4	CIWEntityNotExistException	処理しようとしたオブジェクトが存在しない場合
5	CIWStateException	状態や属性の変更に失敗した場合

注意事項

- JDBC コネクションおよび CIWServer オブジェクトにnull を指定した場合は，例外（java.lang.IllegalArgumentException）が発生します。
- JDBC コネクションおよび CIWServer オブジェクトが関連づけられていない場合の動作は保証しません。
- 指定されたビジネスプロセス定義が存在しない場合や案件投入可能（状態が活性かつ案件投入期間内）でない場合，例外（CIWEntityNotExistException）が発生します。
- ビジネスプロセス定義バージョンにnull が指定された場合，指定されたビジネスプロセス定義の案件投入可能（状態が活性かつ案件投入期間内）な最新バージョンで案件投入します。案件投入するビジネスプロセス定義に，指定された作業定義名の開始（タイマー）が定義されていない場合，例外（CIWEntityNotExistException）が発生します。
- 作業定義名に空文字列またはnull を指定した場合は，例外（java.lang.IllegalArgumentException）が発生します。
- 案件名は，「＜ビジネスプロセス定義名＞_＜バージョン（前ゼロ 4 桁）＞_＜ yyyymmddHHMMss 形式の時刻＞」になります。
- ビジネスプロセス定義名に空文字列またはnull を指定した場合は，例外（java.lang.IllegalArgumentException）が発生します。

12.4.19　setDeadlineForTimer（タイマーの処理期限を変更）

構文

```
void setDeadlineForTimer(
        java.sql.Connection aDBConnection,
        CIWServer aCIWServer,
        java.lang.Integer aProcessInstanceID,
```

```
        java.lang.Integer aWorkItemID,
        java.util.Date aNewDeadline
)
     throws CIWFatalException,
     CIWTransientException,
     CIWStateException,
     CIWEntityNotExistException
```

機能

案件 ID と作業 ID で指定された作業に対し，指定した日時を作業の処理期限の絶対日時として設定します。

対象となる作業は，タイマーイベントから変換された作業です。

引数

setDeadlineForTimer の引数を次の表に示します。

表 12-43　setDeadlineForTimer の引数

項番	仮引数名	名称	I/O	説明
1	aDBConnection	JDBC コネクション	in	CIWServer オブジェクトに関連づけられた JDBC コネクションを指定します。
2	aCIWServer	CIWServer オブジェクト	in	CIWServer オブジェクトを指定します。
3	aProcessInstanceID	案件 ID	in	対象である作業の案件 ID を指定します。 null は指定できません。
4	aWorkItemID	作業 ID	in	対象である作業の作業 ID を指定します。 null は指定できません。
5	aNewDeadline	変更後の処理期限	in	変更後の処理期限を指定します。 null は指定できません。

戻り値

なし

例外

setDeadlineForTimer で発生する例外を次の表に示します。

表 12-44　setDeadlineForTimer の例外

項番	発生する例外	説明
1	CIWFatalException	処理を続行できない障害が発生した場合

項番	発生する例外	説明
2	CIWTransientException	一時的なエラーが発生した場合
3	CIWStateException	状態の遷移に失敗した場合
4	CIWEntityNotExistException	処理しようとしたオブジェクトが存在しない場合

注意事項

- JDBC コネクションおよび CIWServer オブジェクトにnull を指定した場合は，例外（java.lang.IllegalArgumentException）が発生します。
- JDBC コネクションおよび CIWServer オブジェクトが関連づけられていない場合の動作は保証しません。
- 案件 ID および作業 ID にnull を指定した場合は，例外（java.lang.IllegalArgumentException）が発生します。
- 対象となる作業はタイマーから変換された作業です。タイマー以外から変換された作業を指定した場合は，例外（CIWStateException）が発生します。
- 指定した条件を満たす作業オブジェクトが存在しない場合は，例外（CIWEntityNotExistException）が発生します。
- 対象となる作業の状態は，「未終了」状態です。そのほかの場合は，例外（CIWStateException）が発生します。

12.4.20 createFlowNodeInstanceForAdHocSubProcess（アドホック・サブプロセスのフローノードを生成）

構文

```
java.util.List<CIWBPMNFlowNodeInstance> createFlowNodeInstanceForAdHocSubProcess(
        java.sql.Connection aDBConnection,
        CIWServer aCIWServer,
        java.lang.Integer aProcessInstanceID,
        java.lang.String aFlowNodeID,
        java.lang.String aFlowNodeName,
        java.lang.Integer aMIIndex,
        java.util.Collection<CIWBPMNProcessData<?>> aProcessDataCollection
)
     throws CIWFatalException,
     CIWTransientException,
     CIWTransitionFailedException,
     CIWStateException,
     CIWEntityNotExistException
```

機能

指定したアドホック・サブプロセス内のフローノード（実行中の業務ステップ）を生成します。また，指定したプロセスデータを更新します。

生成するフローノードは，アドホック・サブプロセス内に存在するフロー先端のフローノード（入力シーケンスフローが付与されていないフローノード）だけを対象とします。

引数

createFlowNodeInstanceForAdHocSubProcess の引数を次の表に示します。

表 12-45　createFlowNodeInstanceForAdHocSubProcess の引数

項番	仮引数名	名称	I/O	説明
1	aDBConnection	JDBC コネクション	in	CIWServer オブジェクトに関連づけられた JDBC コネクションを指定します。 null は指定できません。
2	aCIWServer	CIWServer オブジェクト	in	CIWServer オブジェクトを指定します。 null は指定できません。
3	aProcessInstanceID	案件 ID	in	対象であるアドホック・サブプロセスが所属する案件 ID を指定します。 null は指定できません。
4	aFlowNodeID	フローノード ID（BPMN 要素の id 属性値）	in	生成するフローノードのフローノード ID を指定します。 フローノード ID を指定しない場合は，null を指定します。null を指定したときは，必ずaFlowNodeName にフローノード名を指定します。 空文字列は指定できません。
5	aFlowNodeName	フローノード名（BPMN 要素の name 属性値）	in	生成するフローノードのフローノード名を指定します。 フローノード名を指定しない場合は，null を指定します。null を指定したときは，必ずaFlowNodeID にフローノード ID を指定します。
6	aMIIndex	マルチインスタンスインデクス	in	生成するフローノードが存在するアドホック・サブプロセスのマルチインスタンスインデクスを指定します。マルチインスタンスインデクスを指定しない場合は，null を指定します。

項番	仮引数名	名称	I/O	説明
7	aProcessDataCollection	プロセスデータのコレクション	in	プロセスデータを更新しない場合は null を指定します。

戻り値

生成したフローノードのリストを返します。フローノードがマルチインスタンスの場合，生成したすべてのフローノードのリストを返します。

フローノードが生成済みだった場合は，フローノードの生成を実行しないで，生成済みのフローノードのリストを返します。フローノードが生成済みだったかどうかの判定については，「5.3　Java API 利用時の注意事項」の「べき等性について」の説明を参照してください。

例外

createFlowNodeInstanceForAdHocSubProcess で発生する例外を次の表に示します。

表 12-46　createFlowNodeInstanceForAdHocSubProcess の例外

項番	発生する例外	説明
1	CIWFatalException	処理を続行できない障害が発生した場合
2	CIWTransientException	一時的なエラーが発生した場合
3	CIWTransitionFailedException	案件処理中にエラーが発生した場合
4	CIWStateException	状態や属性の変更に失敗した場合
5	CIWEntityNotExistException	処理しようとしたオブジェクトが存在しない場合

注意事項

- JDBC コネクションおよび CIWServer オブジェクトにnull を指定した場合は，例外（java.lang.IllegalArgumentException）が発生します。
- JDBC コネクションおよび CIWServer オブジェクトが関連づけられていない場合の動作は保証しません。
- 案件 ID にnull を指定した場合は，例外（java.lang.IllegalArgumentException）が発生します。
- フローノード ID に空文字を指定した場合は，例外（java.lang.IllegalArgumentException）が発生します。
- フローノード ID およびフローノード名の両方にnull を指定した場合は，例外（java.lang.IllegalArgumentException）が発生します。
- 指定した案件が存在しない場合は，例外（CIWEntityNotExistException）が発生します。
- 指定したフローノードが示す業務ステップ定義が存在しない場合は，例外（CIWEntityNotExistException）が発生します。

- 指定したフローノードが示す業務ステップ定義が複数存在する場合は，例外（CIWStateException）が発生します。
- 対象となるアドホック・サブプロセスが「生成可」状態（アドホック・サブプロセスに対応する作業が「実行開始可能」状態）ではない場合は，例外（CIWStateException）が発生します。
- 対象となるアドホック・サブプロセスの実行方式（Ordering）にSequential を指定した場合，アドホック・サブプロセス内に実行中のフローノードが存在するときは，例外（CIWStateException）が発生します。

12.4.21 changeStateAdHocSubProcess（アドホック・サブプロセスの状態変更）

構文

```
boolean changeStateAdHocSubProcess(
        java.sql.Connection aDBConnection,
        CIWServer aCIWServer,
        java.lang.Integer aProcessInstanceID,
        java.lang.String aAdHocSubProcessID,
        java.lang.String aAdHocSubProcessName,
        java.lang.Integer aMIIndex,
        CIWWorkItem.State aNewState,
        java.util.Collection<CIWBPMNProcessData<?>> aProcessDataCollection
)
     throws CIWFatalException,
     CIWTransientException,
     CIWTransitionFailedException,
     CIWStateException,
     CIWEntityNotExistException
```

機能

指定したアドホック・サブプロセスの状態を変更します。また，指定したプロセスデータを更新します。

アドホック・サブプロセス内のフロー遷移，および BPMN ビジネスプロセス定義ファイルに設定したアドホック・サブプロセスの属性値に関係なく，強制的にアドホック・サブプロセスの状態を変更できます。

引数

changeStateAdHocSubProcess の引数を次の表に示します。

表 12-47　changeStateAdHocSubProcess の引数

項番	仮引数名	名称	I/O	説明
1	aDBConnection	JDBC コネクション	in	CIWServer オブジェクトに関連づけられた JDBC コネクションを指定します。

項番	仮引数名	名称	I/O	説明
1	aDBConnection	JDBC コネクション	in	null は指定できません。
2	aCIWServer	CIWServer オブジェクト	in	CIWServer オブジェクトを指定します。 null は指定できません。
3	aProcessInstanceID	案件 ID	in	対象であるアドホック・サブプロセスが所属する案件 ID を指定します。 null は指定できません。
4	aAdHocSubProcessID	アドホック・サブプロセス ID （BPMN 要素の id 属性値）	in	状態変更するアドホック・サブプロセスのアドホック・サブプロセス ID を指定します。 アドホック・サブプロセス ID を指定しない場合は，null を指定します。null を指定したときは，必ず aAdHocSubProcessName にアドホック・サブプロセス名を指定します。 空文字列は指定できません。
5	aAdHocSubProcessName	アドホック・サブプロセス名 （BPMN 要素の name 属性値）	in	状態変更するアドホック・サブプロセスのアドホック・サブプロセス名を指定します。 アドホック・サブプロセス名を指定しない場合は，null を指定します。null を指定したときは，必ず aAdHocSubProcessID にアドホック・サブプロセス ID を指定します。
6	aMIIndex	マルチインスタンスインデクス	in	状態変更するアドホック・サブプロセスのマルチインスタンスインデクスを指定します。マルチインスタンスインデクスを指定しない場合は，null を指定します。
7	aNewState	アドホック・サブプロセスの状態に対応する作業の状態	in	変更する状態を指定します。 状態は，アドホック・サブプロセスの状態に対応する作業の状態です。値には，CIWWorkItem.State 列挙型の定数を指定できます。 null は指定できません。
8	aProcessDataCollection	プロセスデータのコレクション	in	プロセスデータを更新しない場合は null を指定します。

この API では，アドホック・サブプロセスの状態に対応する作業の状態をaNewState に指定します。アドホック・サブプロセスの状態の詳細については，「1.5.6　アドホック・サブプロセスの状態遷移モデル」を参照してください。

aNewState で指定できる作業の状態を次の表に示します。

現在の状態		指定できる状態		説明
定数	意味	定数	意味	
READY	実行開始可能	PERFORMING	作業者実行	アドホック・サブプロセス内の実行中のフローノード（業務ステップ）がすべて完了したあとに，アドホック・サブプロセスを完了する場合に指定します。
PERFORMING	作業者実行	READY	実行開始可能	アドホック・サブプロセスを「生成不可」状態にしたあとに，再度，アドホック・サブプロセス内のフローノード（業務ステップ）を生成する場合に指定します。
－	未終了	EXECUTED	実行済	アドホック・サブプロセスを強制的に完了する場合に指定します。

（凡例）

定数：CIWWorkItem.State 列挙型の定数を表します。

－：定数がないことを表します。

戻り値

アドホック・サブプロセスの状態変更を実行したかどうかを返します。

true：実行しました

false：アドホック・サブプロセスの状態変更が実行済みだったので実行しませんでした

アドホック・サブプロセスの状態変更が実行済みだったかどうかの判定については，「5.3　Java API 利用時の注意事項」の「べき等性について」の説明を参照してください。

例外

changeStateAdHocSubProcess で発生する例外を次の表に示します。

表 12-48　changeStateAdHocSubProcess の例外

項番	発生する例外	説明
1	CIWFatalException	処理を続行できない障害が発生した場合
2	CIWTransientException	一時的なエラーが発生した場合
3	CIWTransitionFailedException	案件処理中にエラーが発生した場合
4	CIWStateException	状態や属性の変更に失敗した場合
5	CIWEntityNotExistException	処理しようとしたオブジェクトが存在しない場合

注意事項

- JDBC コネクションおよび CIWServer オブジェクトにnull を指定した場合は，例外（java.lang.IllegalArgumentException）が発生します。
- JDBC コネクションおよび CIWServer オブジェクトが関連づけられていない場合の動作は保証しません。
- 案件 ID にnull を指定した場合は，例外（java.lang.IllegalArgumentException）が発生します。
- アドホック・サブプロセス ID に空文字を指定した場合は，例外（java.lang.IllegalArgumentException）が発生します。
- アドホック・サブプロセス ID およびアドホック・サブプロセス名の両方にnull を指定した場合は，例外（java.lang.IllegalArgumentException）が発生します。
- 指定した案件が存在しない場合は，例外（CIWEntityNotExistException）が発生します。
- 指定したアドホック・サブプロセスが存在しない場合は，例外（CIWEntityNotExistException）が発生します。
- 指定したアドホック・サブプロセスが複数存在する場合は，例外（CIWStateException）が発生します。
- 対象となるアドホック・サブプロセスに対応する作業の状態と，変更する状態の組み合わせが不正な場合は，例外（CIWStateException）が発生します。ただし，対象となるアドホック・サブプロセスに対応する作業の状態が変更する状態の場合は，false が返され，例外は発生しません。
- アドホック・サブプロセスを「生成不可」状態（アドホック・サブプロセスに対応する作業は「作業者実行」状態）に変更した際に，アドホック・サブプロセス内に実行中のフローノード（業務ステップ）が存在しない場合，アドホック・サブプロセスを完了（アドホック・サブプロセスに対応する作業を完了）します。
- アドホック・サブプロセスを完了した（アドホック・サブプロセスに対応する作業を完了した）場合は，対象となるアドホック・サブプロセス内に存在する実行中のフローノード（業務ステップ）を強制終了します。

12.4.22 getListProcessDataIndex（リスト型プロセスデータのインデクスを取得）

構文

```
java.lang.Integer getListProcessDataIndex (
        java.sql.Connection aDBConnection,
        java.lang.Integer aProcessInstanceID,
        java.lang.String aProcessDataKey,
        java.lang.Object aProcessDataValue
)
     throws CIWFatalException,
     CIWSQLTransientException,
     CIWEntityNotExistException
```

機能

指定したプロセスデータキー名とプロセスデータ値が示すリスト型プロセスデータのインデクス（リスト型プロセスデータの何番目に登録されているかを示す番号）を取得します。

引数

getListProcessDataIndex の引数を次の表に示します。

表 12-49　getListProcessDataIndex の引数

項番	仮引数名	名称	I/O	説明
1	aDBConnection	JDBC コネクション	in	JDBC コネクションを指定します。 null は指定できません。
2	aProcessInstanceID	案件 ID	in	対象であるプロセスデータの案件 ID を指定します。 null は指定できません。
3	aProcessDataKey	プロセスデータキー名	in	リスト型プロセスデータの全要素を示すプロセスデータキー名を指定します。プロセスデータキー名には"$"と型を示す種別を付けた変数名を指定します。 空文字列およびnull は指定できません。
4	aProcessDataValue	プロセスデータ値	in	プロセスデータ値を指定します。 プロセスデータ値に指定できるクラスを次に示します。 • String • Integer 空文字列は指定できません。 null を指定した場合はnull になります。

戻り値

取得したインデクスを返します。指定したプロセスデータ値がリスト型プロセスデータの中に複数登録されている場合，最初に検出された位置のインデクスを取得します。

例外

getListProcessDataIndex で発生する例外を次の表に示します。

表 12-50　getListProcessDataIndex の例外

項番	発生する例外	説明
1	CIWFatalException	処理を続行できない障害が発生した場合
2	CIWSQLTransientException	DBMS の一時的な要因で処理を実行できない場合

項番	発生する例外	説明
3	CIWEntityNotExistException	処理しようとしたオブジェクトが存在しない場合

注意事項

- JDBC コネクションにnull を指定した場合は，例外（java.lang.IllegalArgumentException）が発生します。
- 案件 ID にnull を指定した場合は，例外（java.lang.IllegalArgumentException）が発生します。
- プロセスデータキー名の書式が不正な場合は，例外（java.lang.IllegalArgumentException）が発生します。不正となる書式を次に示します。
  - プロセスデータキー名の命名規則に従っていない

    プロセスデータキー名の命名規則については，「1.6.3　プロセスデータテーブルの内容」の表「プロセスデータキー名の命名規則とプロセスデータ型の対応関係」を参照してください。
  - プロセスデータキー名の末尾が"{}"ではない
- プロセスデータ値に空文字列を指定した場合は，例外（java.lang.IllegalArgumentException）が発生します。
- 指定したプロセスデータが存在しない場合は，-1 を返します。
- 指定した案件が存在しない場合は，例外（CIWEntityNotExistException）が発生します。
- リスト型プロセスデータのリスト内識別子が連番の場合，取得するインデクスは，プロセスデータ値に対応するリスト内識別子と同値になります。リスト内識別子に欠番がある場合，登録されている順番はそのままで 1 から順に採番した値となります。この場合，リスト内識別子とは一致しません。
- 指定したプロセスデータキー名とプロセスデータ値の型が一致しない場合，型を一致させてからリスト型プロセスデータのインデクス取得処理を実施します。例えば，引数に次の値を指定した場合，プロセスデータ値"3"（数値型）のインデクスを返します。
  - プロセスデータキー名："$NKey{}"（数値型のプロセスデータキー名）
  - プロセスデータ値："3"（文字列型のプロセスデータ値）

  なお，プロセスデータキー名とプロセスデータ値の型が一致しない場合，例外は発生しません。例えば，引数に次の値を指定した場合，数値型のリスト型プロセスデータにプロセスデータ値"ABC"（文字列型）は存在しないため，-1 を返します。
  - プロセスデータキー名："$NKey{}"（数値型のプロセスデータキー名）
  - プロセスデータ値："ABC"

12.4.23　getPIIDListByProcessData（プロセスデータから案件 ID のリストを取得）

構文

```
List<Integer> getPIIDListByProcessData(
        java.sql.Connection aDBConnection,
        boolean aIsSortASC,
        int aOffset,
        int aMax,
        java.util.Collection<CIWBPMNProcessData<?>> aProcessDataCollection
)
     throws CIWFatalException,
     CIWSQLTransientException
```

機能

プロセスデータのコレクションで指定されたプロセスデータに対して，案件 ID を取得します。

指定されたプロセスデータをすべて保持している案件の ID が返されます。

引数

getPIIDListByProcessData の引数を次の表に示します。

表 12-51　getPIIDListByProcessData の引数

項番	仮引数名	名称	I/O	説明
1	aDBConnection	JDBC コネクション	in	JDBC コネクションを指定します。 null は指定できません。
2	aIsSortASC	ソート条件	in	案件 ID を取得する際のソート条件を指定します。 true を指定した場合は，案件 ID を昇順で取得します。false を指定した場合は，案件 ID を降順で取得します。
3	aOffset	オフセット値	in	案件 ID を取得する際のオフセットを指定します。 0 未満の値は指定できません。
4	aMax	最大取得件数	in	案件 ID を取得する際の最大取得件数を指定します。 -1 未満の値は指定できません。 0 を指定した場合は，空のコレクションが返されます。-1 を指定した場合は，全件取得します。

項番	仮引数名	名称	I/O	説明
5	aProcessDataCollection	プロセスデータのコレクション	in	複数件のプロセスデータを持つプロセスデータのコレクションを指定します。 null および空は指定できません。

戻り値

指定されたプロセスデータのコレクションの要素である各プロセスデータをすべて含む案件 ID のリストを返します。

例外

getPIIDListByProcessData で発生する例外を次の表に示します。

表 12-52　getPIIDListByProcessData の例外

項番	発生する例外	説明
1	CIWFatalException	処理を続行できない障害が発生した場合
2	CIWSQLTransientException	DBMS の一時的な要因で処理を実行できない場合

注意事項

- JDBC コネクションにnull を指定した場合は，例外（java.lang.IllegalArgumentException）が発生します。
- aProcessDataCollection のプロセスデータの中にnull が存在する場合は，例外（java.lang.IllegalArgumentException）が発生します。
- aOffset に 0 未満の値を指定した場合は，例外（java.lang.IllegalArgumentException）が発生します。
- aMax に-1 未満の値を指定した場合は，例外（java.lang.IllegalArgumentException）が発生します。
- aProcessDataCollection が空またはnull の場合は，例外（IllegalArgumentException）が発生します。
- aProcessDataCollection にリスト型プロセスデータを指定した場合は，指定したリスト型プロセスデータの全要素を含む案件 ID が返されます。なお，指定したリスト型プロセスデータに重複した要素が存在する場合，重複した個数分の要素を含む案件 ID が返されます。
- aProcessDataCollection に指定できるプロセスデータの数は，32 個までです。33 個以上のプロセスデータを指定した場合の動作は，サポートしません。なお，リスト型プロセスデータを指定した場合は，リストに格納されたプロセスデータ値の個数分を含みます。
- 指定したプロセスデータをすべて含む案件 ID が存在しない場合は，空のリストが返されます。

12.4.24 getProcessDataMapByMultiplePIID（複数の案件 ID からプロセスデータのマップを取得）

構文

```
Map<Integer, Map<String, CIWBPMNProcessData<?>>>
getProcessDataMapByMultiplePIID(
        java.sql.Connection aDBConnection,
        java.util.Collection<Integer> aPIIDCollection,
        java.util.Collection<String> aProcessDataKeyCollection
)
     throws CIWFatalException,
     CIWSQLTransientException
```

機能

案件 ID のコレクションで指定した案件に対して，プロセスデータキー名のコレクションで指定したプロセスデータを取得します。

取得結果は，案件 ID とプロセスデータ情報を含むマップとして返されます。

引数

getProcessDataMapByMultiplePIID の引数を次の表に示します。

表 12-53　getProcessDataMapByMultiplePIID の引数

項番	仮引数名	名称	I/O	説明
1	aDBConnection	JDBC コネクション	in	JDBC コネクションを指定します。 null は指定できません。
2	aPIIDCollection	案件 ID のコレクション	in	案件 ID のコレクションを指定します。 null および空は指定できません。
3	aProcessDataKeyCollection	プロセスデータキー名のコレクション	in	プロセスデータキー名のコレクションを指定します。

戻り値

次の表に示す，案件 ID およびプロセスデータの取得結果を格納したマップを返します。指定したプロセスデータキー名を 1 つも保持していない案件 ID は，戻り値のマップのキーに含まれません。

表 12-54　getProcessDataMapByMultiplePIID の戻り値

キー	値（取得結果を格納するためのマップ）	
	キー	値
案件 ID	プロセスデータキー名	プロセスデータ

キー	値（取得結果を格納するためのマップ）	
	キー	値
（Integer 型）	（String 型）	（CIWBPMNProcessData<?>型）

例外

getProcessDataMapByMultiplePIID で発生する例外を次の表に示します。

表 12-55　getProcessDataMapByMultiplePIID の例外

項番	発生する例外	説明
1	CIWFatalException	処理を続行できない障害が発生した場合
2	CIWSQLTransientException	DBMS の一時的な要因で処理を実行できない場合

注意事項

- JDBC コネクションにnull を指定した場合は，例外（java.lang.IllegalArgumentException）が発生します。
- 案件 ID のコレクションにnull または空を指定した場合は，例外（java.lang.IllegalArgumentException）が発生します。
- 案件 ID のコレクションの要素にnull が含まれた場合は，例外（java.lang.IllegalArgumentException）が発生します。
- プロセスデータキー名のコレクションがnull または空の場合は，指定した案件 ID のコレクションに合致したプロセスデータを全件返します。
- プロセスデータキー名のコレクションの要素にnull が含まれた場合は，例外（java.lang.IllegalArgumentException）が発生します。
- 指定したプロセスデータを保持している案件 ID が存在しない場合は，空のマップが返されます。

12.4.25　getFlowNodeDefinitionsList（フローノード定義のリストを取得）

構文

```
java.util.List<CIWBPMNFlowNodeDefinition> getFlowNodeDefinitionsList(
        java.sql.Connection aDBConnection,
        CIWServer aCIWServer,
        java.lang.String aProcessDefinitionName,
        java.lang.String aFlowNodeID,
        java.lang.String aFlowNodeName,
        java.util.Set<CIWBPMNFlowNodeDefinition.AttributeName> aAttributeNames
)
     throws CIWFatalException,
     CIWTransientException
```

機能

指定したフィルター条件（ビジネスプロセス定義名，フローノード ID，フローノード名）を満たすフローノード定義のリストを取得します。

引数

getFlowNodeDefinitionsList の引数を次の表に示します。

表 12-56　getFlowNodeDefinitionsList の引数

項番	仮引数名	名称	I/O	説明
1	aDBConnection	JDBC コネクション	in	CIWServer オブジェクトに関連づけられた JDBC コネクションを指定します。
2	aCIWServer	CIWServer オブジェクト	in	CIWServer オブジェクトを指定します。
3	aProcessDefinitionName	ビジネスプロセス定義名	in	取得したいフローノード定義が所属するビジネスプロセス定義名を指定します。 空文字列および null は指定できません。
4	aFlowNodeID	フローノード ID（BPMN 要素の id 属性値）	in	取得したいフローノード定義のフローノード ID を指定します。フローノード ID をフィルター条件に指定しない場合は，null を指定します。 空文字列は指定できません。
5	aFlowNodeName	フローノード名（BPMN 要素の name 属性値）	in	取得したいフローノード定義のフローノード名を指定します。フローノード名をフィルター条件に指定しない場合は，null を指定します。
6	aAttributeNames	属性名のセット	in	取得するフローノード定義属性名のセットを指定します。 取得属性名を指定しない場合は，サイズ 0 の属性名のセットまたは null を指定します。取得属性名に null は指定できません。取得属性名に指定されていない場合でも，フローノード ID，フローノード名，フローノードの種類，ref 識別子，calledElement は必ず取得します。取得できる属性は，CIWBPMNFlowNodeDefinition.AttributeName 列挙型で指定できる属性です。

戻り値

取得したフローノード定義のリストを返します。

例外

getFlowNodeDefinitionsList で発生する例外を次の表に示します。

表 12-57　getFlowNodeDefinitionsList の例外

項番	発生する例外	説明
1	CIWFatalException	処理を続行できない障害が発生した場合
2	CIWTransientException	一時的なエラーが発生した場合

注意事項

- JDBC コネクションおよび CIWServer オブジェクトにnull を指定した場合は，例外（java.lang.IllegalArgumentException）が発生します。
- JDBC コネクションおよび CIWServer オブジェクトが関連づけられていない場合の動作は保証しません。
- ビジネスプロセス定義名に空文字列またはnull を指定した場合は，例外（java.lang.IllegalArgumentException）が発生します。
- フローノード ID に空文字列を指定した場合は，例外（java.lang.IllegalArgumentException）が発生します。
- 指定したフィルター条件を満たすフローノード定義が存在しない場合は，空のリストを返します。
- 指定したビジネスプロセス定義名が示すビジネスプロセス定義に複数のバージョンが存在する場合，すべてのバージョンが検索対象となります。
- 検索対象となるフローノードは，「1.3.1　BPMN 連携機能で使用できる BPMN 要素」のアクティビティ，イベント，およびアドホック・サブプロセスとなります。ただし，次に示す BPMN 要素は検索対象外です。
  - 開始（タイプなし）
  - スロー（リンク）
  - キャッチ（リンク）

12.4.26　getFlowNodeInstancesListByPDName（ビジネスプロセス定義名を指定してフローノードのリストを取得）

構文

```
java.util.List<CIWBPMNFlowNodeInstance> getFlowNodeInstancesListByPDName(
        java.sql.Connection aDBConnection,
        CIWServer aCIWServer,
        java.lang.String aProcessDefinitionName,
        java.lang.String aFlowNodeID,
        java.lang.String aFlowNodeName,
        java.util.Set<CIWWorkItem.State> aStates,
        java.lang.Integer aMIIndex,
        java.util.Set<CIWBPMNFlowNodeInstance.AttributeName> aAttributeNames
)
     throws CIWFatalException,
     CIWTransientException
```

機能

指定したフィルター条件（ビジネスプロセス定義名，フローノード ID，フローノード名，作業の状態，マルチインスタンスインデクス）を満たすフローノードのリストを取得します。

指定したビジネスプロセス定義名の案件に所属する，フローノードのリストを取得します。子案件を含めてフローノードを取得する場合，「12.4.28　getFlowNodeInstancesListWithChildPIByPIID（案件 ID を指定して子案件を含めたフローノードのリストを取得）」を参照してください。

引数

getFlowNodeInstancesListByPDName の引数を次の表に示します。

表 12-58　getFlowNodeInstancesListByPDName の引数

項番	仮引数名	名称	I/O	説明
1	aDBConnection	JDBC コネクション	in	CIWServer オブジェクトに関連づけられた JDBC コネクションを指定します。
2	aCIWServer	CIWServer オブジェクト	in	CIWServer オブジェクトを指定します。
3	aProcessDefinitionName	ビジネスプロセス定義名	in	取得したいフローノードが所属する案件のビジネスプロセス定義名を指定します。 空文字列および null は指定できません。
4	aFlowNodeID	フローノード ID（BPMN 要素の id 属性値）	in	取得したいフローノードのフローノード ID を指定します。フロー

項番	仮引数名	名称	I/O	説明
4	aFlowNodeID	フローノード ID（BPMN 要素の id 属性値）	in	ノード ID をフィルター条件に指定しない場合は，null を指定します。 空文字列は指定できません。
5	aFlowNodeName	フローノード名（BPMN 要素の name 属性値）	in	取得したいフローノードのフローノード名を指定します。フローノード名をフィルター条件に指定しない場合は，null を指定します。
6	aStates	作業の状態のセット	in	取得したいフローノードに対応する作業の状態のセットを指定します。 指定できる作業の状態については，マニュアル『uCosminexus Service Coordinator Interactive Workflow AP 開発ガイド』の「付録 B.2　指定できる属性値」を参照してください。指定できない作業の状態は，フィルター条件に指定されません。 作業の状態にnull は指定できません。
7	aMIIndex	マルチインスタンスインデクス	in	取得したいフローノードのマルチインスタンスインデクスを指定します。マルチインスタンスインデクスをフィルター条件に指定しない場合は，null を指定します。
8	aAttributeNames	属性名のセット	in	取得するフローノード属性名のセットを指定します。 取得属性名を指定しない場合は，サイズ 0 の属性名のセットまたはnull を指定します。取得属性名にnull は指定できません。取得属性名に指定されていない場合でも，フローノード ID，フローノード名，フローノードの種類，作業 ID，作業が所属する案件の ID は必ず取得します。取得できる属性は，CIWBPMNFlowNodeInstance.AttributeName 列挙型で指定できる属性です。

戻り値

取得したフローノードのリストを返します。

例外

getFlowNodeInstancesListByPDName で発生する例外を次の表に示します。

表 12-59　getFlowNodeInstancesListByPDName の例外

項番	発生する例外	説明
1	CIWFatalException	処理を続行できない障害が発生した場合
2	CIWTransientException	一時的なエラーが発生した場合

注意事項

- JDBC コネクションおよび CIWServer オブジェクトにnull を指定した場合は，例外（java.lang.IllegalArgumentException）が発生します。
- JDBC コネクションおよび CIWServer オブジェクトが関連づけられていない場合の動作は保証しません。
- ビジネスプロセス定義名に空文字列またはnull を指定した場合は，例外（java.lang.IllegalArgumentException）が発生します。
- フローノード ID に空文字列を指定した場合は，例外（java.lang.IllegalArgumentException）が発生します。
- 作業の状態のセットに空のセットまたはnull を指定した場合は，例外（java.lang.IllegalArgumentException）が発生します。
- 指定したフィルター条件を満たすフローノードが存在しない場合は，空のリストを返します。
- 指定したビジネスプロセス定義名が示すビジネスプロセス定義に複数のバージョンが存在する場合，すべてのバージョンが検索対象となります。
- 検索対象となるフローノードは，「1.3.1　BPMN 連携機能で使用できる BPMN 要素」のアクティビティ，イベント，およびアドホック・サブプロセスとなります。ただし，次に示す BPMN 要素は検索対象外とします。
  - 開始（タイプなし）
  - スロー（リンク）
  - キャッチ（リンク）
- ワーク管理データベース上に同じビジネスプロセス定義名の案件が大量に存在する場合，大量のフローノードが取得される場合があります。取得対象のフローノードだけが検索されるように適切なフィルター条件（フローノード ID，フローノード名，作業の状態）を指定してください。
- すでに登録されたビジネスプロセス定義を変更する際に次の操作をした場合，操作前に生成されたフローノードインスタンスは検索対象外になります。
  - フローノード（BPMN 要素）を削除した場合
  - フローノード（BPMN 要素）の id 属性値または name 属性値を変更した場合

12.4.27　getFlowNodeInstancesListByPIID（案件 ID を指定してフローノードのリストを取得）

構文

```
java.util.List<CIWBPMNFlowNodeInstance> getFlowNodeInstancesListByPIID(
        java.sql.Connection aDBConnection,
        CIWServer aCIWServer,
        java.lang.Integer aProcessInstanceID,
        java.lang.String aFlowNodeID,
        java.lang.String aFlowNodeName,
        java.util.Set<CIWWorkItem.State> aStates,
        java.lang.Integer aMIIndex,
        java.util.Set<CIWBPMNFlowNodeInstance.AttributeName> aAttributeNames
)
     throws CIWFatalException,
     CIWTransientException
```

機能

指定したフィルター条件（案件 ID，フローノード ID，フローノード名，作業の状態，マルチインスタンスインデクス）を満たすフローノードのリストを取得します。

指定した案件 ID の案件に所属する，フローノードのリストを取得します。子案件を含めてフローノードを取得する場合，「12.4.28　getFlowNodeInstancesListWithChildPIByPIID（案件 ID を指定して子案件を含めたフローノードのリストを取得）」を参照してください。

引数

getFlowNodeInstancesListByPIID の引数を次の表に示します。

表 12-60　getFlowNodeInstancesListByPIID の引数

項番	仮引数名	名称	I/O	説明
1	aDBConnection	JDBC コネクション	in	CIWServer オブジェクトに関連づけられた JDBC コネクションを指定します。
2	aCIWServer	CIWServer オブジェクト	in	CIWServer オブジェクトを指定します。
3	aProcessInstanceID	案件 ID	in	取得したいフローノードが所属する案件 ID を指定します。 null は指定できません。
4	aFlowNodeID	フローノード ID（BPMN 要素の id 属性値）	in	取得したいフローノードのフローノード ID を指定します。フローノード ID をフィルター条件に指定しない場合は，null を指定します。 空文字列は指定できません。

項番	仮引数名	名称	I/O	説明
5	aFlowNodeName	フローノード名（BPMN 要素の name 属性値）	in	取得したいフローノードのフローノード名を指定します。フローノード名をフィルター条件に指定しない場合は，null を指定します。
6	aStates	作業の状態のセット	in	取得したいフローノードに対応する作業の状態のセットを指定します。 指定できる作業の状態については，マニュアル『uCosminexus Service Coordinator Interactive Workflow AP 開発ガイド』の「付録 B.2　指定できる属性値」を参照してください。指定できない作業の状態は，フィルター条件に指定されません。 作業の状態をフィルター条件に指定しない場合は，サイズ 0 の作業の状態のセットまたはnull を指定します。作業の状態にnull は指定できません。
7	aMIIndex	マルチインスタンスインデクス	in	取得したいフローノードのマルチインスタンスインデクスを指定します。マルチインスタンスインデクスをフィルター条件に指定しない場合は，null を指定します。
8	aAttributeNames	属性名のセット	in	取得するフローノード属性名のセットを指定します。 取得属性名を指定しない場合は，サイズ 0 の属性名のセットまたはnull を指定します。取得属性名にnull は指定できません。取得属性名に指定されていない場合でも，フローノード ID，フローノード名，フローノードの種類，作業 ID，作業が所属する案件の ID は必ず取得します。取得できる属性は，CIWBPMNFlowNodeInstance.AttributeName 列挙型で指定できる属性です。

戻り値

取得したフローノードのリストを返します。

例外

getFlowNodeInstancesListByPIID で発生する例外を次の表に示します。

表 12-61　getFlowNodeInstancesListByPIID の例外

項番	発生する例外	説明
1	CIWFatalException	処理を続行できない障害が発生した場合
2	CIWTransientException	一時的なエラーが発生した場合

注意事項

- JDBC コネクションおよび CIWServer オブジェクトにnull を指定した場合は，例外（java.lang.IllegalArgumentException）が発生します。
- JDBC コネクションおよび CIWServer オブジェクトが関連づけられていない場合の動作は保証しません。
- 案件 ID にnull を指定した場合は，例外（java.lang.IllegalArgumentException）が発生します。
- フローノード ID に空文字列を指定した場合は，例外（java.lang.IllegalArgumentException）が発生します。
- 指定したフィルター条件を満たすフローノードが存在しない場合は，空のリストを返します。
- 検索対象となるフローノードは，「1.3.1　BPMN 連携機能で使用できる BPMN 要素」のアクティビティ，イベント，およびアドホック・サブプロセスとなります。ただし，次に示す BPMN 要素は検索対象外です。
  - 開始（タイプなし）
  - スロー（リンク）
  - キャッチ（リンク）
- すでに登録されたビジネスプロセス定義を変更する際に次の操作をした場合，操作前に生成されたフローノードインスタンスは検索対象外になります。
  - フローノード（BPMN 要素）を削除した場合
  - フローノード（BPMN 要素）の id 属性値または name 属性値を変更した場合

12.4.28　getFlowNodeInstancesListWithChildPIByPIID（案件 ID を指定して子案件を含めたフローノードのリストを取得）

構文

```
java.util.List<CIWBPMNFlowNodeInstance> getFlowNodeInstancesListWithChildPIByPIID(
        java.sql.Connection aDBConnection,
        CIWServer aCIWServer,
        java.lang.Integer aProcessInstanceID,
        java.lang.String aFlowNodeID,
        java.lang.String aFlowNodeName,
        java.util.Set<CIWWorkItem.State> aStates,
```

```
        java.lang.Integer aMIIndex,
        java.util.Set<CIWBPMNFlowNodeInstance.AttributeName> aAttributeNames
)
    throws CIWFatalException,
    CIWTransientException
```

機能

指定したフィルター条件（案件 ID，フローノード ID，フローノード名，作業の状態，マルチインスタンスインデクス）を満たすフローノードのリストを取得します。

指定した案件 ID の案件に子案件が存在する場合，子案件を含めてフローノードのリストを取得します。子案件のフローノードを取得しない場合，次に示すメソッドを使用してください。

- 12.4.26　getFlowNodeInstancesListByPDName（ビジネスプロセス定義名を指定してフローノードのリストを取得）
- 12.4.27　getFlowNodeInstancesListByPIID（案件 ID を指定してフローノードのリストを取得）

引数

getFlowNodeInstancesListWithChildPIByPIID の引数を次の表に示します。

表 12-62　getFlowNodeInstancesListWithChildPIByPIID の引数

項番	仮引数名	名称	I/O	説明
1	aDBConnection	JDBC コネクション	in	CIWServer オブジェクトに関連づけられた JDBC コネクションを指定します。
2	aCIWServer	CIWServer オブジェクト	in	CIWServer オブジェクトを指定します。
3	aProcessInstanceID	案件 ID	in	取得したいフローノードのルート案件 ID を指定します。 子案件 ID が指定された場合，子案件のルート案件 ID が指定されたと仮定してフローノードのリストを取得します。 null は指定できません。
4	aFlowNodeID	フローノード ID（BPMN 要素の id 属性値）	in	取得したいフローノードのフローノード ID を指定します。フローノード ID をフィルター条件に指定しない場合は，null を指定します。 空文字列は指定できません。
5	aFlowNodeName	フローノード名（BPMN 要素の name 属性値）	in	取得したいフローノードのフローノード名を指定します。フローノード名をフィルター条件に指定しない場合は，null を指定します。

項番	仮引数名	名称	I/O	説明
6	aStates	作業の状態のセット	in	取得したいフローノードに対応する作業の状態のセットを指定します。 指定できる作業の状態については，マニュアル『uCosminexus Service Coordinator Interactive Workflow AP 開発ガイド』の「付録 B.2　指定できる属性値」を参照してください。指定できる作業の状態だけが，フィルター条件に指定されます。作業の状態をフィルター条件に指定しない場合は，サイズ 0 の作業の状態のセットまたはnull を指定します。 作業の状態にnull は指定できません。
7	aMIIndex	マルチインスタンスインデクス	in	取得したいフローノードのマルチインスタンスインデクスを指定します。マルチインスタンスインデクスをフィルター条件に指定しない場合は，null を指定します。
8	aAttributeNames	属性名のセット	in	取得するフローノード属性名のセットを指定します。 取得属性名を指定しない場合は，サイズ 0 の属性名のセットまたはnull を指定します。取得属性名にnull は指定できません。取得属性名に指定されていない場合でも，フローノード ID，フローノード名，フローノードの種類，作業 ID，作業が所属する案件の ID は必ず取得します。取得できる属性は，CIWBPMNFlowNodeInstance.AttributeName 列挙型で指定できる属性です。

戻り値

取得したフローノードのリストを返します。

例外

getFlowNodeInstancesListWithChildPIByPIID で発生する例外を次の表に示します。

表 12-63　getFlowNodeInstancesListWithChildPIByPIID の例外

項番	発生する例外	説明
1	CIWFatalException	処理を続行できない障害が発生した場合

項番	発生する例外	説明
2	CIWTransientException	一時的なエラーが発生した場合

注意事項

- JDBC コネクションおよび CIWServer オブジェクトにnull を指定した場合は，例外（java.lang.IllegalArgumentException）が発生します。
- JDBC コネクションおよび CIWServer オブジェクトが関連づけられていない場合の動作は保証しません。
- 案件 ID にnull を指定した場合は，例外（java.lang.IllegalArgumentException）が発生します。
- フローノード ID に空文字列を指定した場合は，例外（java.lang.IllegalArgumentException）が発生します。
- 指定したフィルター条件を満たすフローノードが存在しない場合は，空のリストを返します。
- 検索対象となるフローノードは，「1.3.1　BPMN 連携機能で使用できる BPMN 要素」のアクティビティ，イベント，およびアドホック・サブプロセスです。ただし，次に示す BPMN 要素は検索対象外です。
  - 開始（タイプなし）
  - スロー（リンク）
  - キャッチ（リンク）
- すでに登録されたビジネスプロセス定義を変更する際に次の操作をした場合，操作前に生成されたフローノードインスタンスは検索対象外になります。
  - フローノード（BPMN 要素）の削除
  - フローノード（BPMN 要素）の id 属性値または name 属性値の変更

12.4.29　getProcessInstancesListByPDName（ビジネスプロセス定義名を指定して案件のリストを取得）

構文

```
java.util.List<CIWProcessInstance> getProcessInstancesListByPDName(
        java.sql.Connection aDBConnection,
        CIWServer aCIWServer,
        java.lang.String aProcessDefinitionName,
        java.util.Set<CIWProcessInstance.State> aStates,
        java.util.Set<CIWProcessInstance.AttributeName> aAttributeNames
)
     throws CIWFatalException,
     CIWTransientException
```

機能

指定したフィルター条件（ビジネスプロセス定義名，案件の状態）を満たす案件のリストを取得します。

引数

getProcessInstancesListByPDName の引数を次の表に示します。

表 12-64　getProcessInstancesListByPDName の引数

項番	仮引数名	名称	I/O	説明
1	aDBConnection	JDBC コネクション	in	CIWServer オブジェクトに関連づけられた JDBC コネクションを指定します。
2	aCIWServer	CIWServer オブジェクト	in	CIWServer オブジェクトを指定します。
3	aProcessDefinitionName	ビジネスプロセス定義名	in	取得したい案件のビジネスプロセス定義名を指定します。 空文字列およびnull は指定できません。
4	aStates	案件の状態のセット	in	取得したい案件の状態のセットを指定します。案件の状態にnull は指定できません。
5	aAttributeNames	属性名のセット	in	取得する案件属性名のセットを指定します。 取得属性名を指定しない場合は，サイズ 0 の属性名のセットまたはnull を指定します。取得属性名にnull は指定できません。取得属性名に指定されていない場合でも，案件の ID は必ず取得します。取得できる属性は，CIWProcessInstance.AttributeName 列挙型で指定できる属性です。

戻り値

取得した案件のリストを返します。

例外

getProcessInstancesListByPDName で発生する例外を次の表に示します。

表 12-65　getProcessInstancesListByPDName の例外

項番	発生する例外	説明
1	CIWFatalException	処理を続行できない障害が発生した場合

項番	発生する例外	説明
2	CIWTransientException	一時的なエラーが発生した場合

注意事項

- JDBC コネクションおよび CIWServer オブジェクトにnull を指定した場合は，例外（java.lang.IllegalArgumentException）が発生します。
- JDBC コネクションおよび CIWServer オブジェクトが関連づけられていない場合の動作は保証しません。
- ビジネスプロセス定義名に空文字列またはnull を指定した場合は，例外（java.lang.IllegalArgumentException）が発生します。
- 案件の状態のセットに空のセットまたはnull を指定した場合は，例外（java.lang.IllegalArgumentException）が発生します。
- 指定したフィルター条件を満たす案件が存在しない場合は，空のリストを返します。
- 指定したビジネスプロセス定義名が示すビジネスプロセス定義に複数のバージョンが存在する場合，すべてのバージョンが検索対象となります。
- ワーク管理データベース上に同じビジネスプロセス定義名の案件が大量に存在する場合，大量の案件が取得されることがあります。取得対象の案件だけが検索されるように適切なフィルター条件（案件の状態）を指定してください。

12.4.30 getProcessInstancesListByPIName（ビジネスプロセス定義名と案件名を指定して案件のリストを取得）

構文

```
java.util.List<CIWProcessInstance> getProcessInstancesListByPIName(
        java.sql.Connection aDBConnection,
        CIWServer aCIWServer,
        java.lang.String aProcessDefinitionName,
        java.lang.String aProcessInstanceName,
        java.util.Set<CIWProcessInstance.State> aStates,
        java.util.Set<CIWProcessInstance.AttributeName> aAttributeNames
)
     throws CIWFatalException,
     CIWTransientException
```

機能

指定したフィルター条件（ビジネスプロセス定義名，案件名，案件の状態）を満たす案件のリストを取得します。

引数

getProcessInstancesListByPIName の引数を次の表に示します。

表 12-66　getProcessInstancesListByPIName の引数

項番	仮引数名	名称	I/O	説明
1	aDBConnection	JDBC コネクション	in	CIWServer オブジェクトに関連づけられた JDBC コネクションを指定します。
2	aCIWServer	CIWServer オブジェクト	in	CIWServer オブジェクトを指定します。
3	aProcessDefinitionName	ビジネスプロセス定義名	in	取得したい案件のビジネスプロセス定義名を指定します。 空文字列およびnull は指定できません。
4	aProcessInstanceName	案件名（案件キー）	in	取得したい案件の案件名を指定します。案件名が未設定の案件を取得する場合はnull を指定します。ただし，ワーク管理データベースが ORACLE の場合，null は指定できません。 空文字列は指定できません。
5	aStates	案件の状態のセット	in	取得したい案件の状態のセットを指定します。 案件の状態をフィルター条件に指定しない場合は，サイズ 0 の案件の状態のセットまたはnull を指定します。案件の状態にnull は指定できません。
6	aAttributeNames	属性名のセット	in	取得する案件属性名のセットを指定します。 取得属性名を指定しない場合は，サイズ 0 の属性名のセットまたはnull を指定します。取得属性名にnull は指定できません。取得属性名が指定されていない場合でも，案件の ID は必ず取得します。取得できる属性は，CIWProcessInstance.AttributeName 列挙型で指定できる属性です。

戻り値

取得した案件のリストを返します。

例外

getProcessInstancesListByPIName で発生する例外を次の表に示します。

表 12-67　getProcessInstancesListByPIName の例外

項番	発生する例外	説明
1	CIWFatalException	処理を続行できない障害が発生した場合
2	CIWTransientException	一時的なエラーが発生した場合

注意事項

- JDBC コネクションおよび CIWServer オブジェクトにnull を指定した場合は，例外（java.lang.IllegalArgumentException）が発生します。
- JDBC コネクションおよび CIWServer オブジェクトが関連づけられていない場合の動作は保証しません。
- ビジネスプロセス定義名に空文字列またはnull を指定した場合は，例外（java.lang.IllegalArgumentException）が発生します。
- 案件名に空文字列を指定した場合は，例外（java.lang.IllegalArgumentException）が発生します。
- 指定したフィルター条件を満たす案件が存在しない場合は，空のリストを返します。
- 指定したビジネスプロセス定義名が示すビジネスプロセス定義に複数のバージョンが存在する場合，すべてのバージョンが検索対象となります。

12.4.31　getCallActivityChildPI（コールアクティビティから投入された子案件を取得）

構文

```
CIWProcessInstance getCallActivityChildPI(
        java.sql.Connection aDBConnection,
        CIWServer aCIWServer,
        java.lang.Integer aProcessInstanceID,
        java.lang.Integer aWorkItemID,
        java.util.Set<CIWProcessInstance.AttributeName> aAttributeNames
)
     throws CIWFatalException,
     CIWTransientException,
     CIWEntityNotExistException
```

機能

案件 ID と作業 ID で指定したコールアクティビティから投入された子案件を取得します。

引数

getCallActivityChildPI の引数を次の表に示します。

表 12-68　getCallActivityChildPI の引数

項番	仮引数名	名称	I/O	説明
1	aDBConnection	JDBC コネクション	in	CIWServer オブジェクトに関連づけられた JDBC コネクションを指定します。
2	aCIWServer	CIWServer オブジェクト	in	CIWServer オブジェクトを指定します。
3	aProcessInstanceID	案件 ID	in	取得したい子案件を投入したコールアクティビティの案件 ID を指定します。 null は指定できません。
4	aWorkItemID	作業 ID	in	取得したい子案件を投入したコールアクティビティの作業 ID を指定します。 null は指定できません。
5	aAttributeNames	属性名のセット	in	取得したい子案件の属性名のセットを指定します。 取得したい子案件の属性名を指定しない場合は，サイズ 0 の属性名のセットまたはnull を指定します。取得したい属性名としてnull は指定できません。 取得したい子案件の属性名として指定されていない場合でも，案件の ID は必ず取得します。 取得できる属性は，CIWProcessInstance.AttributeName 列挙型で指定できる属性です。

戻り値

案件 ID と作業 ID で指定したコールアクティビティから投入された子案件の案件オブジェクトを返します。

案件 ID と作業 ID で指定したコールアクティビティから投入された子案件が存在しない場合，null を返します。

例外

getCallActivityChildPI で発生する例外を次の表に示します。

表 12-69　getCallActivityChildPI の例外

項番	発生する例外	説明
1	CIWFatalException	処理を続行できない障害が発生した場合
2	CIWTransientException	一時的なエラーが発生した場合
3	CIWEntityNotExistException	処理しようとしたオブジェクトが存在しない場合

注意事項

- JDBC コネクションおよび CIWServer オブジェクトにnull を指定した場合は，例外（java.lang.IllegalArgumentException）が発生します。
- JDBC コネクションと CIWServer オブジェクトが関連づけられていない場合の動作は保証しません。
- 案件 ID にnull を指定した場合は，例外（java.lang.IllegalArgumentException）が発生します。
- 作業 ID にnull を指定した場合は，例外（java.lang.IllegalArgumentException）が発生します。
- 指定した案件 ID と作業 ID の組み合わせの作業が存在しない場合は，例外（CIWEntityNotExistException）が発生します。

12.4.32 getCallActivityParentPI（対象案件の親案件を取得）

構文

```
CIWProcessInstance getCallActivityParentPI(
        java.sql.Connection aDBConnection,
        CIWServer aCIWServer,
        java.lang.Integer aProcessInstanceID,
        java.util.Set<CIWProcessInstance.AttributeName> aAttributeNames
)
     throws CIWFatalException,
     CIWTransientException,
     CIWEntityNotExistException
```

機能

案件 ID で指定した案件の親案件を取得します。

引数

getCallActivityParentPI の引数を次の表に示します。

表 12-70 getCallActivityParentPI の引数

項番	仮引数名	名称	I/O	説明
1	aDBConnection	JDBC コネクション	in	CIWServer オブジェクトに関連づけられた JDBC コネクションを指定します。
2	aCIWServer	CIWServer オブジェクト	in	CIWServer オブジェクトを指定します。
3	aProcessInstanceID	案件 ID	in	取得したい案件から呼ばれている子案件の案件 ID を指定します。 null は指定できません。

項番	仮引数名	名称	I/O	説明
4	aAttributeNames	属性名のセット	in	取得したい案件の属性名のセットを指定します。 取得したい案件の属性名を指定しない場合は，サイズ 0 の属性名のセットまたはnull を指定します。取得したい属性名としてnull は指定できません。 取得したい属性名として指定されていない場合でも，案件の ID は必ず取得します。 取得できる属性は，CIWProcessInstance.AttributeName 列挙型で指定できる属性です。

戻り値

対象となる案件 ID の親案件の案件オブジェクトを返します。親案件がいない案件 ID を指定した場合，null を返します。

例外

getCallActivityParentPI で発生する例外を次の表に示します。

表 12-71　getCallActivityParentPI の例外

項番	発生する例外	説明
1	CIWFatalException	処理を続行できない障害が発生した場合
2	CIWTransientException	一時的なエラーが発生した場合
3	CIWEntityNotExistException	処理しようとしたオブジェクトが存在しない場合

注意事項

- JDBC コネクションおよび CIWServer オブジェクトにnull を指定した場合は，例外（java.lang.IllegalArgumentException）が発生します。
- JDBC コネクションと CIWServer オブジェクトが関連づけられていない場合の動作は保証しません。
- 案件 ID にnull を指定した場合は，例外（java.lang.IllegalArgumentException）が発生します。
- 指定した案件 ID の案件が存在しない場合は，例外（CIWEntityNotExistException）が発生します。

12.4.33　getCallActivityParentWI（対象案件の呼び元の作業を取得）

構文

```
CIWWorkItem getCallActivityParentWI(
        java.sql.Connection aDBConnection,
        CIWServer aCIWServer,
        java.lang.Integer aProcessInstanceID,
        java.util.Set< CIWWorkItem.AttributeName> aAttributeNames
)
     throws CIWFatalException,
     CIWTransientException,
     CIWEntityNotExistException
```

機能

案件 ID で指定した案件の呼び元であるコールアクティビティの作業を取得します。

引数

getCallActivityParentWI の引数を次の表に示します。

表 12-72　getCallActivityParentWI の引数

項番	仮引数名	名称	I/O	説明
1	aDBConnection	JDBC コネクション	in	CIWServer オブジェクトに関連づけられた JDBC コネクションを指定します。
2	aCIWServer	CIWServer オブジェクト	in	CIWServer オブジェクトを指定します。
3	aProcessInstanceID	案件 ID	in	取得したい作業から呼ばれている子案件の案件 ID を指定します。 null は指定できません。
4	aAttributeNames	属性名のセット	in	取得したい作業の属性名のセットを指定します。 取得したい属性名を指定しない場合は，サイズ 0 の属性名のセットまたはnull を指定します。取得したい属性名としてnull は指定できません。 取得したい属性名として指定されていない場合でも，作業の ID と作業が所属する案件の ID は必ず取得します。 取得できる属性は，CIWWorkItem.AttributeName 列挙型で指定できる属性です。

戻り値

対象となる案件 ID の呼び元であるコールアクティビティの作業オブジェクトを返します。親案件がいない案件 ID を指定した場合，null を返します。

例外

getCallActivityParentWI で発生する例外を次の表に示します。

表 12-73　getCallActivityParentWI の例外

項番	発生する例外	説明
1	CIWFatalException	処理を続行できない障害が発生した場合
2	CIWTransientException	一時的なエラーが発生した場合
3	CIWEntityNotExistException	処理しようとしたオブジェクトが存在しない場合

注意事項

- JDBC コネクションおよび CIWServer オブジェクトにnull を指定した場合は，例外（java.lang.IllegalArgumentException）が発生します。
- JDBC コネクションと CIWServer オブジェクトが関連づけられていない場合の動作は保証しません。
- 案件 ID にnull を指定した場合は，例外（java.lang.IllegalArgumentException）が発生します。
- 指定した案件 ID の案件が存在しない場合は，例外（CIWEntityNotExistException）が発生します。

12.5 CIWBPMNProcessData（プロセスデータのインタフェース）

プロセスデータのキー名，値および種別を取得する機能を提供するインタフェースです。

クラス定義

```
public interface CIWBPMNProcessData<T>
```

入れ子のクラス

CIWBPMNProcessData インタフェースの入れ子のクラスを次の表に示します。

表 12-74 CIWBPMNProcessData インタフェースの入れ子のクラス

項番	メソッド名	説明および記述形式
1	CIWBPMNProcessData.Type	プロセスデータ種別
		static enum CIWBPMNProcessData.Type

メソッド

CIWBPMNProcessData インタフェースのメソッドを次の表に示します。

表 12-75 CIWBPMNProcessData インタフェースのメソッド

項番	メソッド名	説明
1	getKey	プロセスデータキー名を取得
2	getValue	プロセスデータ値を取得
3	getType	プロセスデータの種別を取得

12.5.1 getKey

インタフェース名：CIWBPMNProcessData

構文

```
java.lang.String getKey()
```

機能

プロセスデータキー名を取得します。

引数

なし

戻り値

プロセスデータキー名を返します。

12.5.2 getValue

インタフェース名：CIWBPMNProcessData

構文

```
T getValue()
```

機能

プロセスデータ値を取得します。

引数

なし

戻り値

プロセスデータ値<T> CIWBPMNProcessData<T>を返します。

プロセスデータ値がnull の場合はnull を返します。

T に指定できるクラスは次のクラスです。

- String
- Integer
- java.util.List<String>
- java.util.List<Integer>

12.5.3 getType

インタフェース名：CIWBPMNProcessData

構文

```
CIWBPMNProcessData.Type getType()
```

機能

プロセスデータの種別を取得します。

引数

なし

戻り値

プロセスデータの種別を返します。

12.6 CIWBPMNProcessDataFactory（プロセスデータオブジェクトの生成クラス）

CIWBPMNProcessDataFactory は，CIWBPMNProcessData オブジェクトのファクトリクラスであり，複数種別のプロセスデータを生成する機能を提供します。

クラス定義

```
public final class CIWBPMNProcessDataFactory
```

メソッド

CIWBPMNProcessDataFactory クラスのメソッドを次の表に示します。

表 12-76　CIWBPMNProcessDataFactory クラスのメソッド

項番	メソッド名	説明
1	createProcessData	CIWBPMNProcessData オブジェクトを生成

12.6.1 createProcessData

構文

```
<T> CIWBPMNProcessData<T> createProcessData(
    String aProcessDataKey, Object aProcessDataValue)
```

機能

プロセスデータキー名およびプロセスデータの値を指定し，該当する種別のプロセスデータのオブジェクトを生成します。

引数

createProcessData の引数を次の表に示します。

表 12-77　createProcessData の引数

項番	仮引数名	名称	I/O	説明
1	aProcessDataKey	プロセスデータキー名	in	プロセスデータキー名を指定します。プロセスデータキー名には”$”と型を示す種別を付けた変数名を指定します。空文字列およびnull は指定できません。

項番	仮引数名	名称	I/O	説明
2	aProcessDataValue	プロセスデータ値	in	プロセスデータ値を指定します。指定されたプロセスデータキー名の型とプロセスデータ値の型を一致させる必要があります。空文字列は指定できません。null を指定した場合は null になります。

戻り値

プロセスデータ値<T> CIWBPMNProcessData<T>を返します。

T に指定できるクラスは次のクラスです。

- String
- Integer
- java.util.List<String>
- java.util.List<Integer>

なお，プロセスデータを更新または登録できる Java API でリスト型プロセスデータを指定した場合，プロセスデータテーブルのProcessDataName カラム値は次のように決定されます。

- プロセスデータキー名の先頭の"$N"または”$S"が削除される
- プロセスデータキー名の末尾の"{}"内に，先頭のデータから順番にリスト内識別子（番号）が付与される

例外

なし

注意事項

- 指定されたプロセスデータキー名の書式が以下のように不正な場合は，例外（java.lang.IllegalArgumentException）が発生します。
  - プロセスデータキー名の命名規則に従わない

    プロセスデータキー名の命名規則については，「1.6.3　プロセスデータテーブルの内容」の表「プロセスデータキー名の命名規則とプロセスデータ型の対応関係」を参照してください。
  - <SYSTEMID>_PROCESS_DATA_S テーブルおよび<SYSTEMID>_PROCESS_DATA_N テーブルの ProcessDataName カラムのバイト長より長い

    なお，リスト型プロセスデータの全要素を示すプロセスデータキー名の場合の，テーブルに格納されるプロセスデータキー名のバイト長は次のとおりです。

    指定したプロセスデータキー名のバイト長＋リスト内識別子のバイト数
  - 空文字およびnull

- 指定されたプロセスデータ値が以下のように不正な場合，例外（java.lang.IllegalArgumentException）が発生します。
  - <SYSTEMID>_PROCESS_DATA_S テーブルのProcessDataValue カラムのバイト長より長い
  - 長さ 0 のString 型プロセスデータ値
  - リスト型プロセスデータの全要素を示すプロセスデータキー名を指定した際に，リストが空またはnull ただし，リストに格納されたプロセスデータ値がnull の場合，プロセスデータ値はnull になります。
- 指定されたプロセスデータキー名の型とプロセスデータ値の型が一致しない場合は，例外（java.lang.IllegalArgumentException）が発生します。

12.7 CIWBPMNFlowNodeInstance（フローノードのインタフェース）

フローノードオブジェクトの BPMN 要素の属性，およびフローノードオブジェクトに対応する CSCIW の作業の属性を取得する機能を提供するインタフェースです。

クラス定義

```
public interface CIWBPMNFlowNodeInstance
```

入れ子のクラス

CIWBPMNFlowNodeInstance インタフェースの入れ子のクラスを次の表に示します。

表 12-78 CIWBPMNFlowNodeInstance インタフェースの入れ子のクラス

項番	クラス名	説明および記述形式
1	CIWBPMNFlowNodeInstance.AttributeName	フローノードの属性名の列挙型
		static enum CIWBPMNFlowNodeInstance.AttributeName

メソッド

CIWBPMNFlowNodeInstance インタフェースのメソッドを次の表に示します。

表 12-79 CIWBPMNFlowNodeInstance インタフェースのメソッド

項番	メソッド名	説明
1	getFlowNodeID	フローノードでの BPMN 要素のid 属性値を取得
2	getFlowNodeMIIndex	フローノードでのマルチインスタンスインデクスを取得
3	getFlowNodeName	フローノードでの BPMN 要素のname 属性値を取得
4	getFlowNodeType	フローノードでの BPMN 要素の種類を取得
5	getActivityInstanceID	フローノードに対応する CSCIW の作業が所属する業務ステップの ID を取得
6	getProcessDefinitionID	フローノードに対応する CSCIW の作業が所属するビジネスプロセス定義の ID を取得
7	getProcessDefinitionName	フローノードに対応する CSCIW の作業が所属するビジネスプロセス定義の名称を取得
8	getProcessInstanceID	フローノードに対応する CSCIW の作業が所属する案件の ID を取得
9	getProcessInstanceName	フローノードに対応する CSCIW の作業の案件名（案件キー）を取得
10	getWorkItemClosedDate	フローノードに対応する CSCIW の作業の終了日時を取得
11	getWorkItemCreationDate	フローノードに対応する CSCIW の作業の発生日時を取得

項番	メソッド名	説明
12	getWorkItemDeadline	フローノードに対応する CSCIW の作業の処理期限の絶対日時を取得
13	getWorkItemID	フローノードに対応する CSCIW の作業の ID を取得
14	getWorkItemState	フローノードに対応する CSCIW の作業の状態を取得
15	getWorkItemStartDate	フローノードに対応する CSCIW の作業の開始日時を取得
16	getWorkItemParticipant	フローノードに対応する CSCIW の作業の作業者 ID を取得
17	isMultiInstance	フローノードがマルチインスタンスかどうかを判定

12.7.1 getFlowNodeID

構文

```
java.lang.String getFlowNodeID()
```

機能

フローノードでの BPMN 要素の id 属性値を取得します。

引数

なし

戻り値

フローノードでの BPMN 要素の id 属性値を返します。

12.7.2 getFlowNodeMIIndex

構文

```
java.lang.Integer getFlowNodeMIIndex()
     throws CIWFatalException
```

機能

フローノードでのマルチインスタンスインデクスを取得します。

引数

なし

戻り値

フローノードでのマルチインスタンスインデクスを返します。

例外

getFlowNodeMIIndex で発生する例外を次の表に示します。

表 12-80　getFlowNodeMIIndex の例外

項番	発生する例外	説明
1	CIWFatalException	処理を続行できない障害が発生した場合

注意事項

- フローノードがマルチインスタンスではない場合，null を返します。
- フローノードがユーザタスク（マルチインスタンス），サービスタスク（マルチインスタンス），ビジネスルールタスク（マルチインスタンス）またはコールアクティビティ（マルチインスタンス）の場合，マルチインスタンスインデクスを返します。
- フローノードがサブプロセス（マルチインスタンス）に含まれる場合，マルチインスタンスインデクスを返します。
- オブジェクト取得時にマルチインスタンスインデクスの取得を指定していない場合は，例外（CIWFatalException）が発生します。

12.7.3　getFlowNodeName

構文

```
java.lang.String getFlowNodeName()
```

機能

フローノードでの BPMN 要素のname 属性値を取得します。

引数

なし

戻り値

フローノードでの BPMN 要素のname 属性値を返します。

12.7.4 getFlowNodeType

構文

```
CIWBPMNFlowNodeDefinition.Type getFlowNodeType()
```

機能

フローノードでの BPMN 要素の種類を取得します。

引数

なし

戻り値

フローノードでの BPMN 要素の種類を返します。

12.7.5 getActivityInstanceID

構文

```
java.lang.Integer getActivityInstanceID()
     throws CIWFatalException
```

機能

フローノードに対応する CSCIW の作業が所属する業務ステップの ID を取得します。

引数

なし

戻り値

フローノードに対応する CSCIW の作業が所属する業務ステップの ID を返します。

例外

getActivityInstanceID で発生する例外を次の表に示します。

表 12-81 getActivityInstanceID の例外

項番	発生する例外	説明
1	CIWFatalException	処理を続行できない障害が発生した場合

注意事項

- オブジェクト取得時に業務ステップの ID の取得を指定していない場合は，例外（CIWFatalException）が発生します。

12.7.6 getProcessDefinitionID

構文

```
java.lang.Integer getProcessDefinitionID()
     throws CIWFatalException
```

機能

フローノードに対応する CSCIW の作業が所属するビジネスプロセス定義の ID を取得します。

引数

なし

戻り値

フローノードに対応する CSCIW の作業が所属するビジネスプロセス定義の ID を返します。

例外

getProcessDefinitionID で発生する例外を次の表に示します。

表 12-82 getProcessDefinitionID の例外

項番	発生する例外	説明
1	CIWFatalException	処理を続行できない障害が発生した場合

注意事項

- オブジェクト取得時にビジネスプロセス定義の ID の取得を指定していない場合は，例外（CIWFatalException）が発生します。

12.7.7 getProcessDefinitionName

構文

```
java.lang.String getProcessDefinitionName()
     throws CIWFatalException
```

機能

フローノードに対応する CSCIW の作業が所属するビジネスプロセス定義の名称を取得します。

引数

なし

戻り値

フローノードに対応する CSCIW の作業が所属するビジネスプロセス定義の名称を返します。

例外

getProcessDefinitionName で発生する例外を次の表に示します。

表 12-83　getProcessDefinitionName の例外

項番	発生する例外	説明
1	CIWFatalException	処理を続行できない障害が発生した場合

注意事項

- オブジェクト取得時にビジネスプロセス定義の名称の取得を指定していない場合は，例外（CIWFatalException）が発生します。

12.7.8 getProcessInstanceID

構文

```
java.lang.Integer getProcessInstanceID()
```

機能

フローノードに対応する CSCIW の作業が所属する案件の ID を取得します。

引数

なし

戻り値

フローノードに対応する CSCIW の作業が所属する案件の ID を返します。

12.7.9 getProcessInstanceName

構文

```
java.lang.String getProcessInstanceName()
     throws CIWFatalException
```

機能

フローノードに対応する CSCIW の作業の案件名（案件キー）を取得します。

引数

なし

戻り値

フローノードに対応する CSCIW の作業の案件名（案件キー）を返します。

例外

getProcessInstanceName で発生する例外を次の表に示します。

表 12-84 getProcessInstanceName の例外

項番	発生する例外	説明
1	CIWFatalException	処理を続行できない障害が発生した場合

注意事項

- 案件名（案件キー）が未設定の場合，null を返します。
- オブジェクト取得時に案件名（案件キー）の取得を指定していない場合は，例外（CIWFatalException）が発生します。

12.7.10　getWorkItemClosedDate

構文

```
java.util.Date getWorkItemClosedDate()
     throws CIWFatalException
```

機能

フローノードに対応する CSCIW の作業の終了日時を取得します。

引数

なし

戻り値

フローノードに対応する CSCIW の作業の終了日時を返します。

例外

getWorkItemClosedDate で発生する例外を次の表に示します。

表 12-85　getWorkItemClosedDate の例外

項番	発生する例外	説明
1	CIWFatalException	処理を続行できない障害が発生した場合

注意事項

- フローノードに対応する CSCIW の作業が「終了」状態へ遷移した場合に値が設定されます。「終了」状態への遷移の発生前はnull を返します。
- オブジェクト取得時に終了日時の取得を指定していない場合は，例外（CIWFatalException）が発生します。

12.7.11　getWorkItemCreationDate

構文

```
java.util.Date getWorkItemCreationDate()
     throws CIWFatalException
```

機能

フローノードに対応する CSCIW の作業の発生日時を取得します。

引数

なし

戻り値

フローノードに対応する CSCIW の作業の発生日時を返します。

例外

getWorkItemCreationDate で発生する例外を次の表に示します。

表 12-86　getWorkItemCreationDate の例外

項番	発生する例外	説明
1	CIWFatalException	処理を続行できない障害が発生した場合

注意事項

- フローノードに対応する CSCIW の作業が「実行開始」状態へ遷移した場合に値が設定されます。「実行開始」状態への遷移の発生前はnull を返します。
- オブジェクト取得時に発生日時の取得を指定していない場合は，例外（CIWFatalException）が発生します。

12.7.12　getWorkItemDeadline

構文

```
java.util.Date getWorkItemDeadline()
     throws CIWFatalException
```

機能

フローノードに対応する CSCIW の作業の処理期限の絶対日時を取得します。

引数

なし

戻り値

フローノードに対応する CSCIW の作業の処理期限の絶対日時を返します。

例外

getWorkItemDeadline で発生する例外を次の表に示します。

表 12-87　getWorkItemDeadline の例外

項番	発生する例外	説明
1	CIWFatalException	処理を続行できない障害が発生した場合

注意事項

- 処理期限が未設定の場合，null を返します。
- オブジェクト取得時に処理期限の取得を指定していない場合は，例外（CIWFatalException）が発生します。

12.7.13　getWorkItemID

構文

```
java.util.Integer getWorkItemID()
```

機能

フローノードに対応する CSCIW の作業の ID を取得します。

引数

なし

戻り値

フローノードに対応する CSCIW の作業の ID を返します。

12.7.14　getWorkItemState

構文

```
CIWWorkItem.State getWorkItemState()
     throws CIWFatalException
```

機能

フローノードに対応する CSCIW の作業の状態を取得します。

引数

なし

戻り値

フローノードに対応する CSCIW の作業の状態を返します。

例外

getWorkItemState で発生する例外を次の表に示します。

表 12-88　getWorkItemState の例外

項番	発生する例外	説明
1	CIWFatalException	処理を続行できない障害が発生した場合

注意事項

- オブジェクト取得時に状態の取得を指定していない場合は，例外（CIWFatalException）が発生します。

12.7.15　getWorkItemStartDate

構文

```
java.util.Date getWorkItemStartDate()
     throws CIWFatalException
```

機能

フローノードに対応する CSCIW の作業の開始日時を取得します。

引数

なし

戻り値

フローノードに対応する CSCIW の作業の開始日時を返します。

例外

getWorkItemStartDate で発生する例外を次の表に示します。

表 12-89　getWorkItemStartDate の例外

項番	発生する例外	説明
1	CIWFatalException	処理を続行できない障害が発生した場合

注意事項

- フローノードに対応する CSCIW の作業が「作業者実行」状態または「自動実行」状態へ遷移した場合に値が設定されます。「作業者実行」状態または「自動実行」状態への遷移の発生前はnull を返します。
- オブジェクト取得時に開始日時の取得を指定していない場合は，例外（CIWFatalException）が発生します。

12.7.16　getWorkItemParticipant

構文

```
java.lang.String getWorkItemParticipant()
     throws CIWFatalException
```

機能

フローノードに対応する CSCIW の作業の作業者 ID を取得します。

引数

なし

戻り値

フローノードに対応する CSCIW の作業の作業者 ID を返します。

例外

getWorkItemParticipant で発生する例外を次の表に示します。

表 12-90　getWorkItemParticipant の例外

項番	発生する例外	説明
1	CIWFatalException	処理を続行できない障害が発生した場合

注意事項

- 作業者 ID が未設定の場合，null を返します。
- オブジェクト取得時に作業者 ID の取得を指定していない場合は，例外（CIWFatalException）が発生します。

12.7.17　isMultiInstance

構文

```
boolean isMultiInstance()
```

機能

フローノードがマルチインスタンスかどうかを判定します。

引数

なし

戻り値

true：フローノードはマルチインスタンスです

false：フローノードはマルチインスタンスではありません

注意事項

- フローノードがユーザタスク（マルチインスタンス），サービスタスク（マルチインスタンス），ビジネスルールタスク（マルチインスタンス）またはコールアクティビティ（マルチインスタンス）の場合，true を返します。
- フローノードがサブプロセス（マルチインスタンス）に含まれる場合，true を返します。

12.8 CIWBPMNFlowNodeDefinition（フローノード定義のインタフェース）

フローノード定義オブジェクトの BPMN 要素の属性およびフローノード定義オブジェクトに対応する CSCIW の作業定義の属性を取得する機能を提供するインタフェースです。

クラス定義

```
public interface CIWBPMNFlowNodeDefinition
```

入れ子のクラス

CIWBPMNFlowNodeDefinition インタフェースの入れ子のクラスを次の表に示します。

表 12-91　CIWBPMNFlowDefinition インタフェースの入れ子のクラス

項番	クラス名	説明および記述形式
1	CIWBPMNFlowNodeDefinition.AttributeName	フローノード定義の属性名の列挙型
		static enum CIWBPMNFlowNodeDefinition.AttributeName
2	CIWBPMNFlowNodeDefinition.Type	フローノード定義での BPMN 要素の種類
		static enum CIWBPMNFlowNodeDefinition.Type

メソッド

CIWBPMNFlowNodeDefinition インタフェースのメソッドを次の表に示します。

表 12-92　CIWBPMNFlowNodeDefinition インタフェースのメソッド

項番	メソッド名	説明
1	getFlowNodeCalledElement	フローノード定義での BPMN 要素のcalledElement 属性値を取得
2	getFlowNodeID	フローノード定義での BPMN 要素のid 属性値を取得
3	getFlowNodeName	フローノード定義での BPMN 要素のname 属性値を取得
4	getFlowNodeRefID	フローノード定義での BPMN 要素のref 識別子属性値を取得
5	getFlowNodeType	フローノード定義での BPMN 要素の種類を取得
6	getProcessDefinitionID	フローノード定義に対応する CSCIW の作業定義が所属するビジネスプロセス定義の ID を取得
7	getProcessDefinitionName	フローノード定義に対応する CSCIW の作業定義が所属するビジネスプロセス定義の名称を取得
8	getWorkDefinitionID	フローノード定義に対応する CSCIW の作業定義の ID を取得
9	getWorkDefinitionName	フローノード定義に対応する CSCIW の作業定義の名称を取得

項番	メソッド名	説明
10	isMultiInstance	フローノード定義がマルチインスタンスかどうかを判定

12.8.1 getFlowNodeCalledElement

構文

```
java.lang.String getFlowNodeCalledElement()
```

機能

フローノード定義での BPMN 要素のcalledElement 属性値を取得します。

引数

なし

戻り値

フローノード定義での BPMN 要素のcalledElement 属性値を返します。

注意事項

- フローノードがコールアクティビティではない場合，null を返します。

12.8.2 getFlowNodeID

構文

```
java.lang.String getFlowNodeID()
```

機能

フローノード定義での BPMN 要素のid 属性値を取得します。

引数

なし

戻り値

フローノード定義での BPMN 要素のid 属性値を返します。

12.8.3 getFlowNodeName

構文

```
java.lang.String getFlowNodeName()
```

機能

フローノード定義での BPMN 要素のname 属性値を取得します。

引数

なし

戻り値

フローノード定義での BPMN 要素のname 属性値を返します。

12.8.4 getFlowNodeRefID

構文

```
java.lang.String getFlowNodeRefID()
```

機能

フローノード定義での BPMN 要素のref 識別子属性値を取得します。

引数

なし

戻り値

フローノード定義での BPMN 要素のref 識別子属性値を返します。

注意事項

- ref 識別子属性はサービスタスク，開始（メッセージ）などの BPMN 要素の場合に設定できます。フローノードがref 識別子属性を設定できない BPMN 要素の場合，null を返します。

12.8.5 getFlowNodeType

構文

```
CIWBPMNFlowNodeDefinition.Type getFlowNodeType()
```

機能

フローノード定義での BPMN 要素の種類を取得します。

引数

なし

戻り値

フローノード定義での BPMN 要素の種類を返します。

12.8.6 getProcessDefinitionID

構文

```
java.lang.Integer getProcessDefinitionID()
     throws CIWFatalException
```

機能

フローノード定義に対応する CSCIW の作業定義が所属するビジネスプロセス定義の ID を取得します。

引数

なし

戻り値

フローノード定義に対応する CSCIW の作業定義が所属するビジネスプロセス定義の ID を返します。

例外

getProcessDefinitionID で発生する例外を次の表に示します。

表 12-93 getProcessDefinitionID の例外

項番	発生する例外	説明
1	CIWFatalException	処理を続行できない障害が発生した場合

注意事項

- オブジェクト取得時にビジネスプロセス定義の ID の取得を指定していない場合は，例外（CIWFatalException）が発生します。

12.8.7 getProcessDefinitionName

構文

```
java.lang.String getProcessDefinitionName()
     throws CIWFatalException
```

機能

フローノードに対応する CSCIW の作業定義が所属するビジネスプロセス定義の名称を取得します。

引数

なし

戻り値

フローノードに対応する CSCIW の作業定義が所属するビジネスプロセス定義の名称を返します。

例外

getProcessDefinitionName で発生する例外を次の表に示します。

表 12-94 getProcessDefinitionName の例外

項番	発生する例外	説明
1	CIWFatalException	処理を続行できない障害が発生した場合

注意事項

- オブジェクト取得時にビジネスプロセス定義の名称の取得を指定していない場合は，例外（CIWFatalException）が発生します。

12.8.8 getWorkDefinitionID

構文

```
java.lang.Integer getWorkDefinitionID()
     throws CIWFatalException
```

機能

フローノード定義に対応する CSCIW の作業定義の ID を取得します。

引数

なし

戻り値

フローノード定義に対応する CSCIW の作業定義の ID を返します。

例外

getWorkDefinitionID で発生する例外を次の表に示します。

表 12-95 getWorkDefinitionID の例外

項番	発生する例外	説明
1	CIWFatalException	処理を続行できない障害が発生した場合

注意事項

- オブジェクト取得時に作業定義の ID の取得を指定していない場合は，例外（CIWFatalException）が発生します。

12.8.9 getWorkDefinitionName

構文

```
java.lang.String getWorkDefinitionName()
     throws CIWFatalException
```

機能

フローノードに対応する CSCIW の作業定義の名称を取得します。

引数

なし

戻り値

フローノードに対応する CSCIW の作業定義の名称を返します。

例外

getWorkDefinitionName で発生する例外を次の表に示します。

項番	発生する例外	説明
1	CIWFatalException	処理を続行できない障害が発生した場合

注意事項

- オブジェクト取得時に作業定義の名称の取得を指定していない場合は，例外（CIWFatalException）が発生します。

12.8.10　isMultiInstance

構文

```
boolean isMultiInstance()
```

機能

フローノード定義がマルチインスタンスかどうかを判定します。

引数

なし

戻り値

true：フローノード定義はマルチインスタンスです

false：フローノード定義はマルチインスタンスではありません

注意事項

- フローノード定義がユーザタスク（マルチインスタンス），サービスタスク（マルチインスタンス），ビジネスルールタスク（マルチインスタンス）またはコールアクティビティ（マルチインスタンス）の場合，true を返します。

- フローノード定義がサブプロセス（マルチインスタンス）に含まれる場合，true を返します。

12.9　BPMN 連携ライブラリが提供する列挙型

12.9.1　CIWBPMNProcessData.Type（プロセスデータ種別の列挙型）

概要

プロセスデータの種別の列挙型です。

定義

```
static enum CIWBPMNProcessData.Type
```

すべての実装されたインタフェース

```
CIWBPMNProcessData
```

定数

CIWBPMNProcessData.Type の列挙型定数を次の表に示します。

表 12-96　CIWBPMNProcessData.Type の列挙型定数

項番	定数名	説明
1	INTEGER	種別の「数値型」を表す定数です。 プロセスデータキー名は”$N”から始まります。
2	STRING	種別の「文字列型」を表す定数です。 プロセスデータキー名は”$S”から始まります。
3	SINGLE_LIST_INTEGER	種別の「リスト型（数値）の 1 要素」を表す定数です。 プロセスデータキー名は”$N”から始まります。末尾は”{<リスト内識別子 >}”で終わります。
4	SINGLE_LIST_STRING	種別の「リスト型（文字列）の 1 要素」を表す定数です。 プロセスデータキー名は”$S”から始まります。末尾は”{<リスト内識別子 >}”で終わります。
5	ALL_LIST_INTEGER	種別の「リスト型（数値）の全要素」を表す定数です。 プロセスデータキー名は”$N”から始まります。末尾は”{}”で終わります。
6	ALL_LIST_STRING	種別の「リスト型（文字列）の全要素」を表す定数です。 プロセスデータキー名は”$S”から始まります。末尾は”{}”で終わります。

12.9.2 CIWBPMNFlowNodeInstance.AttributeName（フローノードの属性名の列挙型）

概要

フローノードの属性名の列挙型です。

定義

```
static enum CIWBPMNFlowNodeInstance.AttributeName
```

すべての実装されたインタフェース

```
CIWBPMNFlowNodeInstance
```

定数

CIWBPMNFlowNodeInstance.AttributeName の列挙型定数を次の表に示します。

表 12-97　CIWBPMNFlowNodeInstance.AttributeName の列挙型定数

項番	定数名	説明
1	FLOW_NODE_ID	BPMN 要素のid 属性を表す定数です。
2	FLOW_NODE_MIINDEX	フローノードのマルチインスタンスインデクスを表す定数です。
3	FLOW_NODE_NAME	BPMN 要素のname 属性を表す定数です。
4	FLOW_NODE_TYPE	BPMN 要素の種類を表す定数です。
5	ACTIVITY_INSTANCE_ID	フローノードに対応する CSCIW の作業が所属する業務ステップの ID を表す定数です。
6	PROCESS_DEFINITION_ID	フローノードに対応する CSCIW の作業が所属するビジネスプロセス定義の ID を表す定数です。
7	PROCESS_DEFINITION_NAME	フローノードに対応する CSCIW の作業が所属するビジネスプロセス定義の名称を表す定数です。
8	PROCESS_INSTANCE_ID	フローノードに対応する CSCIW の作業が所属する案件の ID を表す定数です。
9	PROCESS_INSTANCE_NAME	フローノードに対応する CSCIW の作業の案件名（案件キー）を表す定数です。
10	WORK_ITEM_CLOSED_DATE	フローノードに対応する CSCIW の作業の終了日時を表す定数です。
11	WORK_ITEM_CREATION_DATE	フローノードに対応する CSCIW の作業の発生日時を表す定数です。
12	WORK_ITEM_DEADLINE	フローノードに対応する CSCIW の作業の処理期限の絶対日時を表す定数です。

項番	定数名	説明
13	WORK_ITEM_ID	フローノードに対応する CSCIW の作業の ID を表す定数です。
14	WORK_ITEM_STATE	フローノードに対応する CSCIW の作業の状態を表す定数です。
15	WORK_ITEM_START_DATE	フローノードに対応する CSCIW の作業の開始日時を表す定数です。
16	WORK_ITEM_PARTICIPANT	フローノードに対応する CSCIW の作業の作業者 ID を表す定数です。

12.9.3 CIWBPMNFlowNodeDefinition.AttributeName（フローノード定義の属性名の列挙型）

概要

フローノード定義の属性名の列挙型です。

定義

```
static enum CIWBPMNFlowNodeDefinition.AttributeName
```

すべての実装されたインタフェース

```
CIWBPMNFlowNodeDefinition
```

定数

CIWBPMNFlowNodeDefinition.AttributeName の列挙型定数を次の表に示します。

表 12-98 CIWBPMNFlowNodeDefinition.AttributeName の列挙型定数

項番	定数名	説明
1	FLOW_NODE_CALLED_ELEMENT	BPMN 要素の calledElement 属性を表す定数です。
2	FLOW_NODE_ID	BPMN 要素の id 属性を表す定数です。
3	FLOW_NODE_NAME	BPMN 要素の name 属性を表す定数です。
4	FLOW_NODE_REF_ID	BPMN 要素の ref 識別子属性を表す定数です。
5	FLOW_NODE_TYPE	BPMN 要素の種類を表す定数です。
6	PROCESS_DEFINITION_ID	フローノード定義に対応する CSCIW の作業定義が所属するビジネスプロセス定義の ID を表す定数です。
7	PROCESS_DEFINITION_NAME	フローノード定義に対応する CSCIW の作業定義が所属するビジネスプロセス定義の名称を表す定数です。

項番	定数名	説明
8	WORK_DEFINITION_ID	フローノード定義に対応する CSCIW の作業定義の ID を表す定数です。
9	WORK_DEFINITION_NAME	フローノード定義に対応する CSCIW の作業定義の名称を表す定数です。

12.9.4 CIWBPMNFlowNodeDefinition.Type（フローノード定義における BPMN 要素の種類の列挙型）

概要

フローノード定義での BPMN 要素の種類の列挙型です。

定義

```
static enum CIWBPMNFlowNodeDefinition.Type
```

すべての実装されたインタフェース

```
CIWBPMNFlowNodeDefinition
```

定数

CIWBPMNFlowNodeDefinition.Type の列挙型定数を次の表に示します。

表 12-99 CIWBPMNFlowNodeDefinition.Type の列挙型定数

項番	定数名	説明
1	USER_TASK	BPMN 要素のユーザタスクを表す列挙型です。
2	SERVICE_TASK	BPMN 要素のサービスタスクを表す列挙型です。
3	BUSINESS_RULE_TASK	BPMN 要素のビジネスルールタスクを表す列挙型です。
4	CALL_ACTIVITY	BPMN 要素のコールアクティビティを表す列挙型です。
5	MESSAGE_BOUNDARY_INTERRUPT	BPMN 要素の境界中断（メッセージ）を表す列挙型です。
6	MESSAGE_BOUNDARY_NONINTERRUPT	BPMN 要素の境界非中断（メッセージ）を表す列挙型です。
7	MESSAGE_CATCH	BPMN 要素のキャッチ（メッセージ）を表す列挙型です。
8	MESSAGE_SUBSTART_INTERRUPT	BPMN 要素のイベント・サブプロセス中断開始（メッセージ）を表す列挙型です。
9	MESSAGE_SUBSTART_NONINTERRUPT	BPMN 要素のイベント・サブプロセス非中断開始（メッセージ）を表す列挙型です。

項番	定数名	説明
10	MESSAGE_TOPSTART	BPMN 要素の開始（メッセージ）を表す列挙型です。
11	TIMER_BOUNDARY_INTERRUPT	BPMN 要素の境界中断（タイマー）を表す列挙型です。
12	TIMER_BOUNDARY_NONINTERRUPT	BPMN 要素の境界非中断（タイマー）を表す列挙型です。
13	TIMER_CATCH	BPMN 要素のキャッチ（タイマー）を表す列挙型です。
14	TIMER_SUBSTART_INTERRUPT	BPMN 要素のイベント・サブプロセス中断開始（タイマー）を表す列挙型です。
15	TIMER_SUBSTART_NONINTERRUPT	BPMN 要素のイベント・サブプロセス非中断開始（タイマー）を表す列挙型です。
16	TIMER_TOPSTART	BPMN 要素の開始（タイマー）を表す列挙型です。
17	ADHOC_SUBPROCESS	BPMN 要素のアドホック・サブプロセスを表す列挙型です。
18	UNDEFINED	上記以外で未定義の BPMN 要素を表す列挙型です。この列挙型に該当する BPMN 要素の種類の判別はできません。

13 Java オブジェクト呼び出しに使用するクラス

アプリケーション呼び出しサービスが Java オブジェクトを呼び出す際に使用するインタフェースの詳細について説明します。

13.1 Java オブジェクト呼び出しに使用するクラスの一覧

アプリケーション呼び出しサービスが Java オブジェクトを呼び出す際に使用する，クラスおよびインタフェースを次に示します。

パッケージ：jp.co.Hitachi.soft.csciw.bpmn.callback のクラスおよびインタフェース一覧

項番	クラス名・インタフェース名	説明
1	CIWBpmnWorkApplication	アプリケーション呼び出しサービスが呼び出す Java オブジェクトのインタフェース

13.2 CIWBpmnWorkApplication（Java オブジェクト呼び出しのインタフェース）

アプリケーション呼び出しサービスによる，Java オブジェクト呼び出し用のインタフェースです。アプリケーション呼び出しサービスから呼び出される Java オブジェクトには，CIWBpmnWorkApplication インタフェースを実装してください。

メソッド

CIWBpmnWorkApplication インタフェースのメソッドを次の表に示します。

表 13-1　CIWBpmnWorkApplication インタフェースのメソッド

項番	メソッド名	説明
1	execute	Java オブジェクトの業務処理を呼び出します。

13.2.1 execute

構文

```
Map<String,Object> execute(
            Map<String, Object> aParameters )
throws CIWUserException
```

機能

Java オブジェクトの業務処理を呼び出します。

引数

execute の引数を次の表に示します。

表 13-2　execute の引数

項番	引数名	名称	I/O	説明
1	aParameters	アプリケーション呼び出し情報マップ	in	アプリケーション呼び出し情報ファイルに指定したプロパティの値を取得します。

戻り値

java.util.Map を継承するクラスのインスタンスを指定してください。

Map のキーには，アプリケーション呼び出し情報ファイルに指定した java.return.key.<key 要素値>の key 要素値をすべて指定してください。また，Map の値には，String 型，Integer 型，List<String>型，またはList<Integer>型のどれかの型の値を指定してください。

アプリケーション呼び出し情報ファイルに java.return.key.<key 要素値>の指定がない場合，戻り値には空の Map（java.util.Collections.emptyMap()）を指定してください。

例外

execute メソッドでエラーを検出した場合は，CIWUserException をスローしてください。

CIWUserException クラスは Java オブジェクト呼び出し時の例外クラスです。詳細については，マニュアル『uCosminexus Service Coordinator Interactive Workflow AP 開発ガイド』の「CIWUserException（Java オブジェクト呼び出し時にエラーが発生した場合）」を参照してください。

注意事項

- このメソッド内では CSCIW の更新系 API を実行できません。
- execute メソッド内でトランザクションを開始した場合，必ずexecute メソッド内でトランザクションを決着してください。
- 次に示す場合は，アプリケーション呼び出しサービスによるリトライ対象となります。詳細については，「1.8.9　障害時の動作」を参照してください。
  - execute メソッドが例外をスローした
  - アプリケーション呼び出し情報ファイルの java.return.key.<key 要素値>に指定したkey 要素値が，戻り値のキーに含まれていない
  - 戻り値の Map の値が空文字
  - 戻り値の Map の値がプロセスデータの更新に使用できない型
  - 戻り値がnull

14 XPath 拡張関数リファレンス

BPMN 連携機能が提供する XPath 拡張関数について説明します。

14.1　XPath 拡張関数の概要

XPath 拡張関数は，BPMN ビジネスプロセス定義での排他ゲートウェイの評価式などに記述します。

BPMN 連携機能が提供する Xpath 拡張関数を次に示します。

表 14-1　提供する Xpath 拡張関数

項番	関数名	説明
1	csciw:list-size	リスト型プロセスデータの要素数を取得する。
2	csciw:list-contains	リスト型プロセスデータに指定した要素が存在するかを判定する。
3	csciw:exists	プロセスデータが存在するかを判定する。

14.1.1　csciw:list-size

構文

```
csciw:list-size ('processDataKey')
```

機能

指定されたプロセスデータキー名に従って，リスト型プロセスデータの要素数を取得します。

引数

csciw:list-size の引数を次の表に示します。

表 14-2　csciw:list-size の引数

項番	仮引数名	名称	I/O	説明
1	processDataKey	プロセスデータキー名	in	リスト型プロセスデータの全要素を示すプロセスデータキー名を文字列型（''囲み）で指定します。プロセスデータキー名には"$"と型を示す種別を付けた変数名を指定してください。 空文字列は指定できません。

戻り値

リスト型プロセスデータの要素数を数値型で返します。

リスト型プロセスデータのリスト内識別子に欠番がある場合，データベースに登録されているリスト型プロセスデータの要素数だけを取得します。

指定されたプロセスデータキー名のプロセスデータが存在しない場合は 0 を返します。

注意事項

- 指定されたプロセスデータキー名の書式が次のような場合は，例外が発生します。
  - プロセスデータキー名の命名規則に従っていない
  - プロセスデータキー名の末尾が"{}"ではない

関連項目

- 1.6.4　プロセスデータの利用方法

14.1.2　csciw:list-contains

構文

```
csciw:list-contains('processDataKey', procesDataValue)
```

機能

プロセスデータキー名とプロセスデータ値を指定し，プロセスデータキー名が示すリスト型プロセスデータにプロセスデータ値が含まれるかを判定します。

引数

csciw:list-contains の引数を次の表に示します。

表 14-3　csciw:list-contains の引数

項番	仮引数名	名称	I/O	説明
1	processDataKey	プロセスデータキー名	in	リスト型プロセスデータの全要素を示すプロセスデータキー名を文字列型（''囲み）で指定します。プロセスデータキー名には"$"と型を示す種別を付けた変数名を指定してください。 空文字列は指定できません。
2	procesDataValue	プロセスデータ値	in	プロセスデータ値を文字列型（''囲み）または数値型で指定します。

項番	仮引数名	名称	I/O	説明
2	procesDataValue	プロセスデータ値	in	空文字列は指定できません。

戻り値

リスト型プロセスデータにプロセスデータ値が含まれるかを boolean 型で返します。プロセスデータ値が含まれる場合は true を返し，含まれない場合は false を返します。

指定されたプロセスデータキー名のリスト型プロセスデータが 1 つも存在しない場合は false を返します。

注意事項

- 指定されたプロセスデータキー名の書式が次のような場合は，例外が発生します。
  - プロセスデータキー名の命名規則に従っていない
  - プロセスデータキー名の末尾が"{}"ではない
- プロセスデータ値に空文字列を指定した場合は，例外が発生します。
- 指定されたプロセスデータキー名とプロセスデータ値の型が一致しない場合は，型を一致させてからリスト型プロセスデータにプロセスデータ値が含まれるかを判定します。例えば，引数に次の値を指定した場合は，リスト型プロセスデータにプロセスデータ値"1"（数値型） が含まれるとき，true を返します。
  - プロセスデータキー名："$NKey{}"（数値型のプロセスデータキー名）
  - プロセスデータ値："1"（文字列型のプロセスデータ値）

  プロセスデータキー名とプロセスデータ値の型が一致しない場合，例外は発生しません。例えば，引数に次の値を指定した場合は，数値型のリスト型プロセスデータにプロセスデータ値"ABC"（文字列型）は存在しないため，false を返します。
  - プロセスデータキー名："$NKey{}"（数値型のプロセスデータキー名）
  - プロセスデータ値："ABC"（文字列型のプロセスデータ値）
- 指定したプロセスデータ値が数値型の場合の注意事項を次に示します。
  - プロセスデータ値が int 値の範囲外の場合，int 値に丸めた値になります。
  - プロセスデータ値が小数点以下を含む値の場合，小数点以下を切り捨てます。

14.1.3 csciw:exists

構文

```
csciw:exists('processDataKey')
```

機能

プロセスデータキー名を指定し，プロセスデータキー名が示すプロセスデータが存在するかを判定します。

引数

csciw:exists の引数を次の表に示します。

表 14-4　csciw:exists の引数

項番	仮引数名	名称	I/O	説明
1	processDataKey	プロセスデータキー名	in	プロセスデータキー名を文字列型（''囲み）で指定します。プロセスデータキー名には"$"と型を示す種別を付けた変数名を指定してください。 空文字列は指定できません。

戻り値

プロセスデータが存在するかを boolean 型で返します。プロセスデータが存在する場合は true を返し，存在しない場合は false を返します。

リスト型プロセスデータの全要素を示すキー名を指定した場合，指定したリスト型プロセスデータの要素が 1 件以上存在するときは true を返します。

注意事項

- 指定されたプロセスデータキー名の書式が次のような場合は，例外が発生します。
  - プロセスデータキー名の命名規則に従っていない

15 設定ファイル

BPMN 連携機能に関する設定ファイルの詳細について説明します。

15.1　設定するファイルの一覧

BPMN 連携機能を使用するときに設定するファイルの一覧を示します。

表 15-1　設定するファイルの一覧

ファイル	ファイル名	ファイルで設定する内容
共通設定ファイル	csciwbpmnconf.properties	BPMN 連携機能全体の動作
アプリケーション呼び出し情報ファイル	メッセージの場合 　<messageRefの値>.properties オペレーションの場合 　<operationRefの値>.properties エラーの場合 　<errorRefの値>.properties	アプリケーションの呼び出し先の情報および送受信するデータの情報
REST アプリケーション呼び出し用ヘッダファイル	任意	REST アプリケーション呼び出しを行う際のリクエストの HTTP ヘッダに指定する内容
コールアクティビティ情報ファイル	<calledElementの値>.properties	コールアクティビティから子案件を投入する際に使用する情報
REST アプリケーション呼び出しスキーマ変換用スタイルシート	任意	リクエストまたはレスポンスのメッセージボディを，異なる XML スキーマに変換する際に使用するスタイルシート
uCosminexus Business Process Developer 設定ファイル	ciwbpdev.properties	BPMN ビジネスプロセス定義ファイルの変換コマンド（ciwtransbpmn）のログに関する情報

関連項目

- 15.2　共通設定ファイル
- 15.3　アプリケーション呼び出し情報ファイル
- 15.4　REST アプリケーション呼び出し用ヘッダファイル
- 15.5　コールアクティビティ情報ファイル
- 15.6　REST アプリケーション呼び出しスキーマ変換用スタイルシート
- 15.7　uCosminexus Business Process Developer 設定ファイル

15.2　共通設定ファイル

BPMN 連携機能全体の動作を設定する共通設定ファイルの詳細について説明します。

15.2.1　共通設定ファイルの概要

共通設定ファイルとは，ファイルの格納先ディレクトリやメッセージの出力サイズなど，BPMN 連携機能全体の動作を設定するファイルを指します。

15.2.2　共通設定ファイルの設定個所

共通設定ファイルの設定個所について説明します。

共通設定ファイルは，次のパスに格納してください。

- CSCIW_CONF_DIR 環境変数が設定されていない場合

  Windows の場合

  ```
  %CSCIW_HOME%¥bpmn¥conf¥csciwbpmnconf.properties
  ```

  UNIX の場合

  ```
  ${CSCIW_HOME}/bpmn/conf/csciwbpmnconf.properties
  ```

- CSCIW_CONF_DIR 環境変数が設定されている場合

  共通設定ファイルの格納先を，対応するシステム ID のセットアッププロパティファイル（csciwsetup.properties）と同じ格納先にしてください。

  Windows の場合

  ```
  %CSCIW_CONF_DIR%￥csciwbpmnconf.properties
  ```

  UNIX の場合

  ```
  ${CSCIW_CONF_DIR}/csciwbpmnconf.properties
  ```

15.2.3　共通設定ファイルの読み込みタイミング

共通設定ファイルの読み込みタイミングについて説明します。

読み込みタイミングは，パラメタのカテゴリごとに異なります。各パラメタがどのカテゴリに属するかについては，「15.2.5　共通設定ファイルに指定する内容」を参照してください。

カテゴリごとの読み込みタイミングを次に示します。

- パラメタのカテゴリが「BPMN 連携機能共通」のとき

J2EE サーバを起動後，次のどれかの操作を初めてした時に読み込まれます（J2EE サーバを起動後，1回だけ読み込まれます）。

・BPMN 連携ライブラリの初期化（CIWBPMNLibAdmin クラスの initializeCIWBPMNLib メソッドの実行）

・REST サービスの開始

・アプリケーション呼び出しサービスの開始

- パラメタのカテゴリが「REST API」のとき

  REST サービスを開始した時に読み込まれます。

- パラメタのカテゴリが「アプリケーション呼び出しサービス」のとき

  アプリケーション呼び出しサービスを開始した時に読み込まれます。

一度読み込まれた共通設定ファイルのパラメタを変更する方法については，「9.2　ファイルに格納した設定情報を変更する」の共通設定ファイルの内容を変更した場合についての記述を参照してください。

15.2.4　共通設定ファイルの記述規則

共通設定ファイルの記述時の規則について説明します。

共通設定ファイルは，Java のプロパティファイルの仕様と同じです。値に「￥」を記述する場合，「￥」の前に「￥」をエスケープ文字として記述してください。また，ファイルのエンコーディングは UTF-8 とし，BOM を付けないで保存してください。

CSCIW 02-20 以前で使用していた共通設定ファイル（Unicode 形式のファイルを含む）を，そのまま使用することもできます。

15.2.5　共通設定ファイルに指定する内容

共通設定ファイルに指定する内容について説明します。

指定するパラメタの一覧（共通設定ファイル）

共通設定ファイルに指定するパラメタの一覧を示します。

表 15-2　指定するパラメタの一覧（共通設定ファイル）

項番	パラメタ名	カテゴリ	指定内容
1	BpmnLogFileDir	BPMN 連携機能 共通	ログ出力先ディレクトリ
2	BpmnDefinitionFileDir	REST API	BPMN ビジネスプロセス定義ファイル格納先ディレクトリ

項番	パラメタ名	カテゴリ	指定内容
3	BpmnCallInformationFileDir	BPMN 連携機能共通	アプリケーション呼び出し情報ファイルの格納先ディレクトリおよびコールアクティビティ情報ファイルの格納先ディレクトリ
4	AppCallServicePollingInterval	アプリケーション呼び出しサービス	アプリケーション呼び出しサービスのポーリング間隔
5	RestServiceResponseMaxCount	REST API	デフォルト最大取得数
6	RestServiceUserDescription	REST API	デフォルトユーザ記述子
7	RestServiceTransactionTimeout	REST API	REST サービスの API が呼び出される際のトランザクションタイムアウト値
8	RestServiceParamStatementCheck	REST API	REST API のフィルター条件とソート条件に指定された内容に，キーワードが含まれるかチェックするかどうかの値
9	RestServiceParamPreventStatements	REST API	REST API のフィルター条件とソート条件に指定された内容に，含まれているかチェックするキーワード
10	RestServiceMsgFileNum	REST API	REST サービスメッセージファイルの面数
11	RestServiceTraceFileNum	REST API	REST サービストレースファイルの面数
12	RestServiceMsgFileSize	REST API	REST サービスメッセージファイルの出力サイズ
13	RestServiceTraceFileSize	REST API	REST サービストレースファイルの出力サイズ
14	RestServiceMsgOutputThreashold	REST API	REST サービスメッセージ出力レベル
15	RestServiceTraceOutputThreashold	REST API	REST サービストレース出力レベル
16	RestServiceMsgTimeToDelete	REST API	REST サービスメッセージファイルを削除するまでの日数
17	RestServiceTraceTimeToDelete	REST API	REST サービストレースファイルを削除するまでの日数
18	AppCallServiceMsgFileNum	アプリケーション呼び出	アプリケーション呼び出しサービスメッセージファイルの面数

項番	パラメタ名	カテゴリ	指定内容
18	AppCallServiceMsgFileNum	しサービス	アプリケーション呼び出しサービスメッセージファイルの面数
19	AppCallServiceTraceFileNum	アプリケーション呼び出しサービス	アプリケーション呼び出しサービストレースファイルの面数
20	AppCallServiceMsgFileSize	アプリケーション呼び出しサービス	アプリケーション呼び出しサービスメッセージファイルの出力サイズ
21	AppCallServiceTraceFileSize	アプリケーション呼び出しサービス	アプリケーション呼び出しサービストレースファイルの出力サイズ
22	AppCallServiceMsgOutputThreashold	アプリケーション呼び出しサービス	アプリケーション呼び出しサービスメッセージ出力レベル
23	AppCallServiceTraceOutputThreashold	アプリケーション呼び出しサービス	アプリケーション呼び出しサービストレース出力レベル
24	AppCallServiceMsgTimeToDelete	アプリケーション呼び出しサービス	アプリケーション呼び出しサービスメッセージファイルを削除するまでの日数
25	AppCallServiceTraceTimeToDelete	アプリケーション呼び出しサービス	アプリケーション呼び出しサービストレースファイルを削除するまでの日数
26	AppCallServiceTransactionTimeout	アプリケーション呼び出しサービス	アプリケーション呼び出しサービスのトランザクションタイムアウト値

項番	パラメタ名	カテゴリ	指定内容
27	AppCallServiceDebugRestBody	アプリケーション呼び出しサービス	アプリケーション呼び出しサービスと REST アプリケーション間で送受信するボディデータのトレースファイル出力の有無
28	AppCallServiceInitialInterval	アプリケーション呼び出しサービス	アプリケーション呼び出しサービスの初回ポーリング間隔
29	TerminateCompatMode	BPMN 連携機能共通	強制終了イベントに遷移した際の動作について，CSCIW のバージョン 03-10 までとの互換性の有無
30	AppCallServiceSendID	アプリケーション呼び出しサービス	アプリケーション呼び出しサービスが REST アプリケーションに HTTP ヘッダとして呼び出し元の案件 ID と作業 ID を送信するかどうか

(1) BpmnLogFileDir

共通設定ファイルに指定するBpmnLogFileDir パラメタについて，説明します。

ユーザ指定値

値の範囲	パラメタの指定の省略	指定を省略したときの仮定値
-	可	＜ Windows の場合＞ %CSCIW_HOME%¥log ＜ UNIX の場合＞ ${CSCIW_HOME}/log

（凡例）：

-：指定できる値の範囲に，制限はありません。

説明

次のログファイルの出力先ディレクトリを指定します。

- REST サービストレースファイル
- REST サービスメッセージファイル
- アプリケーション呼び出しサービストレースファイル
- アプリケーション呼び出しサービスメッセージファイル

(2) BpmnDefinitionFileDir

共通設定ファイルに指定するBpmnDefinitionFileDir パラメタについて，説明します。

ユーザ指定値

値の範囲	パラメタの指定の省略	指定を省略したときの仮定値
-	可	＜ Windows の場合＞ %CSCIW_HOME%¥bpmn¥bpmnxml ＜ UNIX の場合＞ ${CSCIW_HOME}/bpmn/bpmnxml

（凡例）：

-：指定できる値の範囲に，制限はありません。

説明

BPMN ビジネスプロセス定義ファイルを格納するディレクトリを指定します。

(3) BpmnCallInformationFileDir

共通設定ファイルに指定するBpmnCallInformationFileDir パラメタについて，説明します。

ユーザ指定値

値の範囲	パラメタの指定の省略	指定を省略したときの仮定値
-	可	＜ Windows の場合＞ %CSCIW_HOME%¥bpmn¥callinfo ＜ UNIX の場合＞ ${CSCIW_HOME}/bpmn/callinfo

（凡例）：

-：指定できる値の範囲に，制限はありません。

説明

アプリケーション呼び出しサービスが呼び出し処理を実行する際に読み込む「アプリケーション呼び出し情報ファイル」と，BPMN 連携機能がコールアクティビティに関して処理する際に読み込む「コールアクティビティ情報ファイル」の格納先ディレクトリを指定します。

アプリケーション呼び出しサービスは，このパラメタで設定されたディレクトリ配下のmsg フォルダ，ope フォルダ，sig フォルダ，およびerr フォルダに格納された各アプリケーション呼び出し情報ファイルを読み込みます。また，BPMN 連携機能は，設定されたディレクトリ配下のcall フォルダに格納されたコー

ルアクティビティ情報ファイルを読み込みます。そのため，このパラメタで指定したディレクトリ直下に，次のフォルダがすべて存在している必要があります。

- call
- err
- msg
- ope
- sig

(4) AppCallServicePollingInterval

共通設定ファイルに指定するAppCallServicePollingIntervalパラメタについて，説明します。

ユーザ指定値

値の範囲	パラメタの指定の省略	指定を省略したときの仮定値
1～86,400の整数	可	60

説明

アプリケーション呼び出しサービスのポーリング間隔を秒単位で指定します。

このパラメタで指定した秒数ごとに，アプリケーション呼び出しサービスのプロセスを実行します。

このパラメタに短い秒数を指定した場合，データベースの検索が頻繁に発生するため，60以上の値を指定することを推奨します。

(5) RestServiceResponseMaxCount

共通設定ファイルに指定するRestServiceResponseMaxCountパラメタについて，説明します。

ユーザ指定値

値の範囲	パラメタの指定の省略	指定を省略したときの仮定値
-1～2,147,483,647の整数	可	100

説明

デフォルト最大取得数を指定します。

RESTサービスの一覧取得APIが呼び出される際に最大取得数が指定されていない場合，このパラメタで指定した値がデフォルト値として使用されます。

-1を指定した場合，すべてのデータを取得します。

(6) RestServiceUserDescription

共通設定ファイルに指定するRestServiceUserDescriptionパラメタについて，説明します。

ユーザ指定値

値の範囲	パラメタの指定の省略	指定を省略したときの仮定値
1～32バイトの文字列	可	csciwws

説明

デフォルトユーザ記述子を指定します。

RESTサービスのAPIが呼び出される際にユーザ記述子が指定されていない場合，このパラメタで指定した値がデフォルト値として使用されます。

(7) RestServiceTransactionTimeout

共通設定ファイルに指定するRestServiceTransactionTimeoutパラメタについて，説明します。

ユーザ指定値

値の範囲	パラメタの指定の省略	指定を省略したときの仮定値
0～2,147,483,647の整数	可	0

説明

RESTサービスのAPIが呼び出される際のトランザクションタイムアウト値を秒単位で指定します。

0を指定した場合，J2EEサーバの設定値（J2EEサーバ用ユーザプロパティファイルusrconf.propertiesのejbserver.jta.TransactionManager.defaultTimeOutの設定値）が指定されたと見なされます。

(8) RestServiceParamStatementCheck

共通設定ファイルに指定するRestServiceParamStatementCheckパラメタについて，説明します。

ユーザ指定値

値の範囲	パラメタの指定の省略	指定を省略したときの仮定値
trueまたはfalse	可	false

説明

REST APIのフィルター条件とソート条件に指定された内容に，「RestServiceParamPreventStatements」で指定したキーワードが含まれているか，チェックするかどうかを指定します。

true を指定した場合，キーワードが含まれているかチェックします。フィルター条件またはソート条件にキーワードが含まれていると，エラーになります。

false を指定した場合，キーワードのチェックはしません。

共通設定ファイルの初期値には，true が設定されています。

(9) RestServiceParamPreventStatements

共通設定ファイルに指定するRestServiceParamPreventStatements パラメタについて，説明します。

ユーザ指定値

値の範囲	パラメタの指定の省略	指定を省略したときの仮定値
半角英数字，アンダースコア，およびコンマ	可	空文字

説明

「RestServiceParamStatementCheck」にtrue が指定されている場合，フィルター条件とソート条件に指定された内容に含まれているかチェックするキーワードを指定します。

キーワードを複数指定する場合はコンマ区切りで指定してください。大文字と小文字は区別しません。

共通設定ファイルの初期値には，SELECT が設定されています。

(10) RestServiceMsgFileNum

共通設定ファイルに指定するRestServiceMsgFileNum パラメタについて，説明します。

ユーザ指定値

値の範囲	パラメタの指定の省略	指定を省略したときの仮定値
1～16 の整数	可	2

説明

REST サービスメッセージファイルの面数を指定します。

ラップアラウンドモードの場合，出力面数を指定してください。シフトモードの場合，バックアップ面数を指定してください。

(11) RestServiceTraceFileNum

共通設定ファイルに指定するRestServiceTraceFileNum パラメタについて，説明します。

ユーザ指定値

値の範囲	パラメタの指定の省略	指定を省略したときの仮定値
1～16 の整数	可	4

説明

REST サービストレースファイルの面数を指定します。

ラップアラウンドモードの場合，出力面数を指定してください。シフトモードの場合，バックアップ面数を指定してください。

(12) RestServiceMsgFileSize

共通設定ファイルに指定するRestServiceMsgFileSize パラメタについて，説明します。

ユーザ指定値

値の範囲	パラメタの指定の省略	指定を省略したときの仮定値
100,000～2,147,483,647 の整数	可	2097152

説明

REST サービスメッセージファイルの出力サイズをバイト単位で指定します。

(13) RestServiceTraceFileSize

共通設定ファイルに指定するRestServiceTraceFileSize パラメタについて，説明します。

ユーザ指定値

値の範囲	パラメタの指定の省略	指定を省略したときの仮定値
100,000～2,147,483,647 の整数	可	16777216

説明

REST サービストレースファイルの出力サイズをバイト単位で指定します。

(14) RestServiceMsgOutputThreashold

共通設定ファイルに指定するRestServiceMsgOutputThreashold パラメタについて，説明します。

ユーザ指定値

値の範囲	パラメタの指定の省略	指定を省略したときの仮定値
-1～1,000 の整数	可	20

説明

REST サービスメッセージ出力レベルを指定します。

0 以上の値を指定すると，指定した値以下のレベルの REST サービスメッセージが出力されます。

-1 を指定した場合，REST サービスメッセージは出力されません。

(15) RestServiceTraceOutputThreashold

共通設定ファイルに指定するRestServiceTraceOutputThreashold パラメタについて，説明します。

ユーザ指定値

値の範囲	パラメタの指定の省略	指定を省略したときの仮定値
-1～1,000 の整数	可	20

説明

REST サービストレース出力レベルを指定します。

0 以上の値を指定すると，指定した値以下のレベルの REST サービストレースが出力されます。

-1 を指定した場合，REST サービストレースは出力されません。

(16) RestServiceMsgTimeToDelete

共通設定ファイルに指定するRestServiceMsgTimeToDelete パラメタについて，説明します。

ユーザ指定値

値の範囲	パラメタの指定の省略	指定を省略したときの仮定値
-1～24,855 の整数	可	-1

説明

REST サービスメッセージファイルを削除するまでの日数を指定します。

このパラメタは，ローテーション種別がラップアラウンドモードの場合だけ有効です。

0 を指定した場合，出力された REST サービスメッセージファイルはすぐに削除されます。

-1 を指定した場合，日数が経過してもファイルは削除されません。

(17) RestServiceTraceTimeToDelete

共通設定ファイルに指定するRestServiceTraceTimeToDelete パラメタについて，説明します。

ユーザ指定値

値の範囲	パラメタの指定の省略	指定を省略したときの仮定値
-1～24,855 の整数	可	-1

説明

REST サービストレースファイルを削除するまでの日数を指定します。

このパラメタは，ローテーション種別がラップアラウンドモードの場合だけ有効です。

0 を指定した場合，出力された REST サービストレースファイルはすぐに削除されます。

-1 を指定した場合，日数が経過してもファイルは削除されません。

(18) AppCallServiceMsgFileNum

共通設定ファイルに指定するAppCallServiceMsgFileNum パラメタについて，説明します。

ユーザ指定値

値の範囲	パラメタの指定の省略	指定を省略したときの仮定値
1～16 の整数	可	2

説明

アプリケーション呼び出しサービスメッセージファイルの面数を指定します。

ラップアラウンドモードの場合，出力面数を指定してください。シフトモードの場合，バックアップ面数を指定してください。

(19) AppCallServiceTraceFileNum

共通設定ファイルに指定するAppCallServiceTraceFileNum パラメタについて，説明します。

ユーザ指定値

値の範囲	パラメタの指定の省略	指定を省略したときの仮定値
1～16 の整数	可	4

説明

アプリケーション呼び出しサービストレースファイルの面数を指定します。

ラップアラウンドモードの場合，出力面数を指定してください。シフトモードの場合，バックアップ面数を指定してください。

(20) AppCallServiceMsgFileSize

共通設定ファイルに指定するAppCallServiceMsgFileSizeパラメタについて，説明します。

ユーザ指定値

値の範囲	パラメタの指定の省略	指定を省略したときの仮定値
100,000～2,147,483,647の整数	可	2097152

説明

アプリケーション呼び出しサービスメッセージファイルの出力サイズをバイト単位で指定します。

(21) AppCallServiceTraceFileSize

共通設定ファイルに指定するAppCallServiceTraceFileSizeパラメタについて，説明します。

ユーザ指定値

値の範囲	パラメタの指定の省略	指定を省略したときの仮定値
100,000～2,147,483,647の整数	可	16777216

説明

アプリケーション呼び出しサービストレースファイルの出力サイズをバイト単位で指定します。

(22) AppCallServiceMsgOutputThreashold

共通設定ファイルに指定するAppCallServiceMsgOutputThreasholdパラメタについて，説明します。

ユーザ指定値

値の範囲	パラメタの指定の省略	指定を省略したときの仮定値
-1～1,000の整数	可	20

説明

アプリケーション呼び出しサービスメッセージ出力レベルを指定します。

0 以上の値を指定すると，指定した値以下のレベルのアプリケーション呼び出しサービスメッセージが出力されます。

-1 を指定した場合，アプリケーション呼び出しサービスメッセージは出力されません。

(23) AppCallServiceTraceOutputThreashold

共通設定ファイルに指定するAppCallServiceTraceOutputThreashold パラメタについて，説明します。

ユーザ指定値

値の範囲	パラメタの指定の省略	指定を省略したときの仮定値
-1～1,000 の整数	可	20

説明

アプリケーション呼び出しサービストレース出力レベルを指定します。

0 以上の値を指定すると，指定した値以下のレベルのアプリケーション呼び出しサービストレースが出力されます。

-1 を指定した場合，アプリケーション呼び出しサービストレースは出力されません。

(24) AppCallServiceMsgTimeToDelete

共通設定ファイルに指定するAppCallServiceMsgTimeToDelete パラメタについて，説明します。

ユーザ指定値

値の範囲	パラメタの指定の省略	指定を省略したときの仮定値
-1～24,855 の整数	可	-1

説明

アプリケーション呼び出しサービスメッセージファイルを削除するまでの日数を指定します。

このパラメタは，ローテーション種別がラップアラウンドモードの場合だけ有効です。

0 を指定した場合，出力されたアプリケーション呼び出しサービスメッセージファイルはすぐに削除されます。

-1 を指定した場合，日数が経過してもファイルは削除されません。

(25) AppCallServiceTraceTimeToDelete

共通設定ファイルに指定するAppCallServiceTraceTimeToDelete パラメタについて，説明します。

ユーザ指定値

値の範囲	パラメタの指定の省略	指定を省略したときの仮定値
-1～24,855 の整数	可	-1

説明

アプリケーション呼び出しサービストレースファイルを削除するまでの日数を指定します。

このパラメタは，ローテーション種別がラップアラウンドモードの場合だけ有効です。

0 を指定した場合，出力されたアプリケーション呼び出しサービストレースファイルはすぐに削除されます。

-1 を指定した場合，日数が経過してもファイルは削除されません。

(26) AppCallServiceTransactionTimeout

共通設定ファイルに指定するAppCallServiceTransactionTimeout パラメタについて，説明します。

ユーザ指定値

値の範囲	パラメタの指定の省略	指定を省略したときの仮定値
0～2,147,483,647 の整数	可	0

説明

アプリケーション呼び出しサービスのトランザクションタイムアウト値を秒単位で指定します。

0 を指定した場合，J2EE サーバの設定値（J2EE サーバ用ユーザプロパティファイルusrconf.properties のejbserver.jta.TransactionManager.defaultTimeOut の設定値）が指定されたと見なされます。

1 以上の値を指定した場合，トランザクションタイムアウト値（秒）が指定されたと見なされます。

(27) AppCallServiceDebugRestBody

共通設定ファイルに指定するAppCallServiceDebugRestBody パラメタについて，説明します。

ユーザ指定値

値の範囲	パラメタの指定の省略	指定を省略したときの仮定値
true またはfalse	可	false

説明

アプリケーション呼び出しサービスと REST アプリケーションの間で送受信するボディデータ（リクエストボディとレスポンスボディ）の内容を，トレースファイルに出力するかどうかを指定します。

true を指定した場合，ボディデータの内容を XML 形式でトレースファイルに出力します。XSLT スタイルシートを使用してボディデータをスキーマ変換した場合は，変換前と変換後の両方の内容を出力します。なお，true を指定すると，アプリケーション呼び出しのスループットが低下するおそれがあります。

false を指定した場合，ボディデータの内容をトレースファイルに出力しません。

(28) AppCallServiceInitialInterval

共通設定ファイルに指定するAppCallServiceInitialInterval パラメタについて，説明します。

ユーザ指定値

値の範囲	パラメタの指定の省略	指定を省略したときの仮定値
0～86,400 の整数	可	0

説明

アプリケーション呼び出しサービスの開始後，最初のポーリングを行うまでの時間を秒単位で指定します。指定された秒数の経過後，アプリケーション呼び出しを定期的に実行します。

このパラメタは，アプリケーション呼び出しサービスの開始後，一定時間経過後にアプリケーション呼び出しを実行したい場合だけ指定してください。

(29) TerminateCompatMode

共通設定ファイルに指定するTerminateCompatMode パラメタについて，説明します。

ユーザ指定値

値の範囲	パラメタの指定の省略	指定を省略したときの仮定値
true またはfalse	可	true

説明

強制終了イベントに遷移した際の動作を CSCIW のバージョン 03-10 までと互換性を持たせるかどうかを指定します。

true（03-10 までと同じ動作）を指定した場合，サブプロセス内およびイベント・サブプロセス内の強制終了イベントに遷移した際に，案件自体を強制終了します。

false を指定した場合，サブプロセス内およびイベント・サブプロセス内の強制終了イベントに遷移した際に，サブプロセス内だけを強制終了します。

(30) AppCallServiceSendID

共通設定ファイルに指定するAppCallServiceSendID パラメタについて，説明します。

ユーザ指定値

値の範囲	パラメタの指定の省略	指定を省略したときの仮定値
true またはfalse	可	false

説明

アプリケーション呼び出しサービスが REST アプリケーションに HTTP ヘッダとして呼び出し元の案件 ID と作業 ID を送信するかどうかを指定します。

true を指定した場合，呼び出し元の案件 ID と作業 ID を送信します。次のキーを HTTP ヘッダとして送信します。

- x-hitachi-csciw-piid
- x-hitachi-csciw-wiid

false を指定した場合，呼び出し元の案件 ID と作業 ID を送信しません。

15.3　アプリケーション呼び出し情報ファイル

アプリケーション呼び出しサービスの動作を設定するアプリケーション呼び出し情報ファイルの詳細について説明します。

15.3.1　アプリケーション呼び出し情報ファイルの概要

アプリケーション呼び出し情報ファイルには，アプリケーション呼び出しサービスの呼び出し先の情報や，呼び出す際に送受信するデータの情報を，ref 識別子ごとに設定します。

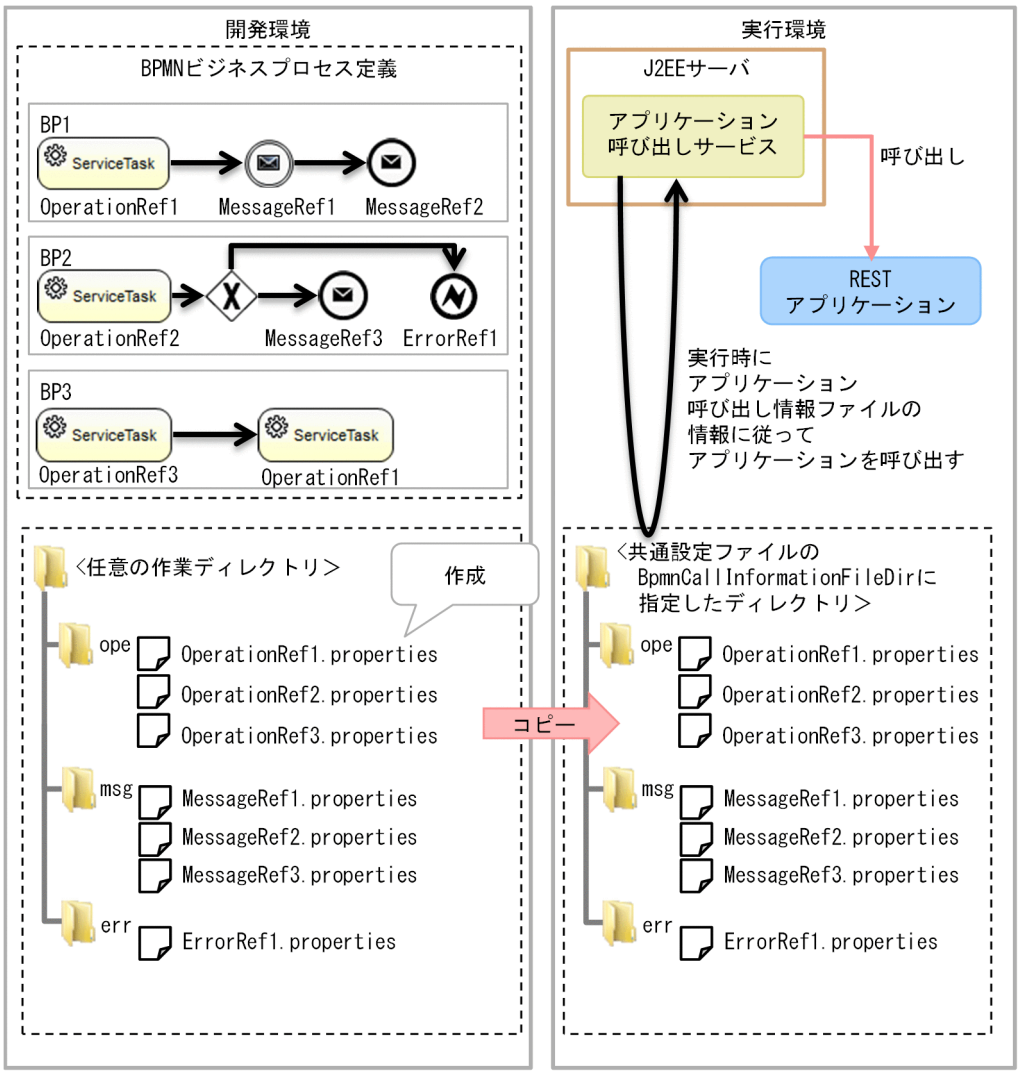
15.3.2　アプリケーション呼び出し情報ファイルの設定個所

アプリケーション呼び出し情報ファイルは，ref 識別子単位で作成し，共通設定ファイルで指定したディレクトリ内のサブディレクトリ下にそれぞれ格納します。

アプリケーション呼び出し情報ファイルは，BPMN ビジネスプロセス定義を作成する開発環境で作成します。開発環境で作成したアプリケーション呼び出し情報ファイルは，実行環境にコピーしてください。

アプリケーション呼び出し情報ファイルの配置例を次の図に示します。

図 15-1　アプリケーション呼び出し情報ファイルの配置例



なお，アプリケーション呼び出し情報ファイルは，BPMN 要素に定義する ref 識別子単位に作成します。各アプリケーション呼び出し情報ファイルには，呼び出し対象の BPMN 要素に対応するファイル名を付けてください。ただし，タイマーイベントはアプリケーション呼び出し情報ファイルを使用しないため，作成する必要はありません。

作成したアプリケーション呼び出し情報ファイルの格納先として，共通設定ファイルの BpmnCallInformationFileDir プロパティで指定したディレクトリ内に，サブディレクトリ（ope，msg，err）を作成します。

次に示す表に従って，アプリケーション呼び出し情報ファイルのファイル名を付けて，該当するサブディレクトリに各ファイルを格納してください。

表 15-3　呼び出し対象の BPMN 要素およびアプリケーション呼び出し情報ファイルのファイル名（ファイルパス）

項番	呼び出し対象の BPMN 要素	アプリケーション呼び出し情報ファイルのファイル名（ファイルパス）
1	サービスタスク	<BpmnCallInformationFileDirの指定値>/ope/<operationRefの値>.properties
2	ビジネスルールタスク	
3	メッセージイベント	<BpmnCallInformationFileDirの指定値>/msg/<messageRefの値>.properties
4	エラーイベント	<BpmnCallInformationFileDirの指定値>/err/<errorRefの値>.properties

ヒント

呼び出し対象の BPMN 要素がエラーイベントの場合，アプリケーション呼び出し情報ファイルの作成を省略できます。アプリケーション呼び出し情報ファイルの作成を省略した場合，すべてのプロパティを省略した場合の設定で，アプリケーション呼び出しが行われます。

呼び出し対象の BPMN 要素がエラーイベント以外の場合は，アプリケーション呼び出し情報ファイルを必ず作成してください。アプリケーション呼び出し情報ファイルが存在しない場合，エラーが発生します（アプリケーション呼び出しは中止します）。

重要

- 「図 15-1　アプリケーション呼び出し情報ファイルの配置例」のOperationRef1 のように，複数の BPMN ビジネスプロセス定義（BP1，BP3）で同じ ref 識別子を定義している場合，共通で使用するOperationRef1.properties ファイルをope フォルダの下に配置してください。
- 大文字と小文字だけが異なる同じ名称の ref 識別子は使用しないでください。

15.3.3　アプリケーション呼び出し情報ファイルの記述規則

アプリケーション呼び出し情報ファイルの記述規則を次に示します。

- java.util.Properties クラスが扱えるフォーマットで記載してください。
- 文字コードは，UTF-8 を使用してください。ただし，BOM 付き UTF-8 では記述できません。
- プロパティの各キーは，大文字と小文字が区別されます。
- プロパティの各キーは，1 つだけ指定してください（同一のキーは複数指定できません）。同一のキーを複数指定した場合，どの設定値が有効になるかは不定です。
- プロパティの記載順は任意です。また，プロパティが読み込まれる順番は不定です。
- プロパティのキーに，アプリケーション呼び出し情報ファイルで使用できるプロパティで定義されていないキー名を指定した場合，アプリケーションの呼び出し時にエラーが発生します。

- プロパティでファイルパスを指定する場合，パスの区切り文字には"/"を使用してください。
- プロパティの値の先頭に"$$"を記載した場合，"$"に変換されます。
- プロパティの値の先頭に"@@""を記載した場合，"@"に変換されます。
- プロパティの値が空の場合，エラーが発生します。

アプリケーション呼び出し情報ファイルが次に示すケースに該当する場合，アプリケーション呼び出しサービスは，該当する ref 識別子のアプリケーション呼び出しを中止します。

- 指定必須のプロパティのキーが指定されていない
- プロパティに不正な値が指定されている

関連項目

- 15.3.14　REST アプリケーションの呼び出しの場合にファイルに指定する内容
- 15.3.15　Java オブジェクトの呼び出しの場合にファイルに指定する内容
- 15.3.16　メッセージによる案件投入の場合にファイルに指定する内容
- 15.3.17　メッセージによる他案件の呼び出しの場合にファイルに指定する内容
- 15.3.18　メッセージによる自案件の呼び出しの場合にファイルに指定する内容
- 15.3.19　エラーによる自案件の呼び出しの場合にファイルに指定する内容

15.3.4　プロセスデータの設定方法（アプリケーション呼び出し情報ファイル）

アプリケーション呼び出し情報ファイルにプロセスデータキー名を記述することで，呼び出し先にプロセスデータを渡したり，呼び出し先からプロセスデータを受け取ったりすることができます。

プロセスデータの設定時の動作

プロセスデータキー名をプロパティの値として指定した場合，アプリケーション呼び出しサービスによって，プロセスデータキー名は，対応するプロセスデータ値に置き換えられます。

プロセスデータキー名をプロパティのキーとして指定した場合，アプリケーション呼び出しサービスによって，該当するプロセスデータが更新されます。

プロセスデータの使用例

アプリケーション呼び出し情報ファイルにプロセスデータキー名を記述して，プロセスデータを更新する例を示します。

この例では，REST アプリケーション呼び出しで呼び出す REST アプリケーションのレスポンスのボディの内容を，プロセスデータテーブルに格納するように，アプリケーション呼び出し情報ファイルを設定しています。

アプリケーション呼び出し情報ファイルの設定例（一部抜粋）

```
rest.response.pi.$SReplyUser=rest.response.body.key.userdescription
```

REST アプリケーションのレスポンスのボディの返却例（一部抜粋）

```
<data>
  <key>userdescription</key>
  <value>User001</value>
</data>
```

上記の場合，アプリケーション呼び出し情報ファイルの内容に従って，レスポンスのkey 要素の値がuserdescription であるvalue 要素の値が，プロセスデータReplyUser に設定されます。

プロセスデータの使用規則

アプリケーション呼び出し情報ファイルでプロセスデータを使用する場合は，次の規則に従ってください。

- プロセスデータキー名の前後に，プロセスデータキー名，組み込み変数，その他の文字列を連続して記載することはできません。また，半角空白などで文字列を区切ることもできません。

  誤った記載例（通常の文字列として扱われる）

  ```
  abc$Skey1
  ```

  誤った記載例（エラーが発生）

  ```
  $Skey1$Skey2
  ```

  ただし，rest.request.url プロパティの値の中では，特定の文字で区切ることで，プロセスデータキー名の前後に組み込み変数や文字列を連続して記載できます。
- プロパティの値の先頭に"$$"を記載した場合，プロセスデータではなく通常の文字列として扱われます。
- 1 つのアプリケーション呼び出し情報ファイル内に，プロセスデータは 255 個まで指定できます。
- 指定したプロセスデータキー名に対応したプロセスデータがプロセスデータテーブルに存在しない場合，アプリケーション呼び出しの実行時にエラーが発生します。
- プロパティのキーに指定した呼び出し元案件のプロセスデータキー名は，プロパティの値に記載できません。記載した場合，アプリケーション呼び出しの実行時にエラーが発生します。

  誤った記載例（エラーが発生）

  ```
  rest.request.body.key.offset=$Skey1
  ```

  ```
  rest.response.pi.$Skey1=rest.response.body.key.result
  ```
- プロセスデータの値がnull の場合，null 値として扱われます。ただし，rest.request.url プロパティの値の中にnull が含まれている場合は，空文字として扱われます。
- リスト型プロセスデータ間の代入をした場合（プロパティキーとプロパティ値の両方にリスト型プロセスデータの全要素を示すプロセスデータキー名を指定した場合），更新対象（プロパティキー側）のリスト型プロセスデータのリスト内識別子は，順番はそのままで 1 から連番で付与されます。参照元（プ

ロパティ値側）のリスト内識別子が連番ではない場合，更新対象（プロパティキー側）のリスト内識別子は，参照元（プロパティ値側）のリスト内識別子と一致しません。

ヒント

rest.request.url プロパティについては，「15.3.14　REST アプリケーションの呼び出しの場合にファイルに指定する内容」の「(3)　rest.request.url」を参照してください。

関連項目

- 1.6　プロセスデータとは

15.3.5　アプリケーション呼び出し情報ファイルで使用できる組み込み変数

組み込み変数は，CSCIW で管理する案件や作業の属性値を指定するための変数です。アプリケーション呼び出し情報ファイルに記載した組み込み変数は，呼び出し元の案件や作業の属性値に変換されます。

使用できる組み込み変数

項番	組み込み変数	説明
1	@PIID	案件 ID
2	@PIName	案件名
3	@PICreator	案件投入者
4	@PIDeadline	案件処理期限
5	@PDefName	ビジネスプロセス定義名
6	@WIID	作業 ID
7	@WDefName	作業定義名
8	@NULL	null 値

組み込み変数の使用規則

アプリケーション呼び出し情報ファイルで組み込み変数を使用する場合は，次の規則に従ってください。

- 組み込み変数名の前後に，ほかの組み込み変数名やその他の文字列を連続して記載することはできません。また，半角空白などで文字列を区切ることもできません。

  誤った記載例（通常の文字列として扱われる）

  ```
  abc@PIName
  ```

  誤った記載例（エラーが発生）

  ```
  @PIName@PIID
  ```

ただし，rest.request.url プロパティの値の中では，特定の文字で区切ることで，組み込み変数の前後にほかの組み込み変数や文字列を連続して記載できます。

- プロパティの値の先頭に"@"を記載し，上記の表「使用できる組み込み変数」の「組み込み変数」列以外の文字列を指定した場合は，アプリケーションの呼び出し時にエラーが発生します。

  誤った記載例

  - @ABC
  - @PINameABC

- プロパティの値の先頭に"@@"を記載した場合は，組み込み変数ではなく通常の文字列として扱われます。
- 組み込み変数の対象となる属性の値が null の場合，null 値として扱われます。ただし，rest.request.url プロパティの値の中に含まれている場合は，空文字として扱われます。
- 案件処理期限は，"1970/01/01 00:00:00 GMT"を起点とした通算秒に置き換えられます。代入先には，文字列型プロセスデータを指定してください。数値型プロセスデータを指定した場合，案件処理期限に"2038/01/19 03:14:07 GMT"より先の日時を指定するとエラーになります。

ヒント

rest.request.url プロパティについては，「15.3.14　REST アプリケーションの呼び出しの場合にファイルに指定する内容」の「(3)　rest.request.url」を参照してください。

15.3.6　HTTP ヘッダを記述したファイルの指定方法（アプリケーション呼び出し情報ファイル）

HTTP ヘッダを記述したファイルとは，REST アプリケーションを呼び出す際のリクエストの HTTP ヘッダに指定する内容を記述するファイルのことです。HTTP ヘッダを記述したファイルを作成した場合は，HTTP ヘッダを記述したファイルのファイルパスを，アプリケーション呼び出し情報ファイルの rest.request.header.filepath プロパティに指定してください。

ヒント

rest.request.header.filepath プロパティについては，「15.3.14　REST アプリケーションの呼び出しの場合にファイルに指定する内容」の「(4)　rest.request.header.filepath」を参照してください。

HTTP ヘッダを記述したファイルの指定規則を次に示します。

- java.util.Properties クラスが扱えるフォーマットで記載してください。
- 文字コードは UTF-8 を使用してください。ただし，BOM 付き UTF-8 では記述しないでください。

- ファイル中に記載する要素の数や種類は，HTTP ヘッダの仕様に合わせてください。複数回指定できない要素を 2 行以上指定しているなど，HTTP ヘッダの仕様と異なる記述をした場合の動作は保証しません。

HTTP ヘッダの記述例を次に示します。

HTTP ヘッダの記述例

```
Accept=application/xml
```

15.3.7　アプリケーション呼び出し情報ファイルの読み込みタイミング

アプリケーション呼び出し情報ファイルは，対応する ref 識別子のアプリケーション呼び出しが最初に実行されるタイミングで，読み込まれます。一度読み込まれたファイルの内容は，アプリケーション呼び出しサービス内で保持されます。

また，アプリケーション呼び出し情報の一部として読み込まれた次のファイルについても同様に，一度読み込まれたファイルの内容は，アプリケーション呼び出しサービス内で保持されます。

- REST アプリケーション用の HTTP ヘッダファイル
- REST アプリケーション用のスキーマ変換に使用されるスタイルシート（リクエストボディ用およびレスポンスボディ用）

ファイルの再読み込みについては，「9.2　ファイルに格納した設定情報を変更する」の，アプリケーション呼び出し情報ファイル，REST アプリケーション呼び出し用ヘッダファイル，または REST アプリケーション呼び出しスキーマ変換用スタイルシートの内容を変更した場合についての説明を参照してください。

15.3.8　アプリケーション呼び出し情報ファイルとリクエストボディの関係

アプリケーション呼び出し情報ファイルに記述した内容に従って，対応するプロセスデータテーブルのプロセスデータ値がリクエストボディに渡されます。リクエストボディは，REST アプリケーションに渡されます。

単一型プロセスデータがリクエストボディに渡される場合

単一型プロセスデータがリクエストボディに渡される場合の，アプリケーション呼び出し情報ファイルの記述例を次に示します。

■アプリケーション呼び出し情報ファイルの記述例

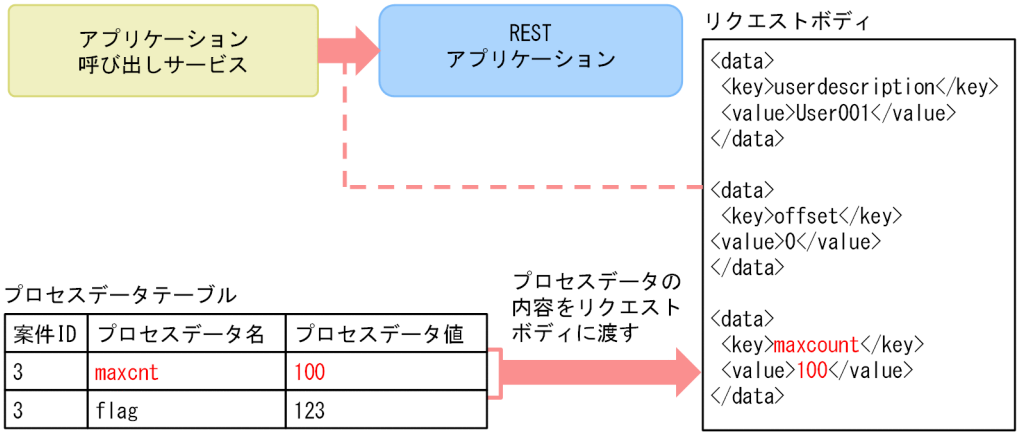
```
rest.request.body.key.maxcount=$Nmaxcnt
rest.request.body.key.userdescription=User001
rest.request.body.key.offset=0
```

[説明]

1 行目で，rest.request.body.key.maxcount プロパティの値として，プロセスデータキー名"$Nmaxcnt"を指定しています。

アプリケーション呼び出し情報ファイルを上記の例のとおり記述した場合の，REST アプリケーションにリクエストボディが渡されるまでの流れを，次の図に示します。

図 15-2　REST アプリケーションにリクエストボディが渡されるまでの流れ（単一型プロセスデータがリクエストボディに渡される場合）



[説明]

アプリケーション呼び出し情報ファイルの内容に従って，key 要素の値が"maxcount"，value 要素の値が"100"のデータが，プロセスデータテーブルからリクエストボディに渡されます。プロセスデータテーブルからデータを受け取ったリクエストボディは，REST アプリケーションに渡されます。

リスト型プロセスデータの全要素がリクエストボディに渡される場合

リスト型プロセスデータの全要素がリクエストボディに渡される場合の，アプリケーション呼び出し情報ファイルの記述例を次に示します。

■アプリケーション呼び出し情報ファイルの記述例

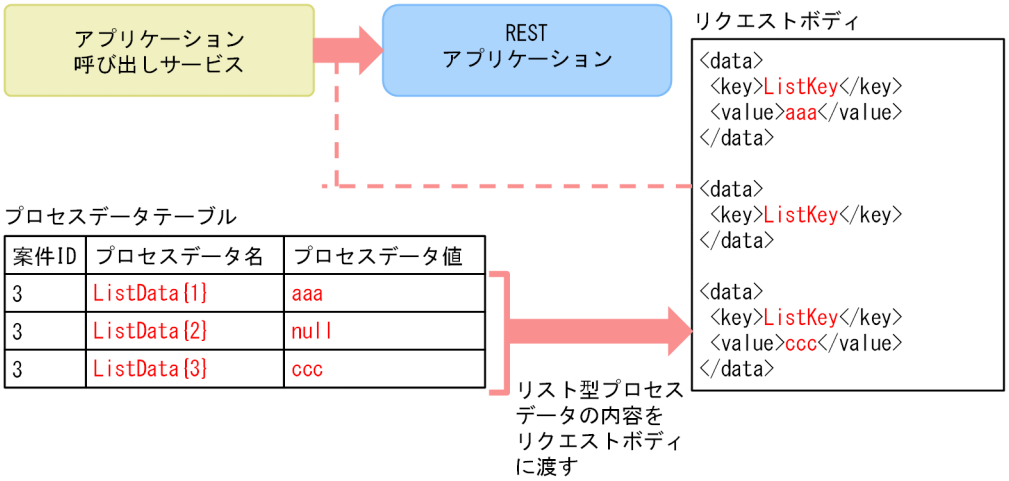
```
rest.request.body.key.ListKey=$SListData{}
```

[説明]

rest.request.body.key.ListKey プロパティの値として，プロセスデータキー名"$SListData{}"を指定しています。

アプリケーション呼び出し情報ファイルを上記の例のとおり記述した場合の，REST アプリケーションにリクエストボディが渡されるまでの流れを，次の図に示します。

図 15-3　REST アプリケーションにリクエストボディが渡されるまでの流れ（リスト型プロセスデータの全要素がリクエストボディに渡される場合）



[説明]

アプリケーション呼び出し情報ファイルの内容に従って，key 要素の値が"ListKey"，value 要素の値が{'aaa', null, 'ccc'}のデータが，プロセスデータテーブルからリクエストボディに渡されます。プロセスデータテーブルからデータを受け取ったリクエストボディは，REST アプリケーションに渡されます。

リスト型プロセスデータの 1 要素がリクエストボディに渡される場合（リスト内識別子が固定値のとき）

リスト内識別子が固定値のリスト型プロセスデータの 1 要素がリクエストボディに渡される場合の，アプリケーション呼び出し情報ファイルの記述例を次に示します。

■アプリケーション呼び出し情報ファイルの記述例

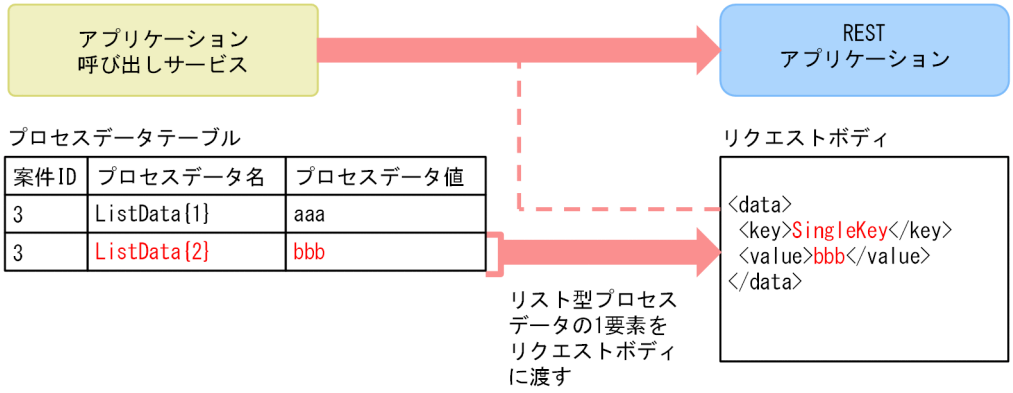
```
rest.request.body.key.SingleKey=$SListData{2}
```

[説明]

rest.request.body.key.SingleKey プロパティの値として，プロセスデータキー名"$SListData{2}"を指定しています。

アプリケーション呼び出し情報ファイルを上記の例のとおり記述した場合の，REST アプリケーションにリクエストボディが渡されるまでの流れを，次の図に示します。

図 15-4　REST アプリケーションにリクエストボディが渡されるまでの流れ（リスト内識別子が固定値のリスト型プロセスデータの 1 要素がリクエストボディに渡される場合）



[説明]

アプリケーション呼び出し情報ファイルの内容に従って，key 要素の値が"SingleKey"，value 要素の値が"bbb"のデータが，プロセスデータテーブルからリクエストボディに渡されます。プロセスデータテーブルからデータを受け取ったリクエストボディは，REST アプリケーションに渡されます。

リスト型プロセスデータの 1 要素がリクエストボディに渡される場合（リスト内識別子が"@MIIndex"のとき）

リスト内識別子が"@MIIndex"のリスト型プロセスデータの 1 要素がリクエストボディに渡される場合の，アプリケーション呼び出し情報ファイルの記述例を次に示します。

■アプリケーション呼び出し情報ファイルの記述例

```
rest.request.body.key.SingleKey=$SListData{@MIIndex}
```

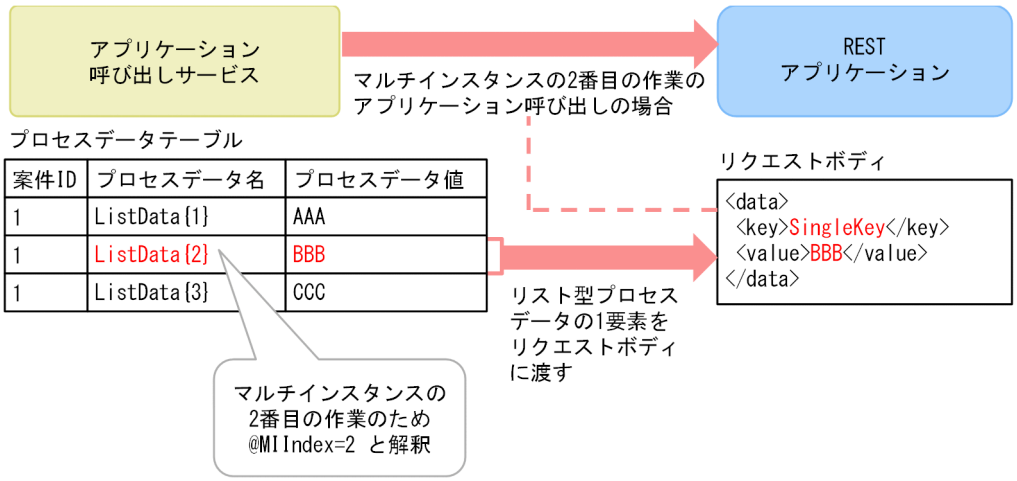
[説明]

rest.request.body.key.SingleKey プロパティの値として，プロセスデータキー名"$SListData{@MIIndex}"を指定しています。

なお，プロセスデータテーブルでのプロセスデータキー名"$SListData{@MIIndex}"に対応する値は，{'AAA', 'BBB', 'CCC'}であるとします。また，マルチインスタンスの 2 番目の作業（@MIIndex=2）の呼び出しであるとします。

アプリケーション呼び出し情報ファイルを上記の例のとおり記述した場合の，REST アプリケーションにリクエストボディが渡されるまでの流れを，次の図に示します。

図 15-5　REST アプリケーションにリクエストボディが渡されるまでの流れ（リスト内識別子が"@MIIndex"のリスト型プロセスデータの 1 要素がリクエストボディに渡される場合）



[説明]

アプリケーション呼び出し情報ファイルの内容に従って，key 要素の値が"SingleKey"，value 要素の値が"BBB"のデータが，プロセスデータテーブルからリクエストボディに渡されます。プロセスデータテーブルからデータを受け取ったリクエストボディは，REST アプリケーションに渡されます。

15.3.9　アプリケーション呼び出し情報ファイルとレスポンスボディの関係

アプリケーション呼び出し情報ファイルに記述した内容に従ったレスポンスボディの内容で，対応するプロセスデータテーブルのプロセスデータ値が更新されます。

単一型プロセスデータを更新する場合

単一型プロセスデータを更新する場合の，アプリケーション呼び出し情報ファイルの記述例を次に示します。

■アプリケーション呼び出し情報ファイルの記述例

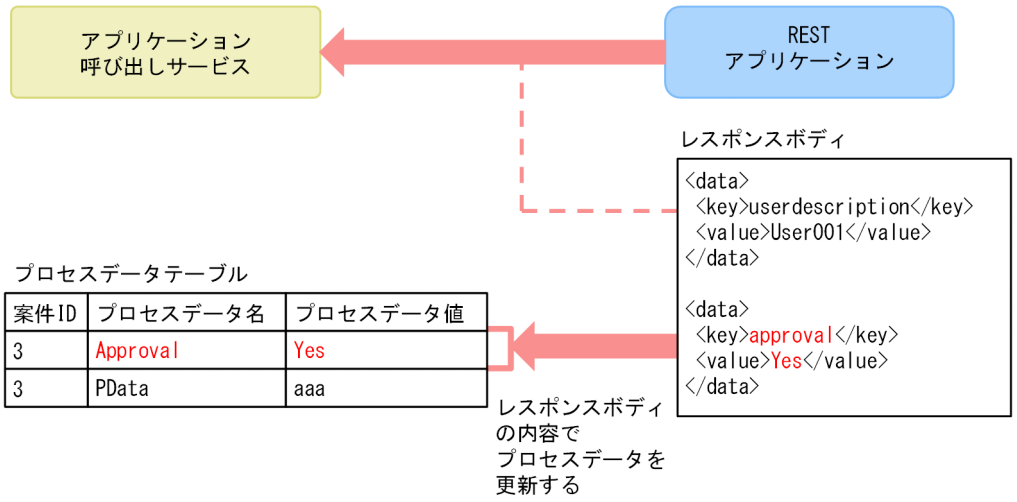
```
rest.response.pi.$SApproval=rest.response.body.key.approval
```

[説明]

REST アプリケーション呼び出しのレスポンスボディの，key 要素の値が"approval"であるvalue 要素の値で，プロセスデータキー名"$SApproval"に対応するプロセスデータ値を更新するように指定します。

アプリケーション呼び出し情報ファイルを上記の例のとおり記述した場合の，単一型プロセスデータが更新されるまでの流れを，次の図に示します。

図 15-6　プロセスデータが更新されるまでの流れ（単一型プロセスデータの場合）



[説明]

レスポンスボディの内容に従って，プロセスデータテーブルのプロセスデータキー名が"$SApproval"であるプロセスデータ値が"Yes"に更新されます。

リスト型プロセスデータの全要素を更新する場合

リスト型プロセスデータの全要素を更新する場合の，アプリケーション呼び出し情報ファイルの記述例を次に示します。

■アプリケーション呼び出し情報ファイルの記述例

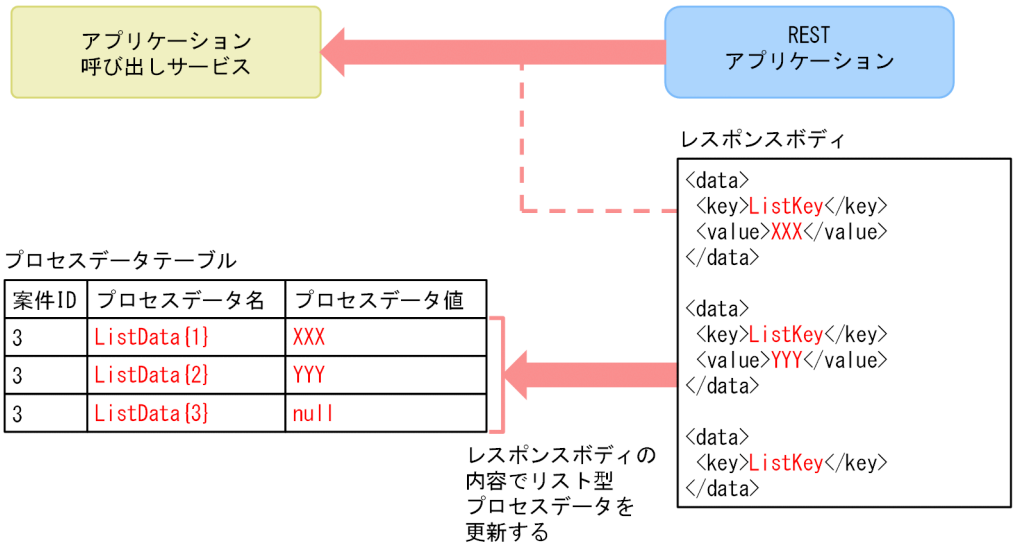
```
rest.response.pi.$SListData{}=rest.request.body.key.ListKey
```

[説明]

REST アプリケーション呼び出しのレスポンスボディの，key 要素の値が"ListKey"であるvalue 要素の値で，プロセスデータキー名"$SListData{}"に対応するプロセスデータ値を更新するように指定します。

アプリケーション呼び出し情報ファイルを上記の例のとおり記述した場合の，リスト型プロセスデータの全要素が更新されるまでの流れを，次の図に示します。

図 15-7　プロセスデータが更新されるまでの流れ（リスト型プロセスデータの全要素の場合）



[説明]

レスポンスボディの内容に従って，プロセスデータテーブルのプロセスデータキー名が"$SListData{}"に対応するプロセスデータ値が{'XXX', 'YYY', null}に更新されます。

リスト型プロセスデータの 1 要素を更新する場合（リスト内識別子が"@MIIndex"のとき）

リスト内識別子が"@MIIndex"のリスト型プロセスデータの 1 要素を更新する場合の，アプリケーション呼び出し情報ファイルの記述例を次に示します。

■アプリケーション呼び出し情報ファイルの記述例

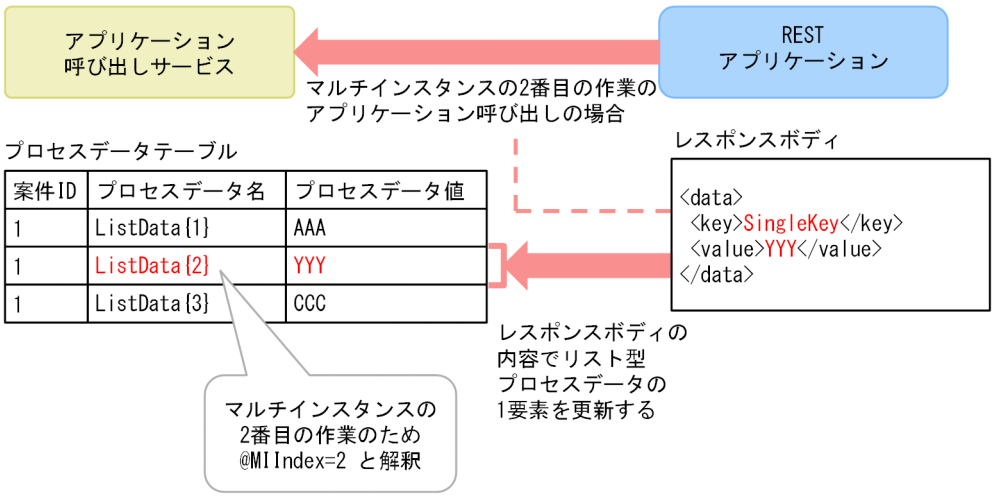
```
rest.response.pi.$SListData{@MIIndex}=rest.request.body.key.SingleKey
```

[説明]

REST アプリケーション呼び出しのレスポンスボディの，key 要素の値が"SingleKey"であるvalue 要素の値で，プロセスデータキー名"$SListData{@MIIndex}"に対応するプロセスデータ値を更新するように指定します。

アプリケーション呼び出し情報ファイルを上記の例のとおり記述した場合の，リスト内識別子が"@MIIndex"のリスト型プロセスデータの 1 要素を更新するまでの流れを，次の図に示します。

図 15-8　プロセスデータが更新されるまでの流れ（リスト内識別子が"@MIIndex"のリスト型プロセスデータの 1 要素の場合）



[説明]

マルチインスタンスの 2 番目の作業（@MIIndex=2）の呼び出しであるとします。

このとき，レスポンスボディの内容に従って，プロセスデータテーブルのプロセスデータキー名が"$SListData{2}"に対応するプロセスデータ値が"YYY"に更新されます。

15.3.10　アプリケーション呼び出し情報ファイルと Java オブジェクトに渡すデータの関係

アプリケーション呼び出し情報ファイルに記述した内容に従って，対応するプロセスデータテーブルのプロセスデータ値を，Java オブジェクトのメソッドの引数として Java オブジェクトに渡します。

単一型プロセスデータを Java オブジェクトのメソッドの引数に渡す場合

単一型プロセスデータを Java オブジェクトのメソッドの引数に渡す場合の，アプリケーション呼び出し情報ファイルの記述例を次に示します。

■アプリケーション呼び出し情報ファイルの記述例

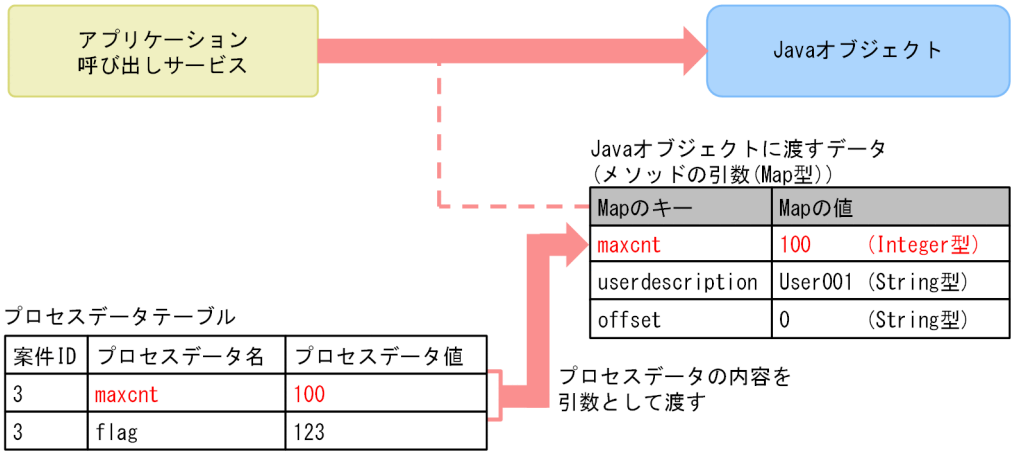
```
java.invoke.key.maxcount=$Nmaxcnt
java.invoke.key.userdescription=User001
java.invoke.key.offset=0
```

[説明]

1 行目で，java.invoke.key.maxcount プロパティの値として，プロセスデータキー名"$Nmaxcnt"を指定しています。

アプリケーション呼び出し情報ファイルを上記の例のとおり記述した場合の，Java オブジェクトにメソッドの引数が渡されるまでの流れを，次の図に示します。

図 15-9　Java オブジェクトにメソッドの引数が渡されるまでの流れ（単一型プロセスデータをメソッドの引数に渡す場合）



[説明]

アプリケーション呼び出し情報ファイルの内容に従って，key 要素の値が"maxcount"，value 要素の値が"100"のデータが，プロセスデータテーブルからメソッドの引数（Map 型）に渡されます。メソッドの引数（Map 型）は，Java オブジェクトに渡されます。

リスト型プロセスデータの全要素を Java オブジェクトのメソッドの引数に渡す場合

リスト型プロセスデータの全要素を Java オブジェクトのメソッドの引数に渡す場合の，アプリケーション呼び出し情報ファイルの記述例を次に示します。

■アプリケーション呼び出し情報ファイルの記述例

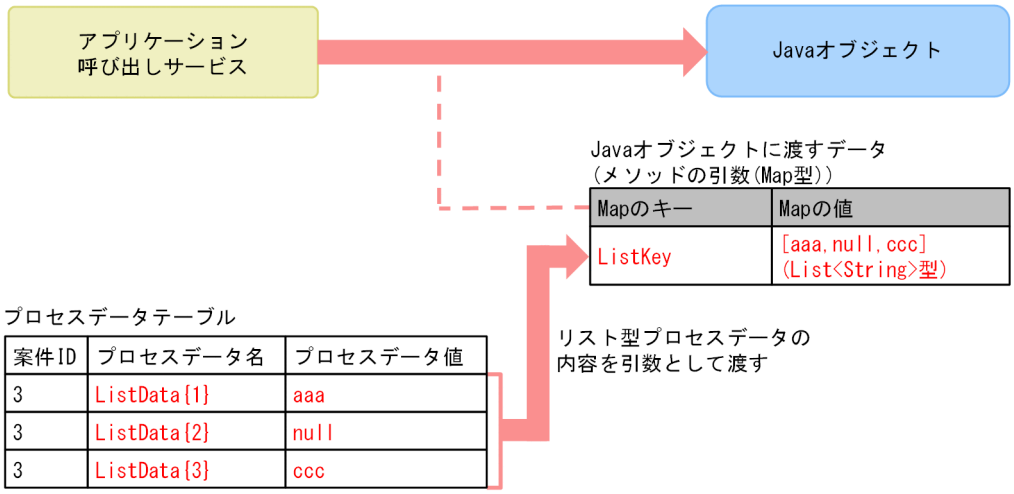
```
java.invoke.key.ListKey=$SListData{}
```

[説明]

java.invoke.key.ListKey プロパティの値として，プロセスデータキー名"$SListData{}"を指定しています。

アプリケーション呼び出し情報ファイルを上記の例のとおり記述した場合の，Java オブジェクトにメソッドの引数が渡されるまでの流れを，次の図に示します。

図 15-10　Java オブジェクトにメソッドの引数が渡されるまでの流れ（リスト型プロセスデータの全要素をメソッドの引数に渡す場合）



[説明]

アプリケーション呼び出し情報ファイルの内容に従って，key 要素の値が"ListKey"，value 要素の値が{'aaa'，null，'ccc'}のデータが，プロセスデータテーブルからメソッドの引数（Map 型）に渡されます。メソッドの引数（Map 型）は，Java オブジェクトに渡されます。

リスト型プロセスデータの 1 要素を Java オブジェクトのメソッドの引数に渡す場合（リスト内識別子が固定値のとき）

リスト内識別子が固定値のリスト型プロセスデータの 1 要素を Java オブジェクトのメソッドの引数に渡す場合の，アプリケーション呼び出し情報ファイルの記述例を次に示します。

■アプリケーション呼び出し情報ファイルの記述例

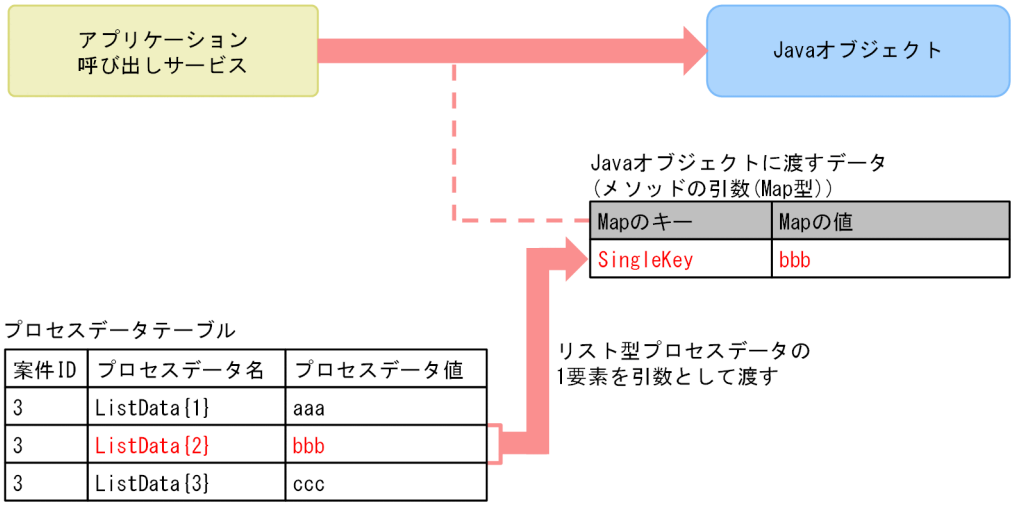
```
java.invoke.key.SingleKey=$SListData{2}
```

[説明]

java.invoke.key.SingleKey プロパティの値として，プロセスデータキー名"$SListData{2}"を指定しています。

アプリケーション呼び出し情報ファイルを上記の例のとおり記述した場合の，Java オブジェクトにメソッドの引数が渡されるまでの流れを，次の図に示します。

図 15-11　Java オブジェクトにメソッドの引数が渡されるまでの流れ（リスト内識別子が固定値のリスト型プロセスデータの 1 要素をメソッドの引数に渡す場合）



[説明]

アプリケーション呼び出し情報ファイルの内容に従って，key 要素の値が"SingleKey"，value 要素の値が"bbb"のデータが，プロセスデータテーブルからメソッドの引数（Map 型）に渡されます。メソッドの引数（Map 型）は，Java オブジェクトに渡されます。

リスト型プロセスデータの 1 要素を Java オブジェクトのメソッドの引数に渡す場合（リスト内識別子が"@MIIndex"のとき）

リスト内識別子が"@MIIndex"のリスト型プロセスデータの 1 要素を Java オブジェクトのメソッドの引数に渡す場合の，アプリケーション呼び出し情報ファイルの記述例を次に示します。

■アプリケーション呼び出し情報ファイルの記述例

```
java.invoke.key.SingleKey=$SListData{@MIIndex}
```

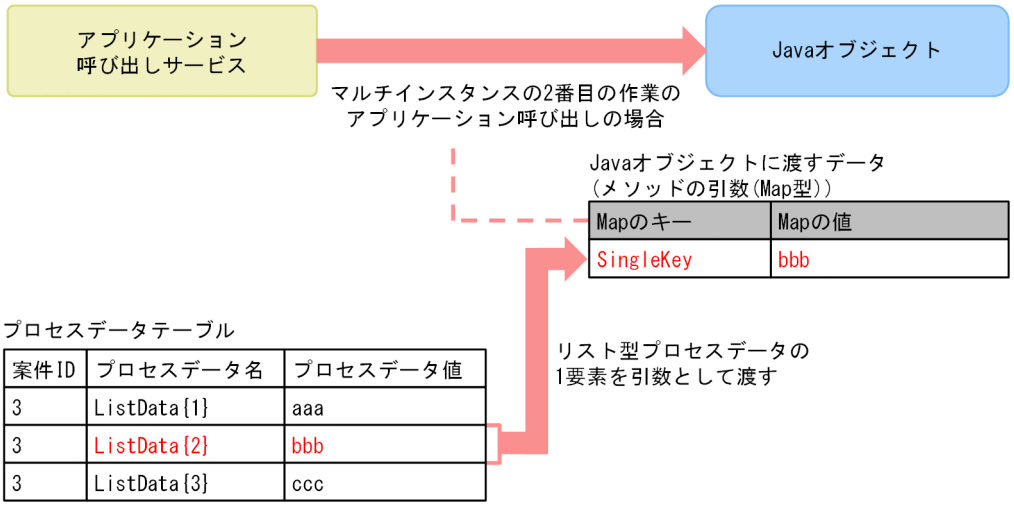
[説明]

java.invoke.key.SingleKey プロパティの値として，プロセスデータキー名"$SListData{@MIIndex}"を指定しています。

なお，プロセスデータテーブルでのプロセスデータキー名"$SListData{@MIIndex}"に対応する値は，{'aaa', 'bbb 'ccc'}であるとします。また，マルチインスタンスの 2 番目の作業（@MIIndex=2）の呼び出しであるとします。

アプリケーション呼び出し情報ファイルを上記の例のとおり記述した場合の，Java オブジェクトにメソッドの引数が渡されるまでの流れを，次の図に示します。

図 15-12　Java オブジェクトにメソッドの引数が渡されるまでの流れ（リスト内識別子が"@MIIndex"のリスト型プロセスデータの 1 要素をメソッドの引数に渡す場合）



[説明]

アプリケーション呼び出し情報ファイルの内容に従って，key 要素の値が"SingleKey"，value 要素の値が"bbb"のデータが，プロセスデータテーブルからメソッドの引数（Map 型）に渡されます。メソッドの引数（Map 型）は，Java オブジェクトに渡されます。

15.3.11　アプリケーション呼び出し情報ファイルと Java オブジェクトから渡すデータの関係

アプリケーション呼び出し情報ファイルに記述した内容に従った Java オブジェクトのメソッドの戻り値の内容で，対応するプロセスデータテーブルのプロセスデータ値が更新されます。

単一型プロセスデータを更新する場合

単一型プロセスデータを更新する場合の，アプリケーション呼び出し情報ファイルの記述例を次に示します。

■アプリケーション呼び出し情報ファイルの記述例

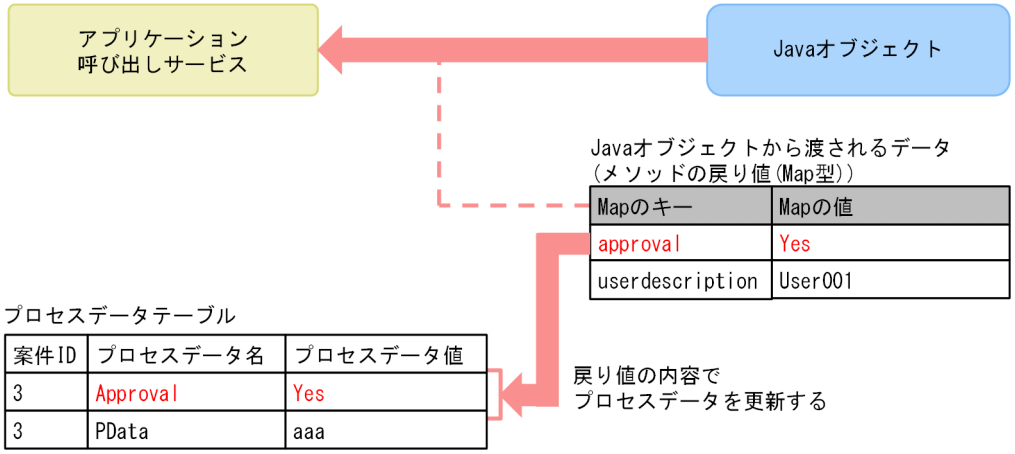
```
java.return.pi.$SApproval=java.return.key.apporoval
```

[説明]

Java オブジェクトのメソッドの戻り値の，key 要素の値が"approval"であるvalue 要素の値で，プロセスデータキー名"$SApproval"に対応するプロセスデータ値を更新するように指定します。

アプリケーション呼び出し情報ファイルを上記の例のとおり記述した場合の，単一型プロセスデータが更新されるまでの流れを，次の図に示します。

図 15-13　プロセスデータが更新されるまでの流れ（単一型プロセスデータの場合）



[説明]

メソッドの戻り値（Map 型）の内容に従って，プロセスデータテーブルのプロセスデータキー名が"$SApproval"であるプロセスデータ値が"Yes"に更新されます。

リスト型プロセスデータの全要素を更新する場合

リスト型プロセスデータの全要素を更新する場合の，アプリケーション呼び出し情報ファイルの記述例を次に示します。

■アプリケーション呼び出し情報ファイルの記述例

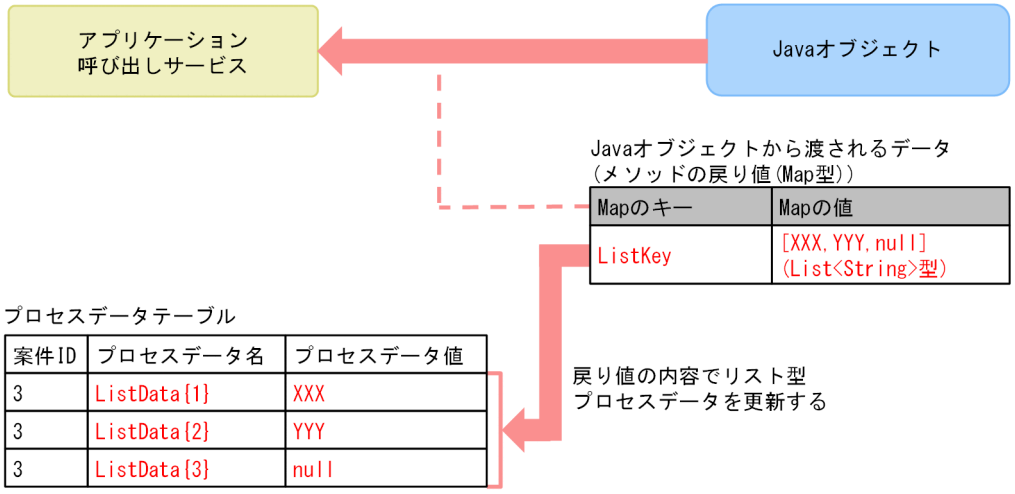
```
java.return.pi.$SListData{}=java.return.key.ListKey
```

[説明]

Java オブジェクトのメソッドの戻り値の，key 要素の値が"ListKey"であるvalue 要素の値で，プロセスデータキー名"$SListData{}"に対応するプロセスデータ値を更新するように指定します。

アプリケーション呼び出し情報ファイルを上記の例のとおり記述した場合の，リスト型プロセスデータの全要素が更新されるまでの流れを，次の図に示します。

図 15-14　プロセスデータが更新されるまでの流れ（リスト型プロセスデータの全要素の場合）



[説明]

メソッドの戻り値（Map 型）の内容に従って，プロセスデータテーブルのプロセスデータキー名が"$SListData{}"に対応するプロセスデータ値が{'XXX', 'YYY', null}に更新されます。

リスト型プロセスデータの 1 要素を更新する場合（リスト内識別子が"@MIIndex"のとき）

リスト内識別子が"@MIIndex"のリスト型プロセスデータの 1 要素を更新する場合の，アプリケーション呼び出し情報ファイルの記述例を次に示します。

■アプリケーション呼び出し情報ファイルの記述例

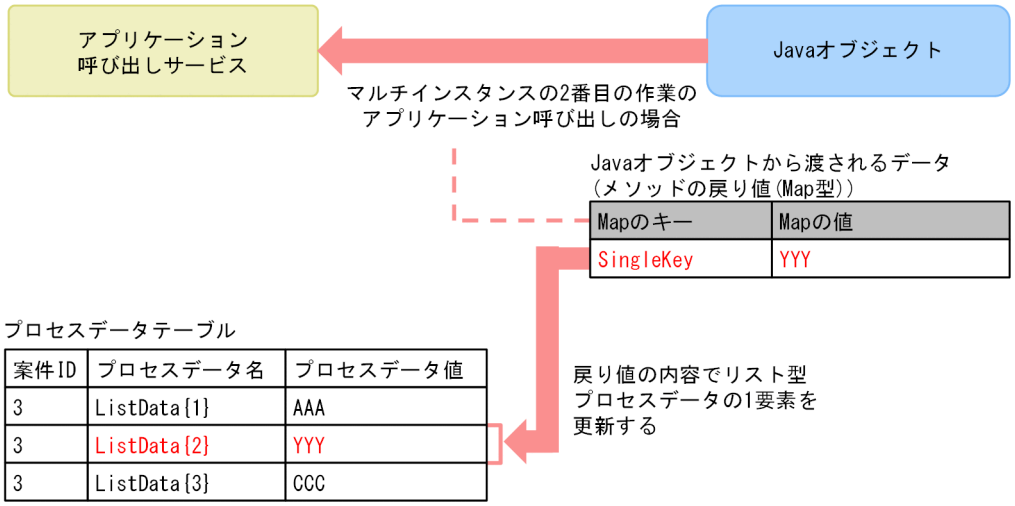
```
java.return.pi.$SListData{@MIIndex}=java.return.key.SingleKey
```

[説明]

Java オブジェクトのメソッドの戻り値の，key 要素の値が"SingleKey"であるvalue 要素の値で，プロセスデータキー名"$SListData{@MIIndex}"に対応するプロセスデータ値を更新するように指定します。

アプリケーション呼び出し情報ファイルを上記の例のとおり記述した場合の，リスト内識別子が"@MIIndex"のリスト型プロセスデータの 1 要素を更新するまでの流れを，次の図に示します。

図 15-15　プロセスデータが更新されるまでの流れ（リスト内識別子が"@MIIndex"のリスト型プロセスデータの 1 要素の場合）



［説明］

マルチインスタンスの 2 番目の作業（@MIIndex=2）の呼び出しであるとします。

このとき，メソッドの戻り値（Map 型）の内容に従って，プロセスデータテーブルのプロセスデータキー名"$SListData{2}"に対応するプロセスデータ値"YYY"に更新されます。

15.3.12　アプリケーション呼び出し情報ファイルに指定するパラメタの一覧

指定するパラメタの一覧（アプリケーション呼び出し情報ファイル）

アプリケーション呼び出し情報ファイルに指定するパラメタの一覧を示します。

表 15-4　指定するパラメタの一覧（アプリケーション呼び出し情報ファイル）

項番	キー名	カテゴリ	指定内容
1	type (REST)	REST アプリケーションの呼び出し	アプリケーション呼び出しの種類（REST）
2	rest.request.method	REST アプリケーションの呼び出し	REST アプリケーションのメソッド
3	rest.request.url	REST アプリケーションの呼び出し	REST アプリケーションの URL
4	rest.request.header.filepath	REST アプリケーションの呼び出し	HTTP ヘッダを記述したファイルのファイルパス
5	rest.request.stylesheet.filepath	REST アプリケーションの呼び出し	リクエストボディ用スキーマ変換スタイルシートのファイルパス
6	rest.request.read.timeout	REST アプリケーションの呼び出し	REST アプリケーションの呼び出し時の読み込みタイムアウト値

項番	キー名	カテゴリ	指定内容
7	rest.request.connect.timeout	REST アプリケーションの呼び出し	REST アプリケーション呼び出し時の接続タイムアウト値
8	rest.request.body.key.<key要素値>	REST アプリケーションの呼び出し	リクエストボディに格納する値
9	rest.response.stylesheet.filepath	REST アプリケーションの呼び出し	レスポンスボディ用スキーマ変換スタイルシートのファイルパス
10	rest.response.pi.<$変数（形式1）>	REST アプリケーションの呼び出し	登録する呼び出し元案件のプロセスデータの値
11	rest.response.pi.<$変数（形式2）>	REST アプリケーションの呼び出し	登録する呼び出し元案件のプロセスデータの値
12	rest.request.idempotency	REST アプリケーションの呼び出し	REST アプリケーションがべき等性を保証しているかどうか
13	type（JAVA）	Java オブジェクトの呼び出し	アプリケーション呼び出しの種類（JAVA）
14	java.invoke.class	Java オブジェクトの呼び出し	Java オブジェクトのクラス名
15	java.invoke.key.<key要素値>	Java オブジェクトの呼び出し	Java オブジェクトのメソッドの引数に指定する値
16	java.return.pi.<$変数（形式1）>	Java オブジェクトの呼び出し	登録する呼び出し元案件のプロセスデータの値
17	java.return.pi.<$変数（形式2）>	Java オブジェクトの呼び出し	登録する呼び出し元案件のプロセスデータの値
18	type（NEW_PI_MESSAGE）	メッセージによる案件の投入	アプリケーション呼び出しの種類（NEW_PI_MESSAGE）
19	new.bp.name	メッセージによる案件の投入	新規に投入する案件のビジネスプロセス定義名
20	new.bp.version	メッセージによる案件の投入	新規に投入する案件のビジネスプロセス定義バージョン
21	new.pi.name	メッセージによる案件の投入	新規に投入する案件の案件名
22	new.pi.<$変数（形式1）>	メッセージによる案件の投入	登録する，新規に投入する案件のプロセスデータ値
23	new.pi.<$変数（形式2）>	メッセージによる案件の投入	登録する，新規に投入する案件のプロセスデータ値
24	type（OTHER_PI_MESSAGE）	メッセージによる他案件の呼び出し	アプリケーション呼び出しの種類（OTHER_PI_MESSAGE）
25	other.pi.id	メッセージによる他案件の呼び出し	呼び出し先案件の案件 ID

項番	キー名	カテゴリ	指定内容
26	other.pi.<$変数（形式1）>	メッセージによる他案件の呼び出し	登録する呼び出し先案件のプロセスデータ値
27	other.pi.<$変数（形式2）>	メッセージによる他案件の呼び出し	登録する呼び出し先案件のプロセスデータ値
28	type（SELF_PI_MESSAGE）	メッセージによる自案件の呼び出し	アプリケーション呼び出しの種類（SELF_PI_MESSAGE）
29	self.pi.<$変数（形式1）>	メッセージによる自案件の呼び出し	登録する呼び出し元案件のプロセスデータ値
30	self.pi.<$変数（形式2）>	メッセージによる自案件の呼び出し	登録する呼び出し元案件のプロセスデータ値
31	type（ERROR）	エラーによる自案件の呼び出し	アプリケーション呼び出しの種類（ERROR）
32	self.pi.<$変数（形式1）>	エラーによる自案件の呼び出し	登録する呼び出し元案件のプロセスデータ値
33	self.pi.<$変数（形式2）>	エラーによる自案件の呼び出し	登録する呼び出し元案件のプロセスデータ値

15.3.13　キー名と値で使用されている変数（アプリケーション呼び出し情報ファイル）

「表 15-4　指定するパラメタの一覧（アプリケーション呼び出し情報ファイル）」内のキー名と値で使用されている変数の意味を，次の表に示します。

表 15-5　キー名と値で使用されている変数（アプリケーション呼び出し情報ファイル）

項番	変数	説明
1	<$変数（形式1）>	1. 単一型プロセスデータのプロセスデータキー名（"{"および"}"を含まない） 指定例 $Skey 2. リスト型プロセスデータの 1 要素を表すプロセスデータキー名（末尾が"{<リスト内識別子>}"で終わる） 指定例 $Skey{5}
2	<$変数（形式2）>	リスト型プロセスデータの全要素を表すプロセスデータキー名（末尾が"{}"で終わる） 指定例 $Skey{}
3	<key要素値>	[REST アプリケーション呼び出しの場合]

項番	変数	説明

3	<key要素値>	REST アプリケーションの呼び出し時に使用されるボディデータ（リクエストボディまたはレスポンスボディ）の，data 要素中のkey 要素の値
		次に示すボディデータの記述例の"aaa"に該当する値です。
		`<data>`
		`  <key>aaa</key>`
		`  <value>bbb</value>`
		`</data>`

		[Java オブジェクト呼び出しの場合]
		アプリケーション呼び出しサービスと Java オブジェクトでやり取りするデータ（execute メソッドの引数または戻り値に格納されるデータで java.util.Map 型）のキー
		次に示すやり取りするデータの記述例の"aaa"に該当する値です。
		`+----+----+`
		`|キー| 値 |`
		`+----+----+`
		`|aaa |bbb |`
		`+----+----+`

15.3.14　REST アプリケーションの呼び出しの場合にファイルに指定する内容

アプリケーション呼び出しサービスは，REST アプリケーションを呼び出して，呼び出し元の作業を完了させます。

アプリケーション呼び出し情報ファイルの格納先ファイルパス

REST アプリケーションの呼び出しについて設定する，アプリケーション呼び出し情報ファイルの格納先ファイルパスは，次のどちらかです。

- <BpmnCallInformationFileDirプロパティの指定値>/ope/<operationRefの値>.properties
- <BpmnCallInformationFileDirプロパティの指定値>/msg/<messageRefの値>.properties

アプリケーション呼び出し情報ファイルの記述例

REST アプリケーションの呼び出しについて設定する場合の，アプリケーション呼び出し情報ファイルの記述例を次に示します。

```
type=REST
rest.request.method=POST
rest.request.url=http://hostname/app/function
rest.request.header.filepath= /home/csciw/header.properties
rest.request.body.key.offset=0
rest.request.body.key.userdescription=@PIName
rest.request.body.key.maxcount=$Nmaxcnt
rest.response.pi.$SApproval=rest.response.body.key.approval
```

(1) type（REST）

キーの説明

アプリケーション呼び出しの種類を指定します。このキーは指定必須です。

指定値の説明

"REST"を指定してください。

REST タイプのアプリケーションにアクセスします。

値の形式ごとの指定可否を次の表に示します。

形式	指定可否
組み込み変数	不可
$変数（形式 1）	不可
$変数（形式 2）	不可

注意事項

大文字と小文字は区別します。

関連項目

- 15.3.13　キー名と値で使用されている変数（アプリケーション呼び出し情報ファイル）

(2) rest.request.method

キーの説明

REST アプリケーションのメソッドを指定します。このキーは指定必須です。

指定値の説明

呼び出し対象の REST アプリケーションのメソッド名を指定します。次のどれかを指定してください。

GET：リソースを取得するリクエストの場合

POST：リソースを追加するリクエストの場合

PUT：リソースを更新するリクエストの場合

DELETE：リソースを削除するリクエストの場合

値の形式ごとの指定可否を次の表に示します。

形式	指定可否
組み込み変数	不可
$変数（形式 1）	不可
$変数（形式 2）	不可

注意事項

大文字と小文字は区別します。

関連項目

- 15.3.13　キー名と値で使用されている変数（アプリケーション呼び出し情報ファイル）

(3)　rest.request.url

キーの説明

REST アプリケーションの URL を指定します。このキーは指定必須です。

指定値の説明

呼び出し対象の REST アプリケーションの URL を指定します。

記述例

```
http://host1:80/apppath1/apppath2
```

プロセスデータキー名および組み込み変数を混合して利用する場合

URL の以下の部分にプロセスデータキー名，および組み込み変数を記述できます。

- パスのディレクトリ名およびファイル名
- クエリパラメタのキーおよび値

記述例

```
http://host1:80/apppath1/apppath2/@PIName?doc1id=$Sid1&doc2id=$Sid2
```

マルチインスタンスのサービスタスクでリスト型プロセスデータの 1 要素を利用する場合

末尾が"{@MIIndex}"のプロセスデータキー名をプロパティ値に指定してください。

値の形式ごとの指定可否を次の表に示します。

形式	指定可否
組み込み変数	可

形式	指定可否
$変数（形式 1）	可
$変数（形式 2）	不可

注意事項

- このプロパティに指定したプロセスデータおよび組み込み変数の値は，次のとおり URI エンコードされます。なお，URI エンコードの文字コードは UTF-8 とします。
  - 半角スペースは"%20"にエンコードされる
  - 半角スペース以外の文字はjava.net.URLEncoder クラスの仕様に従いエンコードされる
- プロセスデータおよび組み込み変数の値以外は，自動では URI エンコードされません。RFC 2396 で定義されている文字を使用し，必要に応じて URI エンコードした値を指定してください。

関連項目

- 15.3.13　キー名と値で使用されている変数（アプリケーション呼び出し情報ファイル）

(4)　rest.request.header.filepath

キーの説明

アプリケーション呼び出しサービスのリクエストに HTTP ヘッダを含める場合に指定します。

指定を省略した場合，ヘッダ部が空のリクエストを送信します。

指定値の説明

HTTP ヘッダを記載したヘッダファイルのパスを，絶対パスで指定します。

ヘッダファイルについては「15.3.6　HTTP ヘッダを記述したファイルの指定方法（アプリケーション呼び出し情報ファイル）」を参照してください。

値の形式ごとの指定可否を次の表に示します。

形式	指定可否
組み込み変数	不可
$変数（形式 1）	不可
$変数（形式 2）	不可

注意事項

OS が UNIX，かつ，root ユーザ以外の任意の OS ユーザが CSCIW の業務アプリケーションを実行する場合， このプロパティに指定したファイルの所有者，およびグループもあわせて変更してください。

関連項目

- 15.3.13　キー名と値で使用されている変数（アプリケーション呼び出し情報ファイル）

(5)　rest.request.stylesheet.filepath

キーの説明

アプリケーション呼び出しサービスのリクエストボディの，XML スキーマ変換をする場合に指定します。

指定を省略した場合，スキーマ変換をしません。

指定値の説明

XSLT で記述されたスタイルシートのファイルパスを，絶対パスで指定します。

値の形式ごとの指定可否を次の表に示します。

形式	指定可否
組み込み変数	不可
$変数（形式 1）	不可
$変数（形式 2）	不可

注意事項

OS が UNIX，かつ，root ユーザ以外の任意の OS ユーザが CSCIW の業務アプリケーションを実行する場合， このプロパティに指定したファイルの所有者，およびグループもあわせて変更してください。

関連項目

- 15.3.13　キー名と値で使用されている変数（アプリケーション呼び出し情報ファイル）

(6)　rest.request.read.timeout

キーの説明

REST アプリケーション呼び出し時の読み込みタイムアウト値を変更する場合に指定します。

指定を省略した場合，タイムアウト値は Cosminexus の J2EE サーバ単位の通信タイムアウトを設定する `ejbserver.javaee.jaxrs.config.client.readTimeout` キーの設定に従います。

指定値の説明

REST アプリケーション呼び出し時のクライアントソケットの読み込みタイムアウト値を，0～2,147,483,647 の範囲で指定します。単位はミリ秒です。

タイムアウトしない設定をする場合，0 を指定してください。

値の形式ごとの指定可否を次の表に示します。

形式	指定可否
組み込み変数	不可
$変数（形式 1）	不可
$変数（形式 2）	不可

注意事項

REST アプリケーション呼び出しのタイムアウト値を Cosminexus の設定に従わせる場合は，このプロパティを設計および指定する必要はありません。

関連項目

- 15.3.13　キー名と値で使用されている変数（アプリケーション呼び出し情報ファイル）

(7)　rest.request.connect.timeout

キーの説明

REST アプリケーション呼び出し時の接続タイムアウト値を変更する場合に指定します。

指定を省略した場合，タイムアウト値は Cosminexus の J2EE サーバ単位の通信タイムアウトを設定する `ejbserver.javaee.jaxrs.config.client.connectTimeout` キーの設定に従います。

指定値の説明

REST アプリケーション呼び出し時のクライアントソケットの接続タイムアウト値を，0～2,147,483,647 の範囲で指定します。単位はミリ秒です。

タイムアウトしない設定をする場合，0 を指定してください。

値の形式ごとの指定可否を次の表に示します。

形式	指定可否
組み込み変数	不可
$変数（形式 1）	不可

形式	指定可否
$変数（形式 2）	不可

注意事項

REST アプリケーション呼び出しのタイムアウト値を Cosminexus の設定に従わせる場合は，このプロパティを設計および指定する必要はありません。

関連項目

- 15.3.13　キー名と値で使用されている変数（アプリケーション呼び出し情報ファイル）

(8)　rest.request.body.key.<key 要素値>

キーの説明

アプリケーション呼び出しサービスのリクエストボディに格納する値を指定します。

`rest.request.method` が POST，または PUT の場合に指定できます。

指定を省略した場合，リクエストボディが空のリクエストを送信します。

指定値の説明

リクエストボディに渡す value 要素の値を指定してください。指定された<key 要素値>を key 要素の値に，また指定されたプロパティの値を value 要素の値に持つ data 要素を，リクエストボディに追加します。

XML スキーマについては「6.1.3　ボディデータスキーマ（XML）」を参照してください。

マルチインスタンスのサービスタスクでリスト型プロセスデータの 1 要素をリクエストボディに渡す場合
　末尾が"`{@MIIndex}`"のプロセスデータキー名をプロパティ値に指定してください。

値の形式ごとの指定可否を次の表に示します。

形式	指定可否
組み込み変数	可
$変数（形式 1）	可
$変数（形式 2）	可

注意事項

プロパティ値にプロセスデータの値が null のプロセスデータキー名や組み込み変数を指定した場合，value 要素のないリクエストボディが作成されます。

関連項目

- 15.3.13　キー名と値で使用されている変数（アプリケーション呼び出し情報ファイル）

(9)　rest.response.stylesheet.filepath

キーの説明

REST アプリケーションのレスポンスボディの XML スキーマ変換をする場合に指定します。

指定を省略した場合，スキーマ変換をしません。

指定値の説明

XSLT で記述されたスタイルシートのファイルパスを絶対パスで指定します。

値の形式ごとの指定可否を次の表に示します。

形式	指定可否
組み込み変数	不可
$変数（形式 1）	不可
$変数（形式 2）	不可

注意事項

OS が UNIX，かつ，root ユーザ以外の任意の OS ユーザが CSCIW の業務アプリケーションを実行する場合， このプロパティに指定したファイルの所有者，およびグループもあわせて変更してください。

関連項目

- 15.3.13　キー名と値で使用されている変数（アプリケーション呼び出し情報ファイル）

(10)　rest.response.pi.<$変数（形式 1）>

キーの説明

呼び出し元案件のプロセスデータ<$変数（形式 1）>を登録する場合に指定します。

マルチインスタンスのサービスタスクでリスト型プロセスデータの 1 要素を登録する場合

末尾が"{@MIIndex}"のプロセスデータキー名を，プロパティキーの<$変数（形式 1）>の部分に指定してください。

指定値の説明

プロセスデータに登録する値を指定します。

REST アプリケーションのレスポンスボディをプロセスデータに格納する場合
　値に「rest.response.body.key.<key 要素値>」を指定してください。

値の形式ごとの指定可否を次の表に示します。

形式	指定可否
組み込み変数	可
$変数（形式 1）	可
$変数（形式 2）	不可

注意事項

rest.response.body.key.<key 要素値>を指定した場合の注意事項を示します。

- REST アプリケーション呼び出しのレスポンスボディに，<key 要素値>に該当する key 要素がないとき，REST アプリケーション呼び出しは成功しますが，作業完了でエラーとなります。
- <key 要素値>に該当する key 要素が 1 つあり，対応する value 要素がないとき，プロセスデータ値には null が登録されます。
- <key 要素値>に該当する key 要素が 1 つあり，value 要素の値が空のとき，REST アプリケーション呼び出しは成功しますが，作業完了でエラーとなります。
- <key 要素値>に該当する key 要素が 2 つ以上あるとき，REST アプリケーション呼び出しは成功しますが，作業完了でエラーとなります。

関連項目

- 15.3.13　キー名と値で使用されている変数（アプリケーション呼び出し情報ファイル）

(11)　rest.response.pi.<$変数（形式 2）>

キーの説明

呼び出し元案件のプロセスデータ<$変数（形式 2）>を登録する場合に指定します。

指定値の説明

プロセスデータに登録する値を指定します。

REST アプリケーションのレスポンスボディをプロセスデータに格納する場合
　値に「rest.response.body.key.<key 要素値>」を指定してください。

値の形式ごとの指定可否を次の表に示します。

形式	指定可否
組み込み変数	不可
$変数（形式 1）	不可
$変数（形式 2）	可

注意事項

rest.response.body.key.<key 要素値>を指定した場合の注意事項を示します。

- REST アプリケーション呼び出しのレスポンスボディに，<key 要素値>に該当する key 要素がないとき，REST アプリケーション呼び出しは成功しますが，作業完了でエラーとなります。
- REST アプリケーション呼び出しのレスポンスボディに，<key 要素値>に該当する key 要素があり，かつ対応する value 要素がないとき，プロセスデータ値には null が登録されます。
- REST アプリケーション呼び出しのレスポンスボディに，<key 要素値>に該当する key 要素があり，かつ対応する value 要素が空のとき，REST アプリケーション呼び出しは成功しますが，作業完了でエラーとなります。

関連項目

- 15.3.13　キー名と値で使用されている変数（アプリケーション呼び出し情報ファイル）

(12)　rest.request.idempotency

キーの説明

REST アプリケーションがべき等性を保証していない場合に`false` を指定します。

指定を省略した場合，REST アプリケーションがべき等性を保証しているものとして，アプリケーション呼び出しサービスが REST アプリケーションを複数回呼び出すことがあります。

指定値の説明

REST アプリケーションがべき等性を保証しているかどうかを指定します。

`true` の場合

べき等性を保証している場合に指定します。アプリケーション呼び出しサービスが REST アプリケーションを複数回呼び出すことがあります。

`false` の場合

べき等性を保証していない場合に指定します。アプリケーション呼び出しサービスが REST アプリケーションを複数回呼び出すことはありません。

値の形式ごとの指定可否を次の表に示します。

形式	指定可否
組み込み変数	不可
$変数（形式 1）	不可
$変数（形式 2）	不可

注意事項

通常，このプロパティを設定しないでください。このプロパティにfalse を指定する場合の注意事項を次に示します。

- REST アプリケーションの呼び出しで障害が発生した場合，呼び出し回数が 0 回になる場合があります。
- REST アプリケーションの呼び出しで障害が発生した場合，ciwchgapwork コマンドを使用して，作業の状態を変更する必要があります。REST アプリケーションを再実行する場合，作業を実行開始可能状態に戻してください。REST アプリケーションを再実行しない場合，作業を完了させ案件を遷移させてください。再実行するかどうかは REST アプリケーションの実行結果を基に決定してください。
- アプリケーション呼び出し制御情報のリトライ回数とリトライ間隔は有効になりません。

関連項目

- 15.3.13　キー名と値で使用されている変数（アプリケーション呼び出し情報ファイル）

15.3.15　Java オブジェクトの呼び出しの場合にファイルに指定する内容

アプリケーション呼び出しサービスは，Java オブジェクトを呼び出して，呼び出し元の作業を完了させます。

アプリケーション呼び出し情報ファイルの格納先ファイルパス

Java オブジェクトの呼び出しについて設定する，アプリケーション呼び出し情報ファイルの格納先ファイルパスを次に示します。

```
<BpmnCallInformationFileDirプロパティの指定値>/ope/<operationRefの値>.properties
```

アプリケーション呼び出し情報ファイルの記述例

Java オブジェクトの呼び出しについて設定する場合の，アプリケーション呼び出し情報ファイルの記述例を次に示します。

```
type=JAVA
java.invoke.class=sample.appcall.JavaCallTest
```

```
java.invoke.key.offset=0
java.invoke.key.userdescription=@PIName
java.invoke.key.maxcount=$Nmaxcnt
java.return.pi.$SApproval=java.return.key.approval
```

(1) type（JAVA）

キーの説明

アプリケーション呼び出しの種類を指定します。このキーは指定必須です。

指定値の説明

"JAVA"を指定してください。

Java オブジェクトのメソッドを呼び出します。

値の形式ごとの指定可否を次の表に示します。

形式	指定可否
組み込み変数	不可
$変数（形式 1）	不可
$変数（形式 2）	不可

注意事項

大文字と小文字は区別します。

関連項目

- 15.3.13　キー名と値で使用されている変数（アプリケーション呼び出し情報ファイル）

(2) java.invoke.class

キーの説明

CSCIW が提供する Java オブジェクト呼び出し用のインタフェースを継承した Java クラスの，完全修飾クラス名を指定します。このキーは指定必須です。

指定値の説明

呼び出し対象の Java オブジェクトのクラス名を，パッケージ名も含めて指定します。

指定例

```
sample.v1.TestClass
```

値の形式ごとの指定可否を次の表に示します。

形式	指定可否
組み込み変数	不可
$変数（形式 1）	不可
$変数（形式 2）	不可

注意事項

なし

関連項目

- 15.3.13　キー名と値で使用されている変数（アプリケーション呼び出し情報ファイル）

(3)　java.invoke.key.<key 要素値>

キーの説明

Java オブジェクトのメソッドの引数に指定する値を指定します。

指定を省略した場合，Java オブジェクトのメソッドの引数として要素数が 0 の Map を使用します。

指定値の説明

Java オブジェクトのメソッドに渡す Map 引数の値を指定してください。指定された<key 要素値>を Map のキーに，また指定されたプロパティの値を Map の値に使用します。

マルチインスタンスのサービスタスクで，リスト型プロセスデータの 1 要素を引数として渡す場合は，末尾が"{@MIIndex}"のプロセスデータキー名をプロパティ値に指定してください。

値の形式ごとの指定可否を次の表に示します。

形式	指定可否
組み込み変数	可
$変数（形式 1）	可
$変数（形式 2）	可

注意事項

Map 引数の値の型は，プロパティの値によって異なります。

- プロパティの値が<$変数（形式 1）>の場合

- 文字列型のとき：String 型
- 数値型のとき：Integer 型

- プロパティの値が<$変数（形式 2）>の場合
  - 文字列型のとき：List<String>型
  - 数値型のとき：List<Integer>型
- プロパティの値が組み込み変数の場合：String 型
- プロパティの値がその他の文字列の場合：String 型

関連項目

- 15.3.13　キー名と値で使用されている変数（アプリケーション呼び出し情報ファイル）

(4)　java.return.pi.<$変数（形式 1）>

キーの説明

呼び出し元案件のプロセスデータ<$変数（形式 1）>を登録する場合に指定します。

マルチインスタンスのサービスタスクでリスト型プロセスデータの 1 要素を登録する場合は，末尾が"{@MIIndex}"のプロセスデータキー名を，プロパティキーの<$変数（形式 1）>の部分に指定してください。

指定値の説明

プロセスデータに登録する値を指定します。

メソッドの戻り値からプロセスデータに格納する場合は，値に「java.return.key.<key 要素値>」を指定してください。

値の形式ごとの指定可否を次の表に示します。

形式	指定可否
組み込み変数	可
$変数（形式 1）	可
$変数（形式 2）	不可

注意事項

なし

関連項目

- 15.3.13　キー名と値で使用されている変数（アプリケーション呼び出し情報ファイル）

(5)　java.return.pi.<$変数（形式 2）>

キーの説明

呼び出し元案件のプロセスデータ<$変数（形式 2）>を登録する場合に指定します。

指定値の説明

プロセスデータに登録する値を指定します。

メソッドの戻り値からプロセスデータに格納する場合は，値に「java.return.key.<key 要素値>」を指定してください

値の形式ごとの指定可否を次の表に示します。

形式	指定可否
組み込み変数	不可
$変数（形式 1）	不可
$変数（形式 2）	可

注意事項

なし

関連項目

- 15.3.13　キー名と値で使用されている変数（アプリケーション呼び出し情報ファイル）

15.3.16　メッセージによる案件投入の場合にファイルに指定する内容

アプリケーション呼び出しサービスは，アプリケーション呼び出し情報ファイルで指定したビジネスプロセス定義に，新規案件を投入します。そのあとで，呼び出し元の作業を完了させます。

アプリケーション呼び出し情報ファイルの格納先ファイルパス

メッセージによる案件の投入について設定する，アプリケーション呼び出し情報ファイルの格納先ファイルパスを次に示します。

```
<BpmnCallInformationFileDirプロパティの指定値>/msg/<messageRefの値>.properties
```

アプリケーション呼び出し情報ファイルの記述例

メッセージによる案件の投入について設定する場合の，アプリケーション呼び出し情報ファイルの記述例を次に示します。

```
type=NEW_PI_MESSAGE
new.bp.name=BP0001
new.pi.name=@PIName
new.pi.$SFlag=$SMyFlag
```

(1) type（NEW_PI_MESSAGE）

キーの説明

アプリケーション呼び出しの種類を指定します。このキーは指定必須です。

指定値の説明

"NEW_PI_MESSAGE"を指定してください。

メッセージタイプの開始イベントを持つビジネスプロセスの新規案件を投入します。

値の形式ごとの指定可否を次の表に示します。

形式	指定可否
組み込み変数	不可
$変数（形式 1）	不可
$変数（形式 2）	不可

注意事項

大文字と小文字は区別します。

関連項目

- 15.3.13　キー名と値で使用されている変数（アプリケーション呼び出し情報ファイル）

(2) new.bp.name

キーの説明

新規投入案件のビジネスプロセス定義名を指定します。このキーは指定必須です。

指定値の説明

投入案件のビジネスプロセス定義名を指定します。

値の形式ごとの指定可否を次の表に示します。

形式	指定可否
組み込み変数	可
$変数（形式 1）	可
$変数（形式 2）	不可

注意事項

なし

関連項目

- 15.3.13　キー名と値で使用されている変数（アプリケーション呼び出し情報ファイル）

(3)　new.bp.version

キーの説明

新規投入案件のビジネスプロセス定義のバージョンを指定します。

指定を省略した場合，指定したビジネスプロセス定義名の中で投入できる最新のバージョンが使用されます。

指定値の説明

投入案件のビジネスプロセス定義のバージョンを整数で指定します。

値の形式ごとの指定可否を次の表に示します。

形式	指定可否
組み込み変数	可
$変数（形式 1）	可
$変数（形式 2）	不可

注意事項

なし

関連項目

- 15.3.13　キー名と値で使用されている変数（アプリケーション呼び出し情報ファイル）

(4)　new.pi.name

キーの説明

新規投入案件の案件名を指定します。指定を省略した場合，案件名は null になります。

指定値の説明

投入案件の案件名を指定します。

値の形式ごとの指定可否を次の表に示します。

形式	指定可否
組み込み変数	可
$変数（形式 1）	可
$変数（形式 2）	不可

注意事項

なし

関連項目

- 15.3.13　キー名と値で使用されている変数（アプリケーション呼び出し情報ファイル）

(5)　new.pi.<$変数（形式 1）>

キーの説明

新規投入案件のプロセスデータ<$変数（形式 1）>を登録する場合に指定します。

指定値の説明

プロセスデータに登録する値を指定します。

値の形式ごとの指定可否を次の表に示します。

形式	指定可否
組み込み変数	可
$変数（形式 1）	可

形式	指定可否
$変数（形式 2）	不可

注意事項

なし

関連項目

- 15.3.13　キー名と値で使用されている変数（アプリケーション呼び出し情報ファイル）

(6)　new.pi.<$変数（形式 2）>

キーの説明

新規投入案件のプロセスデータ<$変数（形式 2）>を登録する場合に指定します。

指定値の説明

プロセスデータに登録する値を指定します。

値の形式ごとの指定可否を次の表に示します。

形式	指定可否
組み込み変数	不可
$変数（形式 1）	不可
$変数（形式 2）	可

注意事項

なし

関連項目

- 15.3.13　キー名と値で使用されている変数（アプリケーション呼び出し情報ファイル）

(7)　self.pi.<$変数（形式 1）>

キーの説明

呼び出し元案件のプロセスデータ<$変数（形式 1）>を登録する場合に指定します。

指定値の説明

プロセスデータに登録する値を指定します。

値の形式ごとの指定可否を次の表に示します。

形式	指定可否
組み込み変数	可
$変数（形式 1）	可
$変数（形式 2）	不可

注意事項

なし

関連項目

- 15.3.13　キー名と値で使用されている変数（アプリケーション呼び出し情報ファイル）

(8)　self.pi.<$変数（形式 2）>

キーの説明

呼び出し元案件のプロセスデータ<$変数（形式 2）>を登録する場合に指定します。

指定値の説明

プロセスデータに登録する値を指定します。

値の形式ごとの指定可否を次の表に示します。

形式	指定可否
組み込み変数	不可
$変数（形式 1）	不可
$変数（形式 2）	可

注意事項

なし

関連項目

- 15.3.13　キー名と値で使用されている変数（アプリケーション呼び出し情報ファイル）

15.3.17　メッセージによる他案件の呼び出しの場合にファイルに指定する内容

アプリケーション呼び出しサービスは，アプリケーション呼び出し情報ファイルで指定した案件 ID を持つ案件中の作業のうち，呼び出し元と同じ ref 識別子を作業者 ID に持つ作業を完了させます。そのあとで，呼び出し元の作業も完了させます。

アプリケーション呼び出し情報ファイルの格納先ファイルパス

メッセージによる他案件の呼び出しについて設定する，アプリケーション呼び出し情報ファイルの格納先ファイルパスを次に示します。

```
<BpmnCallInformationFileDirプロパティの指定値>/msg/<messageRefの値>.properties
```

アプリケーション呼び出し情報ファイルの記述例

メッセージによる他案件の呼び出しについて設定する場合の，アプリケーション呼び出し情報ファイルの記述例を次に示します。

```
type=OTHER_PI_MESSAGE
other.pi.id=$NtargetKey
other.pi.$SFlag=$SMyFlag
```

(1)　type（OTHER_PI_MESSAGE）

キーの説明

アプリケーション呼び出しの種類を指定します。このキーは指定必須です。

指定値の説明

"OTHER_PI_MESSAGE"を指定してください。

メッセージを受け取ることができる状態の案件に対してメッセージをスローします。

値の形式ごとの指定可否を次の表に示します。

形式	指定可否
組み込み変数	不可
$変数（形式 1）	不可
$変数（形式 2）	不可

注意事項

大文字と小文字は区別します。

関連項目

- 15.3.13　キー名と値で使用されている変数（アプリケーション呼び出し情報ファイル）

(2)　other.pi.id

キーの説明

呼び出し先案件の案件 ID を指定します。このキーは指定必須です。

指定値の説明

呼び出し先案件の案件 ID を格納したプロセスデータキー名を指定します。

値の形式ごとの指定可否を次の表に示します。

形式	指定可否
組み込み変数	可
$変数（形式 1）	可
$変数（形式 2）	不可

注意事項

なし

関連項目

- 15.3.13　キー名と値で使用されている変数（アプリケーション呼び出し情報ファイル）

(3)　other.pi.<$変数（形式 1）>

キーの説明

呼び出し先案件のプロセスデータ<$変数（形式 1）>を登録する場合に指定します。

指定値の説明

プロセスデータに登録する値を指定します。

値の形式ごとの指定可否を次の表に示します。

形式	指定可否
組み込み変数	可
$変数（形式 1）	可

形式	指定可否
$変数（形式 2）	不可

注意事項

メッセージがキャッチされなかった場合，プロセスデータは登録されません。

関連項目

- 15.3.13　キー名と値で使用されている変数（アプリケーション呼び出し情報ファイル）

(4)　other.pi.<$変数（形式 2）>

キーの説明

呼び出し先案件のプロセスデータ<$変数（形式 2）>を登録する場合に指定します。

指定値の説明

プロセスデータに登録する値を指定します。

値の形式ごとの指定可否を次の表に示します。

形式	指定可否
組み込み変数	不可
$変数（形式 1）	不可
$変数（形式 2）	可

注意事項

メッセージがキャッチされなかった場合，プロセスデータは登録されません。

関連項目

- 15.3.13　キー名と値で使用されている変数（アプリケーション呼び出し情報ファイル）

(5)　self.pi.<$変数（形式 1）>

キーの説明

呼び出し元案件のプロセスデータ<$変数（形式 1）>を登録する場合に指定します。

指定値の説明

プロセスデータに登録する値を指定します。

値の形式ごとの指定可否を次の表に示します。

形式	指定可否
組み込み変数	可
$変数（形式 1）	可
$変数（形式 2）	不可

注意事項

メッセージがキャッチされなくても，プロセスデータは登録されます。

関連項目

- 15.3.13　キー名と値で使用されている変数（アプリケーション呼び出し情報ファイル）

(6)　self.pi.<$変数（形式 2）>

キーの説明

呼び出し元案件のプロセスデータ<$変数（形式 2）>を登録する場合に指定します。

指定値の説明

プロセスデータに登録する値を指定します。

値の形式ごとの指定可否を次の表に示します。

形式	指定可否
組み込み変数	不可
$変数（形式 1）	不可
$変数（形式 2）	可

注意事項

メッセージがキャッチされなくても，プロセスデータは登録されます。

関連項目

- 15.3.13　キー名と値で使用されている変数（アプリケーション呼び出し情報ファイル）

15.3.18　メッセージによる自案件の呼び出しの場合にファイルに指定する内容

アプリケーション呼び出しサービスは，自案件にある作業のうち，呼び出し元と同じ ref 識別子を作業者 ID に持つ作業を完了させます。そのあとで，呼び出し元の作業も完了させます。

アプリケーション呼び出し情報ファイルの格納先ファイルパス

メッセージによる自案件の呼び出しについて設定する，アプリケーション呼び出し情報ファイルの格納先ファイルパスを次に示します。

```
<BpmnCallInformationFileDirプロパティの指定値>/msg/<messageRefの値>.properties
```

アプリケーション呼び出し情報ファイルの記述例

メッセージによる自案件の呼び出しについて設定する場合の，アプリケーション呼び出し情報ファイルの記述例を次に示します。

```
type=SELF_PI_MESSAGE
self.pi.$SMyFlag=SUCCESS
```

(1)　type（SELF_PI_MESSAGE）

キーの説明

アプリケーション呼び出しの種類を指定します。このキーは指定必須です。

指定値の説明

"SELF_PI_MESSAGE"を指定してください。

アプリケーション呼び出しをする案件に対してメッセージをスローします。

値の形式ごとの指定可否を次の表に示します。

形式	指定可否
組み込み変数	不可
$変数（形式 1）	不可
$変数（形式 2）	不可

注意事項

大文字と小文字は区別します。

関連項目

- 15.3.13　キー名と値で使用されている変数（アプリケーション呼び出し情報ファイル）

(2)　self.pi.<$変数（形式 1）>

キーの説明

呼び出し元案件のプロセスデータ<$変数（形式 1）>を登録する場合に指定します。

指定値の説明

プロセスデータに登録する値を指定します。

値の形式ごとの指定可否を次の表に示します。

形式	指定可否
組み込み変数	可
$変数（形式 1）	可
$変数（形式 2）	不可

注意事項

メッセージがキャッチされなくても，プロセスデータは登録されます。

関連項目

- 15.3.13　キー名と値で使用されている変数（アプリケーション呼び出し情報ファイル）

(3)　self.pi.<$変数（形式 2）>

キーの説明

呼び出し元案件のプロセスデータ<$変数（形式 2）>を登録する場合に指定します。

指定値の説明

プロセスデータに登録する値を指定します。

値の形式ごとの指定可否を次の表に示します。

形式	指定可否
組み込み変数	不可
$変数（形式 1）	不可

形式	指定可否
$変数（形式 2）	可

注意事項

メッセージがキャッチされなくても，プロセスデータは登録されます。

関連項目

- 15.3.13　キー名と値で使用されている変数（アプリケーション呼び出し情報ファイル）

15.3.19　エラーによる自案件の呼び出しの場合にファイルに指定する内容

アプリケーション呼び出しサービスは，自案件にある作業のうち，呼び出し元と同じ ref 識別子を作業者 ID に持つ作業を完了させます。そのあとで，呼び出し元の作業も完了させます。

アプリケーション呼び出し情報ファイルの格納先ファイルパス

エラーによる自案件の呼び出しについて設定する，アプリケーション呼び出し情報ファイルの格納先ファイルパスを次に示します。

```
<BpmnCallInformationFileDirプロパティの指定値>/err/<errorRefの値>.properties
```

アプリケーション呼び出し情報ファイルの記述例

エラーによる自案件の呼び出しについて設定する場合の，アプリケーション呼び出し情報ファイルの記述例を次に示します。

```
type=ERROR
self.pi.$SMyFlag=ERROR
ancestor.pi.$SFromChildFlag=ERROR
```

(1)　type（ERROR）

キーの説明

アプリケーション呼び出しの種類を指定します。

指定値の説明

"ERROR"を指定してください。

アプリケーション呼び出しをする案件に対してエラーをスローします。

値の形式ごとの指定可否を次の表に示します。

形式	指定可否
組み込み変数	不可
$変数（形式 1）	不可
$変数（形式 2）	不可

キー省略時のデフォルト値

ERROR

注意事項

大文字と小文字は区別します。

関連項目

- 15.3.13　キー名と値で使用されている変数（アプリケーション呼び出し情報ファイル）

(2)　self.pi.<$変数（形式 1）>

キーの説明

呼び出し元案件のプロセスデータ<$変数（形式 1）>を登録する場合に指定します。

指定値の説明

プロセスデータに登録する値を指定します。

値の形式ごとの指定可否を次の表に示します。

形式	指定可否
組み込み変数	可
$変数（形式 1）	可
$変数（形式 2）	不可

注意事項

- エラーがキャッチされなくてもプロセスデータは登録されます。
- 親案件に対してエラーをスローした場合，親案件のプロセスデータは登録されないで，呼び出し元案件のプロセスデータだけが登録されます。

関連項目

- 15.3.13　キー名と値で使用されている変数（アプリケーション呼び出し情報ファイル）

(3)　self.pi.<$変数（形式 2）>

キーの説明

呼び出し元案件のプロセスデータ<$変数（形式 2）>を登録する場合に指定します。

指定値の説明

プロセスデータに登録する値を指定します。

値の形式ごとの指定可否を次の表に示します。

形式	指定可否
組み込み変数	不可
$変数（形式 1）	不可
$変数（形式 2）	可

注意事項

- エラーがキャッチされなくてもプロセスデータは登録されます。
- 親案件に対してエラーをスローした場合，親案件のプロセスデータは登録されないで，呼び出し元案件のプロセスデータだけが登録されます。

関連項目

- 15.3.13　キー名と値で使用されている変数（アプリケーション呼び出し情報ファイル）

(4)　ancestor.pi.<$変数（形式 1）>

キーの説明

エラーをキャッチした案件（自案件は除く）のプロセスデータ<$変数（形式 1）>を登録する場合に指定します。

コールアクティビティによって呼び出されるビジネスプロセス定義の「終了（エラー）」でだけ利用できます。コールアクティビティで呼び出されるビジネスプロセス定義以外で指定したときは，プロセスデータは登録されません。

指定値の説明

プロセスデータに登録する値を指定します。

値の形式ごとの指定可否を次の表に示します。

形式	指定可否
組み込み変数	可
$変数（形式 1）	可
$変数（形式 2）	不可

注意事項

なし

関連項目

- 15.3.13　キー名と値で使用されている変数（アプリケーション呼び出し情報ファイル）

(5)　ancestor.pi.<$変数（形式 2）>

キーの説明

エラーをキャッチした案件（自案件は除く）のプロセスデータ<$変数（形式 2）>を登録する場合に指定します。

コールアクティビティで呼び出されるビジネスプロセス定義の「終了（エラー）」でだけ利用できます。コールアクティビティで呼び出されるビジネスプロセス定義以外で指定したときは，プロセスデータは登録されません。

指定値の説明

プロセスデータに登録する値を指定します。

値の形式ごとの指定可否を次の表に示します。

形式	指定可否
組み込み変数	不可
$変数（形式 1）	不可
$変数（形式 2）	可

注意事項

なし

関連項目

- 15.3.13　キー名と値で使用されている変数（アプリケーション呼び出し情報ファイル）

15.3.20　タイマーイベント

タイマーイベントは，アプリケーション呼び出し情報ファイルを使用しません。

15.4　REST アプリケーション呼び出し用ヘッダファイル

REST アプリケーション呼び出し用ヘッダファイルの詳細については，「15.3.6　HTTP ヘッダを記述したファイルの指定方法（アプリケーション呼び出し情報ファイル）」を参照してください。

15.5　コールアクティビティ情報ファイル

コールアクティビティから子案件を投入する際に使用する情報を定義する，コールアクティビティ情報ファイルの詳細について説明します。

15.5.1　コールアクティビティ情報ファイルの概要

コールアクティビティ情報ファイルには，子案件の投入先のビジネスプロセス定義名や，親案件と子案件との間でやり取りするプロセスデータのマッピングを定義します。

15.5.2　コールアクティビティ情報ファイルの設定個所

コールアクティビティ情報ファイルの設定個所について説明します。

コールアクティビティ情報ファイルは，共通設定ファイルのBpmnCallInformationFileDir プロパティで指定したディレクトリ内のcall の下に格納します。

格納パスを示します。

```
<BpmnCallInformationFileDirプロパティの指定値>/call/<calledElementの値>.properties
```

15.5.3　コールアクティビティ情報ファイルの記述規則

コールアクティビティ情報ファイルの記述規則を示します。

- java.util.Properties クラスが扱えるフォーマットで記載してください。
- 文字コードは，UTF-8 を使用してください。ただし，BOM 付き UTF-8 では記述できません。
- プロパティの各キーは，大文字と小文字が区別されます。
- プロパティの各キーは，1 つだけ指定してください（同一のキーは複数指定できません）。同一のキーを複数指定した場合，どの設定値が有効になるかは不定です。
- プロパティの記載順は任意です。また，プロパティが読み込まれる順番は不定です。
- プロパティのキーに，「15.5.9　コールアクティビティ情報ファイルに指定する内容」の説明中で定義されていないキー名を指定した場合，アプリケーションの呼び出し時にエラーが発生します。
- プロパティの値の先頭に"$$"を記載した場合，"$"に変換されます。
- プロパティの値の先頭に"@@"を記載した場合，"@"に変換されます。
- プロパティの値が空の場合，エラーが発生します。

15.5.4　プロセスデータの使用方法（コールアクティビティ情報ファイル）

コールアクティビティ情報ファイルでのプロセスデータの使用方法について説明します。

コールアクティビティ情報ファイルでプロセスデータを使用すると，次に示す処理ができます。

- 子案件の投入時に，子案件プロセスデータを作成する
- 子案件の投入時に，親案件プロセスデータを参照する
- 子案件の完了時に，親案件プロセスデータを作成または更新する
- 子案件の完了時に，子案件プロセスデータを参照する

コールアクティビティ情報ファイルでのプロセスデータの使用規則を次に示します。

- プロセスデータキー名の前後に，プロセスデータキー名，組み込み変数，またはその他の文字列を連続して記載することはできません。
- 1 つのコールアクティビティ情報ファイルにはプロセスデータを 255 個まで含めることができます。256 個以上のプロセスデータは指定できません。
- プロパティキーが"child."で始まるプロパティの値にプロセスデータキー名を指定した場合，親案件のプロセスデータを表します。プロパティ値に指定したプロセスデータが親案件に存在しない場合は，子案件の投入時にエラーが発生します。
- プロパティキーが"parent."で始まるプロパティの値にプロセスデータキー名を指定した場合，子案件のプロセスデータを表します。プロパティ値に指定したプロセスデータが子案件に存在しない場合は，子案件の完了時にエラーが発生します。
- プロパティ値に指定したプロセスデータの値がnull の場合，null 値として扱います。
- リスト型プロセスデータ間の代入をした場合（プロパティキーとプロパティ値の両方にリスト型プロセスデータの全要素を示すプロセスデータキー名を指定した場合），更新対象（プロパティキー側）のリスト型プロセスデータのリスト内識別子は，順番はそのままで 1 から連番で付与されます。参照元（プロパティ値側）のリスト内識別子が連番ではない場合，更新対象（プロパティキー側）のリスト内識別子は，参照元（プロパティ値側）のリスト内識別子と一致しません。

15.5.5　コールアクティビティ情報ファイルで使用できる組み込み変数

組み込み変数は，CSCIW で管理する案件や作業の属性値を指定するための変数です。コールアクティビティ情報ファイルに記載した組み込み変数は，案件や作業の属性値に変換されます。

コールアクティビティ情報ファイルでの組み込み変数の使用規則を次に示します。

- 組み込み変数名の前後に，ほかの組み込み変数名やその他の文字列を連続して記載することはできません。
- プロパティキーが"child."で始まるプロパティの値に組み込み変数を指定した場合，親案件の組み込み変数を表します。指定できる組み込み変数を次の表で示します。

表 15-6　指定できる組み込み変数（子案件の投入時）

項番	組み込み変数	説明
1	@PIID	案件 ID
2	@PIName	案件名
3	@PICreator	案件投入者
4	@PIDeadline	案件処理期限
5	@PDefName	ビジネスプロセス定義名
6	@WIID	作業 ID
7	@WDefName	作業定義名
8	@NULL	null 値

- プロパティキーが"parent."で始まるプロパティの値に組み込み変数を指定した場合，子案件の組み込み変数を表します。指定できる組み込み変数を次の表で示します。

表 15-7　指定できる組み込み変数（子案件の完了時）

項番	組み込み変数	説明
1	@PIID	案件 ID
2	@PIName	案件名
3	@PICreator	案件投入者
4	@PIDeadline	案件処理期限
5	@PDefName	ビジネスプロセス定義名
6	@NULL	null 値

- プロパティの値の先頭に"@"を記載し，「表 15-6　指定できる組み込み変数（子案件の投入時）」または「表 15-7　指定できる組み込み変数（子案件の完了時）」に記載されていない文字列を指定した場合，エラーが発生します。
- 組み込み変数の対象となる属性の値がnull の場合，null 値として扱います。
- 案件処理期限は，"1970/01/01 00:00:00GMT"を起点とした通算秒に置き換わります。代入先には，文字列型プロセスデータを指定してください。数値型プロセスデータを指定した場合，案件処理期限に"2038/01/19 03:14:07 GMT"より先の日時を指定するとエラーになります。

15.5.6　コールアクティビティ情報ファイルの読み込みタイミング

コールアクティビティ情報ファイルの読み込みタイミングについて説明します。

コールアクティビティ情報ファイルは，コールアクティビティに遷移して子案件を投入する際に，読み込みキャッシュします。再読み込みに関しては，「9.2　ファイルに格納した設定情報を変更する」のコールアクティビティ情報ファイルの内容を変更した場合についての記述を参照してください。

15.5.7　コールアクティビティ情報ファイルに指定するパラメタの一覧

指定するパラメタの一覧（コールアクティビティ情報ファイル）

コールアクティビティ情報ファイルに指定するパラメタの一覧を示します。

表 15-8　指定するパラメタの一覧（コールアクティビティ情報ファイル）

項番	キー名	指定内容
1	child.bp.name	子案件のビジネスプロセス定義名
2	child.bp.version	ビジネスプロセスのバージョン
3	child.pi.name	子案件の案件名
4	child.pi.<$変数（形式1）>	子案件のプロセスデータに登録する値
5	child.pi.<$変数（形式2）>	子案件のプロセスデータに登録する値
6	parent.pi.<$変数（形式1）>	親案件のプロセスデータに登録する値
7	parent.pi.<$変数（形式2）>	親案件のプロセスデータに登録する値

15.5.8　キー名と値で使用されている変数（コールアクティビティ情報ファイル）

「表 15-8　指定するパラメタの一覧（コールアクティビティ情報ファイル）」内のキー名と値で使用されている変数の意味を，次の表に示します。

表 15-9　キー名と値で使用されている変数（コールアクティビティ情報ファイル）

項番	変数	説明
1	<$変数（形式1）>	1. 単一型プロセスデータのプロセスデータキー名（"{"および"}"を含まない） 指定例 $Skey 2. リスト型プロセスデータの 1 要素を表すプロセスデータキー名（末尾が"{<リスト内識別子>}"で終わる） 指定例 $Skey{5}

項番	変数	説明
2	<$変数（形式2）>	リスト型プロセスデータの全要素を表すプロセスデータキー名（末尾が"{}"で終わる） 指定例 $Skey{}

15.5.9　コールアクティビティ情報ファイルに指定する内容

コールアクティビティ情報ファイルに指定するパラメタについて説明します。

(1)　child.bp.name

キーの説明

子案件のビジネスプロセス定義名を指定します。このキーは指定必須です。

指定値の説明

ビジネスプロセス定義名を指定します。

値の形式ごとの指定可否を次の表に示します。

形式	指定可否
組み込み変数	可
$変数（形式 1）	可
$変数（形式 2）	不可

注意事項

なし

関連項目

- 15.5.8　キー名と値で使用されている変数（コールアクティビティ情報ファイル）

(2)　child.bp.version

キーの説明

ビジネスプロセスのバージョンを指定します。

指定を省略した場合，child.bp.name で指定したビジネスプロセス定義の中で投入できる最新のバージョンが使用されます。

指定値の説明

子案件のビジネスプロセス定義のバージョンを整数で指定します。

null 値（組み込み変数値またはプロセスデータ値が null）を指定した場合

child.bp.name で指定したビジネスプロセス定義の中で投入可能な最新のバージョンが使用されます。

値の形式ごとの指定可否を次の表に示します。

形式	指定可否
組み込み変数	可
$変数（形式 1）	可
$変数（形式 2）	不可

注意事項

なし

関連項目

- 15.5.8　キー名と値で使用されている変数（コールアクティビティ情報ファイル）

(3)　child.pi.name

キーの説明

子案件の案件名を指定します。

指定を省略した場合，<親案件 ID>_<yyyymmddHHMMss 形式の時刻>になります。

指定値の説明

子案件の案件名を指定します。

マルチインスタンスのコールアクティビティでリスト型プロセスデータの 1 要素を参照する場合

末尾が"{@MIIndex}"のプロセスデータキー名をプロパティ値に指定してください。

値の形式ごとの指定可否を次の表に示します。

形式	指定可否
組み込み変数	可

形式	指定可否
$変数（形式 1）	可
$変数（形式 2）	不可

注意事項

なし

関連項目

- 15.5.8　キー名と値で使用されている変数（コールアクティビティ情報ファイル）

(4)　child.pi.<$変数（形式 1）>

キーの説明

子案件のプロセスデータ<$変数（形式 1）>を登録する場合に指定します。

指定値の説明

プロセスデータに登録する値を指定します。

記載例

```
child.pi.$Sparam=aaa
```

プロセスデータキー名，組み込み変数を指定した場合

親案件の変数値で，子案件のプロセスデータを作成します。

記載例

```
child.pi.$Sparam=$Svalue
```

マルチインスタンスのコールアクティビティでリスト型プロセスデータの 1 要素を参照する場合

末尾が"{@MIIndex}"のプロセスデータキー名をプロパティ値に指定してください。

値の形式ごとの指定可否を次の表に示します。

形式	指定可否
組み込み変数	可
$変数（形式 1）	可
$変数（形式 2）	不可

注意事項

なし

関連項目

- 15.5.8　キー名と値で使用されている変数（コールアクティビティ情報ファイル）

(5)　child.pi.<$変数（形式 2）>

キーの説明

子案件のプロセスデータ<$変数（形式 2）>を登録する場合に指定します。

指定値の説明

プロセスデータに登録する値を指定します。

親案件のリスト型プロセスデータ値で，子案件のプロセスデータを作成します。

記載例

```
child.pi.$Slist{}=$Svalue{}
```

値の形式ごとの指定可否を次の表に示します。

形式	指定可否
組み込み変数	不可
$変数（形式 1）	不可
$変数（形式 2）	可

注意事項

なし

関連項目

- 15.5.8　キー名と値で使用されている変数（コールアクティビティ情報ファイル）

(6)　parent.pi.<$変数（形式 1）>

キーの説明

親案件のプロセスデータ<$変数（形式 1）>を登録する場合に指定します。

マルチインスタンスのコールアクティビティでリスト型プロセスデータの 1 要素を登録する場合

末尾が"{@MIIndex}"のプロセスデータキー名を，プロパティキーの<$変数（形式 1）>の部分に指定してください。

指定値の説明

プロセスデータに登録する値を指定します。

記載例

```
parent.pi.$Sresult=success
```

プロセスデータキー名，組み込み変数を指定した場合

子案件の変数値で，親案件のプロセスデータを登録します。

記載例

```
parent.pi.$Sresult=$Svalue
```

値の形式ごとの指定可否を次の表に示します。

形式	指定可否
組み込み変数	可
$変数（形式 1）	可
$変数（形式 2）	不可

注意事項

なし

関連項目

- 15.5.8　キー名と値で使用されている変数（コールアクティビティ情報ファイル）

(7)　parent.pi.<$変数（形式 2）>

キーの説明

親案件のプロセスデータ<$変数（形式 2）>を登録する場合に指定します。

指定値の説明

プロセスデータに登録する値を指定します。

子案件のリスト型プロセスデータ値で，親案件のプロセスデータを登録します。

記載例

```
parent.pi.$Sresult{}=$List{}
```

値の形式ごとの指定可否を次の表に示します。

形式	指定可否
組み込み変数	不可
$変数（形式 1）	不可
$変数（形式 2）	可

注意事項

なし

関連項目

- 15.5.8　キー名と値で使用されている変数（コールアクティビティ情報ファイル）

15.6　REST アプリケーション呼び出しスキーマ変換用スタイルシート

REST アプリケーション呼び出しスキーマ変換用スタイルシートについては，「6.1.5　XML 形式のボディデータのスキーマ変換」を参照してください。

15.7　uCosminexus Business Process Developer 設定ファイル

uCosminexus Business Process Developer 設定ファイルには，BPMN ビジネスプロセス定義ファイルの変換コマンド（ciwtransbpmn）のメッセージやトレースの出力先ディレクトリ，ファイルサイズなどの情報を定義します。同一のマシンに CSCIW がインストールされていても，変換コマンドのメッセージおよびトレースの設定は uCosminexus Business Process Developer 設定ファイルを使用します。

uCosminexus Business Process Developer 設定ファイルの設定個所

uCosminexus Business Process Developer 設定ファイルは，次のパスに格納してください。

```
<uCosminexus Business Process Developerのインストールディレクトリ>¥conf¥ciwbpdev.properties
```

uCosminexus Business Process Developer 設定ファイルの設定内容

uCosminexus Business Process Developer 設定ファイルの設定内容を次に示します。

表 15-10　uCosminexus Business Process Developer 設定ファイルの設定内容

キー※1	内容	設定
CmdMsgFileDir	変換コマンドメッセージファイル出力先ディレクトリ 変換コマンドメッセージファイルの出力先ディレクトリを指定します。 デフォルトの出力先ディレクトリは次のとおりです。 <uCosminexus Business Process Developerのインストールディレクトリ>¥log	任意
CmdMsgFileNum	変換コマンドメッセージファイル出力面数 変換コマンドメッセージファイルの出力面数を，1～16 の整数で指定します。ただし，ローテーション種別がシフトモードの場合は，バックアップ面数を指定します。 デフォルト値は2 です。	任意
CmdMsgFileSize	変換コマンドメッセージファイル出力サイズ 変換コマンドメッセージファイルの出力サイズを，100000～2147483647 の整数で指定します。（単位：バイト） デフォルト値は2097152 です。	任意
CmdMsgOutputThreashold	変換コマンドメッセージ出力レベル 変換コマンドメッセージ出力レベルを，-1～1000 の数値で指定します。 デフォルト値は20 です。 • 出力しない場合：-1 を指定 • 値以下のレベルを出力する場合：0 以上の値を指定	任意
CmdMsgTimeToDelete	変換コマンドメッセージファイル削除までの日数 変換コマンドメッセージファイルを削除するまでの日数を，-1～24855 の数値で指定します。 ローテーション種別がラップアラウンドモードの場合だけ有効になります。 デフォルト値は-1 です。	任意

キー※1	内容	設定
CmdMsgTimeToDelete	• 削除しない場合：-1 を指定 • 常に削除する場合：0 を指定※2 • 経過日数で削除する場合：1 以上の値を指定	任意
CmdTraceFileDir	変換コマンドトレースファイル出力先ディレクトリ 変換コマンドトレースファイルの出力先ディレクトリを指定します。 デフォルトの出力先ディレクトリは次のとおりです。 <uCosminexus Business Process Developerのインストールディレクトリ>¥log	任意
CmdTraceFileNum	変換コマンドトレースファイル出力面数 変換コマンドトレースファイルの出力面数を，1～16 の整数で指定します。ただし，ローテーション種別がシフトモードの場合は，バックアップ面数を指定します。 デフォルト値は4 です。	任意
CmdTraceFileSize	変換コマンドトレースファイル出力サイズ 変換コマンドトレースファイルの出力サイズを，100000～2147483647 の整数で指定します。(単位：バイト) デフォルト値は16777216 です。	任意
CmdTraceOutputThreashold	変換コマンドトレース出力レベル 変換コマンドトレース出力レベルを，-1～1000 の数値で指定します。 デフォルト値は20 です。 • 出力しない場合：-1 を指定 • 値以下のレベルを出力する場合：0 以上の値を指定	任意
CmdTraceTimeToDelete	変換コマンドトレースファイル削除までの日数 変換コマンドトレースファイルを削除するまでの日数を，-1～24855 の数値で指定します。 ローテーション種別がラップアラウンドモードの場合だけ有効になります。 デフォルト値は-1 です。 • 削除しない場合：-1 を指定 • 常に削除する場合：0 を指定※2 • 経過日数で削除する場合：1 以上の値を指定	任意

注※1

記載しているキー以外の内容を指定した場合，指定したキーおよび値は無視されます。

注※2

0 を指定した場合，複数のプロセスを実行したときに，ほかのプロセスのファイルも削除するので注意してください。

16 障害対策

ワークフローシステムに障害が発生した場合の対処方法について説明します。

16.1　障害情報の取得

CSCIW の障害対策については，マニュアル『uCosminexus Service Coordinator Interactive Workflow システム構築・運用ガイド』の「8.　障害対策」を参照してください。CSCIW のトレースファイル，およびメッセージファイルについては，マニュアル『uCosminexus Service Coordinator Interactive Workflow システム構築・運用ガイド』の「8.2.1　CSCIW の出力情報の取得」を参照してください。

ここでは，BPMN 連携機能のトレースファイル，およびメッセージファイルについて説明します。

マニュアル『uCosminexus Service Coordinator Interactive Workflow メッセージ』やマニュアル『uCosminexus Service Coordinator Interactive Workflow コマンド』を参照する場合で，参照先に CSCIW-Definer の記述があるときは，マニュアル内の「CSCIW-Definer」の表記を「BPMN エディタ」に置き換えてお読みください。

トレースファイルおよびメッセージファイルのファイル名

BPMN 連携ライブラリには，トレース・メッセージ機能があります。

トレース・メッセージ機能を使用して，次のサービスのログを出力します。

1. REST サービス
2. アプリケーション呼び出しサービス

業務プログラムから呼び出された BPMN 連携ライブラリ内の処理中に発生するトレース・メッセージ自体は，CSCIW 本体のトレース・メッセージ機能を使用してログを出力します。ログは，CSCIW 本体と同一のファイルに出力されます。

REST サービスおよびアプリケーション呼び出しサービスから呼び出された BPMN 連携ライブラリ内の処理中に発生するトレースは，各サービス用のトレースファイルにログが出力されます。メッセージは，CSCIW 本体と同一のファイルにログが出力されます。

BPMN 連携ライブラリのトレース・メッセージ機能の出力先ファイルパス，ログレベル，ログの出力面数，ログの出力サイズ，およびファイル削除までの日数は，BPMN 連携機能の共通設定ファイルで設定します。詳細は，「15.2　共通設定ファイル」を参照してください。

BPMN 連携ライブラリのトレース・メッセージ機能によって出力される，トレースファイルおよびメッセージファイルのファイル名の一覧を表に示します。

表 16-1　トレースファイルおよびメッセージファイルのファイル名一覧

対象	出力種別	ローテーション種別	ファイル名※
REST サービス	トレース	ラップアラウンド	RESTSVC_<システム ID>_<ロック ID>_TRC_<通番>.log

対象	出力種別	ローテーション種別	ファイル名※
REST サービス	トレース	シフト	RESTSVC_<システム ID>_<ロック ID>_TRC<通番>.log
	メッセージ	ラップアラウンド	RESTSVC_<システム ID>_<ロック ID>_MSG_<通番>.log
		シフト	RESTSVC_<システム ID>_<ロック ID>_MSG<通番>.log
アプリケーション呼び出しサービス	トレース	ラップアラウンド	APPCALLSVC_<システム ID>_<アプリケーション番号>_<ロック ID>_TRC_<通番>.log
		シフト	APPCALLSVC_<システム ID>_<アプリケーション番号>_<ロック ID>_TRC<通番>.log
	メッセージ	ラップアラウンド	APPCALLSVC_<システム ID>_<アプリケーション番号>_<ロック ID>_MSG_<通番>.log
		シフト	APPCALLSVC_<システム ID>_<アプリケーション番号>_<ロック ID>_MSG<通番>.log

注※

- ロック ID

  OS のプロセス間で排他を取るためのロック ID が入ります。

- アプリケーション番号

  J2EE サーバにインポートしたアプリケーション呼び出しサービスの J2EE アプリケーション名の，末尾の番号が入ります。詳細については，「付録 C.6　チューニングの設定方法」の「(2)　アプリケーション呼び出しサービスの稼働数の設定」を参照してください。

17 ワークフローシステムをバージョンアップする

ワークフローシステムのバージョンアップ方法について説明します。バージョンアップ前から後への移行方法には，既存のワークフローシステムを引き継ぐ方法と，ワークフローシステムを新規に構築する方法があります。どちらの方法もワーク管理データベースに保持するデータ（案件，ビジネスプロセス定義，環境設定）を引き継ぎます。

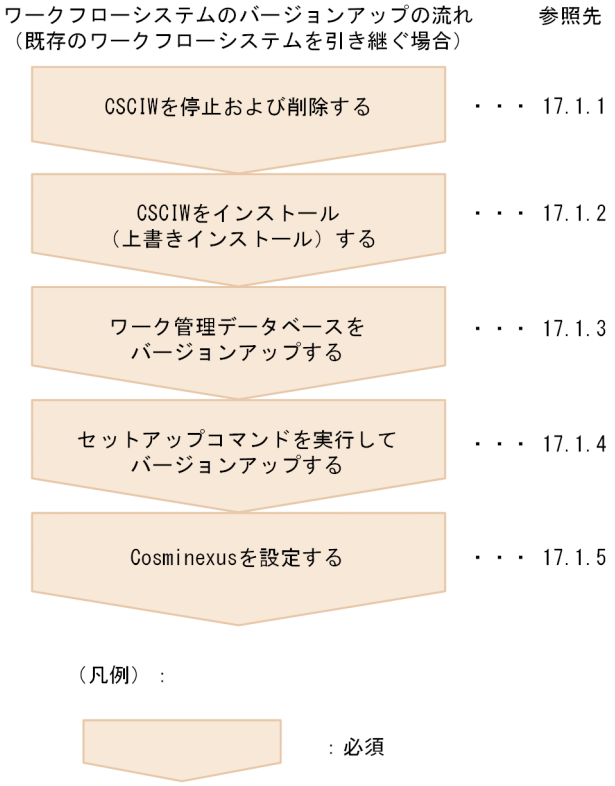
17.1　ワークフローシステムをバージョンアップする（既存のワークフローシステムを引き継ぐ場合）

既存のワークフローシステムを引き継ぐ場合の，ワークフローシステムのバージョンアップ手順を説明します。

ワークフローシステムのバージョンアップの流れ（既存のワークフローシステムを引き継ぐ場合）を次の図で示します。

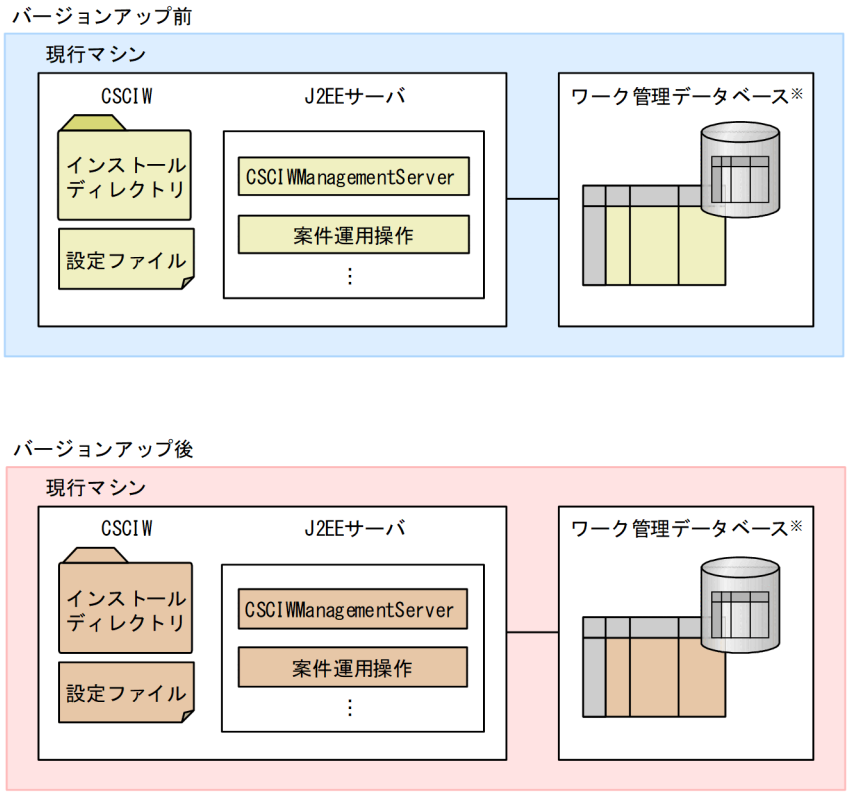
なお，図中の数字は，対応する内容を説明している項の番号を表しています。

図 17-1　ワークフローシステムのバージョンアップの流れ（既存のワークフローシステムを引き継ぐ場合）



既存のワークフローシステムを引き継ぐ場合の概念図を次に示します。

図 17-2　ワークフローシステムのバージョンアップ（既存のワークフローシステムを引き継ぐ場合）



関連項目

- 17.1.1　CSCIW を停止および削除する
- 17.1.2　CSCIW をインストール（上書きインストール）する
- 17.1.3　ワーク管理データベースをバージョンアップする
- 17.1.4　セットアップコマンドを実行してバージョンアップする
- 17.1.5　Cosminexus を設定する

17.1.1　CSCIW を停止および削除する

CSCIW の停止および削除について説明します。

(1)　アプリケーション呼び出しサービスを停止および削除する

アプリケーション呼び出しサービスを停止および削除します。

関連項目

- 8.6　アプリケーション呼び出しサービスを停止および削除する

(2)　業務アプリケーションを停止する

CSCIW を使用している Java アプリケーションを停止します。また，J2EE サーバ上で実行している J2EE アプリケーションも停止します。J2EE アプリケーションは次のどちらかの手順で停止します。

運用管理ポータルを使用する場合

1. 運用管理ポータルにログインする。
2. 運用管理ポータルで［論理サーバのアプリケーション管理］アンカーをクリックする。
3. ツリーペインで次の順にクリックする。
   ［<操作対象の運用管理ドメイン名>］－［論理 J2EE サーバ］－［J2EE サーバ］－［<操作対象の J2EE サーバ名>］－［アプリケーション］
4. 右ペインの［開始／停止］タブをクリックし，［J2EE アプリケーションの開始／停止］画面を表示する。
5. 停止する業務アプリケーションの［停止］アンカーをクリックする。
6. 内容を確認して，［はい］ボタンをクリックする。

サーバ管理コマンドを使用する場合

1. cjstopapp コマンドを使用して，J2EE サーバ上で実行している J2EE アプリケーションを停止します。
   J2EE アプリケーションの停止方法については，マニュアル『Cosminexus 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「J2EE アプリケーションの停止」の説明を参照してください。
   cjstopapp コマンドの指定形式を次に示します。

   ```
   cjstopapp 〈サーバ名称〉 -name 〈業務アプリケーション名〉
   ```

   cjstopapp コマンドの詳細については，マニュアル『Cosminexus アプリケーションサーバ　リファレンス コマンド編』を参照してください。

(3)　案件運用操作を停止および削除する

案件運用操作を停止および削除します。

関連項目

- 8.3　案件運用操作を停止および削除する

(4)　ビジネスプロセスオペレータを停止および削除する

ビジネスプロセスオペレータを使用している場合，ビジネスプロセスオペレータを停止および削除します。

関連項目

- 8.4　ビジネスプロセスオペレータを停止および削除する

(5)　REST サービスを停止および削除する

REST サービスを使用している場合，REST サービスを停止および削除します。

関連項目

- 8.5　REST サービスを停止および削除する

(6)　CSCIWManagementServer を停止および削除する

CSCIWManagementServer を停止および削除します。

関連項目

- 8.7　CSCIWManagementServer を停止および削除する

(7)　J2EE サーバを停止する

CSCIW を使用している J2EE サーバを停止します。

関連項目

- 8.9　J2EE サーバを停止する

17.1.2　CSCIW をインストール（上書きインストール）する

最新の CSCIW をインストール（上書きインストール）してください。CSCIW のインストールについては，マニュアル『uCosminexus Service Coordinator Interactive Workflow システム構築・運用ガイド』の「4.2.2　CSCIW のインストール」を参照してください。

メモ

バージョンアップ後，Cosminexus J2EE サーバの V9 互換モードを使用する場合，CSCIW を上書きインストールしたあとに，マニュアル『uCosminexus Service Coordinator Interactive Workflow システム構築・運用ガイド』の「付録 J.1　互換モード用のファイルへの入れ替え」を参照して，ファイルを入れ替えてください。

17.1.3　ワーク管理データベースをバージョンアップする

CSCIW をバージョンアップする場合に必要な，ワーク管理データベースのバージョンアップ手順を示します。

ワーク管理データベースをバージョンアップするときは，次に示す作業を実施してください。また，テーブルやインデクスは，SQL スクリプトファイルを使用して追加および更新してください。

SQL スクリプトファイルはバージョンごとに段階的に実行する必要があります。例えば，CSCIW の 02-10 から 03-30 へバージョンアップする場合，次の順序で実行します。

1. 02-10 から 02-20 へ移行する SQL スクリプトファイルを編集および実行
2. 02-20 から 02-30 へ移行する SQL スクリプトファイルを編集および実行
3. 02-30 から 03-00 へ移行する SQL スクリプトファイルを編集および実行
4. 03-00 から 03-10 へ移行する SQL スクリプトファイルを編集および実行
5. 03-10 から 03-11 へ移行する SQL スクリプトファイルを編集および実行
6. 03-11 から 03-20 へ移行する SQL スクリプトファイルを編集および実行
7. 03-20 から 03-30 へ移行する SQL スクリプトファイルを編集および実行

(1)　ワーク管理データベースをバージョンアップする（HiRDB の場合）

使用しているワーク管理データベースが HiRDB の場合の手順を示します。

操作手順

1. 次に示す SQL スクリプトファイルを編集する。
   - 02-10 から 02-20 へ移行する場合

     ```
     <CSCIWインストールディレクトリ>/sql/vup_0210to0220_hirdb.sql
     ```

     ```
     <CSCIWインストールディレクトリ>/sql/vupex_0210to0220_hirdb.sql
     ```

   - 02-20 から 02-30 へ移行する場合

     ```
     <CSCIWインストールディレクトリ>/sql/vup_0220to0230_hirdb.sql
     ```

     ```
     <CSCIWインストールディレクトリ>/sql/vupex_0220to0230_hirdb.sql
     ```

   - 02-30 から 03-00 へ移行する場合

     ```
     <CSCIWインストールディレクトリ>/sql/vup_0230to0300_hirdb.sql
     ```

   - 03-00 から 03-10 へ移行する場合

     ```
     <CSCIWインストールディレクトリ>/sql/vup_0300to0310_hirdb.sql
     ```

```
<CSCIWインストールディレクトリ>/sql/vupex_0300to0310_hirdb.sql
```

- 03-10 から 03-11 へ移行する場合

```
<CSCIWインストールディレクトリ>/sql/vup_0310to0311_hirdb.sql
```

- 03-11 から 03-20 へ移行する場合

```
<CSCIWインストールディレクトリ>/sql/vup_0311to0320_hirdb.sql
```

- 03-20 から 03-30 へ移行する場合

```
<CSCIWインストールディレクトリ>/sql/vup_0320to0330_hirdb.sql
```

SQL スクリプトファイルの書き換えが必要な文字列，および書き換える内容の詳細については，ワーク管理データベースを作成するときと同じです。

2. SQL スクリプトファイルを実行する。

編集した SQL スクリプトファイルの実行方法については，ワーク管理データベースを作成するときと同じです。

3. ワーク管理データベースへのアクセス権限を付与する。

ワーク管理データベースを移行する前にアクセス権限を付与していた場合，アクセス権限を付与し直す必要があります。

関連項目

- 7.2　ワーク管理データベースを作成する（HiRDB の場合）
- 7.5　データベースへのアクセス権限を付与する（HiRDB の場合）

(2)　ワーク管理データベースをバージョンアップする（ORACLE の場合）

使用しているワーク管理データベースが ORACLE の場合の手順を示します。

操作手順

1. 次に示す SQL スクリプトファイルを編集する。

- 02-30 から 03-00 へ移行する場合

```
<CSCIWインストールディレクトリ>/sql/vup_0230to0300_oracle.sql
```

- 03-00 から 03-10 へ移行する場合

```
<CSCIWインストールディレクトリ>/sql/vup_0300to0310_oracle.sql
```

```
<CSCIWインストールディレクトリ>/sql/vupex_0300to0310_oracle.sql
```

- 03-10 から 03-11 へ移行する場合

```
<CSCIWインストールディレクトリ>/sql/vup_0310to0311_oracle.sql
```

- 03-11 から 03-20 へ移行する場合

```
<CSCIWインストールディレクトリ>/sql/vup_0311to0320_oracle.sql
```

- 03-20 から 03-30 へ移行する場合

```
<CSCIWインストールディレクトリ>/sql/vup_0320to0330_oracle.sql
```

SQL スクリプトファイルの書き換えが必要な文字列，および書き換える内容の詳細については，ワーク管理データベースを作成するときと同じです。

2. SQL スクリプトファイルを実行する。

編集した SQL スクリプトファイルの実行方法については，ワーク管理データベースを作成するときと同じです。

3. ワーク管理データベースへのアクセス権限を付与する。

ワーク管理データベースを移行する前にアクセス権限を付与していた場合，アクセス権限を付与し直す必要があります。

関連項目

- 7.3　ワーク管理データベースを作成する（ORACLE の場合）
- 7.6　データベースへのアクセス権限を付与する（ORACLE の場合）

(3)　ワーク管理データベースをバージョンアップする（PostgreSQL の場合）

使用しているワーク管理データベースが PostgreSQL の場合の手順を示します。

操作手順

1. 次に示す SQL スクリプトファイルを編集する。

- 02-30 から 03-00 へ移行する場合

```
<CSCIWインストールディレクトリ>/sql/vup_0230to0300_postgresql.sql
```

- 03-00 から 03-10 へ移行する場合

```
<CSCIWインストールディレクトリ>/sql/vup_0300to0310_postgresql.sql
```

```
<CSCIWインストールディレクトリ>/sql/vupex_0300to0310_postgresql.sql
```

- 03-10 から 03-11 へ移行する場合

```
<CSCIWインストールディレクトリ>/sql/vup_0310to0311_postgresql.sql
```

- 03-11 から 03-20 へ移行する場合

```
<CSCIWインストールディレクトリ>/sql/vup_0311to0320_postgresql.sql
```

- 03-20 から 03-30 へ移行する場合

```
<CSCIWインストールディレクトリ>/sql/vup_0320to0330_postgresql.sql
```

SQL スクリプトファイルの書き換えが必要な文字列，および書き換える内容の詳細については，ワーク管理データベースを作成するときと同じです。

2. SQL スクリプトファイルを実行する。

編集した SQL スクリプトファイルの実行方法については，ワーク管理データベースを作成するときと同じです。

3. ワーク管理データベースへのアクセス権限を付与する。

ワーク管理データベースを移行する前にアクセス権限を付与していた場合，アクセス権限を付与し直す必要があります。

関連項目

- 7.4　ワーク管理データベースを作成する（PostgreSQL の場合）
- 7.7　データベースへのアクセス権限を付与する（PostgreSQL の場合）

17.1.4　セットアップコマンドを実行してバージョンアップする

CSCIW をバージョンアップする場合に必要な，セットアップコマンドの設定方法を示します。

(1)　環境変数を設定する

COSMINEXUS_HOME 環境変数およびCSCIW_HOME 環境変数を設定します。

環境変数に設定する内容は，CSCIW の実行環境を初期化する場合と同じです。バージョンアップ前と同じ内容を設定してください。

関連項目

- 7.8　CSCIW の実行環境を初期化する

(2)　セットアッププロパティファイルを設定する

セットアッププロパティファイルを設定します。

セットアッププロパティファイルに設定する内容は，CSCIW の実行環境を初期化する場合と同じです。バージョンアップ前と同じ内容を設定してください。

関連項目

- 7.8　CSCIW の実行環境を初期化する

(3)　コマンド用環境設定ファイルを設定する

コマンド用環境設定ファイルを設定します。

コマンド用環境設定ファイルに設定する内容は，CSCIW の実行環境を初期化する場合と同じです。バージョンアップ前と同じ内容を設定してください。

関連項目

- 7.8　CSCIW の実行環境を初期化する

(4)　コマンドを実行して実行環境をバージョンアップする

CSCIW の実行環境は，ciwsetenv（環境の構築または削除）コマンドを実行してバージョンアップします。なお，ciwsetenv（環境の構築または削除）コマンドは，すべてのマシンで実行してください。

次に示す形式でコマンドを実行すると，システム設定プロパティファイルの内容は，指定した環境構築ファイルの内容に書き換えられます。なお，環境構築ファイルに設定する項目「SystemDBPassword」および「SystemDBURL」については，同じワーク管理データベースに接続できる範囲で，値を変更できます。

```
ciwsetenv -sid 〈システムID〉 -vup 〈環境構築ファイル名〉
```

ciwsetenv（環境の構築または削除）コマンドについては，マニュアル『uCosminexus Service Coordinator Interactive Workflow コマンド』を参照してください。

関連項目

- 7.8　CSCIW の実行環境を初期化する

17.1.5　Cosminexus を設定する

Cosminexus を設定します。

(1)　コンテナ拡張ライブラリに JAR ファイルを取り込む

コンテナ拡張ライブラリに jar を追加します。

メモ

バージョンアップ後，Cosminexus J2EE サーバの V9 互換モードを使用する場合，コンテナ拡張ライブラリに取り込むライブラリの変更が必要です。

関連項目

- 7.9.1　コンテナ拡張ライブラリに JAR ファイルを取り込む

(2)　CSCIWManagementServer を再設定する

CSCIWManagementServer を再設定します。

関連項目

- 7.9.5　CSCIWManagementServer に関する設定をする

(3)　REST サービスを再設定する

REST サービスを再設定します。

関連項目

- 7.9.7　REST サービスに関する設定をする

(4)　ビジネスプロセスオペレータを再設定する

ビジネスプロセスオペレータを再設定します。

関連項目

- 7.9.8　ビジネスプロセスオペレータに関する設定をする

(5)　案件運用操作を再設定する

案件運用操作を再設定します。

関連項目

- 7.9.9　案件運用操作に関する設定をする

(6)　アプリケーション呼び出しサービスを再設定する

アプリケーション呼び出しサービスを再設定します。

関連項目

- 7.9.6　アプリケーション呼び出しサービスに関する設定をする

17.2　ワークフローシステムをバージョンアップする（ワークフローシステムを新規に構築する場合）

ワークフローシステムを新規に構築する場合の，ワークフローシステムのバージョンアップ手順を説明します。

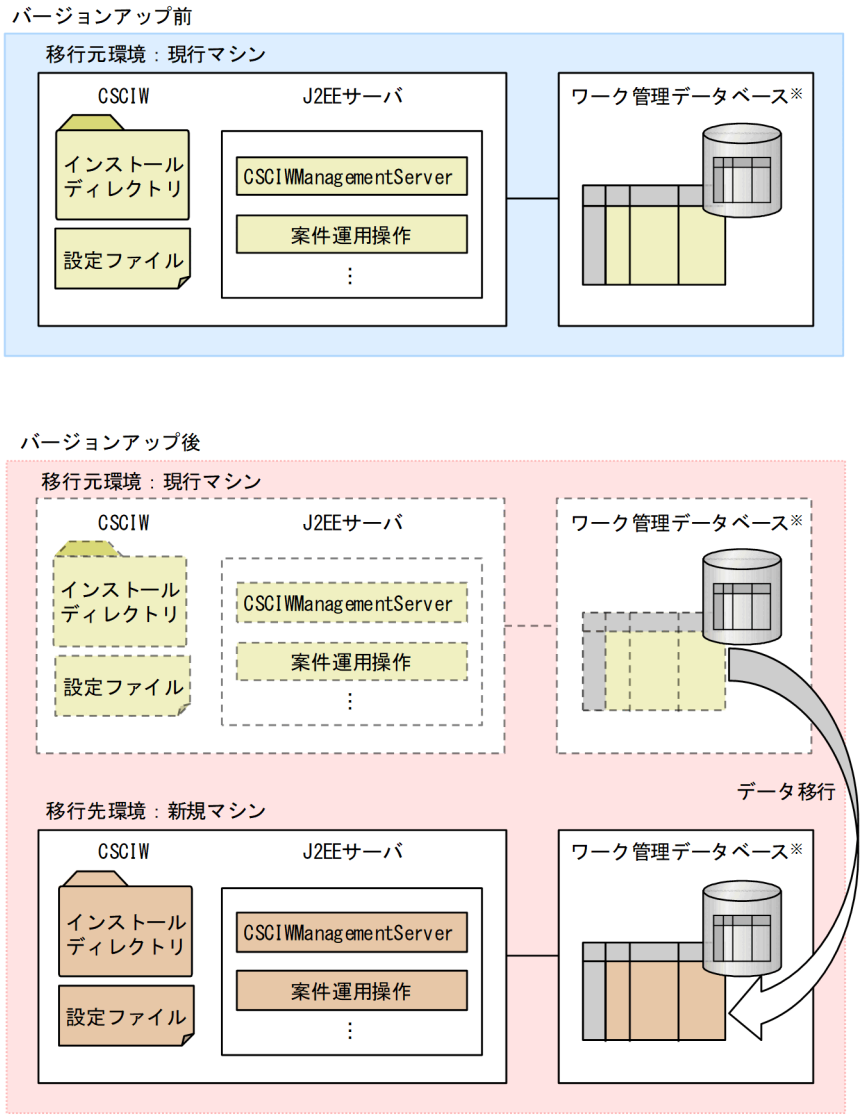
ワークフローシステムのバージョンアップの流れ（ワークフローシステムを新規に構築する場合）を次の図で示します。

なお，図中の数字は，対応する内容を説明している項の番号を表しています。

図 17-3　ワークフローシステムのバージョンアップの流れ（ワークフローシステムを新規に構築する場合）

移行先環境（新規マシン）に新しいバージョンの CSCIW をインストールして，ワークフローシステムを新規に構築する場合の概念図を次に示します。

図 17-4　ワークフローシステムのバージョンアップ（ワークフローシステムを新規に構築する場合）



関連項目

- 17.2.1　移行元環境の CSCIW を停止する
- 17.2.2　CSCIW をインストール（新規インストール）する
- 17.2.3　ワーク管理データベースを構築する
- 17.2.4　ワーク管理データベースのデータを移行する
- 17.2.5　ワーク管理データベースをバージョンアップする
- 17.2.6　セットアップコマンドを実行してバージョンアップする

- 17.2.7　Cosminexus を設定する

17.2.1　移行元環境の CSCIW を停止する

移行元環境の CSCIW を停止します。

停止後に ciwcleanup コマンドを実行し，ciwlistsid コマンドで各プロセスの状態が Stopped であることを確認します。

関連項目

- 10.2　業務を停止する

17.2.2　CSCIW をインストール（新規インストール）する

最新の CSCIW をインストール（新規インストール）してください。CSCIW のインストールについては，マニュアル『uCosminexus Service Coordinator Interactive Workflow システム構築・運用ガイド』の「4.2.2　CSCIW のインストール」を参照してください。

メモ

- バージョンアップ後，Cosminexus J2EE サーバの V9 互換モードを使用する場合，CSCIW を新規インストールしたあとに，マニュアル『uCosminexus Service Coordinator Interactive Workflow システム構築・運用ガイド』の「付録 J.1　互換モード用のファイルへの入れ替え」を参照して，ファイルを入れ替えてください。
- Windows の場合，移行元環境と同じフォルダにインストールしてください。

(1)　移行元環境から移行先環境へファイルを移行

次のファイルを移行元環境から取得し，移行先環境に格納してください。格納先は移行元環境と同じディレクトリにしてください。また，必要に応じて，ファイルの内容を変更してください。

1. セットアッププロパティファイル
2. コマンド用環境設定ファイル
   JDBC のクラスパスが変更になっている場合は，コマンド用環境設定ファイルの`CSCIWCMD_JVM_CLPATH`の設定を変更してください。
3. 共通設定ファイル
4. アプリケーション呼び出し情報ファイル

アプリケーション呼び出しサービスが呼び出す REST アプリケーションの URL に変更がある場合は，rest.request.url プロパティに指定した URL を変更してください。

5. REST アプリケーション呼び出し用ヘッダファイル
6. REST アプリケーション呼び出しスキーマ変換用スタイルシート
7. コールアクティビティ情報ファイル
8. BPMN ビジネスプロセス定義ファイル※

注※

次に示すどちらかの場合に移行が必要です（BPMN ビジネスプロセス定義ファイルを BPMN ビジネスプロセス定義ファイル格納先ディレクトリに格納している場合は，移行が必要になります。このディレクトリは，共通設定ファイルのBpmnDefinitionFileDir パラメタの指定値であり，03-00 以前の環境，または 03-00 以前からバージョンアップした環境でこのディレクトリを使用します）。

- 03-00 以前からバージョンアップするとき
- 03-10 以降からバージョンアップするときで，かつ過去に 03-00 以前からバージョンアップしたことがあるとき

17.2.3　ワーク管理データベースを構築する

移行先環境へ移行元環境と同じバージョンのワーク管理データベースを作成します。

(1)　ワーク管理データベースを構築する（HiRDB の場合）

使用しているワーク管理データベースが HiRDB の場合の手順を示します。

操作手順

1. ワーク管理データベースを作成する。

   ワーク管理データベースのテーブルやインデクスを HiRDB に作成します。移行元環境と同じバージョンのものを作成してください。

   DBMS の機能を使用して，ワーク管理データベースのテーブル定義，インデクスを移行することもできます。テーブル定義，インデクスと同時にテーブルのデータも移行する場合は，「17.2.4　ワーク管理データベースのデータを移行する」の手順を実施する必要はありません。

関連項目

- 7.2　ワーク管理データベースを作成する（HiRDB の場合）

(2)　ワーク管理データベースを構築する（ORACLE の場合）

使用しているワーク管理データベースが ORACLE の場合の手順を示します。

操作手順

1. ワーク管理データベースを作成する。

   ワーク管理データベースのテーブルやインデクスを ORACLE に作成します。移行元環境と同じバージョンのものを作成してください。

   DBMS の機能を使用して，ワーク管理データベースのテーブル定義，インデクスを移行することもできます。テーブル定義，インデクスと同時にテーブルのデータも移行する場合は，「17.2.4　ワーク管理データベースのデータを移行する」の手順を実施する必要はありません。

関連項目

- 7.3　ワーク管理データベースを作成する（ORACLE の場合）

(3)　ワーク管理データベースを構築する（PostgreSQL の場合）

使用しているワーク管理データベースが PostgreSQL の場合の手順を示します。

操作手順

1. ワーク管理データベースを作成する。

   ワーク管理データベースのテーブルやインデクスを PostgreSQL に作成します。移行元環境と同じバージョンのものを作成してください。

   DBMS の機能を使用して，ワーク管理データベースのテーブル定義，インデクスを移行することもできます。テーブル定義，インデクスと同時にテーブルのデータも移行する場合は，「17.2.4　ワーク管理データベースのデータを移行する」の手順を実施する必要はありません。

関連項目

- 7.4　ワーク管理データベースを作成する（PostgreSQL の場合）

17.2.4　ワーク管理データベースのデータを移行する

移行元環境のワーク管理データベースの，すべてのテーブルのデータを，移行先環境のワーク管理データベースへ移行します。

データの移行については，各データベースのマニュアルを参照してください。

17.2.5　ワーク管理データベースをバージョンアップする

CSCIW をバージョンアップする場合に必要な，ワーク管理データベースのバージョンアップ手順を示します。

ワーク管理データベースをバージョンアップをするときは，次に示す作業を実施してください。また，テーブルやインデクスは，SQL スクリプトファイルを使用して追加および更新してください。

SQL スクリプトファイルはバージョンごとに段階的に実行する必要があります。例えば，CSCIW の 02-10 から 03-30 へバージョンアップする場合，次の順序で実行します。

1. 02-10 から 02-20 へ移行する SQL スクリプトファイルを編集および実行
2. 02-20 から 02-30 へ移行する SQL スクリプトファイルを編集および実行
3. 02-30 から 03-00 へ移行する SQL スクリプトファイルを編集および実行
4. 03-00 から 03-10 へ移行する SQL スクリプトファイルを編集および実行
5. 03-10 から 03-11 へ移行する SQL スクリプトファイルを編集および実行
6. 03-11 から 03-20 へ移行する SQL スクリプトファイルを編集および実行
7. 03-20 から 03-30 へ移行する SQL スクリプトファイルを編集および実行

(1) ワーク管理データベースをバージョンアップする（HiRDB の場合）

使用しているワーク管理データベースが HiRDB の場合の手順を示します。

操作手順

1. 次に示す SQL スクリプトファイルを編集する。

- 02-10 から 02-20 へ移行する場合

```
<CSCIWインストールディレクトリ>/sql/vup_0210to0220_hirdb.sql
```

```
<CSCIWインストールディレクトリ>/sql/vupex_0210to0220_hirdb.sql
```

- 02-20 から 02-30 へ移行する場合

```
<CSCIWインストールディレクトリ>/sql/vup_0220to0230_hirdb.sql
```

```
<CSCIWインストールディレクトリ>/sql/vupex_0220to0230_hirdb.sql
```

- 02-30 から 03-00 へ移行する場合

```
<CSCIWインストールディレクトリ>/sql/vup_0230to0300_hirdb.sql
```

- 03-00 から 03-10 へ移行する場合

```
<CSCIWインストールディレクトリ>/sql/vup_0300to0310_hirdb.sql
```

```
<CSCIWインストールディレクトリ>/sql/vupex_0300to0310_hirdb.sql
```

- 03-10 から 03-11 へ移行する場合

```
<CSCIWインストールディレクトリ>/sql/vup_0310to0311_hirdb.sql
```

- 03-11 から 03-20 へ移行する場合

```
<CSCIWインストールディレクトリ>/sql/vup_0311to0320_hirdb.sql
```

- 03-20 から 03-30 へ移行する場合

```
<CSCIWインストールディレクトリ>/sql/vup_0320to0330_hirdb.sql
```

SQL スクリプトファイルの書き換えが必要な文字列，および書き換える内容の詳細については，ワーク管理データベースを作成するときと同じです。

2. SQL スクリプトファイルを実行する。

編集した SQL スクリプトファイルの実行方法については，ワーク管理データベースを作成するときと同じです。

3. ワーク管理データベースへのアクセス権限を付与する。

ワーク管理データベースを移行する前にアクセス権限を付与していた場合，アクセス権限を付与し直す必要があります。

関連項目

- 7.2　ワーク管理データベースを作成する（HiRDB の場合）
- 7.5　データベースへのアクセス権限を付与する（HiRDB の場合）

(2)　ワーク管理データベースをバージョンアップする（ORACLE の場合）

使用しているワーク管理データベースが ORACLE の場合の手順を示します。

操作手順

1. 次に示す SQL スクリプトファイルを編集する。

- 02-30 から 03-00 へ移行する場合

```
<CSCIWインストールディレクトリ>/sql/vup_0230to0300_oracle.sql
```

- 03-00 から 03-10 へ移行する場合

```
<CSCIWインストールディレクトリ>/sql/vup_0300to0310_oracle.sql
```

```
<CSCIWインストールディレクトリ>/sql/vupex_0300to0310_oracle.sql
```

- 03-10 から 03-11 へ移行する場合

```
<CSCIWインストールディレクトリ>/sql/vup_0310to0311_oracle.sql
```

- 03-11 から 03-20 へ移行する場合

```
<CSCIWインストールディレクトリ>/sql/vup_0311to0320_oracle.sql
```

- 03-20 から 03-30 へ移行する場合

```
<CSCIWインストールディレクトリ>/sql/vup_0320to0330_oracle.sql
```

SQL スクリプトファイルの書き換えが必要な文字列，および書き換える内容の詳細については，ワーク管理データベースを作成するときと同じです。

2. SQL スクリプトファイルを実行する。

編集した SQL スクリプトファイルの実行方法については，ワーク管理データベースを作成するときと同じです。

3. ワーク管理データベースへのアクセス権限を付与する。

ワーク管理データベースを移行する前にアクセス権限を付与していた場合，アクセス権限を付与し直す必要があります。

関連項目

- 7.3　ワーク管理データベースを作成する（ORACLE の場合）
- 7.6　データベースへのアクセス権限を付与する（ORACLE の場合）

(3)　ワーク管理データベースをバージョンアップする（PostgreSQL の場合）

使用しているワーク管理データベースが PostgreSQL の場合の手順を示します。

操作手順

1. 次に示す SQL スクリプトファイルを編集する。

- 02-30 から 03-00 へ移行する場合

```
<CSCIWインストールディレクトリ>/sql/vup_0230to0300_postgresql.sql
```

- 03-00 から 03-10 へ移行する場合

```
<CSCIWインストールディレクトリ>/sql/vup_0300to0310_postgresql.sql
```

```
<CSCIWインストールディレクトリ>/sql/vupex_0300to0310_postgresql.sql
```

- 03-10 から 03-11 へ移行する場合

```
<CSCIWインストールディレクトリ>/sql/vup_0310to0311_postgresql.sql
```

- 03-11 から 03-20 へ移行する場合

```
<CSCIWインストールディレクトリ>/sql/vup_0311to0320_postgresql.sql
```

- 03-20 から 03-30 へ移行する場合

```
<CSCIWインストールディレクトリ>/sql/vup_0320to0330_postgresql.sql
```

SQL スクリプトファイルの書き換えが必要な文字列，および書き換える内容の詳細については，ワーク管理データベースを作成するときと同じです。

2. SQL スクリプトファイルを実行する。

編集した SQL スクリプトファイルの実行方法については，ワーク管理データベースを作成するときと同じです。

3. ワーク管理データベースへのアクセス権限を付与する。

ワーク管理データベースを移行する前にアクセス権限を付与していた場合，アクセス権限を付与し直す必要があります。

関連項目

- 7.4　ワーク管理データベースを作成する（PostgreSQL の場合）
- 7.7　データベースへのアクセス権限を付与する（PostgreSQL の場合）

17.2.6　セットアップコマンドを実行してバージョンアップする

CSCIW をバージョンアップする場合に必要な，セットアップコマンドの設定方法を示します。

(1)　環境変数を設定する

CSCIW_HOME 環境変数を設定します。

環境変数に設定する内容は，CSCIW の実行環境を初期化する場合と同じです。バージョンアップ前と同じ内容を設定してください。

関連項目

- 7.8　CSCIW の実行環境を初期化する

(2)　セットアッププロパティファイルを設定する

セットアッププロパティファイルを設定します。

セットアッププロパティファイルに設定する内容は，CSCIW の実行環境を初期化する場合と同じです。バージョンアップ前と同じ内容を設定してください。

関連項目

- 7.8　CSCIW の実行環境を初期化する

(3)　コマンド用環境設定ファイルを設定する

コマンド用環境設定ファイルを設定します。

コマンド用環境設定ファイルに設定する内容は，CSCIW の実行環境を初期化する場合と同じです。バージョンアップ前と同じ内容を設定してください。

関連項目

- 7.8　CSCIW の実行環境を初期化する

(4)　コマンドを実行して実行環境をバージョンアップする

CSCIW の実行環境は，ciwsetenv（環境の構築または削除）コマンドを実行してバージョンアップします。なお，ciwsetenv（環境の構築または削除）コマンドは，すべてのマシンで実行してください。

次に示す形式でコマンドを実行すると，システム設定プロパティファイルの内容は，指定した環境構築ファイルの内容に書き換えられます。なお，環境構築ファイルに設定する項目「SystemDBPassword」および「SystemDBURL」については，同じワーク管理データベースに接続できる範囲で，値を変更できます。

```
ciwsetenv -sid 〈システムID〉-vup 〈環境構築ファイル名〉
```

ciwsetenv（環境の構築または削除）コマンドについては，マニュアル『uCosminexus Service Coordinator Interactive Workflow コマンド』を参照してください。

関連項目

- 7.8　CSCIW の実行環境を初期化する

17.2.7　Cosminexus を設定する

Cosminexus を設定します。

(1)　コンテナ拡張ライブラリに JAR ファイルを取り込む

コンテナ拡張ライブラリに jar を追加します。

メモ

Cosminexus J2EE サーバの V9 互換モードを使用する場合，コンテナ拡張ライブラリに取り込むライブラリの変更が必要です。

関連項目

- 7.9.1　コンテナ拡張ライブラリに JAR ファイルを取り込む

(2)　環境変数を取り込む

環境変数を取り込みます。

関連項目

- 7.9.2　環境変数を取り込む

(3)　Cosminexus を起動する

Cosminexus を起動します。

関連項目

- 7.9.3　Cosminexus を起動する

(4)　DB Connector を設定する

DB Connector を設定します。

関連項目

- 7.9.4　DB Connector を設定する

(5)　CSCIWManagementServer を設定する

CSCIWManagementServer を設定します。

関連項目

- 7.9.5　CSCIWManagementServer に関する設定をする

(6)　REST サービスを設定する

REST サービスを設定します。

関連項目

- 7.9.7　REST サービスに関する設定をする

(7)　ビジネスプロセスオペレータを設定する

ビジネスプロセスオペレータを設定します。

関連項目

- 7.9.8　ビジネスプロセスオペレータに関する設定をする

(8)　案件運用操作を設定する

案件運用操作を設定します。

関連項目

- 7.9.9　案件運用操作に関する設定をする

(9)　アプリケーション呼び出しサービスを設定する

アプリケーション呼び出しサービスを設定します。

以上で移行先環境のバージョンアップは完了です。

関連項目

- 7.9.6　アプリケーション呼び出しサービスに関する設定をする

17.2.8　データ移行だけを繰り返し行う

移行先環境のバージョンアップが完了したあと，データ移行だけを繰り返し行うことができます。

例えば，次のような場面で利用できます。

- 本番環境のバージョンアップ時に，あらかじめ移行先環境を構築しておき，本番直前はデータ移行だけ行う
- テスト環境のバージョンアップ時に，移行元環境のデータを使用してテストを繰り返し行う

> **メモ**
>
> データ移行だけを繰り返し行うには，移行元環境のバージョンが 02-20 以降である必要があります。

データ移行だけを繰り返し行う手順を次に示します。

1. CSCIW を停止する。

移行元環境および移行先環境の CSCIW を停止します。

2. 移行先環境のテーブルのレコードを削除する。

   このとき，移行先環境の<SYSTEMID>_USER_SETTING_INFO テーブルのデータは削除しないでください。

3. テーブルのデータを移行する。

   <SYSTEMID>_USER_SETTING_INFO 以外のテーブルのデータを移行します。

ヒント

<SYSTEMID>_USER_SETTING_INFO テーブルを対象外にすることで，システム共通環境情報（ciwchgenv コマンドで設定した内容）は，移行元環境から移行先環境に反映されません。

それ以外のすべてのデータ（ビジネスプロセス定義や案件）は反映されます。

付録

付録 A　テーブル容量の見積もり

次の情報を基に，使用する DBMS のマニュアルを参考にデータベースの容量見積もりをしてください。

- テーブルおよびインデクス定義
- レコード数の概算式

付録 A.1　テーブル定義

BPMN 連携機能の使用に伴い追加するテーブルの内容を基に，テーブル容量を見積もってください。見積もりに必要なテーブルの情報を示します。

(1)　テーブルの一覧

表 A-1　BPMN 連携機能に関するテーブル一覧

項番	分類	テーブル名	インデクス名プレフィクス
1	アプリケーション呼び出し開始タイマーテーブル定義	<SYSTEMID>_APPLICATION_START_TIMER	<SYSTEMID>_APPSTIMER
2	プロセスデータ（文字列用）テーブル定義	<SYSTEMID>_PROCESS_DATA_S	<SYSTEMID>_PDS
3	プロセスデータ（数値用）テーブル定義	<SYSTEMID>_PROCESS_DATA_N	<SYSTEMID>_PDN
4	マルチインスタンス管理テーブル定義	<SYSTEMID>_MULTI_INSTANCE_MNG	<SYSTEMID>_MIMNG
5	ビジネスプロセス連携テーブル定義	<SYSTEMID>_BP_RELATION	<SYSTEMID>_BPR
6	サブプロセス用マルチインスタンス管理テーブル定義	<SYSTEMID>_SUB_MULTI_INSTANCE_MNG	<SYSTEMID>_SUBMIMNG
7	アドホック・サブプロセス管理テーブル定義	<SYSTEMID>_ADHOC_SUBPROCESS_MNG	<SYSTEMID>_ASUBMNG
8	BPMN ビジネスプロセス定義	<SYSTEMID>_BPMN_PROCESS_DEF	<SYSTEMID>_BPMNPDEF

(2)　<SYSTEMID>_APPLICATION_START_TIMER の内容

<SYSTEMID>_APPLICATION_START_TIMER の内容を示します。

テーブル名：<SYSTEMID>_APPLICATION_START_TIMER

インデクス名プレフィクス：<SYSTEMID>_APPSTIMER

表 A-2　アプリケーション呼び出し開始タイマーテーブル定義

項番	名前	列名	HiRDB のデータ型	ORACLE のデータ型	POSTGRESQL のデータ型	INDEX※	NULLABLE	省略可否(省略時の値)
1	ビジネスプロセス定義 ID	ProcessDefinitionID	INTEGER	NUMBER(10)	INTEGER	P1-1	N	不可
2	作業定義 ID	WorkDefinitionID	INTEGER	NUMBER(10)	INTEGER	P1-2	N	不可
3	イベント発火時刻	ExecuteDate	DECIMAL(19)	NUMBER(19)	BIGINT	－	－	可(NULL)
4	実行回数	ExecuteNumber	INTEGER	NUMBER(10)	INTEGER	－	N	不可

(凡例)

N：NOT NULL 制約を定義する

－：該当しない

注※

インデクスの形式は，<T><通番>[-<構成順序>]です。

<T>部分のアルファベットの意味は次のとおりです。

P：主キー（ただし，ユニークと NOT NULL の組み合わせで実現）

(3)　<SYSTEMID>_PROCESS_DATA_S の内容

<SYSTEMID>_PROCESS_DATA_S の内容を示します。

テーブル名：<SYSTEMID>_PROCESS_DATA_S

インデクス名プレフィクス：<SYSTEMID>_PDS

表 A-3　プロセスデータ(文字列用)テーブル定義

項番	名前	列名	HiRDB のデータ型	ORACLE のデータ型	POSTGRESQL のデータ型	INDEX※	NULLABLE	省略可否(省略時の値)
1	案件 ID	ProcessInstanceID	INTEGER	NUMBER(10)	INTEGER	P1-1	N	不可
2	プロセスデータ名	ProcessDataName	MVARCHAR(64)	VARCHAR2(64)	VARCHAR(64)	P1-2 N2-1	N	不可
3	プロセスデータ値	ProcessDataValue	MVARCHAR(512)	VARCHAR2(512)	VARCHAR(512)	N2-2	－	可(NULL)
4	状態	State	CHAR(1)	CHAR(1)	CHAR(1)	－	－	可

項番	名前	列名	HiRDB のデータ型	ORACLE のデータ型	POSTGRESQL のデータ型	INDEX※	NULLABLE	省略可否(省略時の値)
4	状態	State	CHAR(1)	CHAR(1)	CHAR(1)	－	－	(NULL)

(凡例)

N：NOT NULL 制約を定義する

－：該当しない

注※

インデクスの形式は，<T><通番>[-<構成順序>]です。

<T>部分のアルファベットの意味は次のとおりです。

N：通常

P：主キー（ただし，ユニークと NOT NULL の組み合わせで実現）

(4) <SYSTEMID>_PROCESS_DATA_N の内容

<SYSTEMID>_PROCESS_DATA_N の内容を示します。

テーブル名：<SYSTEMID>_PROCESS_DATA_N

インデクス名プレフィクス：<SYSTEMID>_PDN

表 A-4 プロセスデータ(数値用)テーブル定義

項番	名前	列名	HiRDB のデータ型	ORACLE のデータ型	POSTGRESQL のデータ型	INDEX※	NULLABLE	省略可否(省略時の値)
1	案件 ID	ProcessInstanceID	INTEGER	NUMBER(10)	INTEGER	P1-1	N	不可
2	プロセスデータ名	ProcessDataName	MVARCHAR(64)	VARCHAR2(64)	VARCHAR(64)	P1-2 N2-1	N	不可
3	プロセスデータ値	ProcessDataValue	INTEGER	NUMBER(10)	INTEGER	N2-2	－	可 (NULL)
4	状態	State	CHAR(1)	CHAR(1)	CHAR(1)	－	－	可 (NULL)

(凡例)

N：NOT NULL 制約を定義する

－：該当しない

注※

インデクスの形式は，<T><通番>[-<構成順序>]です。

<T>部分のアルファベットの意味は次のとおりです。

N：通常

P：主キー（ただし，ユニークと NOT NULL の組み合わせで実現）

(5) <SYSTEMID>_MULTI_INSTANCE_MNG の内容

<SYSTEMID>_MULTI_INSTANCE_MNG の内容を示します。

テーブル名：<SYSTEMID>_MULTI_INSTANCE_MNG

インデクス名プレフィクス：<SYSTEMID>_MIMNG

表 A-5　マルチインスタンス管理テーブル定義

項番	名前	列名	HiRDB のデータ型	ORACLE のデータ型	POSTGRESQL のデータ型	INDEX※	NULLABLE	省略可否（省略時の値）
1	案件 ID	ProcessInstanceID	INTEGER	NUMBER(10)	INTEGER	P1-1	N	不可
2	業務ステップ ID	ActivityInstanceID	INTEGER	NUMBER(10)	INTEGER	P1-2	N	不可
3	作業 ID	WorkItemID	INTEGER	NUMBER(10)	INTEGER	－	N	不可
4	繰り返し回数	LoopCardinality	INTEGER	NUMBER(10)	INTEGER	－	N	不可
5	完了済インスタンス数	CompletedInstanceNumber	INTEGER	NUMBER(10)	INTEGER	－	N	不可
6	インスタンス生成済フラグ	CreatedFlag	CHAR(1)	CHAR(1)	CHAR(1)	－	－	可（NULL）

（凡例）

N：NOT NULL 制約を定義する

－：該当しない

注※

インデクスの形式は，<T><通番>[-<構成順序>]です。

<T>部分のアルファベットの意味は次のとおりです。

P：主キー（ただし，ユニークと NOT NULL の組み合わせで実現）

(6) <SYSTEMID>_BP_RELATION の内容

<SYSTEMID>_BP_RELATION の内容を示します。

テーブル名：<SYSTEMID>_BP_RELATION

インデクス名プレフィクス：<SYSTEMID>_BPR

表 A-6　ビジネスプロセス連携テーブル定義

項番	名前	列名	HiRDB のデータ型	ORACLE のデータ型	POSTGRESQL のデータ型	INDEX※	NULLABLE	省略可否(省略時の値)
1	案件 ID	ProcessInstanceID	INTEGER	NUMBER(10)	INTEGER	P1	N	不可
2	親案件 ID	ParentProcessInstanceID	INTEGER	NUMBER(10)	INTEGER	N3-1 U4-1	N	不可
3	親業務ステップ ID	ParentActivityInstanceID	INTEGER	NUMBER(10)	INTEGER	N3-2	N	不可
4	親作業 ID	ParentWorkItemID	INTEGER	NUMBER(10)	INTEGER	U4-2	N	不可
5	ルート案件 ID	RootProcessInstanceID	INTEGER	NUMBER(10)	INTEGER	N2	N	不可
6	calledElement	CalledElement	MVARCHAR(32)	VARCHAR2(32)	VARCHAR(32)	－	N	不可

（凡例）

N：NOT NULL 制約を定義する

－：該当しない

注※

インデクスの形式は，<T><通番>[-<構成順序>]です。

<T>部分のアルファベットの意味は次のとおりです。

N：通常

P：主キー（ただし，ユニークと NOT NULL の組み合わせで実現）

U：ユニーク

(7)　<SYSTEMID>_SUB_MULTI_INSTANCE_MNG の内容

<SYSTEMID>_SUB_MULTI_INSTANCE_MNG の内容を示します。

テーブル名：<SYSTEMID>_SUB_MULTI_INSTANCE_MNG

インデクス名プレフィクス：<SYSTEMID>_SUBMIMNG

表 A-7　サブプロセス用マルチインスタンス管理テーブル定義

項番	名前	列名	HiRDB のデータ型	ORACLE のデータ型	POSTGRESQL のデータ型	INDEX※	NULLABLE	省略可否(省略時の値)
1	案件 ID	ProcessInstanceID	INTEGER	NUMBER(10)	INTEGER	P1-1	N	不可
2	階層定義 ID	HierarchyDefinitionID	INTEGER	NUMBER(10)	INTEGER	P1-2	N	不可
3	繰り返し回数	LoopCardinality	INTEGER	NUMBER(10)	INTEGER	－	N	不可
4	完了済インスタンス数	CompletedInstanceNumber	INTEGER	NUMBER(10)	INTEGER	－	N	不可
5	実行済フラグ	ExecutedFlag	CHAR(1)	CHAR(1)	CHAR(1)	－	N	不可

（凡例）

N：NOT NULL 制約を定義する

－：該当しない

注※

インデクスの形式は，<T><通番>[-<構成順序>]です。

<T>部分のアルファベットの意味は次のとおりです。

P：主キー（ただし，ユニークと NOT NULL の組み合わせで実現）

(8)　<SYSTEMID>_ADHOC_SUBPROCESS_MNG の内容

<SYSTEMID>_ADHOC_SUBPROCESS_MNG の内容を示します。

テーブル名：<SYSTEMID>_ADHOC_SUBPROCESS_MNG

インデクス名プレフィクス：<SYSTEMID>_ASUBMNG

表 A-8　アドホック・サブプロセス管理テーブル定義

項番	名前	列名	HiRDB のデータ型	ORACLE のデータ型	POSTGRESQL のデータ型	INDEX※	NULLABLE	省略可否(省略時の値)
1	案件 ID	ProcessInstanceID	INTEGER	NUMBER(10)	INTEGER	P1-1	N	不可
2	階層定義 ID	HierarchyDefinitionID	INTEGER	NUMBER(10)	INTEGER	P1-2	N	不可

項番	名前	列名	HiRDB のデータ型	ORACLE のデータ型	POSTGRESQL のデータ型	INDEX※	NULLABLE	省略可否(省略時の値)
3	作業 ID	WorkItemID	INTEGER	NUMBER(10)	INTEGER	－	N	不可
4	実行方式	Ordering	CHAR(1)	CHAR(1)	CHAR(1)	－	N	不可

（凡例）

N：NOT NULL 制約を定義する

－：該当しない

注※

インデクスの形式は，<T><通番>[-<構成順序>]です。

<T>部分のアルファベットの意味は次のとおりです。

P：主キー（ただし，ユニークと NOT NULL の組み合わせで実現）

(9) <SYSTEMID>_BPMN_PROCESS_DEF の内容

<SYSTEMID>_BPMN_PROCESS_DEF の内容を示します。

テーブル名：<SYSTEMID>_BPMN_PROCESS_DEF

インデクス名プレフィクス：<SYSTEMID>_BPMNPDEF

表 A-9　BPMN ビジネスプロセス定義テーブル定義

項番	名前	列名	HiRDB のデータ型	ORACLE のデータ型	POSTGRESQL のデータ型	INDEX※	NULLABLE	省略可否(省略時の値)
1	ビジネスプロセス定義名	ProcessDefName	MVARCHAR(64)	VARCHAR2(64)	VARCHAR(64)	P1-1	N	不可
2	ビジネスプロセス定義バージョン	ProcessDefVersion	SMALLINT	NUMBER(4)	SMALLINT	P1-2	N	不可
3	BPMN ビジネスプロセス定義	BpmnProcessDef	BINARY(4194304)	LONG RAW	BYTEA	－	－	不可

（凡例）

N：NOT NULL 制約を定義する

－：該当しない

注※

インデクスの形式は，<T><通番>[-<構成順序>]です。

<T>部分のアルファベットの意味は次のとおりです。

P：主キー（ただし，ユニークと NOT NULL の組み合わせで実現）

(10) インデクスを追加するテーブルの一覧

インデクスを追加するテーブルの一覧を示します。

BPMN 連携機能を使用する場合，次のテーブルにインデクスを追加します。

- <SYSTEMID>_PROCESS_INSTANCE
- <SYSTEMID>_WORK_ITEM

各テーブルの詳細については，マニュアル『uCosminexus Service Coordinator Interactive Workflow システム構築・運用ガイド』の，次の個所をそれぞれ参照してください。

- 「付録 B.1(1)(w) <SYSTEMID>_PROCESS_INSTANCE の内容」
- 「付録 B.1(1)(y) <SYSTEMID>_WORK_ITEM の内容」

(11) インデクスの一覧

BPMN 連携機能で追加するインデクスの一覧を示します。

表 A-10 BPMN 連携機能で追加するインデクス一覧

テーブル名	インデクス名	制約	列
<SYSTEMID>_PROCESS_DATA_S	IDX_<SYSTEMID>_PDS_P1	U	2
	IDX_<SYSTEMID>_PDS_N2	NU	2
<SYSTEMID>_PROCESS_DATA_N	IDX_<SYSTEMID>_PDN_P1	U	2
	IDX_<SYSTEMID>_PDN_N2	NU	2
<SYSTEMID>_MULTI_INSTANCE_MNG	IDX_<SYSTEMID>_MIMNG_P1	U	2
<SYSTEMID>_BP_RELATION	IDX_<SYSTEMID>_BPR_P1	U	1
	IDX_<SYSTEMID>_BPR_N2	NU	1
	IDX_<SYSTEMID>_BPR_N3	NU	2
	IDX_<SYSTEMID>_BPR_U4	U	2

テーブル名	インデクス名	制約	列
<SYSTEMID>_WORK_ITEM	IDX_<SYSTEMID>_WI_N4	NU	2
	IDX_<SYSTEMID>_WI_N5	NU	2
	IDX_<SYSTEMID>_WI_N6	NU	2
<SYSTEMID>_APPLICATION_START_TIMER	IDX_<SYSTEMID>_APPSTIMER_P1	U	2
<SYSTEMID>_PROCESS_INSTANCE	IDX_<SYSTEMID>_PI_N5	NU	2
<SYSTEMID>_SUB_MULTI_INSTANCE_MNG	IDX_<SYSTEMID>_SUBMIMNG_P1	U	2
<SYSTEMID>_ADHOC_SUBPROCESS_MNG	IDX_<SYSTEMID>_ASUBMNG_P1	U	2
<SYSTEMID>_BPMN_PROCESS_DEF	IDX_<SYSTEMID>_BPMNPDEF_P1	U	2

（凡例）

U：UNIQUE

NU：NONUNIQUE

付録 A.2　レコード数の概算式

データベースの容量は，テーブルのレコード数が影響します。レコード数の見積もりは，各テーブルの概算式で算出してください。

(1)　インスタンステーブルのレコード数

プロセスデータ（文字列用）（テーブル名：<SYSTEMID>_PROCESS_DATA_S）

$$\text{レコード数} = \sum_{\text{全案件数}} \text{（案件に対応するプロセスデータ（文字列）の数）}$$

リスト型プロセスデータの場合，プロセスデータの数は，追加したリスト型プロセスデータの要素数になります。ただし，1 度リスト型プロセスデータを追加したあとに要素数を減少させた場合，減少させた要素はテーブルのレコードに登録されたままになります。そのため，追加する要素の最大数で見積もってください。

プロセスデータ（数値用）（テーブル名：<SYSTEMID>_PROCESS_DATA_N）

$$\text{レコード数} = \sum_{\text{全案件数}} \text{（案件に対応するプロセスデータ（数値）の数）}$$

プロセスデータ（数値）の数＝<業務アプリケーションが追加したプロセスデータ（数値）の数>＋1（BPMN 連携ライブラリが投入時に内部で追加）

リスト型プロセスデータの場合，プロセスデータの数は，追加したリスト型プロセスデータの要素数になります。ただし，1 度リスト型プロセスデータを追加したあとに要素数を減少させた場合，減少させた要素はテーブルのレコードに登録されたままになります。そのため，追加する要素の最大数で見積もってください。

マルチインスタンス管理テーブル（テーブル名：<SYSTEMID>_MULTI_INSTANCE_MNG）

$$\text{レコード数} = \sum_{\text{＜全ビジネスプロセス定義数＞}} (\text{＜案件数＞} \times \text{＜平均マルチインスタンス数＞})$$

案件数：終了している案件も含みます。

平均マルチインスタンス数：BPMN ビジネスプロセス定義ごとに 1 案件で生成される平均マルチインスタンス数です。

なお，マルチインスタンスは，BPMN ビジネスプロセス定義に定義されたマルチインスタンスから変換された業務ステップに遷移するごとに，1 件生成されます。業務ステップ内に複数生成される作業数は含みません。

ビジネスプロセス連携テーブル（テーブル名：<SYSTEMID>_BP_RELATION）

$$\text{レコード数} = \sum_{\text{＜全ビジネスプロセス定義数＞}} (\text{＜案件数＞} \times \text{＜平均コールアクティビティ数＞})$$

案件数：終了している案件も含みます。

平均コールアクティビティ数：ビジネスプロセス定義ごとに 1 案件で生成される平均コールアクティビティ数です。

なお，コールアクティビティにマルチインスタンスが定義されている場合は，マルチインスタンス内で生成されるインスタンス数がコールアクティビティ数になります。

サブプロセス用マルチインスタンス管理テーブル（テーブル名：<SYSTEMID>_SUB_MULTI_INSTANCE_MNG）

$$\text{レコード数} = \sum_{\text{＜全ビジネスプロセス定義数＞}} (\text{＜案件数＞} \times \text{＜平均サブプロセス(マルチインスタンス)数＞})$$

案件数：終了している案件も含みます。

平均サブプロセス（マルチインスタンス）数：ビジネスプロセス定義ごとに 1 案件で生成される平均サブプロセス（マルチインスタンス）数です。

なお，サブプロセス（マルチインスタンス）は，BPMN ビジネスプロセス定義に定義されたサブプロセス（マルチインスタンス）から変換された階層定義に遷移するごとに，1 件生成されます。

アドホック・サブプロセス管理テーブル（テーブル名：<SYSTEMID>_ADHOC_SUBPROCESS_MNG）

$$\text{レコード数} = \sum_{\text{<全ビジネスプロセス定義数>}} (\text{<案件数>} \times \text{<平均アドホック・サブプロセス数>})$$

案件数：終了している案件も含みます。

平均アドホック・サブプロセス数：ビジネスプロセス定義ごとに 1 案件で生成される平均アドホック・サブプロセス数です。

なお，アドホック・サブプロセスは，BPMN ビジネスプロセス定義に定義されたアドホック・サブプロセスから変換された階層定義に遷移するごとに，1 件生成されます。

(2) そのほかのテーブルのレコード数

アプリケーション呼び出し開始タイマーテーブル（テーブル名：<SYSTEMID>_APPLICATION_START_TIMER）

$$\text{レコード数} = \sum_{\text{<全ビジネスプロセス定義数>}} (\text{各ビジネスプロセス定義に含まれる開始(タイマー)の数})$$

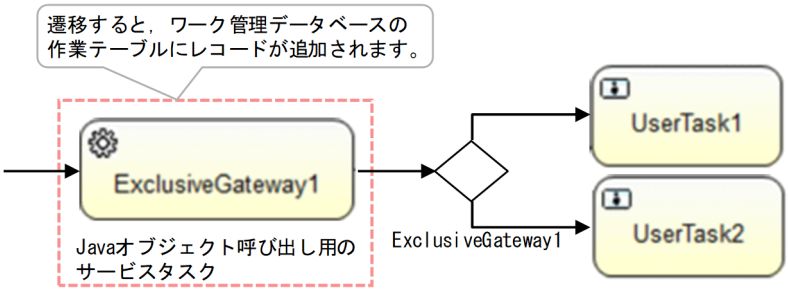
BPMN ビジネスプロセス定義テーブル（テーブル名：<SYSTEMID>_BPMN_PROCESS_DEF）

レコード数＝ビジネスプロセス定義数

付録 B　ゲートウェイの実行履歴をワーク管理データベースに保存する

CSCIW では，業務処理を行わずに作業を完了する Java オブジェクト呼び出しを使用することで，排他ゲートウェイ，並列ゲートウェイ，排他イベントゲートウェイ，サブプロセス，リンクなどの BPMN 要素に遷移した履歴をワーク管理データベースに保存できます。次の図に示すように，排他ゲートウェイの遷移元に，この Java オブジェクト呼び出しを定義したサービスタスクを配置することで，遷移した履歴をワーク管理データベースに保存できます。また，この Java オブジェクト呼び出しを定義したサービスタスクは，アプリケーション呼び出しサービスによって，自動的に完了して次に遷移します。Java API，REST API などで遷移させる必要はありません。

図 B-1　ビジネスプロセスの定義例（ゲートウェイの実行履歴を保存する）



付録 B.1　業務処理を行わずに作業を完了する Java オブジェクトの設定方法

サービスタスクの ref 識別子に対応するアプリケーション呼び出し情報ファイルに，業務処理を行わずに作業を完了する Java オブジェクトの Java クラス名を指定します。アプリケーション呼び出し情報ファイルの設定方法を次に示します。

アプリケーション呼び出し情報ファイルの設定

```
type=JAVA
java.invoke.class=jp.co.Hitachi.soft.csciw.bpmn.callback.java.CIWBpmnEmptyApplication
```

付録 B.2　実行間隔とポーリング間隔の設定方法

アプリケーション呼び出しサービスがサービスタスクを完了する契機は，実行間隔とポーリング間隔によって決まります。詳細については，「1.8.6　実行間隔とポーリング間隔」を参照してください。

実行間隔のデフォルト値は 300 秒，またポーリング間隔のデフォルト値は 60 秒であるため，サービスタスクに遷移してから，完了して次に遷移するまでの間に，待ち時間が発生します。この待ち時間を短くしたい場合は，実行間隔とポーリング間隔に小さい値を設定してください。

ポーリング間隔は，共通設定ファイルの「AppCallServicePollingInterval」に設定してください。AppCallServicePollingInterval の詳細については，「15.2.5　共通設定ファイルに指定する内容」の「(4)　AppCallServicePollingInterval」を参照してください。

実行間隔は，アプリケーション呼び出し制御情報の「実行間隔（ExecuteInterval)」に設定してください。詳細については，マニュアル『uCosminexus Service Coordinator Interactive Workflow コマンド』の「ciwmngap（アプリケーション呼び出し制御情報の管理)」を参照してください。

ref 識別子共通設定単位で，「実行間隔（ExecuteInterval)」を変更する手順を次に示します。

1. アプリケーション呼び出し制御情報を出力する

   ciwmngap コマンドを実行して，アプリケーション呼び出し制御情報をファイルに出力します。

   ■コマンドの実行例

   ```
   ciwmngap -sid システムID -list > アプリケーション呼び出し制御情報ファイル名
   ```

2. アプリケーション呼び出し制御情報ファイルを編集する

   アプリケーション呼び出し制御情報ファイルで，サービスタスクの ref 識別子共通設定の実行間隔（ExecuteInterval）に小さい値（次の例では 0）を指定します。

   ■ファイルの作成例

   ```
   #UPDATEOPTION,REFTYPE,REF,EXECUTEINTERVAL,RETRYINTERVAL,RETRYCOUNT,WORKITEMMAX,RECOVERYTI
   ME
   #,err,(Common),300,300,0,10000,1500
   #,msg,(Common),300,300,0,10000,1500
   U,ope,(Common),0,300,0,10000,1500
   …
   ```

3. アプリケーション呼び出し制御情報を更新する

   アプリケーション呼び出し制御情報ファイルを指定してciwmngap コマンドを実行します。

   ■コマンドの実行例

   ```
   ciwmngap -sid システムID -chg -apdf アプリケーション呼び出し制御情報ファイル名
   ```

付録 B.3　障害発生時の対処手順

障害が発生すると，サービスタスクに対応する作業が「作業者実行」状態になり，案件遷移が止まるおそれがあります。この場合，該当の作業を完了させ，案件を次に遷移させてください。

対処手順については，「10.5.2　アプリケーション呼び出しをしないで案件を遷移させる」を参照してください。

付録 C　アプリケーション呼び出しサービスの性能チューニング

アプリケーション呼び出しサービスのスループットに影響するチューニング項目を示します。

付録 C.1　性能チューニングが必要となるケース

次に示す場合は，アプリケーション呼び出しサービスの性能チューニングの実施を検討してください。

ケース 1

単位時間当たりの呼び出し対象となる作業の数が多い。

例：1 秒間に 20 件以上の作業が発生している。

ケース 2

ref 識別子の数が多い。

ケース 1 では，アプリケーション呼び出しサービスが単位時間当たりに処理できる作業の量よりも，アプリケーション呼び出しの対象となる作業の量が多くなることによって，作業の遷移が滞留することがあります。

例えば，次に示す要件に該当する場合，1 秒間に 20 件以上の作業が発生すると，作業の遷移が滞留します。

- REST アプリケーションの平均応答時間：0.4（単位：秒）
- 1 作業当たりの CSCIW の平均処理時間：0.1（単位：秒）
- WorkManager の最大スレッド数：10（デフォルト値）

ケース 2 では，アプリケーション呼び出しサービスの負荷が偏ったり，ref 識別子ごとのアプリケーション呼び出しの待ち時間が長くなったりすることによって，作業の遷移が滞留することがあります。

付録 C.2　チューニング項目と効果

アプリケーション呼び出しサービスのスループットを向上させるチューニング項目を次に示します。

- アプリケーション呼び出しを行うスレッド数を増やす

  アプリケーション呼び出しを行うスレッド数を増やすには，WorkManager の最大スレッド数，またはアプリケーション呼び出しサービスの稼働数を増やします。
- アプリケーション呼び出しを行うスレッドの利用効率を上げる

  アプリケーション呼び出しを行うスレッドの利用効率を上げるには，アプリケーション呼び出しグループを登録します。

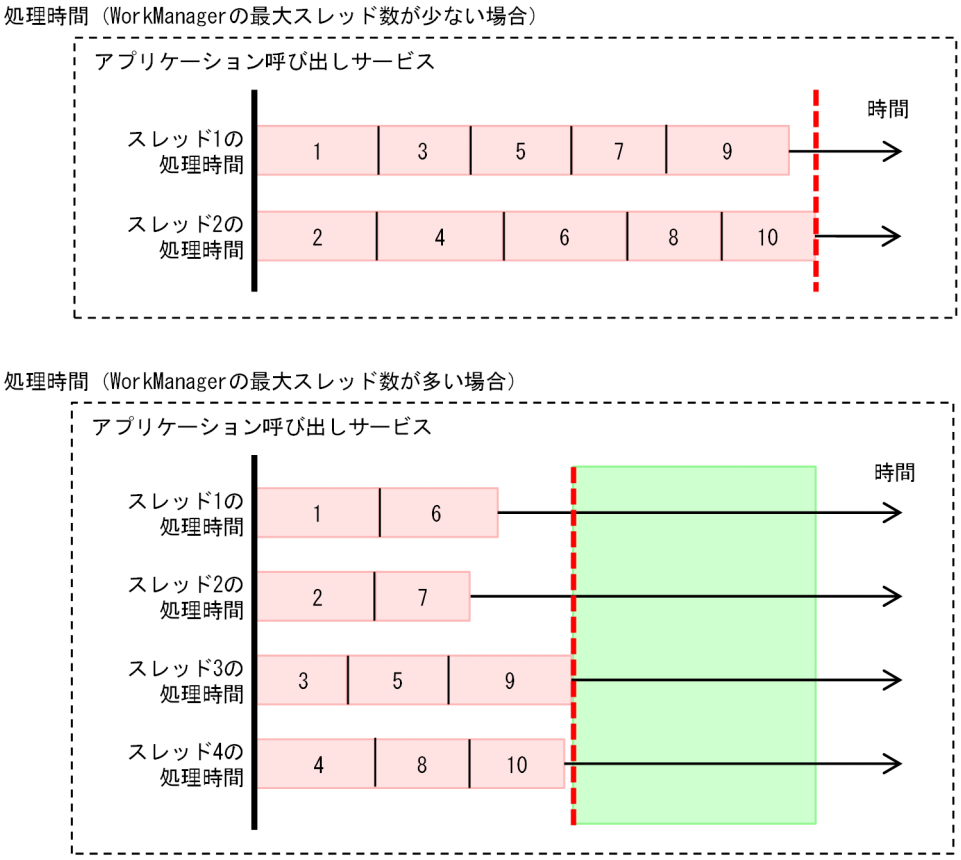
(1) WorkManager の最大スレッド数を増やす

単位時間当たりのアプリケーション呼び出し対象の作業数が多い場合，WorkManager の最大スレッド数を増やすことで，スループットを向上できます。

次に示す条件に該当するケースを例として，WorkManager の最大スレッド数の増加によるスループットの向上の仕組みについて説明します。

- アプリケーション呼び出しの実行間隔ごとに 10 件の作業が発生している。
- 各作業の REST アプリケーション呼び出しに平均 1 秒ずつ掛かっている。

図 C-1　アプリケーション呼び出しサービスのスループットの向上例（WorkManager の最大スレッド数を増やす場合）



［説明］

WorkManager の最大スレッド数が少ない場合：

WorkManager の最大スレッド数が 2 の場合は，アプリケーション呼び出しの完了までに約 5 秒掛かります。

WorkManager の最大スレッド数が多い場合：

WorkManager の最大スレッド数を 4 に倍増させると，WorkManager の最大スレッド数が 2 の場合の約半分の時間で，アプリケーション呼び出しが完了します。

関連項目

- 9.4　WorkManager の最大スレッド数を変更する

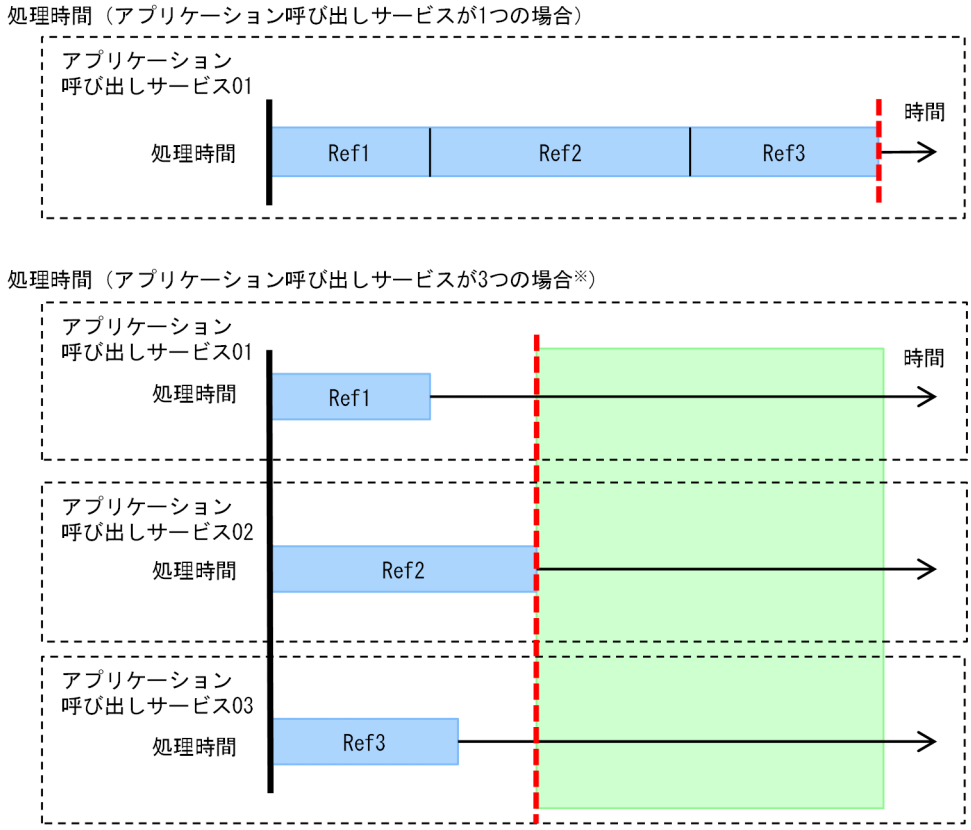
(2)　アプリケーション呼び出しサービスの稼働数を増やす

次に示すどちらかに該当する場合，アプリケーション呼び出しサービスを複数稼働することで，スループットを向上できます。

- ref 識別子の数が多い。
- 単位時間当たりのアプリケーション呼び出し対象の作業数が多い。

次に示す図のケースを例として，アプリケーション呼び出しサービスの稼働数の増加によるスループットの向上の仕組みについて説明します。

図 C-2　アプリケーション呼び出しサービスのスループットの向上例（アプリケーション呼び出しサービスの稼働数を増やす場合）



[説明]

アプリケーション呼び出しサービスが 1 つの場合：

アプリケーション呼び出しサービス 01 が，ref 識別子「Ref1」～「Ref3」すべての REST アプリケーションを，順番に 1 つずつ呼び出します。

アプリケーション呼び出しサービスが 3 つの場合：

アプリケーション呼び出しサービス 01 が ref 識別子「Ref1」の REST アプリケーションを呼び出している間に，アプリケーション呼び出しサービス 02，03 が，それぞれ ref 識別子「Ref2」「Ref3」を同時に処理します。これによって，全体のスループットを向上できます。

ヒント

アプリケーション呼び出しサービスの稼働数を増やす設定方法については，「付録 C.6　チューニングの設定方法」の「(2)　アプリケーション呼び出しサービスの稼働数の設定」を参照してください。

(3)　アプリケーション呼び出しグループを登録する

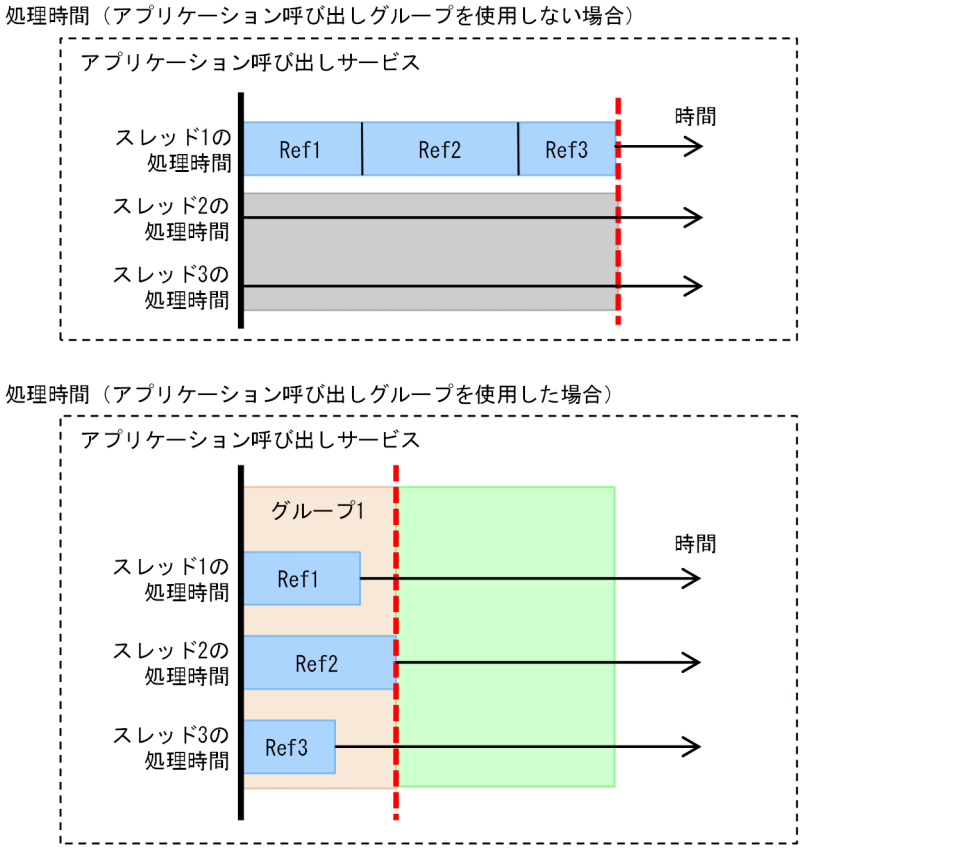
ref 識別子の数が多い場合，アプリケーション呼び出しグループを登録することで，アプリケーション呼び出しを行うスレッドの利用効率が上がります。その結果，スループットを向上できます。

アプリケーション呼び出しグループを登録する場合，次の指針に従うことを推奨します。

- アプリケーション呼び出し制御情報の実行間隔と障害復旧間隔の差が少ない ref 識別子ごとに，グループを分ける。
- 単位時間当たりに発生する各グループの作業数の差が少なくなるように，グループを分ける。

アプリケーション呼び出しグループの登録によるスループットの向上の仕組みを，次の図に示します。

図 C-3　アプリケーション呼び出しサービスのスループットの向上例（アプリケーション呼び出しグループを登録する場合）



注
この図では，ref識別子「Ref1」～「Ref3」は，それぞれ1つの作業を含む想定です。

ヒント

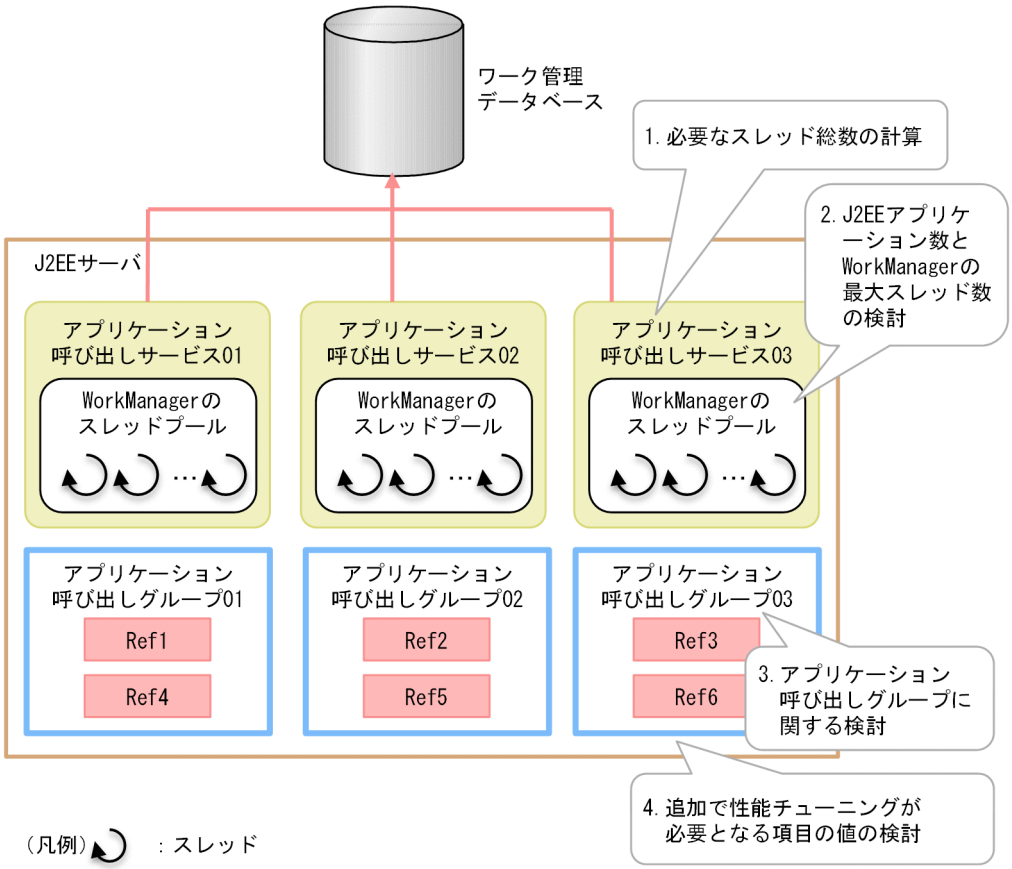
アプリケーション呼び出しグループを登録する設定方法については，「付録 C.6　チューニングの設定方法」の「(3)　アプリケーション呼び出しグループの設定」を参照してください。

付録 C.3　チューニング項目の設計手順の概要

アプリケーション呼び出しサービスの性能チューニングは，次に示す手順で設計します。

1. 必要なスレッド総数の計算
2. J2EE アプリケーション数と WorkManager の最大スレッド数の検討
3. アプリケーション呼び出しグループに関する検討
4. 追加でチューニングが必要となる項目の値の検討

図 C-4　チューニング項目の設計手順



付録 C.4　この設計例で想定するシステム要件

ここでは，想定したシステム要件の場合の設計例を示します。

アプリケーション呼び出しのスループットの目標の決定

アプリケーション呼び出しサービスの性能チューニングの目標を設定します。

ここでは，「1 時間当たり 50,000 件の案件が投入される環境で，アプリケーション呼び出し対象の作業が増え続けないこと（作業の遷移が停滞しないこと）」を目標に設定したと想定します。

性能チューニング対象の環境の決定

単位時間当たりの案件投入数や J2EE サーバの数など，この例で想定するシステム要件を次の表に示します。

表 C-1　この例で想定するシステム要件

項番	項目	値
1	単位時間当たりの案件の投入数	13.888（単位：件／秒） 1 時間当たりの案件の投入数（50,000）から求めた値です。
2	1 案件がシンクに達するまでに作成される呼び出し対象の作業数の平均値	10（単位：件）
3	REST アプリケーションの平均応答時間	0.4（単位：秒）
4	CSCIW の平均処理時間※	0.1（単位：秒）
5	J2EE サーバの数	1
6	ref 識別子の数	200

注※

アプリケーション呼び出しサービスは一件の作業を呼び出すごとに，アプリケーション呼び出しサービスのトレースファイルに次の内容を出力します。

```
行番号 yyyy/mm/dd hh:mm:ss.sss    tid          message(LANG=ja)
1      2019/06/03 09:55:13.601 … 67D96C9E … mark KDIW63604-I The application call …
       …
2      2019/06/03 09:55:13.602 … 67D96C9E … call Client#method()
3      2019/06/03 09:55:13.635 … 67D96C9E … return Client#method()
       …
4      2019/06/03 09:55:13.636 … 67D96C9E … mark call-application success. …
```

「CSCIW の平均処理時間」は，行番号「1」から行番号「4」で掛かった時間から，行番号「2」から行番号「3」で掛かった時間を差し引くことで算出できます。なお，行番号「2」から行番号「3」に掛かった時間は，REST アプリケーションの応答時間です。

付録 C.5　チューニング項目の設計手順の詳細

アプリケーション呼び出しサービスのチューニング項目の設計例を示します。

関連項目

- 付録 C.3　チューニング項目の設計手順の概要
- 付録 C.4　この設計例で想定するシステム要件

(1)　必要なスレッド総数の計算

発生する作業を単位時間内に処理するために必要な，アプリケーション呼び出しのスレッド総数を計算します。

アプリケーション呼び出しに必要なスレッド総数を求める計算式を次に示します。

```
必要なスレッド総数＝A×B×（C＋D）
```

変数の説明

- A：単位時間当たりの案件の投入数
- B：1 案件がシンクに達するまでに作成される呼び出し対象の作業数の平均値
- C：REST アプリケーションの平均応答時間
- D：CSCIW の平均処理時間

この例では各値に「表 C-1　この例で想定するシステム要件」の値を想定しているため，アプリケーション呼び出しに必要なスレッド総数を求める計算式に，次のとおり値を代入します。その結果，この例での必要なスレッド総数は，小数点以下を切り上げた「70」になります。

```
A×B×（C＋D）
＝13.888×10×（0.4＋0.1）
＝69.444..
＝70
```

(2)　J2EE アプリケーション数と WorkManager の最大スレッド数の検討

アプリケーション呼び出しサービスの J2EE アプリケーション数と WorkManager の最大スレッド数を検討します。

次の計算式を満たすよう，アプリケーション呼び出しサービスの J2EE アプリケーション数と WorkManager の最大スレッド数を決定します。

```
<必要なスレッド総数>≦<J2EEアプリケーション数>×<WorkManagerの最大スレッド数>
```

この例では，「付録 C.5　チューニング項目の設計手順の詳細」の「(1)　必要なスレッド総数の計算」で求めた 70 以上を必要なスレッド総数と想定して，J2EE アプリケーション数と WorkManager の最大スレッド数の組み合わせを検討します。

必要なスレッド総数が 70 以上の場合の，J2EE アプリケーション数と WorkManager の最大スレッド数の組み合わせの例を次に示します。

- J2EE アプリケーション数：8
- WorkManager の最大スレッド数：10（デフォルト値）

(3)　アプリケーション呼び出しグループに関する検討

WorkManager のスレッドの利用効率を上げるため，アプリケーション呼び出しサービスの J2EE アプリケーション数に応じて，アプリケーション呼び出しグループの数を検討します。

アプリケーション呼び出しグループの数の目安は，次に示す計算式で求められます。

〈アプリケーション呼び出しグループの数〉＝〈アプリケーション呼び出しサービスのJ2EEアプリケーション数〉×E

変数の説明

E：1.0～2.0

ヒント

アプリケーション呼び出しグループの数は，アプリケーション呼び出しサービスの J2EE アプリケーション数以上作成する必要があります。アプリケーション呼び出しグループの数が不足していると，アプリケーション呼び出しサービスごとに処理が分散されないことから，スループットが向上しないことがあります。

また，アプリケーション呼び出しグループの数が多過ぎても効果は上がりません。アプリケーション呼び出しグループの数は，多くても，アプリケーション呼び出しサービスの J2EE アプリケーション数の 2 倍程度までが適切です。

上記から決定したアプリケーション呼び出しグループの数に沿うように，「付録 C.2　チューニング項目と効果」の「(3)　アプリケーション呼び出しグループを登録する」で示した指針に従って，グループ定義を検討します。

「表 C-1　この例で想定するシステム要件」で示したとおり，この例では，ref 識別子の数を 200 と想定しています。ref 識別子の数が 200 の場合の，アプリケーション呼び出しグループの構成例を示します。

- ref 識別子の数：200（システム要件）
- アプリケーション呼び出しグループの数：8
- 1 グループ当たりの ref 識別子数：25

なお，アプリケーション呼び出し制御情報に登録したグループの実行間隔当たりの作業発生数が，WorkManager の最大スレッド数よりも少ない場合，WorkManager に空きスレッドが発生するおそれがあります。その場合，スループットが十分に向上しないことがあります。

(4)　追加でチューニングが必要となる項目の値の検討

アプリケーション呼び出しサービスの J2EE アプリケーション数と WorkManager の最大スレッド数のほかに，必要な項目をチューニングします。

アプリケーション呼び出しサービスの J2EE アプリケーション数と WorkManager の最大スレッド数を決めたあとに，値の変更が必要な項目と，各値を求める計算式を次の表に示します。

表 C-2　追加でチューニングが必要となる項目と各値を求める計算式

項番	チューニング対象の項目	計算式	計算結果の例※1
1	データベースコネクション数	<アプリケーション呼び出しサービスのJ2EEアプリケーション数>×<WorkManagerの最大スレッド数>	80
2	TimerService での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※2	<アプリケーション呼び出しサービスのJ2EEアプリケーション数>	8

注※1

アプリケーション呼び出しサービスの J2EE アプリケーション数を 8，および WorkManager の最大スレッド数を 10 と想定した場合の計算結果です。

注※2

TimerService での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，J2EE サーバのユーザプロパティファイル（usrconf.properties）の次のプロパティに設定します。

```
ejbserver.ejb.timerservice.maxCallbackThreads=<1～100の整数>
```

詳細については，マニュアル『Cosminexus アプリケーションサーバ リファレンス 定義編(サーバ定義)』の「J2EE サーバ用ユーザプロパティを設定するパラメタ」を参照してください。

関連項目

- 3.1.2　データベースコネクション数を見積もる

付録 C.6　チューニングの設定方法

関連項目

- 付録 C.2　チューニング項目と効果

(1)　WorkManager の最大スレッド数の設定

アプリケーション呼び出しを行うスレッド数を増やすために，WorkManager の最大スレッド数を増やします。

WorkManager の最大スレッド数の設定方法の詳細については，「9.4　WorkManager の最大スレッド数を変更する」を参照してください。

(2) アプリケーション呼び出しサービスの稼働数の設定

1 つの J2EE サーバでアプリケーション呼び出しサービスを複数稼働させる場合に，アプリケーション呼び出しサービスを追加します。ここでは，アプリケーション呼び出しサービスの追加および削除の手順について説明します。

アプリケーション呼び出しサービスの追加手順

アプリケーション呼び出しサービスの追加手順を次に示します。

1 つの J2EE サーバでアプリケーション呼び出しサービスを複数稼働させる場合は，実行環境を構築したあとで，次に示す手順を繰り返し実施してください。

1. アプリケーション呼び出しサービスをインポートする。

   cjimportapp コマンドを使用して，アプリケーション呼び出しサービスをインポートします。cjimportapp コマンドの詳細については，マニュアル『Cosminexus アプリケーションサーバ　リファレンス コマンド編』を参照してください。

   ■cjimportapp コマンドの指定形式

   ```
   cjimportapp 〈サーバ名称〉 -f 〈earファイルパス〉
   ```

   > **メモ**
   >
   > 1 つの J2EE サーバに同一のear ファイルを重複してインポートできません。

   アプリケーション呼び出しサービスのear ファイルは，次に示すディレクトリに格納されています。

   - Windows の場合

     ```
     <CSCIWのインストールディレクトリ>¥lib
     ```

   - UNIX の場合

     ```
     /opt/hitachi/CSCIW/lib
     ```

   なお，アプリケーション呼び出しサービスには，インポートしたear ファイルごとに異なる J2EE アプリケーション名が設定されます。

   アプリケーション呼び出しサービスのear ファイル名と，対応する J2EE アプリケーション名の一覧を次の表に示します。

表 C-3　アプリケーション呼び出しサービスの ear ファイル名および J2EE アプリケーション名

項番	ear ファイル名	J2EE アプリケーション名
1	csciwapsrv.ear	CSCIWBpmnAPService01
2	csciwapsrv02.ear	CSCIWBpmnAPService02
3	csciwapsrv03.ear	CSCIWBpmnAPService03

項番	ear ファイル名	J2EE アプリケーション名
4	csciwapsrv04.ear	CSCIWBpmnAPService04
5	csciwapsrv05.ear	CSCIWBpmnAPService05
6	csciwapsrv06.ear	CSCIWBpmnAPService06
7	csciwapsrv07.ear	CSCIWBpmnAPService07
8	csciwapsrv08.ear	CSCIWBpmnAPService08

2. アプリケーション呼び出しサービスのプロパティを設定する。

cjgetappprop コマンドおよびcjsetappprop コマンドを使用して，アプリケーション呼び出しサービスのプロパティを設定します。cjgetappprop コマンドおよびcjsetappprop コマンドの詳細については，マニュアル『Cosminexus アプリケーションサーバ　リファレンス コマンド編』を参照してください。

まず，cjgetappprop コマンドを実行して，J2EE アプリケーションの属性ファイル（XML ファイル）を取得します。

■cjgetappprop コマンドの指定形式

```
cjgetappprop <サーバ名称> -name <J2EEアプリケーション名> -type all -c <属性ファイルパス>
```

取得した XML ファイルをテキストエディタで開きます。XML ファイル内の次に示すresource-ref 要素のlinked-to 要素に，DB Connector のデータソース表示名を記載します。

```
<resource-ref>
   <description xml:lang="ja-JP"></description>
   <res-ref-name>jdbc/CSCIWApServiceDataSource</res-ref-name>
   <res-type>javax.sql.DataSource</res-type>
   <res-auth>Container</res-auth>
   <res-sharing-scope>Shareable</res-sharing-scope>
   <linked-to>データソース表示名</linked-to>
 </resource-ref>
```

編集した XML ファイルを指定したcjsetappprop コマンドを実行して，アプリケーション呼び出しサービスのプロパティを設定します。

■cjsetappprop コマンドの指定形式

```
cjsetappprop <サーバ名称> -name <J2EEアプリケーション名> -type all -c <属性ファイルパス>
```

3. アプリケーション呼び出しサービスを開始する。

cjstartapp コマンドを使用して，アプリケーション呼び出しサービスを開始します。cjstartapp コマンドの詳細については，マニュアル『Cosminexus アプリケーションサーバ　リファレンス コマンド編』を参照してください。

■cjstartapp コマンドの指定形式

```
cjstartapp <サーバ名称> -name <J2EEアプリケーション名>
```

アプリケーション呼び出しサービスの削除手順

性能チューニングのために追加したアプリケーション呼び出しサービスが不要になった場合に，削除する手順を次に示します。

なお，1 つの J2EE サーバに設定したアプリケーション呼び出しサービスを複数削除する場合は，次に示す手順を繰り返し実施してください。

1. アプリケーション呼び出しサービスを停止する。

   cjstopapp コマンドを使用して，アプリケーション呼び出しサービスを停止します。cjstopapp コマンドの詳細については，マニュアル『Cosminexus アプリケーションサーバ　リファレンス コマンド編』を参照してください。

   ■cjstopapp コマンドの指定形式

   ```
   cjstopapp 〈サーバ名称〉 -name 〈J2EEアプリケーション名〉
   ```

2. アプリケーション呼び出しサービスを削除する。

   cjdeleteapp コマンドを使用して，アプリケーション呼び出しサービスを削除します。cjdeleteapp コマンドの詳細については，マニュアル『Cosminexus アプリケーションサーバ　リファレンス コマンド編』を参照してください。

   ■cjdeleteapp コマンドの指定形式

   ```
   cjdeleteapp 〈サーバ名称〉 -name 〈J2EEアプリケーション名〉
   ```

注意事項

- 同一プロセス内のすべてのアプリケーション呼び出しサービスに対して，同じ共通設定ファイルおよびアプリケーション呼び出し情報ファイルを使用してください（アプリケーション呼び出しサービスごとに異なるファイルを使用しないでください）。
- 次に示すファイルの内容を変更した場合は，同一プロセス内のすべてのアプリケーション呼び出しサービスを再開始してください。
  - 共通設定ファイル
  - アプリケーション呼び出し情報ファイル
  - アプリケーション呼び出しで使用するヘッダファイル
  - アプリケーション呼び出しで使用するスタイルシート

関連項目

- 7.　ワークフローシステムを構築する

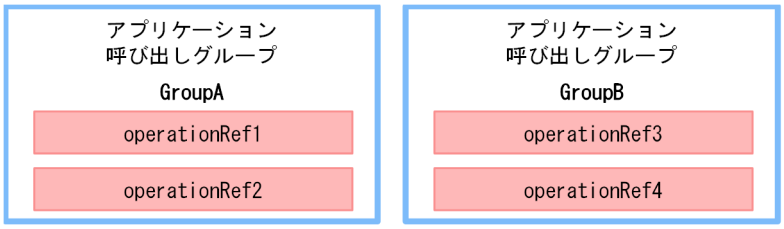
(3)　アプリケーション呼び出しグループの設定

WorkManager のスレッドの利用効率を上げるために，アプリケーション呼び出しグループを作成します。ここでは，アプリケーション呼び出しグループの設定および削除の手順について説明します。

アプリケーション呼び出しグループの構成例

アプリケーション呼び出しグループの設定手順および削除手順の説明で想定する，アプリケーション呼び出しグループの構成例を次の図に示します。

図 C-5　アプリケーション呼び出しグループの構成例



アプリケーション呼び出しグループの設定手順

アプリケーション呼び出し制御情報，およびアプリケーション呼び出しグループ定義を追加して，アプリケーション呼び出しグループを設定します。

1. アプリケーション呼び出し制御情報を追加する。

   アプリケーション呼び出し制御情報ファイルを作成し，アプリケーション呼び出し制御情報を追加します。アプリケーション呼び出し制御情報ファイルについては，マニュアル『uCosminexus Service Coordinator Interactive Workflow コマンド』の「ciwmngap（アプリケーション呼び出し制御情報の管理)」を参照してください。

   ■アプリケーション呼び出し制御情報ファイルの作成例

   ```
   #UPDATEOPTION,REFTYPE,REF,EXECUTEINTERVAL,RETRYINTERVAL,RETRYCOUNT,WORKITEMMAX,RECOVERYTI
   ME
   U,grp,GroupA,20,150,0,10000,1000
   U,grp,GroupB,40,200,0,10000,3000
   ```

   [説明]

   　U：アプリケーション呼び出し制御情報の追加を意味しています。

   次に，アプリケーション呼び出し制御情報ファイルを指定したciwmngap コマンドを実行して，アプリケーション呼び出し制御情報を反映させます。ciwmngap コマンドについては，マニュアル『uCosminexus Service Coordinator Interactive Workflow コマンド』を参照してください。

   ■ciwmngap コマンドの指定形式

   ```
   ciwmngap -sid <システムID> -chg -apdf <アプリケーション呼び出し制御情報ファイル名>
   ```

2. アプリケーション呼び出しグループ定義を追加する。

アプリケーション呼び出しグループ定義ファイルを作成し，アプリケーション呼び出しグループ定義を追加します。アプリケーション呼び出しグループ定義ファイルについては，マニュアル『uCosminexus Service Coordinator Interactive Workflow コマンド』の「ciwmngapgrp（アプリケーション呼び出しグループの管理)」を参照してください。

■アプリケーション呼び出しグループ定義ファイルの作成例

```
#UPDATEOPTION,REFTYPE,REF,GROUPNAME
U,ope,operationRef01,GroupA
U,ope,operationRef02,GroupA
U,ope,operationRef03,GroupB
U,ope,operationRef04,GroupB
```

[説明]

U：アプリケーション呼び出しグループ定義の追加を意味しています。

次に，アプリケーション呼び出しグループ定義ファイルを指定したciwmngapgrp コマンドを実行して，アプリケーション呼び出しグループ定義の内容を反映させます。ciwmngapgrp コマンドについては，マニュアル『uCosminexus Service Coordinator Interactive Workflow コマンド』を参照してください。

■ciwmngapgrp コマンドの指定形式

```
ciwmngapgrp -sid 〈システムID〉 -chg -apgf 〈アプリケーション呼び出しグループ定義ファイル〉
```

重要

アプリケーション呼び出しグループ定義に登録したグループは，アプリケーション呼び出し制御情報にも登録してください。アプリケーション呼び出し制御情報に登録しないと，アプリケーション呼び出しグループ定義の該当するグループに登録した ref 識別子が呼び出されません。

アプリケーション呼び出しグループの削除手順

性能チューニングのために設定したアプリケーション呼び出しグループが不要になった場合，アプリケーション呼び出し制御情報，およびアプリケーション呼び出しグループ定義を削除して，アプリケーション呼び出しグループを削除します。

1. アプリケーション呼び出しグループ定義を削除する。

アプリケーション呼び出しグループ定義ファイルを作成し，アプリケーション呼び出しグループ定義を削除します。アプリケーション呼び出しグループ定義ファイルについては，マニュアル『uCosminexus Service Coordinator Interactive Workflow コマンド』の「ciwmngapgrp（アプリケーション呼び出しグループの管理)」を参照してください。

■アプリケーション呼び出しグループ定義ファイルの作成例

```
#UPDATEOPTION,REFTYPE,REF,GROUPNAME
D,ope,operationRef01
D,ope,operationRef02
D,ope,operationRef03
D,ope,operationRef04
```

［説明］

D：アプリケーション呼び出しグループ定義の削除を意味しています。

次に，アプリケーション呼び出しグループ定義ファイルを指定したciwmngapgrp コマンドを実行して，アプリケーション呼び出しグループ定義の内容を反映させます。ciwmngapgrp コマンドについては，マニュアル『uCosminexus Service Coordinator Interactive Workflow コマンド』を参照してください。

■ciwmngapgrp コマンドの指定形式

```
ciwmngapgrp -sid 〈システムID〉 -chg -apgf 〈アプリケーション呼び出しグループ定義ファイル〉
```

2. アプリケーション呼び出し制御情報を削除する。

アプリケーション呼び出し制御情報ファイルを作成し，アプリケーション呼び出し制御情報を削除します。アプリケーション呼び出し制御情報ファイルについては，マニュアル『uCosminexus Service Coordinator Interactive Workflow コマンド』の「ciwmngap（アプリケーション呼び出し制御情報の管理)」を参照してください。

■アプリケーション呼び出し制御情報ファイルの作成例

```
#UPDATEOPTION,REFTYPE,REF,EXECUTEINTERVAL,RETRYINTERVAL,RETRYCOUNT,WORKITEMMAX,RECOVERYTI
ME
D,grp,GroupA
D,grp,GroupB
```

［説明］

D：アプリケーション呼び出し制御情報の削除を意味しています。

次に，アプリケーション呼び出し制御情報ファイルを指定したciwmngap コマンドを実行して，アプリケーション呼び出し制御情報を反映させます。ciwmngap コマンドについては，マニュアル『uCosminexus Service Coordinator Interactive Workflow コマンド』を参照してください。

■ciwmngap コマンドの指定形式

```
ciwmngap -sid 〈システムID〉 -chg -apdf 〈アプリケーション呼び出し制御情報ファイル名〉
```

(4)　追加でチューニングが必要となる項目の設定

追加でチューニングが必要となる項目の設定方法については，「付録 C.5　チューニング項目の設計手順の詳細」の「(4)　追加でチューニングが必要となる項目の値の検討」を参照してください。

付録 D　ビジネスプロセスモニタとは

ビジネスプロセスモニタについて説明します。

ビジネスプロセスモニタは，ビジネスプロセスオペレータに含まれています。モニタ画面の表示方法については，「10.3.1　ビジネスプロセスオペレータを実行する」，および「10.3.2　ビジネスプロセスオペレータの画面構成」を参照してください。

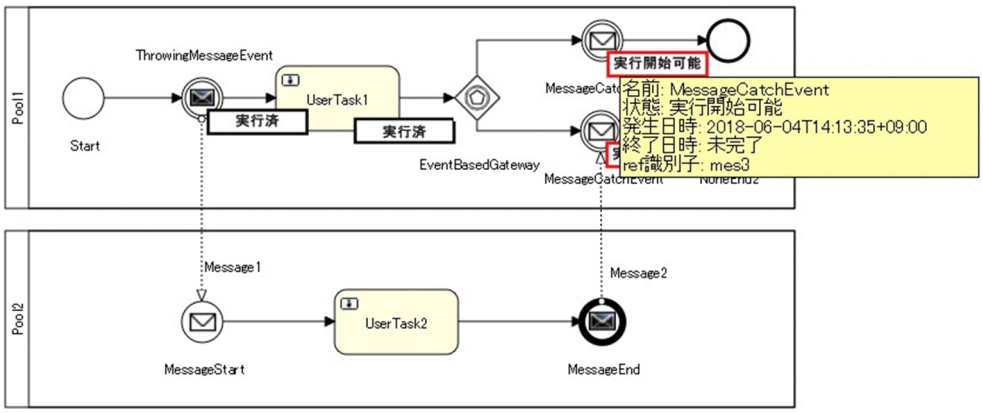
付録 D.1　ビジネスプロセスモニタの表示

ビジネスプロセスモニタの表示例

ビジネスプロセスモニタには，ビジネスプロセス定義（XML）から生成した BPMN のプロセス図と，案件のステータス情報が表示されます。

ビジネスプロセスモニタの例を次に示します。

図 D-1　ビジネスプロセスモニタの例



ビジネスプロセス定義ファイルの内容を基に，BPMN のプロセス図がビジネスプロセスモニタに表示されます。ビジネスプロセス定義に未サポートの要素が含まれる場合は，エラーが表示されます。サポートしている BPMN 要素については，「1.3.1　BPMN 連携機能で使用できる BPMN 要素」を参照してください。

案件のステータス情報は，REST サービスを介して取得され，ビジネスプロセスモニタに表示されます。

ビジネスプロセスモニタに表示されるステータス情報

ビジネスプロセスモニタに表示されるステータス情報の一覧を次の表に示します。

表 D-1　ビジネスプロセスモニタに表示されるステータス情報

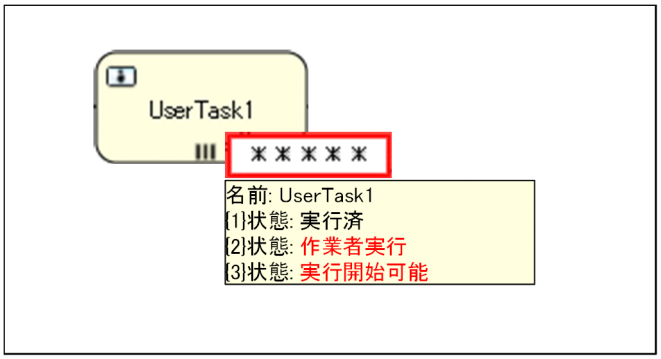
項番	ステータス情報（英語）	ステータス情報（日本語）	説明
1	Name	名前	アドホック・サブプロセス，タスク，またはイベントの名前です。

項番	ステータス情報（英語）	ステータス情報（日本語）	説明
2	Status	状態	詳細については，マニュアル『uCosminexus Service Coordinator Interactive Workflow AP 開発ガイド』の「付録 B.1(6)　作業属性」を参照してください。
3	Participant	作業者	
4	Creation Date	発生日時	
5	End Date	終了日時	
6	Identifier	ref 識別子	次に示すどれかの値を表示します。 • operationRef • messageRef • errorRef

マルチインスタンスの場合のステータス情報の表示例

マルチインスタンスの場合，BPMN 要素のステータスは，インスタンスごとの状態が表示されます。表示例を次に示します。

図 D-2　マルチインスタンスの場合のステータス情報の表示例



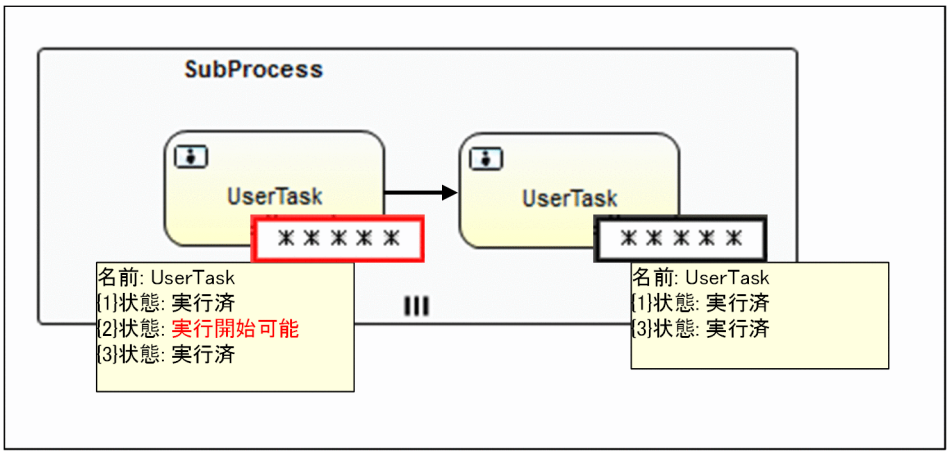
マルチインスタンスの場合，ステータスメッセージは"＊＊＊＊＊"と表示されます。

マルチインスタンスの場合の注意事項

注意事項 1

展開されたサブプロセス（マルチインスタンス）の場合，遷移していないインスタンスのステータスは表示されません。そのため，次に示す表示例のように，"{}"内に表示されるマルチインスタンスインデクスの値が連番にならない場合があります。

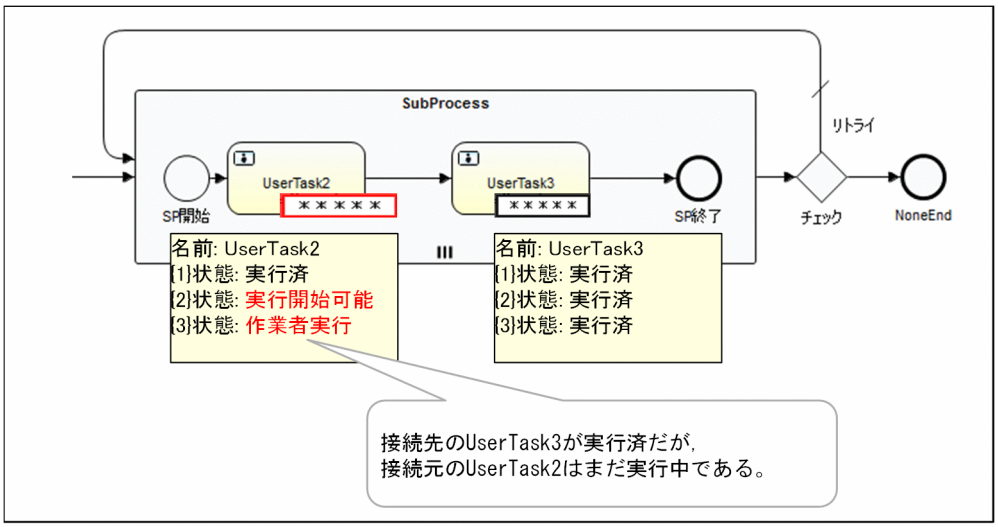
図 D-3　サブプロセス（マルチインスタンス）の場合のステータス情報の表示例



注意事項 2

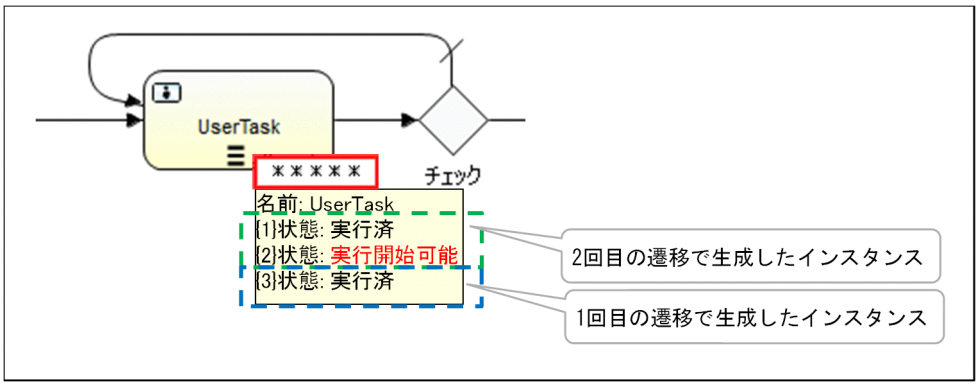
ステータスを取得した結果，"{}"内に表示されるマルチインスタンスインデクスの値が同一のインスタンスを複数取得した場合，マルチインスタンスではない BPMN 要素と同様に，発生日時がいちばん新しいフローノードの状態が表示されます。そのため，複数回実行するマルチインスタンスの場合の BPMN 要素については，次に示す表示例のように，BPMN 要素の状態が前後することがあります。

図 D-4　繰り返し実行時のステータス情報の表示例



また，マルチインスタンスの BPMN 要素を複数回実行するビジネスプロセスの場合，次に示す表示例のように，生成したタイミングが異なるインスタンスが混ざって表示されることがあります。

図 D-5　複数回実行時のステータス情報の表示例



注意事項 3

実行回数が 0 回であるマルチインスタンスの BPMN 要素では，ステータスが表示されません。

注意事項 4

インスタンスの状態は，最大で 50 件まで表示されます。インスタンスが 51 件以上ある場合，末尾に「...」（半角ピリオド 3 個）が表示されます。51 件目以降の状態を確認する場合は，ビジネスプロセスオペレータを使用してください。

注意事項 5

BPMN 要素の位置，ブラウザのサイズ，またはズーム倍率によっては，表示項目がすべては表示できない場合があります。その場合は，ブラウザのサイズ変更やズームアウトをするか，またはビジネスプロセスオペレータを使用して確認してください。

付録 E　ビジネスプロセス開発環境のバージョンアップ

ビジネスプロセス開発環境をバージョンアップする手順を説明します。

付録 E.1　ビジネスプロセス開発環境をバージョンアップする

ビジネスプロセスの開発環境をバージョンアップする場合の手順を示します。

操作手順

1. BPMN エディタをアンインストールする。
   Eclipse から BPMN エディタをアンインストールします。
2. uCosminexus Business Process Developer をアンインストールする。
   旧バージョンの uCosminexus Business Process Developer をアンインストールします。
3. uCosminexus Business Process Developer をインストールする。
   最新バージョンの uCosminexus Business Process Developer をインストールします。
4. BPMN エディタをインストールする。
   Eclipse に BPMN エディタをインストールします。

関連項目

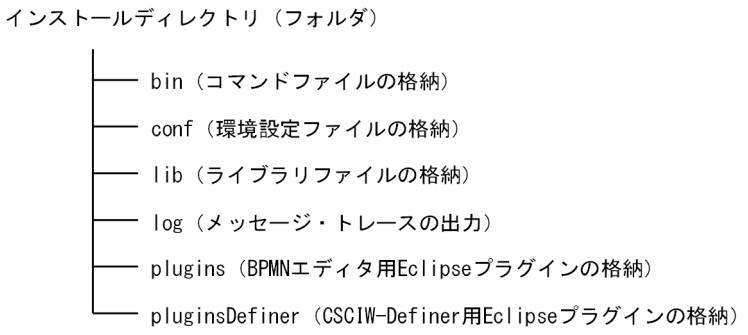
- 2.2　BPMN エディタをインストールする
- 2.8　BPMN エディタをアンインストールする

付録Ｆ　インストールディレクトリ（フォルダ）

CSCIW をインストールした場合のディレクトリ（フォルダ）構成については，マニュアル『uCosminexus Service Coordinator Interactive Workflow システム構築・運用ガイド』の「付録Ｋ　インストールディレクトリ（フォルダ）」を参照してください。

uCosminexus Business Process Developer をインストールした場合のフォルダ構成を，次の図に示します。

インストール時に作成するディレクトリ（フォルダ）



bin

uCosminexus Business Process Developer の各機能の実行形式ファイル，およびコマンドの実行形式ファイルが格納されています。

conf

環境構築時に作成されるコンフィグレーション情報などが格納されています。

lib

uCosminexus Business Process Developer として共通に使用するライブラリが格納されています。

log

uCosminexus Business Process Developer が出力したメッセージやトレース情報を記録したファイルが格納されています。

plugins

BPMN エディタ用の Eclipse プラグインが格納されています。

pluginsDefiner

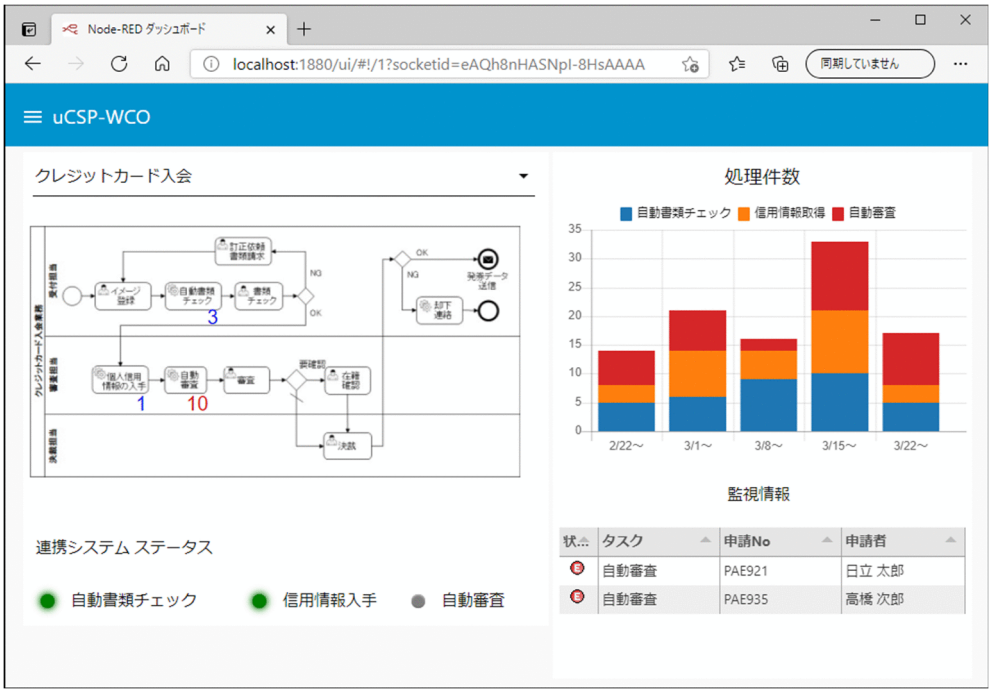
CSCIW-Definer 用の Eclipse プラグインが格納されています。

付録 G　実行履歴の可視化

案件，業務ステップ，および作業の情報を管理するワーク管理データベースは，テーブル構造やデータ型を公開しています。そのため，OSS（Open Source Software）や市販の BI（Business Intelligence）ツールを活用することで，業務データと合わせてグラフィカルに表示し，業務状況の把握や分析ができます。

Node-RED による可視化の例を次に示します。

図 G-1　Node-RED による可視化の例



付録 H　各バージョンの変更内容

付録 H.1　03-20 での変更内容

- 適用 OS に Windows Server 2022 を追加しました。
- ciwmngcr コマンドの BPMN 連携機能での使用可否を変更しました。
- 「登録済みのビジネスプロセス定義を変更する」の説明に，次の説明を追加しました。
  - ビジネスプロセス定義の変更例
  - BPMN エディタで変更可能なビジネスプロセス定義の詳細
- Cosminexus に同梱されている JAX-RS ライブラリについての注意事項を修正しました。
- CSCIWManagementServer の設定手順にコンテキストルートを設定する手順を追加しました。
- REST API のリクエスト例，レスポンス例を修正しました。
- REST API のリクエスト例を修正しました。
- ワーク管理データベースを移行する手順について，CSCIW03-11 から CSCIW03-20 へ移行する場合の手順を追加しました。
- BPMN ビジネスプロセス定義ファイルの移行に関する注意事項を修正しました。
- ワークフローシステムをバージョンアップする手順に，データ移行だけを繰り返し行う手順を追加しました。
- ゲートウェイの実行履歴をワーク管理データベースに保存する方法について，説明を追加しました。

付録 H.2　03-11 での変更内容

- Windows 11 に対応しました。
- 対応する製品から Oracle Database 18c を削除しました。
- 共通設定ファイルのTerminateCompatMode パラメタにfalse を指定している場合の動作を追加しました。
- BPMN 要素の状態遷移モデルの説明を追加しました。
- ciwchgapwork コマンドの実行例と手順を変更しました。
- REST API で操作できるリソースとして業務ステップ定義を追加しました。
- 共通設定ファイルに指定するパラメタに次のパラメタを追加しました。
  - AppCallServiceInitialInterval
  - TerminateCompatMode

- アプリケーション呼び出し情報ファイルに指定するパラメタにrest.request.idempotency を追加しました。
- ワーク管理システムのバージョンアップの概念図を追加しました。
- ワーク管理データベースを移行する手順について，CSCIW03-10 から CSCIW03-11 へ移行する場合の手順を追加しました。

付録 H.3　03-10 での変更内容

- 次の図を変更しました。
  - 各環境での BPMN 連携機能の機能構成
- BPMN エディタ上でciwtransbpmn コマンドを実行して，BPMN ビジネスプロセス定義ファイルを CSCIW のビジネスプロセス定義ファイルに変換できるようになりました。また，ビジネスプロセス定義の，ワーク管理データベースへの登録とワーク管理データベースからの削除を，BPMN エディタからできるようになりました。
- アプリケーション呼び出し情報ファイルに次のパラメタを追加しました。
  - rest.request.connect.timeout
- コマンドを使用したビジネスプロセス定義の登録，削除，変換の説明を変更しました。
- 実行環境にビジネスプロセス定義ファイルを転送する手順を変更しました。
- ビジネスプロセス定義の，ワーク管理データベースへの登録とワーク管理データベースからの削除を，案件運用操作からできるようになりました。
  BPMN ビジネスプロセス定義ファイルの取得の REST API で，ワーク管理データベースから BPMN ビジネスプロセス定義ファイルの内容を取得できるようになりました。
- BPMN ビジネスプロセス定義ファイルのワーク管理データベースへの登録とワーク管理データベースからの削除を，ciwmbgbp コマンドでできるように手順を変更しました。
- BPMN エディタのアンインストール手順を追加しました。
- 構成を変更しました。
- リトライ時の複数回呼び出しに関する説明を修正しました。
- 節，項の構成に合わせてワークフローシステムの構築の流れの図を変更しました。
- 「HiRDB の SQL 最適化についての注意事項」の記載を追加しました。
- 次の 2 とおりの手順で場合分けしました。
  - 運用管理ポータルを使用して設定する場合
  - 定義ファイルおよびサーバ管理コマンドを使用して設定する場合
- Cosminexus の起動の手順を追加した。それに伴い，環境変数の取り込み手順に含まれていた Cosminexus の起動の説明を削除しました。
- 次の 2 とおりの手順で場合分けしました。

  - 運用管理ポータルを使用する場合
  - サーバ管理コマンドを使用する場合
- 運用管理ポータルで設定した場合の記述を追加しました。
- 付録に記載していた CSCIW のバージョンアップ手順のうち，「ビジネスプロセス開発環境のバージョンアップ」以外の手順を 17 章に移動した。また，記述内容を修正しました。
- BPMN ビジネスプロセス定義テーブル（テーブル名：<SYSTEMID>_BPMN_PROCESS_DEF）を追加しました。
- ビジネスプロセス開発環境のバージョンアップ手順を変更しました。
- Kibana による可視化の例から Node-RED による可視化の例に変更しました。

付録 H.4　03-00 での変更内容

- 案件運用操作の更新系の操作をできるようにしました。それに伴い，案件運用操作の設定手順を追加しました。
- BPMN 要素として，ビジネスルールタスクを追加しました。
- ユーザタスクの機能の説明を変更しました。
- アプリケーション呼び出しサービスから呼び出す業務アプリケーションとして，Java オブジェクトを追加しました。
- アプリケーション呼び出しサービスの動作の概要の説明を追加しました。
- アプリケーション呼び出しサービス自身の障害の注意事項として，障害復旧間隔の求め方の説明を追加しました。
- REST アプリケーション呼び出しの通信タイムアウトの設定についての説明を変更しました。
- BPMN エディタのインストール手順を変更しました。
- ciwtransbpmn コマンドを使用して，BPMN ビジネスプロセス定義ファイルを変換する場合の前提条件を削除しました。
- ビジネスプロセス定義の定義内容を，BPMN エディタから CSV ファイルにエクスポートできるようにしました。
- マニュアル『uCosminexus Service Coordinator Interactive Workflow システム構築・運用ガイド』に記述を集約したため，アプリケーション呼び出しサービスに関する次の説明を削除しました。
  - データベースコネクション数の計算式
  - <SYSTEMID>_APPLICATION_LOCK_INFO テーブル
  - <SYSTEMID>_APPLICATION_LOCK_GROUP テーブル
- REST API を使用した業務処理の実装例を，JAX-RS2.0 の API に対応した実装例に変更しました。
- 業務アプリケーションをコンパイルする場合のクラスパスの設定方法を変更しました。

また，Java API 利用時に，uCosminexus Business Process Developer をインストールした環境で業務アプリケーションをコンパイルするときにクラスパスに指定する JAR ファイルの説明を追加しました。

- Java API 利用時の BPMN 連携機能の更新系 API 共通の注意事項の説明を削除しました。
- アプリケーション呼び出しサービスのレスポンスボディを省略する場合に指定するステータスコードの説明を追加しました。
- アプリケーション呼び出しサービスの REST 通信に関する設定の説明を変更しました。
- データベースへのアクセス権限の付与および削除で，SQL スクリプトファイルを実行する手順の記述を変更しました。
- CSCIW の実行環境を初期化時に，セットアッププロパティファイルに「UseApplicationCallService=true」の指定は不要であることを追記しました。
- Cosminexus の J2EE サーバを，V9 互換モードで動作させる場合の注意書きを追加しました。
- コンテナ拡張ライブラリに取り込む JAR ファイルを変更しました。
- DB Connector のリソースアダプタの表示名に CSCIW のデフォルトの値を設定することで，各アプリケーションのデータソース表示名の設定手順を省略できるようにしました。
- REST サービスのリダイレクタの設定の説明をリバースプロシキを配置する設定の説明に変更しました。
- ビジネスプロセスオペレータの設定手順に，リバースプロシキを配置する設定の説明を追加しました。
- ビジネスプロセスオペレータの画面でリスト型プロセスデータを指定する方法について追加しました。
- 次の設定ファイルを追加しました。
  - uCosminexus Business Process Developer 設定ファイル
- REST API のユーザ記述子の説明を変更しました。
- 共通設定ファイルに次のパラメタを追加しました。
  - AppCallServiceDebugRestBody
- 共通設定ファイルの次のパラメタの指定を省略したときの仮定値を変更しました。
  - RestServiceTraceFileNum
  - RestServiceTraceFileSize
  - AppCallServiceTraceFileNum
  - AppCallServiceTraceFileSize
- アプリケーション呼び出し情報ファイル，およびコールアクティビティ情報ファイルの組み込み変数「案件処理期限」の説明に，代入先についての説明を追加しました。
- アプリケーション呼び出し情報ファイルの次のキーに，注意事項を追加しました。
  - rest.request.header.filepath
  - rest.request.stylesheet.filepath
  - rest.response.stylesheet.filepath

- アプリケーション呼び出し情報ファイルの「rest.request.body.key.<key 要素値>」は，rest.request.method が DELETE の場合には指定できないように変更しました。
- REST サービスおよびアプリケーション呼び出しサービスのトレースは各サービス用のトレースファイルに出力される旨，説明を変更しました。
- アプリケーション呼び出しサービスの性能チューニングでの，CSCIW の平均処理時間の求め方の説明を追加しました。
- ビジネスプロセスモニタに表示されるステータス情報にアドホック・サブプロセスを追加しました。
- CSCIW 02-30 から CSCIW 03-00 へ移行する場合の手順を追加しました。
- CSCIW のバージョンアップ手順に，コンテナ拡張ライブラリに JAR ファイルを取り込む手順を追加しました。
- CSCIW のインストールディレクトリのフォルダ構成を変更しました。

付録 H.5　02-30 での変更内容

- Firefox および Chrome に対応しました。
- ワーク管理データベースとして Oracle に対応しました。
- 案件運用操作で運用状況の参照ができるようにしました。それに伴い，案件運用操作に関する設定方法および操作方法についての記述を追加しました。
- ユーザタスクの作業者を動的に割り当てられるようにしました。また，作業者の決定方法についての説明を追加しました。
- BPMN 要素としてアドホック・サブプロセスを追加しました。
- 完了条件（completionCondition）の設定を省略した場合の動作についての説明を追加しました。
- リトライ対象の作業が API やコマンドで「実行開始可能」状態に変更された場合のリトライ回数についての説明を追加しました。
- BPMN ビジネスプロセス定義ファイルの作成時の，次の指定値のチェックのタイミングを変更しました。
  - Loop cardinality プロパティ（マルチインスタンスを使用する場合）
  - 「Name プロパティ値_Id プロパティ値」の規則で生成される文字列

  また，タイマーイベントの設定に関する規則を追加しました。
- BPMN 要素の配置に関する規則を追加しました。
- CSCIW のセットアッププロパティファイルに必要な設定を「BpmnMode=true」に変更しました。
- アプリケーション呼び出しサービスのデータベースコネクション数の計算式を変更しました。
- REST API で業務処理を実装する際の記述例を変更しました。
- Cosminexus V9.0 以前で使用していた Server Plug-in の操作の代替手段の説明を削除しました。

- REST API で，リクエストボディのパラメタをすべて省略する場合の説明を追加しました。また，リクエストボディを省略する場合の注意事項を追加しました。
- Cosminexus アプリケーションサーバの場合だけ，JSON の値に指定された null が空文字として扱われるという説明に変更しました。
- REST API の記載順序を変更しました。
- Java API の`allocateWIEx` メソッドの「レーン名」を「作業者 ID」に変更しました。
- 共通設定ファイルのエンコーディングについての説明を変更しました。
- REST API のフィルター条件とソート条件に指定された内容にキーワードが含まれているかチェックする，次のパラメタを追加しました。
  - `RestServiceParamStatementCheck`
  - `RestServiceParamPreventStatements`
- ワーク管理データベースを移行する手順について，CSCIW 02-20 から CSCIW 02-30 へ移行する場合の手順を追加しました。
- 実行履歴の可視化についての説明を追加しました。

付録 H.6　02-20 での変更内容

- BPMN 連携機能を使用する場合の，システム構成モデルの説明を追加しました。
- BPMN 連携機能で使用できる CSCIW の機能範囲に，`ciwchgapwork` コマンドを追加しました。
- 次の BPMN 要素を追加しました。
  - 開始（タイマー）
  - キャッチ（タイマー）
  - 境界中断（タイマー）
  - 境界非中断（タイマー）
  - イベント・サブプロセス中断開始（タイマー）
  - イベント・サブプロセス非中断開始（タイマー）
  - 折りたたまれたサブプロセス（シーケンシャルマルチインスタンス）
  - 折りたたまれたサブプロセス（パラレルマルチインスタンス）
  - 展開されたサブプロセス（シーケンシャルマルチインスタンス）
  - 展開されたサブプロセス（パラレルマルチインスタンス）
- サブプロセス（マルチインスタンス）の説明を追加しました。
- タイマーイベントの説明を追加しました。
- 次に示すプロセスデータの利用手段を追加し，また表全体の記述を変更しました。

    - BPMN ビジネスプロセス定義のタイマーイベントに設定するタイマールールの記述
    - リスト型プロセスデータのインデクスを取得する API の利用
- サブプロセス（マルチインスタンス）内に定義された分岐条件についての記述を追加しました。
- 次の場合のプロセスデータ使用例を追加しました。
    - マルチインスタンスでの作業実行時
    - マルチインスタンスの場合の検索時
- BPMN 要素ごとのアプリケーション呼び出しが行われるまでの処理の流れの説明を追加しました。
- アプリケーション呼び出しの一時抑止と解除の方法の説明を追加しました。
- リクエストまたはレスポンスのデータスキーマとして JSON を追加しました。
- 次の数の求め方を追加しました。
    - 業務ステップ数
    - 作業数
    - 制御ノード定義の合計
    - 制御ノード定義ごとの遷移元アローの合計
- BPMN 連携ライブラリの参照系 API を使用した場合の実装例を追加しました。
- 次の Java API を追加しました。
    - `allocateWIEx`
    - `freeWI`
    - `setProcessData`
    - `createAndStartPIForTimer`
    - `setDeadlineForTimer`
- ビジネスプロセスオペレータの機能を次のとおり変更しました。
    - ビジネスプロセスオペレータの検索画面で，検索条件の発生日時で時，分，秒を指定できるようになった。
    - 状態変更画面，およびメッセージ送信画面からキーの値を変更できなくなった。
    - プロセスデータ画面で，指定した案件のプロセスデータの値を更新できるようになった。
- リトライ対象から外れた作業を呼び出し対象に戻す方法の説明を追加しました。
- 次の API を追加しました。
    - 条件に一致する作業の作業者割り当てと着手をする
    - 着手した作業を返却する
    - プロセスデータを登録する
    - リスト型プロセスデータのインデクスを取得する

    - ビジネスプロセス定義名から案件を取得する
    - フローノード一覧を取得する
    - フローノード定義一覧を取得する
    - タイマーの処理期限を変更する
    - 案件（タイマー）を生成して開始する
- 次に示す XML スキーマファイルを新規に追加しました。
    - allocateWorkItemRequest.xsd
    - getWorkItemResponse.xsd
    - freeWorkItemRequest.xsd
    - getWorkItemResponse.xsd
    - setProcessDataRequest.xsd
    - getListProcessDataIndexRequest.xsd
    - getListProcessDataIndexResponse.xsd
    - getProcessInstanceListResponse.xsd
    - getFlowNodeInstanceListResponse.xsd
    - getFlowNodeDefinitionListResponse.xsd
    - setWorkItemDeadlineRequest.xsd
    - getWorkItemResponse.xsd
    - createAndStartTimerProcessInstanceRequest.xsd
    - getProcessInstanceResponse.xsd
- 次のインタフェースを追加しました。
    - CIWBPMNFlowNodeInstance
    - CIWBPMNFlowNodeDefinition
- CIWBPMNLib インタフェースに，次のメソッドを追加しました。
    - allocateWIEx
    - freeWI
    - setProcessData
    - createAndStartPIForTimer
    - setDeadlineForTimer
    - getListProcessDataIndex
    - getFlowNodeDefinitionsList
    - getFlowNodeInstancesListByPDName

- getFlowNodeInstancesListByPIID
- getProcessInstancesListByPDName
- getProcessInstancesListByPIName

- BPMN 連携ライブラリが提供する列挙型として，次を追加しました。
  - CIWBPMNFlowNodeInstance.AttributeName
  - CIWBPMNFlowNodeDefinition.AttributeName
  - CIWBPMNFlowNodeDefinition.Type
- 次の XPath 拡張関数を追加しました。
  - csciw:list-contains
  - csciw:exists
- 共通設定ファイルに設定するAppCallServiceTransactionTimeout パラメタを追加しました。
- メッセージによる案件投入の場合にファイルに指定する内容として，次のキーを追加しました。
  - self.pi.<$変数（形式 1）>
  - self.pi.<$変数（形式 2）>
- メッセージによる他案件の呼び出しの場合にファイルに指定する内容として，次のキーを追加しました。
  - self.pi.<$変数（形式 1）>
  - self.pi.<$変数（形式 2）>
- エラーによる自案件の呼び出しの場合にファイルに指定する内容として，次のキーを追加しました。
  - ancestor.pi.<$変数（形式 1）>
  - ancestor.pi.<$変数（形式 2）>
- 次のテーブルを追加しました。
  - <SYSTEMID>_APPLICATION_START_TIMER
  - <SYSTEMID>_SUB_MULTI_INSTANCE_MNG
- 次のテーブルのテーブル定義に列名「State」を追加しました。
  - <SYSTEMID>_PROCESS_DATA_S
  - <SYSTEMID>_PROCESS_DATA_N
- BPMN 連携機能を使用する場合に，インデクスを追加するテーブルの一覧を追加しました。
- インデクスの一覧に次のインデクスを追加しました。
  - IDX_<SYSTEMID>_WI_N5
  - IDX_<SYSTEMID>_WI_N6
  - IDX_<SYSTEMID>_APPSTIMER_P1
  - IDX_<SYSTEMID>_PI_N5

- IDX_<SYSTEMID>_SUBMIMNG_P1

- インスタンステーブルのレコード数の説明に，リスト型プロセスデータの場合の説明を追加しました。
- ビジネスプロセスモニタの単体での使用を非サポートにしました。それに伴い，JavaScript 関数，およびユーザアプリケーション内で使用する場合の構成や使用方法などについての記述を削除しました。
- ワーク管理データベースを移行する手順について，CSCIW 02-10 から CSCIW 02-20 へ移行する場合の手順に変更しました。
- インストールディレクトリ（フォルダ）の説明を追加しました。

用語解説

英字

BPMN エディタ

BPMN 要素を組み合わせて，BPMN ビジネスプロセス定義を新規に作成したり編集したりするツールです。

BPMN ビジネスプロセス定義

業務の流れ（ビジネスプロセス）を BPMN エディタで定義したものです。

BPMN 連携機能

CSCIW で BPMN2.0 をサポートするための機能です。

CSV エクスポート

BPMN エディタの機能です。ビジネスプロセス定義の定義情報を CSV 形式のファイル（BPMN 定義一覧ファイル）にエクスポートします。

ref 識別子

アプリケーション呼び出しサービスによる呼び出しの定義を識別する名称です。BPMN 連携機能を使用する場合，ref 識別子として次の種類があります。

- サービスタスク属性またはビジネスルールタスク属性の`operationRef`
- メッセージイベント定義属性の`messageRef`
- エラーイベント定義属性の`errorRef`

REST アプリケーション

ビジネスプロセスの内容に従って，アプリケーション呼び出しサービスから REST 形式で呼び出される業務アプリケーションのことです。

REST サービス

案件の投入や案件一覧の取得など，リソースに対する操作を REST API として提供するサービスのことです。

WorkManager の最大スレッド数

WorkManager の非デーモン（短寿命）Work スレッドプールの最大スレッド数です。

ア行

アプリケーション呼び出しサービス

BPMN ビジネスプロセス定義で定義された，サービスタスクやスロー（メッセージ）などの要素について，REST アプリケーションの呼び出しやメッセージの送信などを行うサービスのことです。

親案件

コールアクティビティから子案件を投入した案件です。

カ行

子案件

コールアクティビティから投入された案件です。

コールアクティビティ情報ファイル

コールアクティビティから子案件を投入する際に使用する情報を定義するファイルです。ビジネスプロセス定義名やプロセスデータのマッピングを定義します。

タ行

単一型プロセスデータ

プロセスデータ値が文字列型および数値型のプロセスデータのことです。

ハ行

ビジネスプロセスモニタ

BPMN ビジネスプロセス定義から生成した BPMN のプロセス図と，案件のステータス情報を表示するサービスです。

プロセスデータ

ビジネスプロセスの遷移の制御に使用するデータです。プロセスデータ値の型の種類を次に示します。

- 文字列型
- 数値型
- リスト型

マ行

マルチインスタンスインデクス

マルチインスタンスで生成された各インスタンスを識別するための番号のことです。各インスタンスを生成するときに付与されます。

ラ行

リスト型プロセスデータ

文字列型と数値型のプロセスデータ値の集合を，リストに格納したプロセスデータのことです。

リスト内識別子

リスト型プロセスデータ内に存在する各プロセスデータ値を識別するための数値です。リスト内識別子はプロセスデータキー名の"{}"内に付与されます。

ルート案件

API やアプリケーション呼び出しサービスから投入された案件のことです。

索引

記号

A

B

C

D

E

F

G

H

I

J

M

N

S

T

U

W

X

あ

い

う

え

お

か

き

く

け

こ

さ

し

す

せ

そ

た

ち

て

と

は

ひ

ふ

へ

ほ

ま

み

め

も

ゆ

株式会社 日立製作所
〒 100-8280　東京都千代田区丸の内一丁目 6 番 6 号