HITACHI
Inspire the Next

Cosminexus V11 BPM/ESB 基盤 サービスプラットフォーム ファーストステップガイド

手引・操作書

3021-3-J41-60

前書き

■ 対象製品

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

■ 輸出時の注意

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

なお、不明な場合は、弊社担当営業にお問い合わせください。

■ 商標類

■ ランゲージパックの使用について

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

■ 発行

2024 年 9 月 3021-3-J41-60

■ 著作権

変更内容

変更内容(3021-3-J41-60) uCosminexus Service Architect 11-50，uCosminexus Service Platform 11-50

追加・変更内容	変更個所
新規商品手配システム ProductStock_REST に全面改訂した。	－
Service Platform の適用 OS に Amazon Linux 2023 を追加した。	－

単なる誤字・脱字などはお断りなく訂正しました。

はじめに

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

目次

1 商品手配システムを開発する前に

この章では，このマニュアルの読み方，およびこのマニュアルで体験できる内容について説明します。

1.1　このマニュアルの読み方

このマニュアルは，ハンズオン形式で実際にマシンを操作しながら，サービスプラットフォームの環境構築からシステムの開発，実行までの操作を体験するためのものです。ハンズオンで開発するのは商品手配システムです。

2 章では，このマニュアルで紹介する商品手配システムの内容，および商品手配システムの構成要素について説明しています。

3 章では，サービスプラットフォームの開発・テスト環境を構築する手順について説明しています。

4 章では，サービス部品の構築について説明しています。

5 章では，3 章および 4 章で準備したサービスプラットフォームの開発環境を使用して，商品手配システムの開発から実行までの一連の流れについて説明しています。

6 章では，3 章～5 章で準備した商品手配システムの削除，サービス部品の削除，テスト環境の停止，開発環境のアンセットアップ，および製品をアンインストールする手順について説明しています。

それぞれの章は通読型となっていますので，マシンを操作しながら，順番に読み進めてください。

ポイント

画面や操作の説明で使用している記号について

マニュアルの画面や操作の説明で使用している記号を次に示します。

記号	意味
＜　＞	＜＞内の名称がユーザの環境によって異なることを表します。
「　」	入力値，可変値，またはメッセージなどのユーザが入力する内容を表します。

使用している構文要素記号について

マニュアルで使用している構文要素の種類と定義を次に示します。

種類	定義
英字	A～Z　a～z
英数字	A～Z　a～z　0～9
文字列	任意の文字の配列

インストールディレクトリについて

Service Architect のデフォルトのインストールディレクトリは，「C:¥Program Files¥Hitachi¥Cosminexus」です。別のディレクトリにインストールする場合には，このマニュアルに記載されている「<Service Architect のインストールディレクトリ>」をご利用のディレクトリに読み替えてください。

Eclipse メニュー表記について

このマニュアルでは，Eclipse Babel Project が提供する BABEL 日本語ランゲージパックを前提とした Eclipse メニュー表記を使用しています。

このため，使用する Eclipse およびランゲージパックのバージョンの差異によって，マニュアル内のメニュー表記が異なる場合があります。

1.2　このマニュアルで体験できること

このマニュアルでは，商品手配システムの環境設定および開発から実行までの操作を体験できます。

それぞれの章で体験できることを次に示します。

- 3 章で体験できること

  サービスプラットフォームを使うための基本的な設定を体験できます。具体的には，製品のインストール，開発環境のセットアップ，およびテスト環境のセットアップです。この章で体験することは，一般的なシステム開発時の設定手順としても使用できます。

- 4 章で体験できること

  サービス部品の構築を体験できます。具体的には，サービス部品用 Web システムの構築，および RESTful Web サービスの構築です。

- 5 章で体験できること

  手順どおりに操作を進めることで，サービスプラットフォームの開発環境を使用して，商品手配システムの開発から実行までの一連の流れを，2 パターン体験できます。

  商品手配システムの開発の中では，ビジネスプロセス，サービスアダプタ，データ変換定義など，サービスプラットフォームを使用する操作を体験できます。また，商品手配システムの実行，およびサービスプラットフォームが提供するデバッグ機能の操作を体験できます。

- 6 章で体験できること

  3 章～5 章で準備した商品手配システムの削除，サービス部品の削除，テスト環境の停止，開発環境のアンセットアップ，および製品のアンインストールを体験できます。

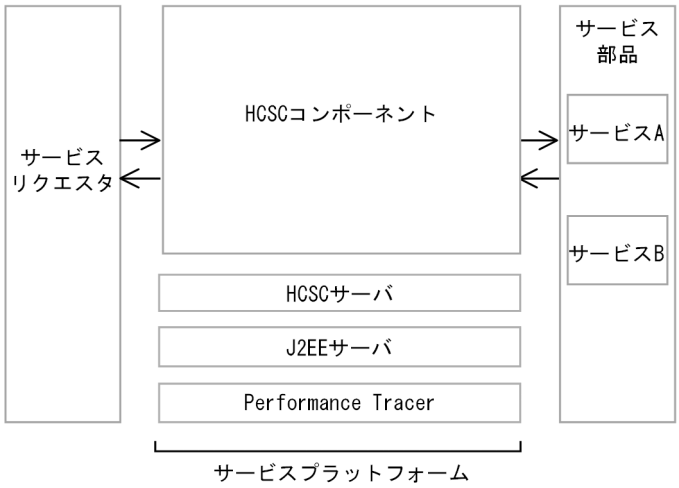
2 商品手配システムの紹介

この章では，商品手配システムの概要について説明します。また，商品手配システムで学んでいただきたいこと，およびその目的についても説明します。

2.1　サービスプラットフォームを利用する一般的なシステム構成

サービスプラットフォームを利用する一般的なシステム構成を次の図に示します。HCSC コンポーネントはサービスリクエスタからの要求を受け取り，サービス部品を呼び出します。また，HCSC コンポーネントは，サービス部品の処理結果を利用し，サービスリクエスタへ応答を返します。

図 2-1　一般的なシステム構成



（凡例）

⟶ ：サービス部品を呼び出す要求・応答の流れ

一般的なシステム構成要素が，それぞれどのような役割を果たしているのかを説明します。

- サービスリクエスタ

  サービスを呼び出すためのクライアントアプリケーションです。サービスプラットフォームを使用する場合，サービスリクエスタは HCSC サーバ上の HCSC コンポーネントへ要求（要求電文）を送ります。

- HCSC コンポーネント

  サービスリクエスタからの要求を受け取り，定義した業務の流れや条件に従って要求を受付，サービスを呼び出すためのものです。受付，ビジネスプロセス，サービスアダプタの総称です。

- サービス部品

  サービスプラットフォームで接続する連携先サービスのことです。

- HCSC サーバ

  ビジネスプロセスやサービスアダプタを管理するサーバ機能です。サービスプラットフォームの中にあります。

- J2EE サーバ

  J2EE アプリケーション（JSP，サーブレット，Enterprise Bean などで構成されるアプリケーション）を実行するためのサーバ機能です。HCSC サーバを起動するのに必要です。

- Performance Tracer

  リクエストの処理過程で各サーバが出力するトレース情報をトレースファイルに出力するプロセスです。

このようなサービスプラットフォームを利用したシステムを構築することで，次の利点があります。

サービスプラットフォームをハブとした，柔軟なシステム構築とシステムの一元管理ができる

サービスプラットフォームでサービスを自由に組み合わせることで，戦略に合わせた柔軟なシステムを構築できます。また，異なるサービス部品間でも，システム内でサービスプラットフォームを経由してから接続されるため，サービスプラットフォームでシステムの一元管理ができます。

サービスプラットフォームを経由することで，異なるプロトコルのサービスを連携できる

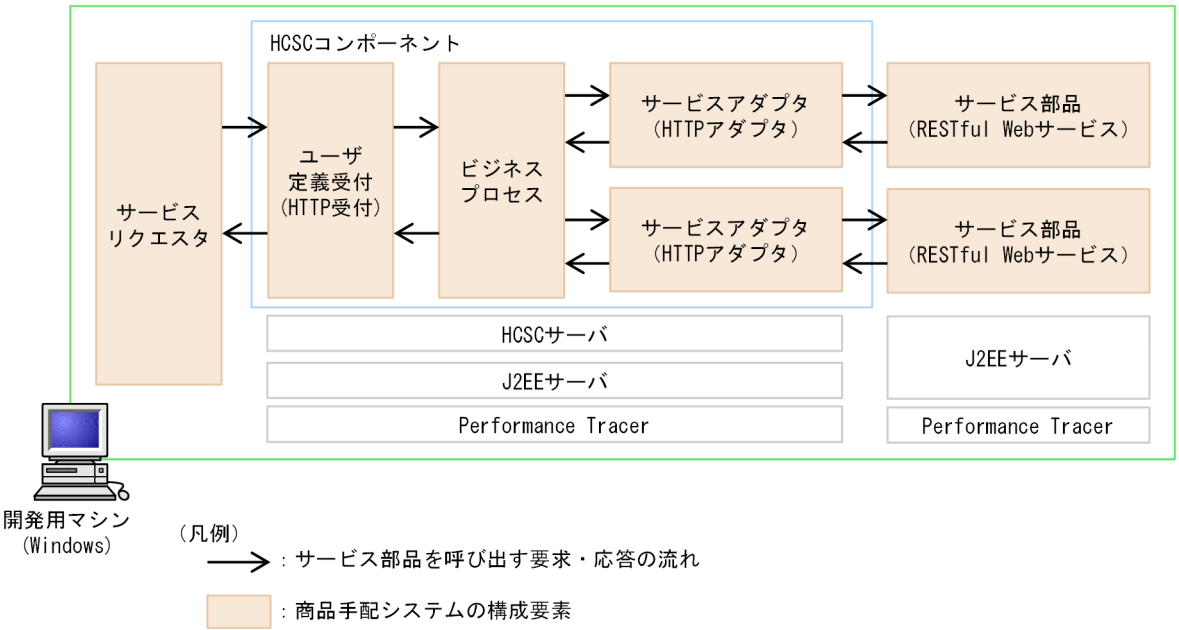
メインフレームで利用されているレガシーなプロトコルから RESTful Web サービスで利用されているモダンなプロトコルまで対応しているので，異なるプロトコルのサービスを連携できます。

このマニュアルで構築する商品手配システムでは，ハブとしての利点を体験できます。

2.2　商品手配システムのシステム構成

商品手配システムはビジネスプロセスからサービスアダプタを介して，商品の在庫を引き当てたり配送を手配したりするサービス部品を呼び出すシステムです。このシステムでは，実際の業務に近い内容での定義を習得することを目的にしています。商品手配システムのシステム構成を次の図に示します。

図 2-2　商品手配システムのシステム構成



商品手配システムの構成要素が，それぞれどのような役割を果たしているのかを説明します。

- **サービスリクエスタ**

  ビジネスプロセスを呼び出すユーザ定義受付に対して要求（要求電文）を送ります。この商品手配システムのサービスリクエスタは，HTTP プロトコルで通信を行います。

- **ユーザ定義受付**

  サービスリクエスタからの要求電文を受け付け，ビジネスプロセスを呼び出すための機能（インタフェース）です。ユーザが任意のインタフェースを定義できます。この商品手配システムはユーザ定義受付の HTTP 受付を使用します。

- **ビジネスプロセス**

  ビジネスプロセスは，複数のサービスの処理順序や処理条件などを定義し，一連の業務の流れとして定義したものです。この商品手配システムでは，サービス部品ごとに異なる電文フォーマットを変換したり，サービス部品からの応答によって処理を分岐したりしています。

- **サービスアダプタ**

  サービスアダプタは，ビジネスプロセスからの要求を受け付けて，サービス部品を呼び出すためのものです。この商品手配システムは HTTP プロトコルで通信を行うため，HTTP アダプタを使用します。

- **サービス部品**

  サービスアダプタから呼び出される RESTful Web サービスです。HTTP プロトコルで通信を行います。

2.3　商品手配システムの処理内容

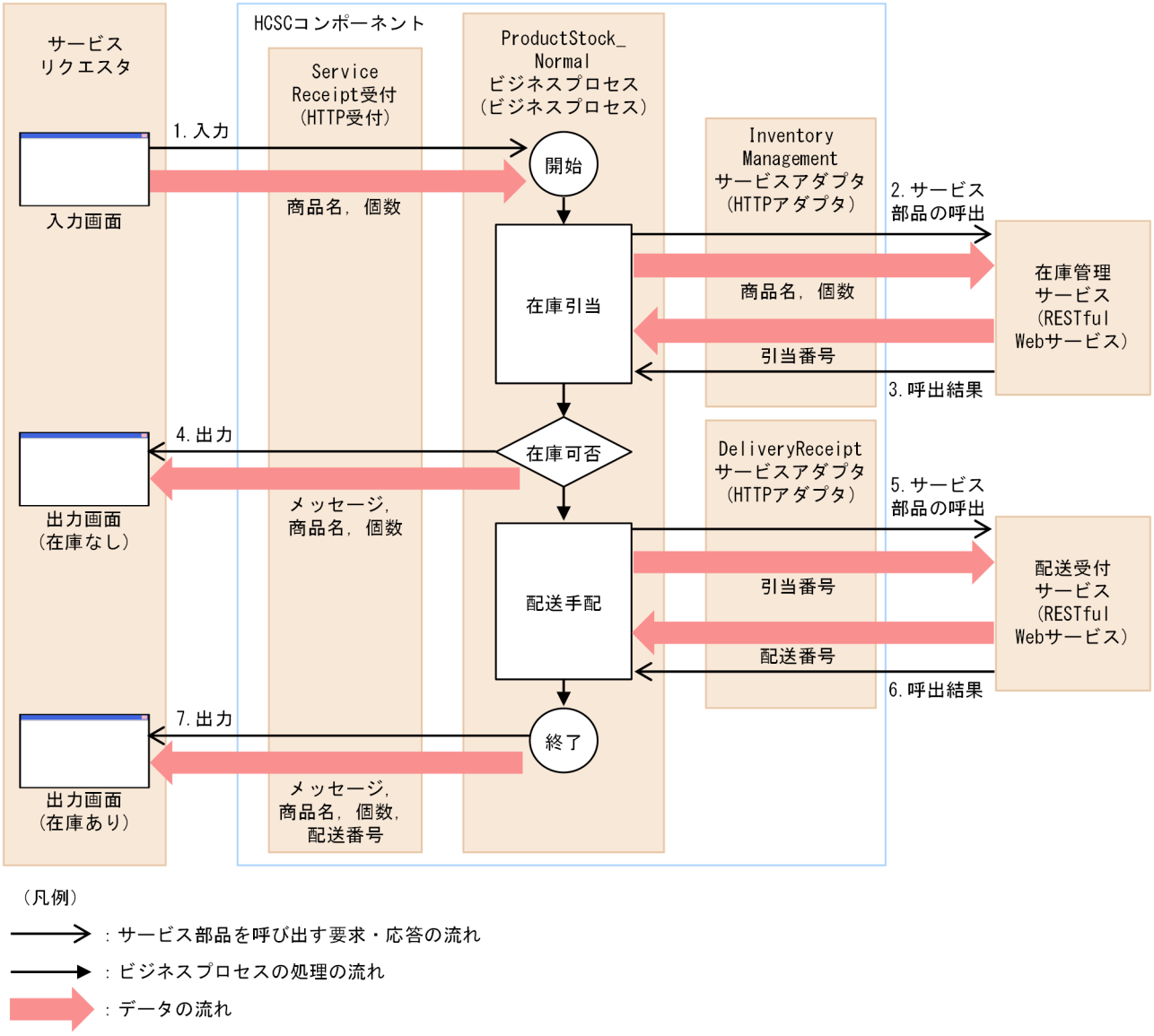
商品手配システムを構成する各要素が，どのような処理を実行しているのかを説明します。

ここで紹介する商品手配システムは，店舗の担当者が業務システムを使って商品を手配する場合に，在庫引当や配送手配の処理をすることを想定しています。

ここでは，正常系の処理を実装した商品手配システム（基本編）の処理内容について説明します。

商品手配システム（基本編）の処理の詳細を次の図に示します。

図 2-3　商品手配システム（基本編）の処理詳細



処理の流れは次のとおりです。

1. 入力画面から商品名と個数を入力します。入力された商品名と個数は，HCSC コンポーネントへのリクエストとして，ServiceReceipt 受付経由で ProductStock_Normal ビジネスプロセスへ送信されます。

2. ProductStock_Normal ビジネスプロセスは，受信したリクエスト（商品名，個数）を使用して，InventoryManagement サービスアダプタ経由で在庫管理サービスを呼び出します。

3. 在庫管理サービスは，呼出結果（引当番号）を InventoryManagement サービスアダプタ経由で ProductStock_Normal ビジネスプロセスに送信します。

4. 在庫がない場合，ProductStock_Normal ビジネスプロセスは，ServiceReceipt 受付経由でレスポンス（在庫なしメッセージ，商品名，および個数）をサービスリクエスタに送信します。サービスリクエスタは受信したレスポンスを出力画面に出力します。

5. 在庫がある場合，ProductStock_Normal ビジネスプロセスは，受信したレスポンス（引当番号）を使用して，DeliveryReceipt サービスアダプタ経由で配送受付サービスを呼び出します。

6. 配送受付サービスは，呼出結果（配送番号）を DeliveryReceipt サービスアダプタ経由で ProductStock_Normal ビジネスプロセスに送信します。

7. ProductStock_Normal ビジネスプロセスは，ServiceReceipt 受付経由でレスポンス（在庫ありメッセージ，商品名，個数，および配送番号）をサービスリクエスタに送信します。サービスリクエスタは受信したレスポンスを出力画面に出力します。

ここで説明した基本編のほかに，異常系の処理を実装した商品手配システム（フォルトハンドリング編）も提供しています。商品手配システム（フォルトハンドリング編）の処理内容については，「5.5.6 ProductStock_FaultHandling ビジネスプロセスの定義」を参照してください。

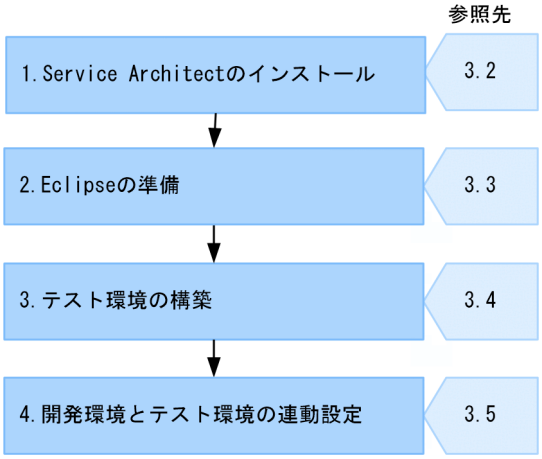
3 商品手配システムの開発・テスト環境を構築する

この章では，商品手配システムの開発・テスト環境の構築について説明します。

3.1　商品手配システムの開発・テスト環境を構築する流れ

商品手配システムの開発・テスト環境を構築する手順を次の図に示します。

図 3-1　商品手配システムの開発・テスト環境を構築する手順



Service Architect とは，HCSC コンポーネントの開発環境およびテスト環境を提供する製品です。Service Architect では Eclipse 経由で HCSC コンポーネントを開発します。Eclipse で HCSC コンポーネントを開発する環境を開発環境といいます。

テスト環境は，Service Architect の HCSC 簡易セットアップ機能を使用して構築します。

1. Service Architect のインストール

   インストーラを使用して，Service Architect をインストールします。詳細については，「3.2　Service Architect のインストール」を参照してください。

2. Eclipse の準備

   Eclipse をインストールして開発環境を構築します。Eclipse のインストールは，Service Architect が提供する Eclipse セットアップ機能を利用します。詳細については，「3.3　Eclipse の準備」を参照してください。

3. テスト環境の構築

   Service Architect の HCSC 簡易セットアップ機能を利用し，テスト環境を構築します。詳細については，「3.4　テスト環境の構築」を参照してください。

4. 開発環境とテスト環境の連動設定

   テスト環境構築後のシステム構成を，開発環境に反映させる必要があります。詳細については，「3.5 開発環境とテスト環境の連動設定」を参照してください。

以降の節で，これらの手順について説明します。

3.2 Service Architect のインストール

Service Architect のインストール手順を次に示します。

注意事項

Service Architect をインストールする前に，すべての Windows アプリケーションを終了させておいてください。また，インストールは，Administrator 権限が設定されたユーザが実施してください。

1. インストール CD-ROM を，CD-ROM ドライブにセットします。

   ［日立総合インストーラ］ダイアログに，「選択されたソフトウェアをインストールします。」と表示されます。

   ［日立総合インストーラ］ダイアログが表示されない場合，エクスプローラを使用して，CD-ROM ディレクトリの「HCD_INST.EXE」をダブルクリックしてください。

2. uCosminexus Service Architect を選択した状態で，［インストール実行］ボタンをクリックします。

   ［インストール処理開始の確認 － 日立総合インストーラ］ダイアログに，「インストールを開始します。よろしいですか？」と表示されます。

3. ［OK］ボタンをクリックします。

   ［uCosminexus Service Architect セットアッププログラムへようこそ］ページが表示されます。

4. ［次へ］ボタンをクリックします。

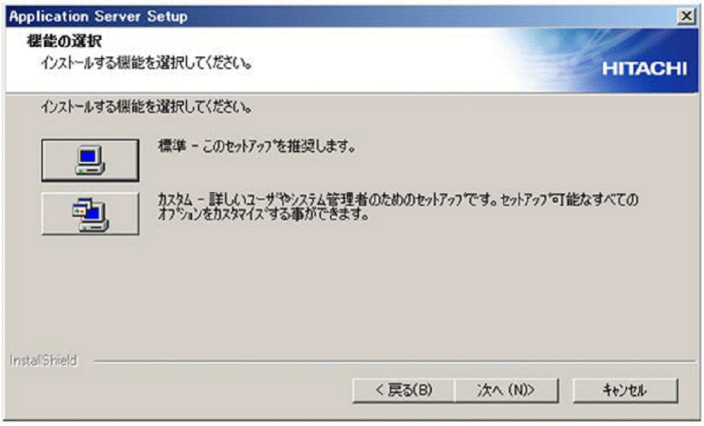
   ［インストール先の選択］ページが表示されます。

5. 必要に応じてインストール先ディレクトリを選択して，［次へ］ボタンをクリックします。

   ポイント

   インストール先ディレクトリを指定する場合，半角英数字で 50 文字までのパス名を指定してください。

   ［機能の選択］ページが表示されます。



6. [標準 - このセットアップを推奨します。] の左にあるボタンをクリックします。

   [ユーザ情報] ページが表示されます。

7. [ユーザ名] および [会社名] を入力して [次へ] ボタンをクリックします。

   [プログラム フォルダの選択] ページが表示されます。

8. 必要に応じてプログラムフォルダ名を変更して，[次へ] ボタンをクリックします。

   [インストールの開始] ページが表示されます。

9. 設定した内容を確認して，問題がなければ [次へ] ボタンをクリックします。

   インストールが開始されます。インストールが完了すると，[セットアップの完了] ダイアログが表示されます。インストールは数分掛かる場合があります。

10. [完了] ボタンをクリックします。

    OS を再起動するかどうかを確認する画面が表示されます。

11. [はい] ボタンをクリックします。

    OS が再起動し，Service Architect のインストールが完了します。

これで，Service Architect はインストールされました。

3.3　Eclipse の準備

ここでは，Eclipse セットアップ機能を使用した開発環境の構築について説明します。また，JDK の確認やローカル変数情報の出力の設定も説明します。

3.3.1　Eclipse をインストールするための準備

Eclipse をインストールするための環境を準備します。Eclipse のアーカイブファイルを準備する手順を次に示します。

1. Eclipse のアーカイブファイルを入手します。

   次に示すどれかの Eclipse のアーカイブファイルを Service Architect に同梱されている添付品または Eclipse.org のダウンロードサイトから入手してください。

Eclipse のバージョン	アーカイブファイル名
Eclipse IDE for Enterprise Java and Web Developers 2021-09（4.21）	eclipse-jee-2021-09-R-win32-x86_64.zip
Eclipse IDE for Enterprise Java and Web Developers 2021-12（4.22）	eclipse-jee-2021-12-R-win32-x86_64.zip

2. Eclipse のアーカイブファイルを格納します。

   入手した Eclipse のアーカイブファイルを次のディレクトリへ格納します。

   <Service Architect のインストールディレクトリ>¥ADP¥Archives

   参考

   添付品の Eclipse ランゲージパックを Eclipse のアーカイブファイルと同じディレクトリに格納すると，Eclipse セットアップ時に Eclipse を日本語化できます。

これで，Eclipse をインストールする準備ができました。

3.3.2　Eclipse セットアップ機能を使用した開発環境の構築

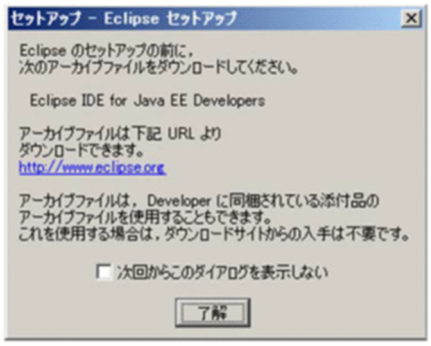
Eclipse セットアップ機能を使用して開発環境を構築します。開発環境を構築する手順を次に示します。この操作は，管理者または管理者特権で実行してください。

1. Windows の［スタート］メニューから，［Cosminexus※］－［Eclipse セットアップ］を選択します。

   Eclipse のアーカイブファイルを準備したかどうかを確認するダイアログが表示されます。

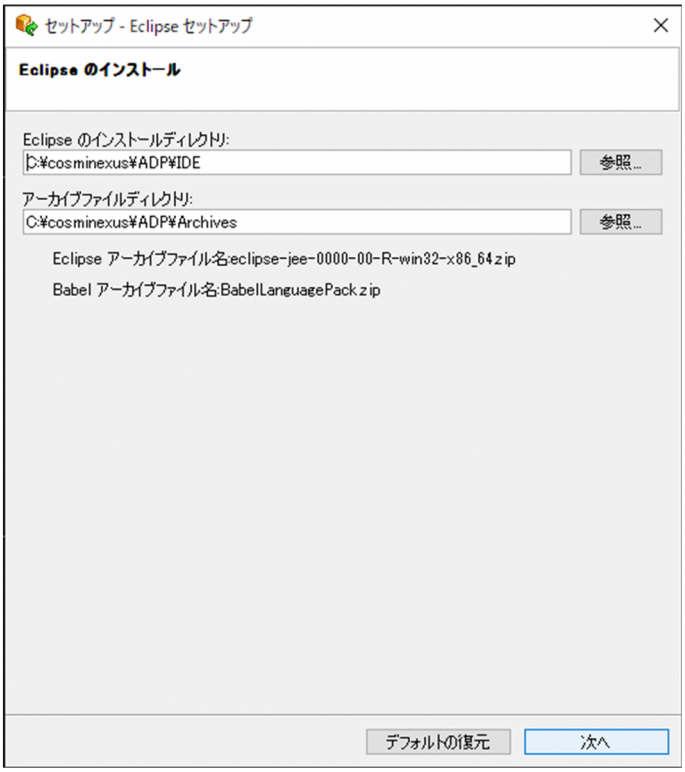
   注※

   プログラムフォルダ名を変更している場合は，変更したプログラムフォルダから選択してください。



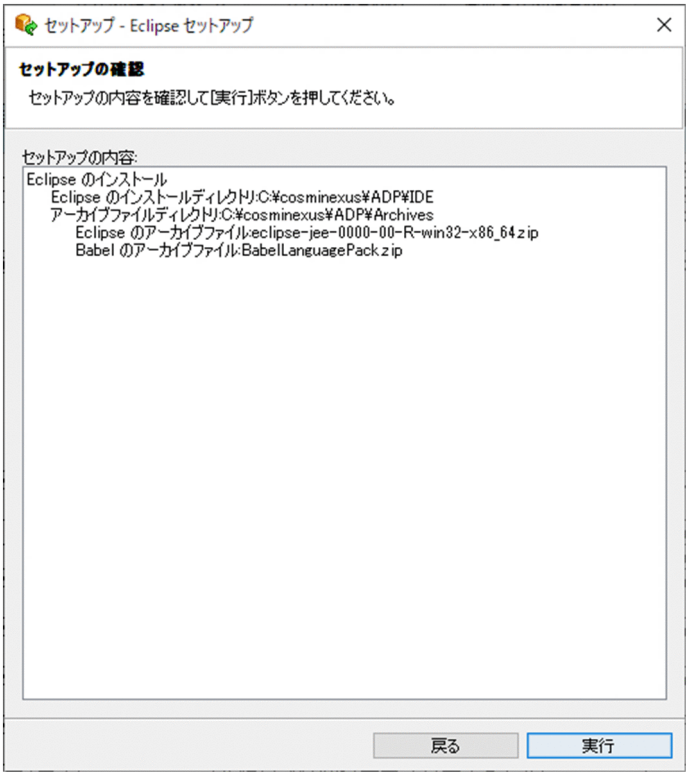
2. 確認ダイアログで，［了解］ボタンをクリックします。

［Eclipse のインストール］ページが表示されます。



3. ［Eclipse のインストール］ページで，Eclipse のインストールディレクトリおよびアーカイブファイルディレクトリを指定して，［次へ］ボタンをクリックします。

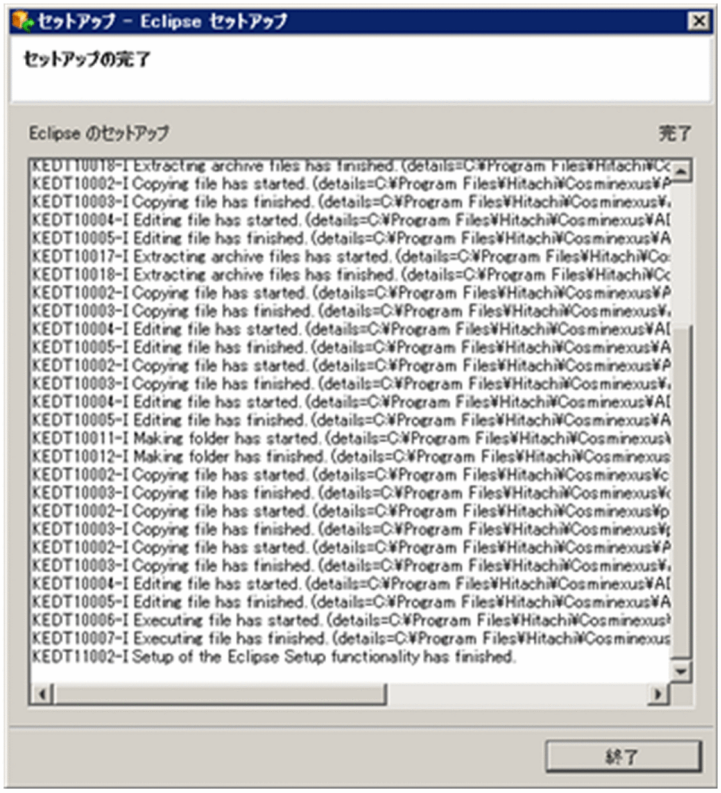
［セットアップの確認］ページが表示されます。



4. [セットアップの確認] ページでセットアップ内容を確認して，[実行] ボタンをクリックします。

[進行状況] ページが表示され，セットアップが実行されます。セットアップが完了すると [セットアップの完了] ページが表示されます。

Eclipse セットアップは数分掛かる場合があります。



5. [セットアップの完了] ページで，[終了] ボタンをクリックします。

   [セットアップ - Eclipse セットアップ] ダイアログが閉じて，Eclipse のセットアップが完了します。デスクトップに Eclipse のショートカットが生成されます。

6. Eclipse のインストールディレクトリ内の eclipse.ini ファイルをエディタで開き，次のオプションを追加します。

   - -vmargs オプション

     -Xbootclasspath/a:<Service Architrct のインストールディレクトリ>¥jaxp¥lib¥csmjaxp.jar;<Service Architrct のインストールディレクトリ>¥jaxp¥lib¥csmstax.jar

   オプション追加後の eclipse.ini ファイルの内容の例を次に示します。

```
-vm
C:¥Program Files¥Hitachi¥Cosminexus¥jdk¥bin¥javaw.exe
   : (略)
--add-modules=ALL-SYSTEM
-Xbootclasspath/a:C:¥Program Files¥Hitachi¥Cosminexus¥jaxp¥lib¥csmjaxp.jar;C:¥Program Fil
es¥Hitachi¥Cosminexus¥jaxp¥lib¥csmstax.jar
```

   eclipse.ini ファイルのデフォルトの格納先を次に示します。

```
<Service Architectのインストールディレクトリ>¥ADP¥IDE¥eclipse
```

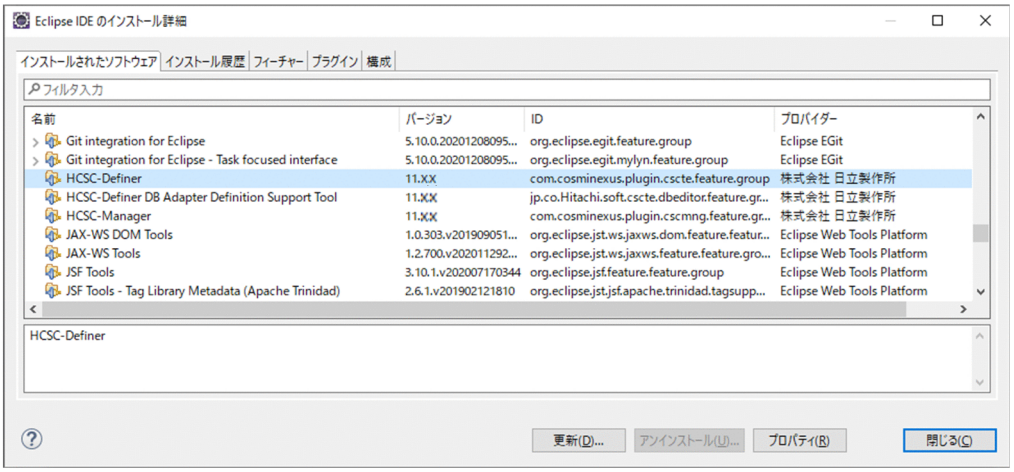
これで，Eclipse 環境が構築できました。

正しくセットアップされたことを確認します。この操作は，管理者または管理者特権で実行してください。

1. Eclipse を起動します。

2. Eclipse のメニューから，[ヘルプ] − [Eclipse IDE について] − [インストール詳細] を選択し，[Eclipse IDE のインストール詳細] ダイアログを表示します。

   HCSC-Definer のバージョン番号が「11.5.0」になっていることを確認します。



参考

セットアップログの確認方法

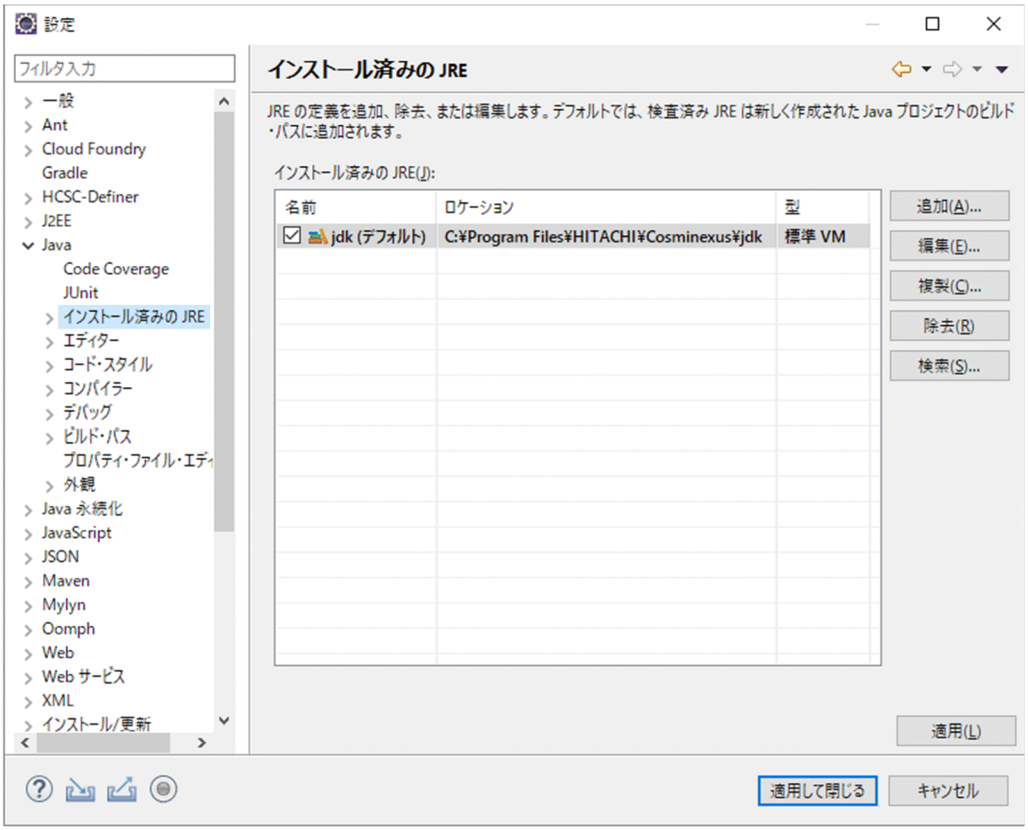
Eclipse セットアップ機能を実行したときのログは，Eclipse のセットアップログに出力されます。セットアップログの確認方法については，「付録 D　Eclipse セットアップ機能実行時の情報の採取」を参照してください。

3.3.3　JDK の確認

Eclipse 上で開発するときに使用する JDK が Service Architect で提供されている JDK かどうかを確認します。確認手順を次に示します。

1. Eclipse のメニューから［ウィンドウ］－［設定］を選択します。
   ［設定］ダイアログが表示されます。

2. 左ペインで［Java］－［インストール済みの JRE］を選択します。
   右ペインに［インストール済みの JRE］ページが表示されます。

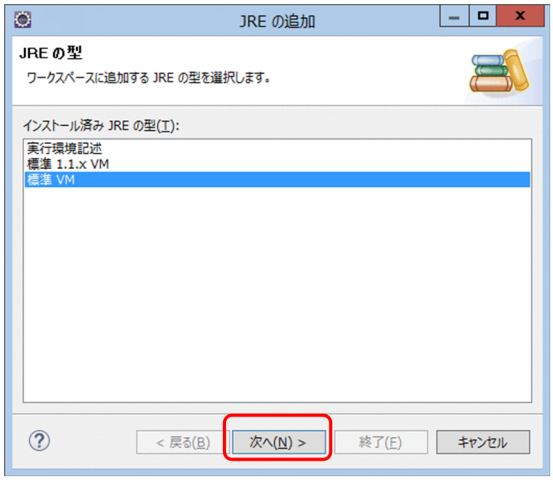


3. ［インストール済みの JRE］が Service Architect で提供されている JDK かどうかを確認します。
   ［ロケーション］に次のパスが表示されているかどうかを確認します。

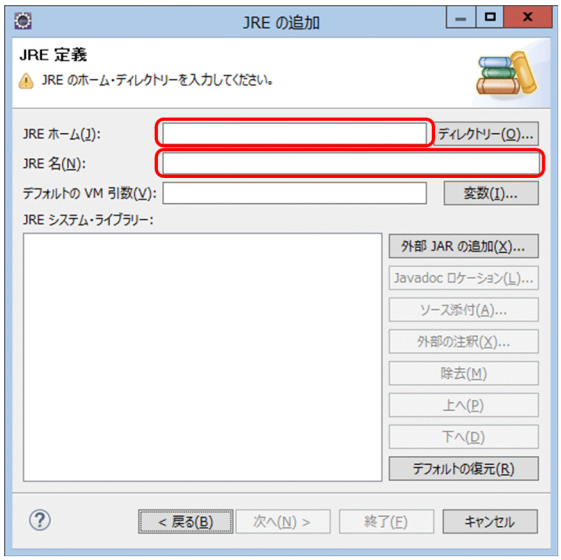
```
<Service Architectのインストールディレクトリ>¥jdk
```

パスが表示されていない場合

［追加］ボタンをクリックします。［JRE の型］ページが表示されます。



「標準 VM」を選択し，［次へ］ボタンをクリックします。［JRE 定義］ページが表示されます。



［JRE ホーム］に「<Service Architect のインストールディレクトリ>¥jdk」を，［JRE 名］に「jdk」を選択し，［終了］ボタンをクリックします。

設定後，［インストール済みの JRE］ページの［名前］にチェックを入れます。

パスが表示されている場合

［名前］にチェックが入っているかどうかを確認します。チェックが入っていない場合はチェックを入れてください。

特に，複数の JDK がインストールされている場合，<Service Architect のインストールディレクトリ>¥jdk にチェックが入っていないことがあります。チェックが入っていない場合は，チェックを入れてください。

4. ［適用］ボタンまたは［適用して閉じる］ボタンをクリックします。

設定が保存されます。

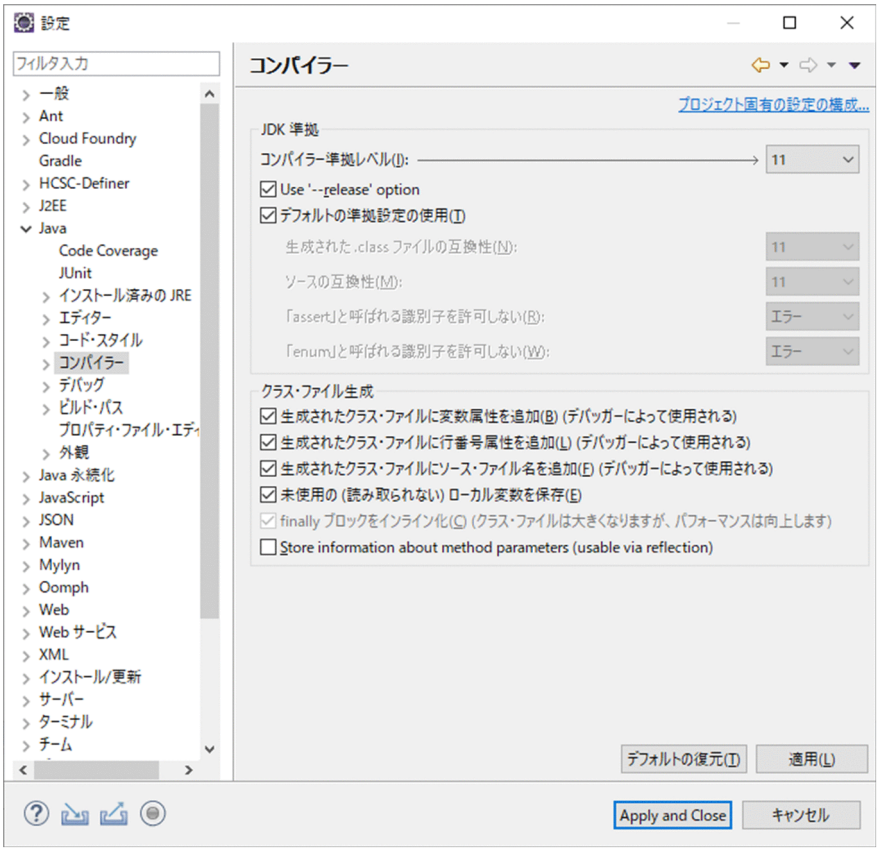
3.3.4　ローカル変数情報の出力の設定

Eclipse のコンパイラーの設定によって，例外発生時に J2EE アプリケーションが持つローカル変数の情報をスタックトレースに出力できます。

ローカル変数情報をスタックトレースに出力する場合のコンパイラーの設定手順を次に示します。

1. Eclipse のメニューから［ウィンドウ］－［設定］を選択します。
   ［設定］ダイアログが表示されます。

2. 左ペインで［Java］－［コンパイラー］を選択します。
   ［コンパイラー］ページが表示されます。



3. 次の項目を指定します。

項目名	設定値
クラス・ファイル生成	[生成されたクラス・ファイルに変数属性を追加（デバッガーによって使用される)]にチェックを入れます。 なお，それ以外の項目については，出力したい内容に合わせてチェックを入れてください。

必要に応じて，次の項目を指定してください。

項目名		設定値
JDK 準拠	デフォルトの準拠設定の使用	コンパイラーに使用する設定を指定します。 • チェックを入れる ［コンパイラー準拠レベル］で指定したレベルに沿った設定が適用されます。 • チェックを入れない 次の項目を手動で指定します。 ［生成された.class ファイルの互換性］ ［ソースの互換性］ ［「assert」と呼ばれる識別子を許可しない］ ［「enum」と呼ばれる識別子を許可しない］

4. ［適用］ボタンまたは［適用して閉じる］ボタンをクリックします。

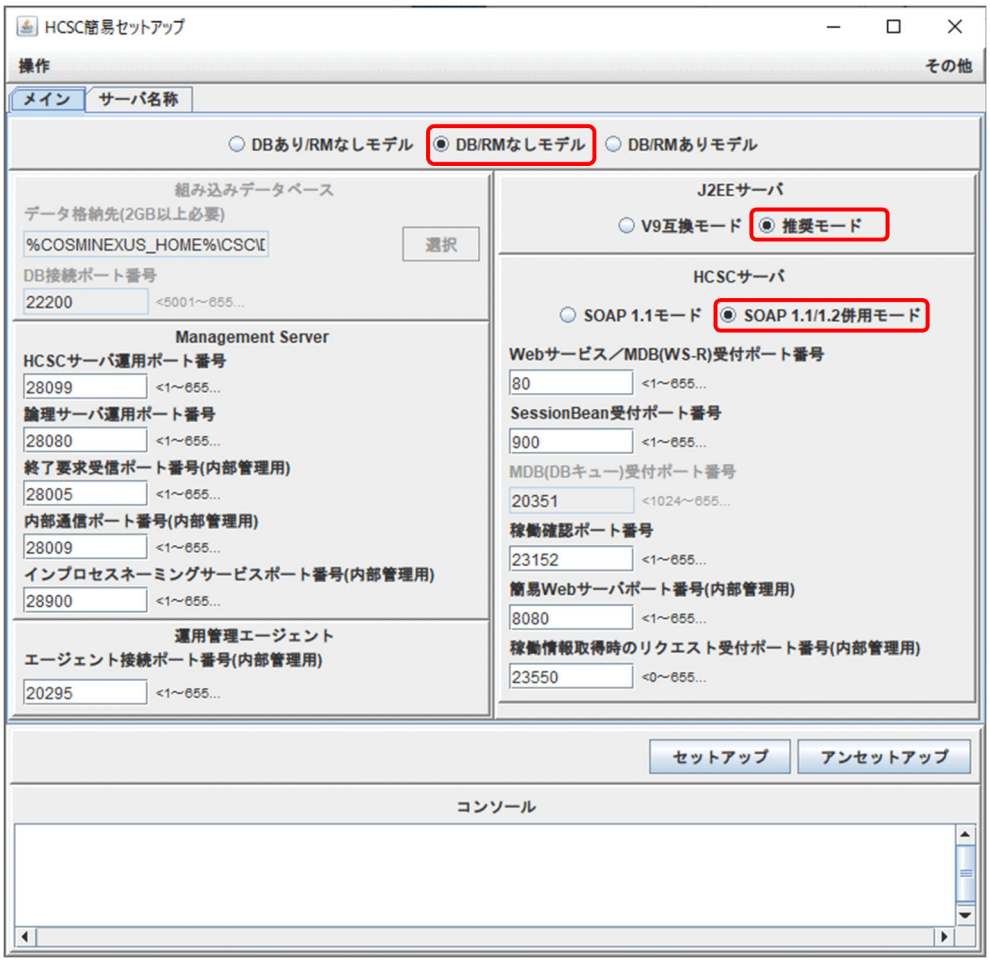
設定が保存されます。

3.4　テスト環境の構築

3.4.1　HCSC 簡易セットアップ

テスト環境を構築するには，**HCSC 簡易セットアップ機能**を使います。テスト環境の構築方法を次に示します。この操作は，管理者または管理者特権で実行してください。

1. Eclipse を起動している場合は，Eclipse を終了します。
2. ［スタート］メニューの［プログラム］から［Cosminexus※］－［テスト環境セットアップ］を選択します。
   HCSC 簡易セットアップの画面の［メイン］ページが表示されます。
   注※
   　プログラムフォルダ名を変更している場合は，変更したプログラムフォルダから選択してください。
3. ［DB/RM なしモデル］ラジオボタンを選択します。
4. ［推奨モード］ラジオボタンを選択します。
5. ［SOAP1.1/1.2 併用モード］ラジオボタンを選択します。



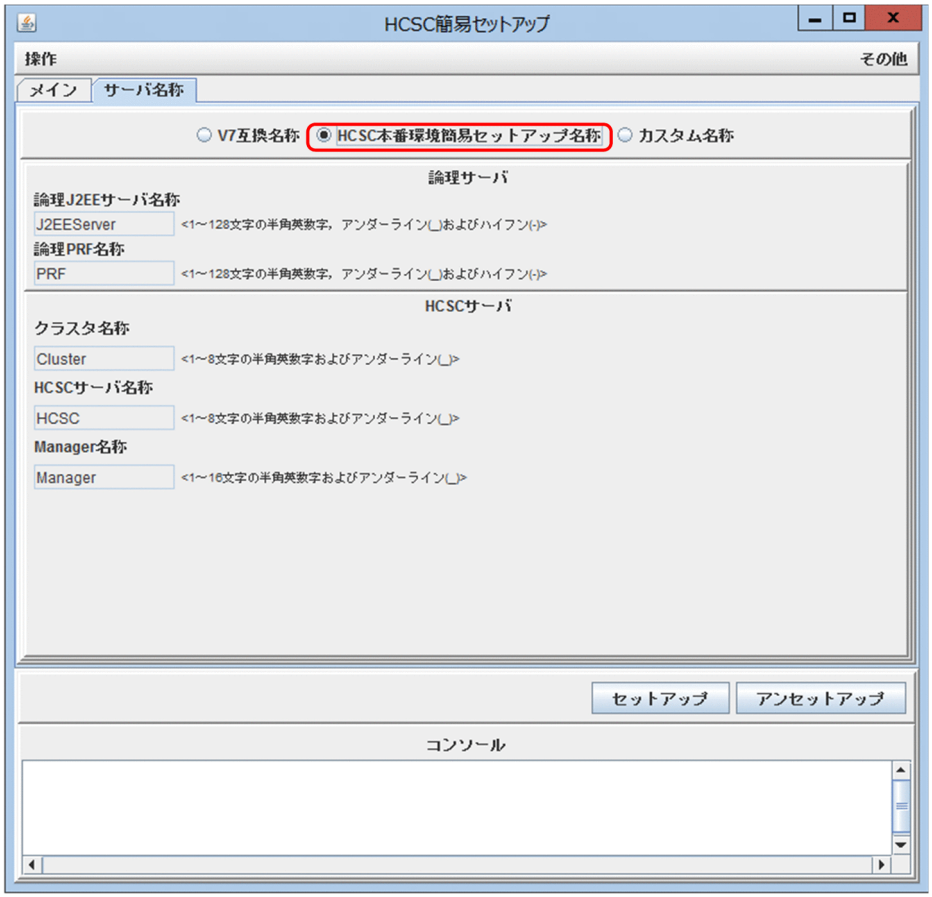
6. [サーバ名称] タブをクリックします。

［サーバ名称］ページが表示されます。

7. [HCSC 本番環境簡易セットアップ名称] ラジオボタンを選択します。

論理サーバおよび HCSC サーバの名称が次のように変更されます。

- 論理 J2EE サーバ名称＝ J2EEServer
- 論理 PRF 名称＝ PRF
- クラスタ名称＝ Cluster
- HCSC サーバ名称＝ HCSC
- Manager 名称＝ Manager



8. [セットアップ] ボタンをクリックします。

テスト環境のセットアップが開始されます。セットアップの状況は，HCSC 簡易セットアップ画面のコンソールに表示されます。コンソールに「HCSC 簡易セットアップ機能のセットアップを終了します」と表示されたら，テスト環境のセットアップは正常に終了です。

注意事項

コンソールにエラーが表示され，テスト環境のセットアップが異常終了した場合，再セットアップする必要があります。異常終了したときに，HCSC 簡易セットアップ画面の［セットアップ］ボタンが活性か，非活性かによって，再セットアップの手順が異なります。

［セットアップ］ボタンが活性の場合

［セットアップ］ボタンをクリックして，もう一度セットアップしてください。

［セットアップ］ボタンが非活性の場合

［アンセットアップ］ボタンをクリックして，アンセットアップします。そのあとで，もう一度セットアップしてください。

9. HCSC 簡易セットアップ画面のメニューから［操作］－［終了］を選択し，HCSC 簡易セットアップ画面を閉じます。

3.4.2　テスト環境の起動

構築したテスト環境を起動する方法を次に示します。この操作は，管理者または管理者特権で実行してください。

1. ［スタート］メニューの［プログラム］から［Cosminexus※］－［テストサーバ起動］を選択して，テスト環境の Performance Tracer，J2EE サーバ，および，標準受付とユーザ定義受付を含む HCSC サーバを起動します。

注※

プログラムフォルダ名を変更している場合は，変更したプログラムフォルダから選択してください。

注意事項

Service Architect の使用を終了する場合は，起動中のテスト環境を停止したあとに Eclipse を終了します。テスト環境を停止する方法については，「6.2.1　テスト環境の停止」を参照してください。

3.5　開発環境とテスト環境の連動設定

3.5.1　開発環境の SOAP モードの設定

開発環境の SOAP モードは「3.4.1　HCSC 簡易セットアップ」で設定したテスト環境の SOAP モードと一致させる必要があります。

開発環境の SOAP モードの設定方法を次に示します。この項の操作は，管理者または管理者特権で実行してください。以降，Eclipse を起動する時は，管理者または管理者特権で実行してください。

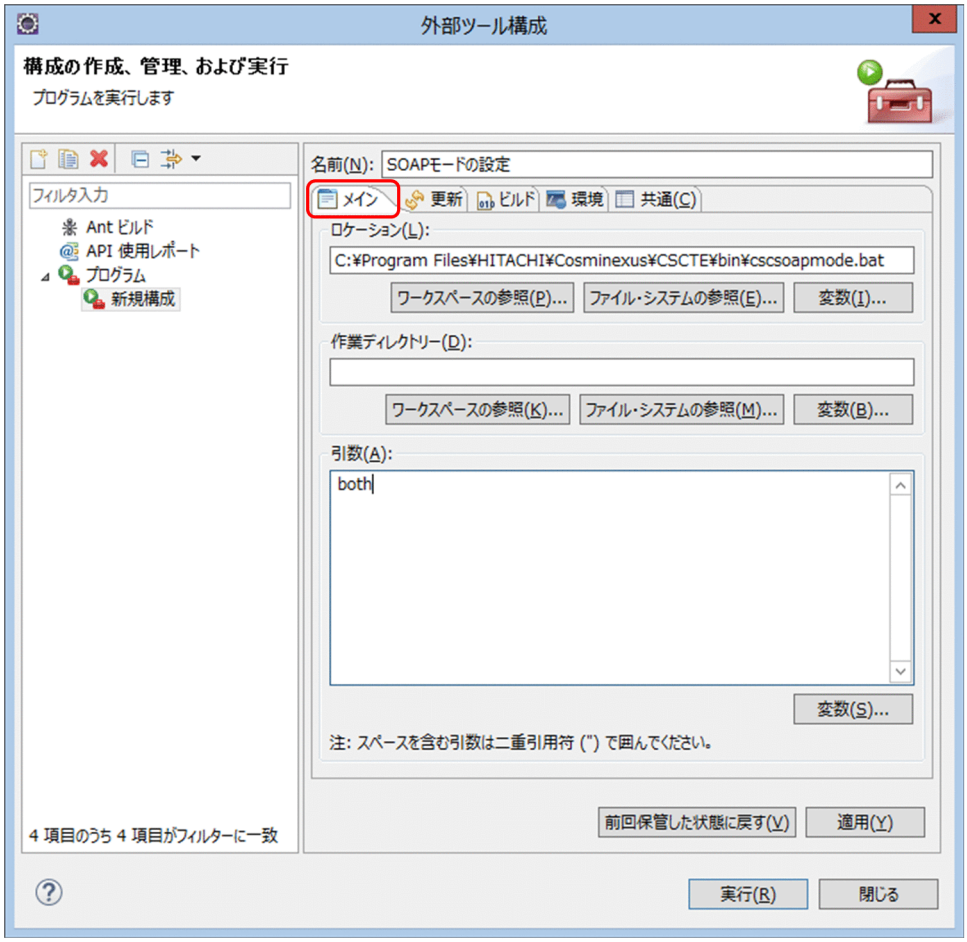
1. Eclipse が終了状態の場合，Eclipse を起動します。

2. Eclipse のメニューから，［ウィンドウ］－［設定］を選択します。
   ［設定］ダイアログが表示されます。

3. ダイアログ左側のツリービューで［HCSC-Definer］を選択します。
   ダイアログ右側の［現在の SOAP モード］に使用している SOAP モードが表示されます。現在の SOAP モードは SOAP1.1/1.2 併用モードかどうかを確認してください。現在の SOAP モードが SOAP1.1 モードの場合だけ以降の手順を実施して SOAP モードを SOAP1.1/1.2 併用モードに設定してください。

4. Eclipse のメニューから［実行］－［外部ツール］－［外部ツールの構成］を選択します。
   ［外部ツール構成］ダイアログが表示されます。

5. 左ペインの［プログラム］を右クリックし，［新規構成］を選択します。
   ［構成の作成、管理、および実行］ページが表示されます。



［メイン］タブを選択して，次の内容を入力します。

項目名	設定値
名前	任意の名称を指定します。 ここでは，「SOAP モードの設定」と指定します。
ロケーション	${env_var:COSMINEXUS_HOME}¥CSCTE¥bin¥cscsoapmode.bat または［ファイル・システムの参照］ボタンから次のファイルを選択します。 <Service Architect のインストールディレクトリ>¥CSCTE¥bin¥cscsoapmode.bat
引数	both

6. ［実行］ボタンをクリックします。

   コマンドが登録，および実行され，コンソールビューに実行結果が表示されます。

7. Eclipse を再起動します。

3.5.2 HCSCTE プロジェクトの作成

開発環境で HCSC コンポーネントを作成・編集するためには，HCSCTE プロジェクトが必要となります。HCSCTE プロジェクトを作成する手順を次に示します。

注意事項

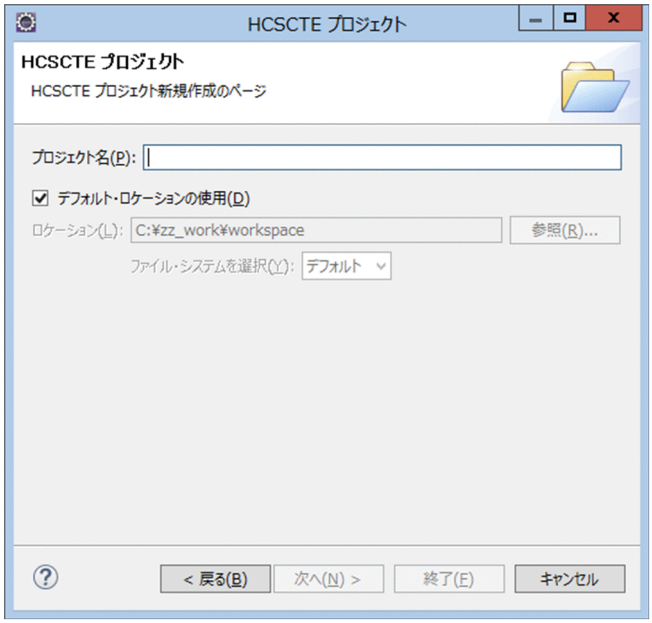
HCSCTE プロジェクトは，1 つのプログラムにつき 1 つ作成してください。複数のプログラムを開発する場合，各プログラムの HCSCTE プロジェクトのワークスペースは分ける必要があります。同じワークスペースに HCSCTE プロジェクトを複数作成すると，プログラムが正しく動作しません。

1. メニューから［ファイル］－［新規］－［プロジェクト］を選択します。

［新規プロジェクト］ダイアログが表示されます。

2. [HCSCTE プロジェクト］を選択して，［次へ］ボタンをクリックします。

［HCSCTE プロジェクト］ダイアログ（HCSCTE プロジェクト新規作成のページ）が表示されます。



3. 次の項目を設定して，［次へ］ボタンをクリックします。

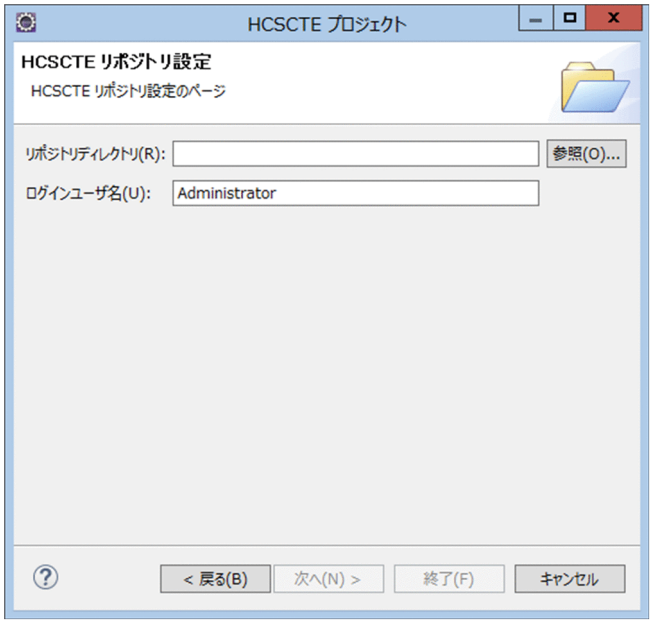
プロジェクト名

任意の名称を指定します。ここでは「HCSCTE」とします。

デフォルト・ロケーションの使用

［デフォルト・ロケーションの使用］チェックボックスにチェックします。

［HCSCTE プロジェクト］ダイアログ（HCSCTE リポジトリ設定）が表示されます。



4. 次の項目を設定して，［終了］ボタンをクリックします。

リポジトリディレクトリ

リポジトリ情報を格納する任意のディレクトリを指定します。ここでは，「C:¥work¥ProductStock_REST¥repository」とします。リポジトリディレクトリを指定する場合は，次の点に注意してください。

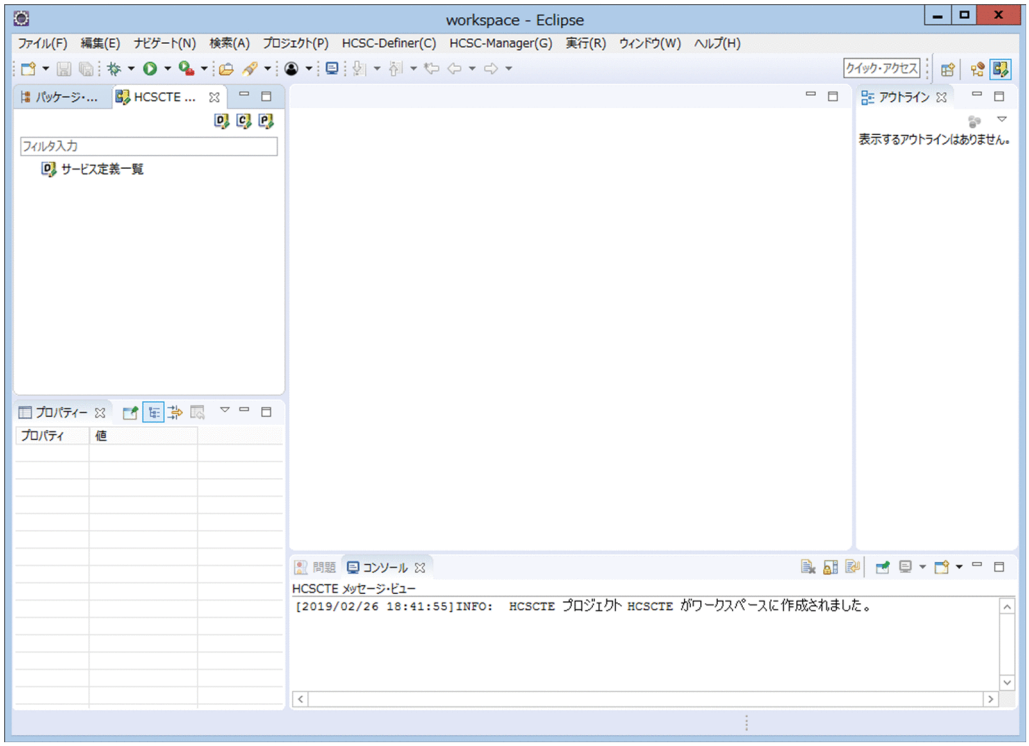
- リポジトリディレクトリのパスとプロジェクトのパスには，同じパスを指定しないでください。
- パスは，絶対パスで指定してください。
- パスの長さは，正規化された絶対パスでチェックされます。

ログインユーザ名

リポジトリログインに使用するユーザ名を指定します。ログインユーザ名は製品内部だけで使用されるため，任意の値で問題ありません。ユーザ名に使用できる文字は半角英数字だけで，長さは 1～16 文字になります。

「関連付けられたパースペクティブを開きますか？」というパースペクティブを切り替えるかどうかを確認するダイアログが表示された場合は，［パースペクティブを開く］ボタンをクリックします。

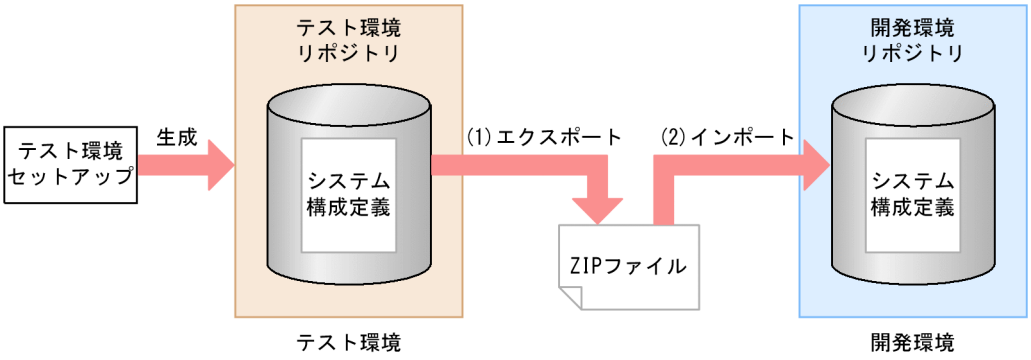
HCSCTE のプロジェクトが作成され，HCSCTE のプロジェクトのパースペクティブが起動します。



3.5.3　テスト環境のシステム構成定義を開発環境に取り込むための設定

開発環境からテスト環境を操作するためには，テスト環境のリポジトリにあるシステム構成定義と開発環境のリポジトリにあるシステム構成定義を同一にする必要があります。システム構成定義の用途の詳細については，マニュアル「サービスプラットフォーム 解説」の「1.1.6　配備定義機能」を参照してください。

テスト環境のシステム構成定義を開発環境に取り込むための手順を次の図に示します。テスト環境の構築で作成されるシステム構成定義を含むリポジトリをエクスポートし，開発環境にインポートします。



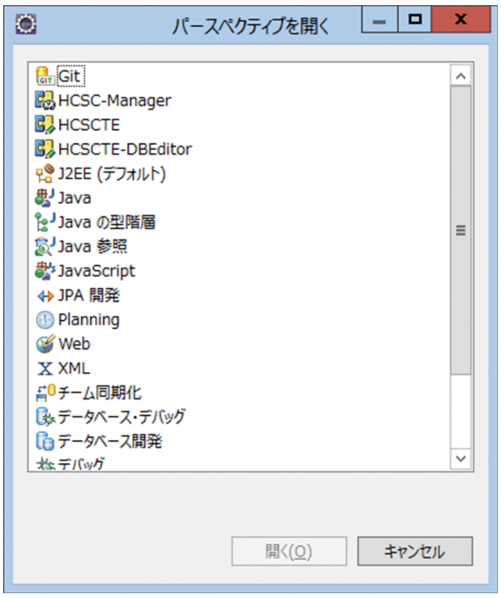
(1)　テスト環境からのシステム構成定義のエクスポート

テスト環境のシステム構成定義を開発環境に取り込むため，テスト環境のリポジトリ情報を ZIP 形式のファイルとしてエクスポートします。エクスポートの手順を次に示します。

ポイント

システム構成定義には HCSC コンポーネントの配備情報も含まれます。HCSC コンポーネントが配備されていない状態のテスト環境リポジトリを保存しておくため，テスト環境構築後に必ず 1 回のみ実行し，エクスポートしたファイルを退避しておいてください。

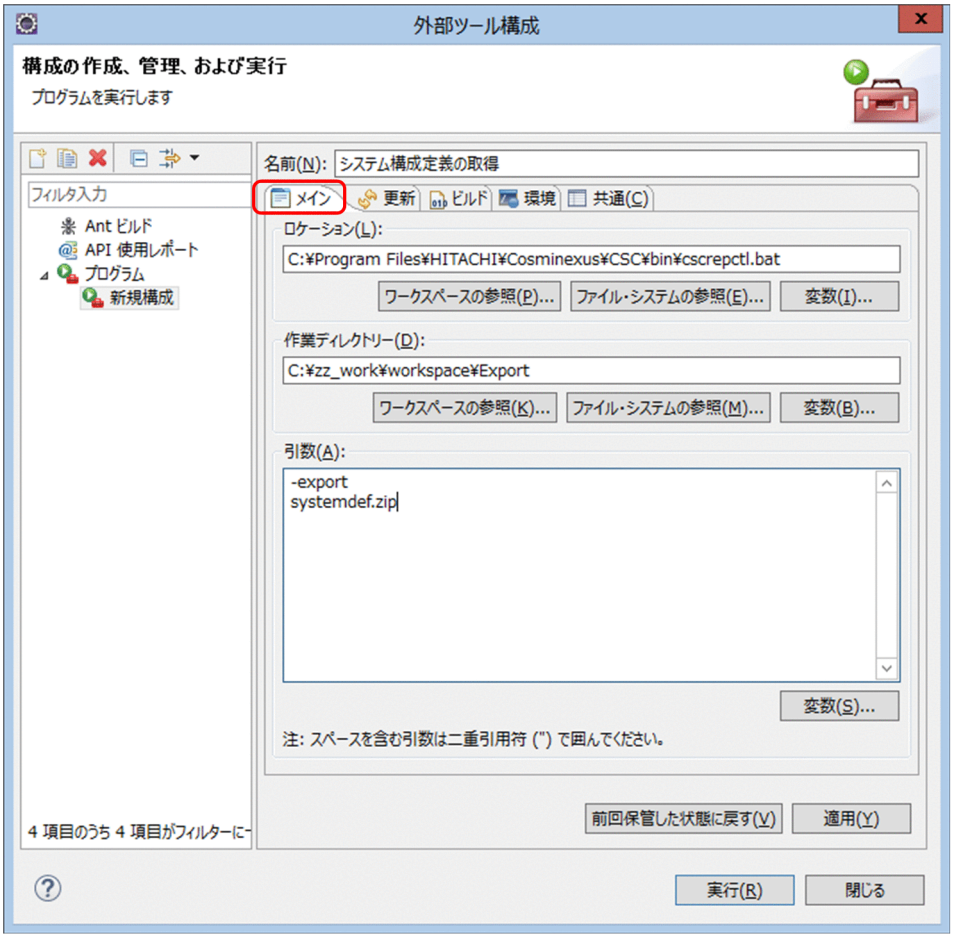
1. Eclipse のメニューから［ウィンドウ］－［パースペクティブを開く］－［その他］を選択します。
   ［パースペクティブを開く］ダイアログが表示されます。



2. [HCSCTE] を選択し，［開く］ボタンをクリックします。
   ［HCSCTE］パースペクティブが表示されます。

3. Eclipse のメニューから［実行］－［外部ツール］－［外部ツールの構成］を選択します。
   ［外部ツール構成］ダイアログが表示されます。

4. 左ペインの［プログラム］をダブルクリックし，［新規構成］を選択します。
   ［構成の作成、管理、および実行］ページが表示されます。



［メイン］タブを選択して，次の内容を入力します。

項目名	設定値
名前	任意の名称を指定します。 ここでは，「システム構成定義の取得」と指定します。
ロケーション	${env_var:COSMINEXUS_HOME}¥CSC¥bin¥cscrepctl.bat または［ファイル・システムの参照］ボタンから次のファイルを選択します。 <Service Architect のインストールディレクトリ>¥CSC¥bin¥cscrepctl.bat
作業ディレクトリー	引数で指定するファイルの出力先ディレクトリを任意で指定します。
引数	-export <任意出力ファイル名>.zip ここでは，「systemdef.zip」と指定します。

5. ［実行］ボタンをクリックします。

コマンドが登録，および実行され，テスト環境のリポジトリ情報が ZIP 形式のファイルとして，指定した出力先ディレクトリにエクスポートされます。

(2)　開発環境へのシステム構成定義のインポート

エクスポートしたテスト環境のリポジトリ情報の中から，システム構成定義だけを開発環境のリポジトリにインポートします。インポートの手順を次に示します。

1. Eclipse のメニューから，［HCSC-Definer］－［定義情報管理］－［全定義情報インポート］を選択します。

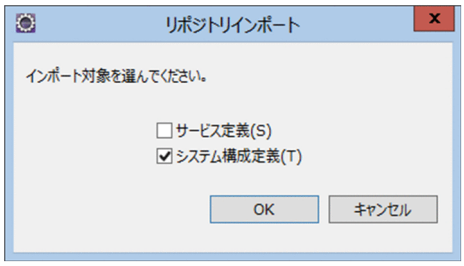
   リポジトリの上書きを確認するダイアログが表示されます。

2. ［はい］ボタンをクリックします。

   リポジトリ情報の ZIP ファイルを選択する［リポジトリインポート］ダイアログが表示されます。テスト環境からエクスポートした「systemdef.zip」の ZIP ファイルを指定します。

3. ［開く］ボタンをクリックします。

   インポート対象の定義情報を選択する［リポジトリインポート］ダイアログが表示されます。



4. ［システム構成定義］のチェックボックスにチェックし，［OK］ボタンをクリックします。

   テスト環境のシステム構成定義が取り込まれると，正常に完了したことを示すダイアログが表示されます。

5. ［OK］ボタンをクリックします。

これで，商品手配システムの開発・テスト環境の構築ができました。

4 サービス部品を構築する

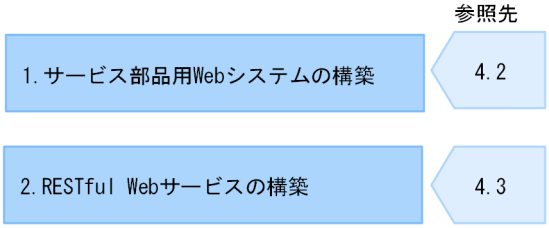
この章では，HCSC コンポーネントと連携するサービス部品の構築方法について説明します。

4.1　サービス部品を構築する流れ

商品手配システムで構築するサービス部品は，Web システムの JAX-RS 機能を使用した RESTful Web サービスです。そのため，RESTful Web サービスを配備するための Web システムを新規に構築します。構築した Web システムに RESTful Web サービスを配備することで，サービスを開始できます。

サービス部品を構築する手順を次の図に示します。

図 4-1　サービス部品を構築する手順



1. サービス部品用 Web システムの構築

Smart Composer 機能を使用して，サービス部品用 Web システムを構築します。詳細については，「4.2　サービス部品用 Web システムの構築」を参照してください。

2. RESTful Web サービスの構築

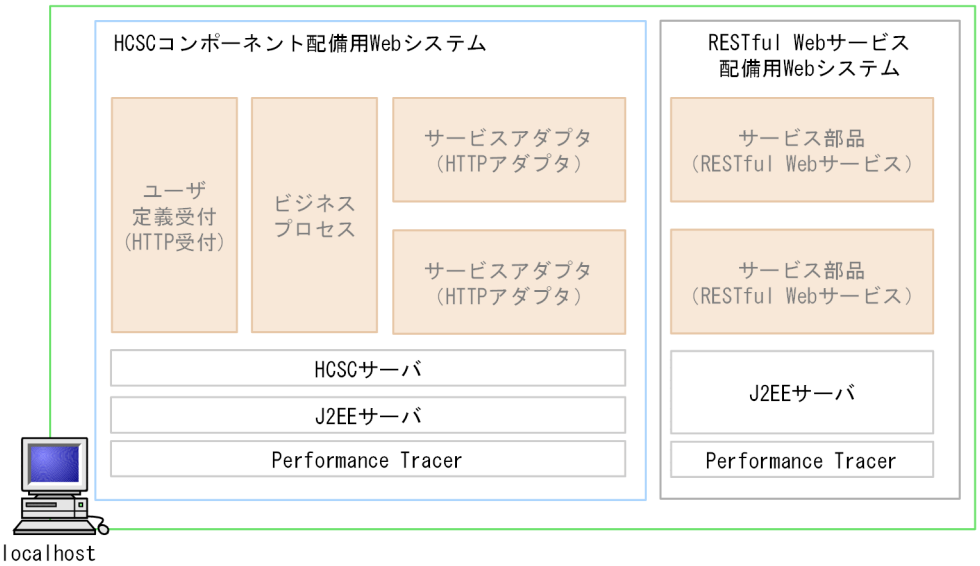
1.で構築した Web システムの J2EE サーバ上に RESTful Web サービスを，コマンドを使用して構築します。詳細については，「4.3　RESTful Web サービスの構築」を参照してください。

4.2　サービス部品用 Web システムの構築

商品手配システムでは，localhost 上に HCSC コンポーネント配備用 Web システムと，RESTful Web サービス配備用 Web システムの 2 つの Web システムを用意します。

HCSC コンポーネント配備用 Web システムと RESTful Web サービス配備用 Web システムの概要を，次の図に示します。

図 4-2　HCSC コンポーネント配備用 Web システムと RESTful Web サービス配備用 Web システムの概要



HCSC コンポーネント配備用 Web システムは，「3.4　テスト環境の構築」で構築済みです。ここでは，RESTful Web サービス配備用 Web システムの構築方法について説明します。

RESTful Web サービス配備用 Web システムは，J2EE サーバ，および Performance Tracer を配置し，Smart Composer 機能を用いて構築します。

RESTful Web サービス配備用 Web システムの構築手順を次に示します。この操作は，管理者または管理者特権で実行してください。

1. cmx_build_system コマンドを実行して，RESTful Web サービス配備用 Web システム（RESTfulWebServiceSystem）を構築します。

   コマンドの実行例を次に示します。

```
”%COSMINEXUS_HOME%¥manager¥bin¥cmx_build_system” -m localhost -u admin -p admin -f ”%COSM
INEXUS_HOME%¥CSCTE¥Samples¥SOAP1.1_1.2mode¥ProductStock_REST¥Server¥jaxrsserver_def.xml”
```

   cmx_build_system コマンドには，簡易構築定義ファイル（jaxrsserver_def.xml）を引数として指定します。簡易構築定義ファイル（jaxrsserver_def.xml）の格納先を次に示します。

```
<Service Architectのインストールディレクトリ>¥CSCTE¥Samples¥SOAP1.1_1.2mode¥ProductStock_
REST¥Server¥jaxrsserver_def.xml
```

cmx_build_system コマンドの詳細については，マニュアル「アプリケーションサーバ リファレンス コマンド編」の「cmx_build_system（Web システムの構築)」を参照してください。

2. cmx_start_target コマンドを実行して，RESTful Web サービス配備用 Web システム（RESTfulWebServiceSystem）を起動します。

コマンドの実行例を次に示します。

```
”%COSMINEXUS_HOME%¥manager¥bin¥cmx_start_target” -m localhost -u admin -p admin -s RESTfu
lWebServiceSystem -mode ALL
```

cmx_start_target コマンドの詳細については，マニュアル「アプリケーションサーバ リファレンス コマンド編」の「cmx_start_target（Web システムまたはサービスユニットの開始)」を参照してください。

4.3　RESTful Web サービスの構築

RESTful Web サービスは，「4.2　サービス部品用 Web システムの構築」で構築した Web システムの J2EE サーバ上で稼働させます。

商品手配システムでは，サンプルプログラムが提供するサービス部品のプロジェクトディレクトリを利用して，在庫管理サービスと配送受付サービスの RESTful Web サービスを作成します。

在庫管理サービス

注文を受けた商品名と個数から在庫数を更新し，引当番号を返します。

配送受付サービス

引当番号を基に，配送を手配して配送番号を返します。

各サービスの構成，およびファイル構成の詳細については，「付録 B　RESTful Web サービスのサービス構成とファイル構成」を参照してください。

また，サービス構築時に使用するコマンドの詳細については，マニュアル「アプリケーションサーバ リファレンス コマンド編」の「2.　J2EE サーバで使用するコマンド」を参照してください。

4.3.1　在庫管理サービスの構築

在庫管理サービスの構築手順を次に示します。この操作は，管理者または管理者特権で実行してください。

1. cjimportapp コマンドを実行し，サービスを配備します。

   cjimportapp コマンドの実行例を次に示します。

```
"%COSMINEXUS_HOME%¥CC¥admin¥bin¥cjimportapp" jaxrsserver -nameserver corbaname::localhost
:901 -f "%COSMINEXUS_HOME%¥CSCTE¥Samples¥SOAP1.1_1.2mode¥ProductStock_REST¥Service¥Invent
oryManagement¥inventorymanagement.ear"
```

2. cjstartapp コマンドを実行し，サービスを開始します。

   cjstartapp コマンドの実行例を次に示します。

```
"%COSMINEXUS_HOME%¥CC¥admin¥bin¥cjstartapp" jaxrsserver -nameserver corbaname::localhost:
901 -name Sample_application_jaxrs_inventorymanagement
```

4.3.2　配送受付サービスの構築

配送受付サービスの構築手順を次に示します。この操作は，管理者または管理者特権で実行してください。

1. cjimportapp コマンドを実行し，サービスを配備します。

   cjimportapp コマンドの実行例を次に示します。

```
"%COSMINEXUS_HOME%¥CC¥admin¥bin¥cjimportapp" jaxrsserver -nameserver corbaname::localhost
:901 -f "%COSMINEXUS_HOME%¥CSCTE¥Samples¥SOAP1.1_1.2mode¥ProductStock_REST¥Service¥Delive
ryReceipt¥delivery.ear"
```

2. cjstartapp コマンドを実行し，サービスを開始します。

cjstartapp コマンドの実行例を次に示します。

```
"%COSMINEXUS_HOME%¥CC¥admin¥bin¥cjstartapp" jaxrsserver -nameserver corbaname::localhost:
901 -name Sample_application_jaxrs_delivery
```

5 システムの開発を体験する

この章では，商品手配システムのサービスアダプタやビジネスプロセスを定義する方法について説明します。

5.1　商品手配システムの開発の手順

この章では，商品手配システムの開発の手順について説明します。

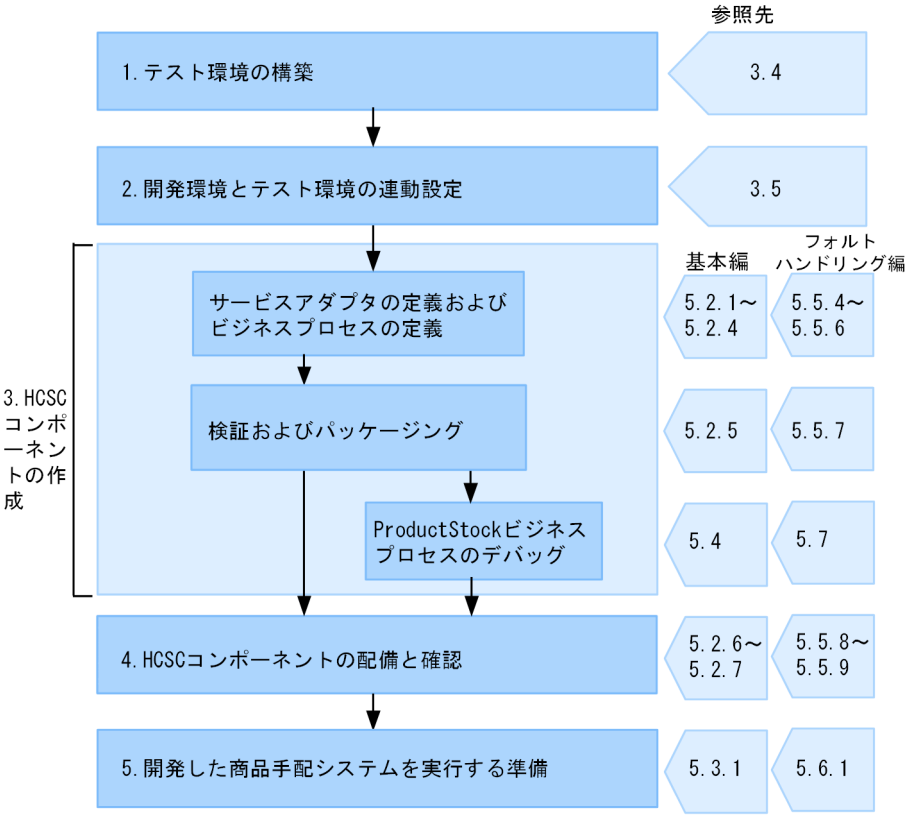
注意事項

下記の手順を行う前に，次の章で説明している手順を実行している必要があります。

- 「3.　商品手配システムの開発・テスト環境を構築する」
- 「4.　サービス部品を構築する」

商品手配システムの開発の手順を次の図に示します。

図 5-1　商品手配システムの開発の手順



1. テスト環境の構築

HCSC 簡易セットアップ機能を使用してテスト環境を構築します。詳細については，「3.4.1　HCSC 簡易セットアップ」を参照してください。テスト環境の構築が終わりましたら，テスト環境を起動してください。詳細については，「3.4.2　テスト環境の起動」を参照してください。

2. 開発環境とテスト環境の連動設定

開発環境の SOAP モードの設定とテスト環境のシステム構成定義を開発環境に取り込むための設定をします。詳細については，「3.5　開発環境とテスト環境の連動設定」を参照してください。

3. HCSC コンポーネントの作成

サービス部品を呼び出すサービスアダプタ，サービスアダプタを経由して複数のサービス部品を呼び出すビジネスプロセスなどの HCSC コンポーネントを作成します。詳細については次の個所を参照してください。

商品手配システムの開発（基本編）の場合

「5.2　商品手配システムの開発（基本編）」

商品手配システムの開発（フォルトハンドリング編）の場合

「5.5　商品手配システムの開発（フォルトハンドリング編）」

また，作成したビジネスプロセスをデバッグする場合は，次の個所を参照してください。

商品手配システムの開発（基本編）の場合

「5.4　ProductStock_Normal ビジネスプロセスのデバッグ」

商品手配システムの開発（フォルトハンドリング編）の場合

「5.7　ProductStock_FaultHandling ビジネスプロセスのデバッグ」

4. HCSC コンポーネントの配備と確認

HCSC コンポーネントをサーバに配備し，起動状態を確認します。詳細については次の個所を参照してください。

商品手配システムの開発（基本編）の場合

配備については「5.2.6　HCSC コンポーネントの配備と開始」を参照してください。

起動状態の確認については「5.2.7　HCSC コンポーネントの配備内容の確認」を参照してください。

商品手配システムの開発（フォルトハンドリング編）の場合

配備については「5.5.8　HCSC コンポーネントの配備と開始」を参照してください。

起動状態の確認については「5.5.9　HCSC コンポーネントの配備内容の確認」を参照してください。

5. 開発した商品手配システムを実行する準備

商品手配システムの動作を確認するための準備を行います。詳細については次の個所を参照してください。

商品手配システムの開発（基本編）の場合

「5.3.1　開発した商品手配システム（基本編）を実行する準備」

商品手配システムの開発（フォルトハンドリング編）の場合

「5.6.1　開発した商品手配システム（フォルトハンドリング編）を実行する準備」

以降の節で，これらの手順について説明します。

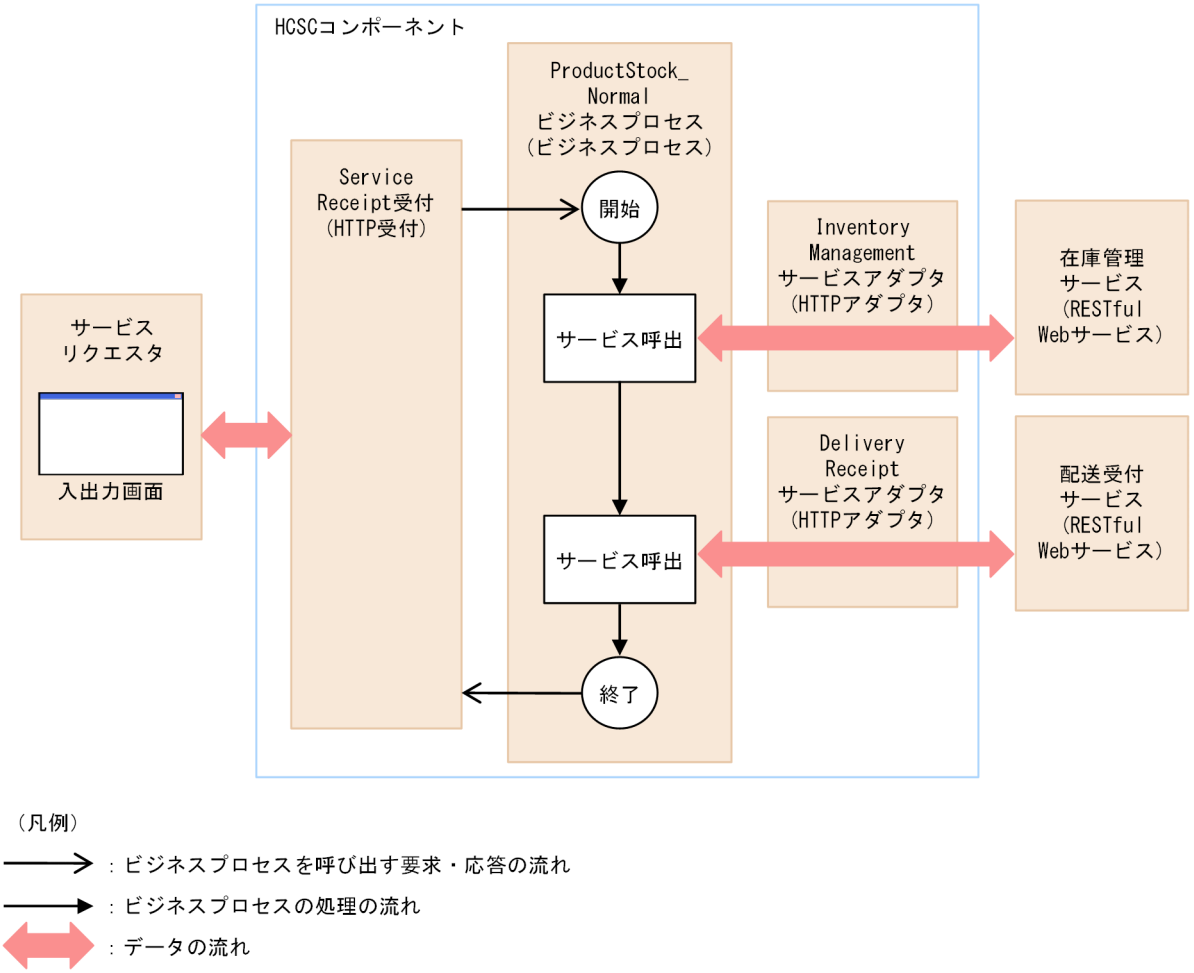
5.2　商品手配システムの開発（基本編）

商品手配システム（基本編）では，次に示す４つのコンポーネントを定義します。

- InventoryManagement サービスアダプタ
- DeliveryReceipt サービスアダプタ
- ProductStock_Normal ビジネスプロセス
- ServiceReceipt 受付

これらのコンポーネントの関係を次の図に示します。

図 5-2　コンポーネントの関係（商品手配システム（基本編））



商品手配システム（基本編）では，ServiceReceipt 受付を ProductStock_Normal ビジネスプロセスに定義します。

ServiceReceipt 受付はサービスリクエスタ側のインタフェースの役割を持ち，サービスリクエスタからの実行要求を受け付けると，ProductStock_Normal ビジネスプロセスにデータを渡します。そして，ビジネスプロセスは定義されたフローで順次実行する役割を持ちます。

このフロー内でサービスアダプタを利用することで，次のサービス部品を呼び出せます。

- InventoryManagement サービスアダプタ
  在庫管理サービスを呼び出します。
- DeliveryReceipt サービスアダプタ
  配送受付サービスを呼び出します。

5.2.1 HTTP アダプタを動作させるためのクラスパスの設定

実行環境で HTTP アダプタを動作させるために，usrconf.cfg（J2EE サーバ用オプション定義ファイル）をエディタで開き，クラスパスを追加します。この項の操作は，管理者または管理者特権で実行してください。

- usrconf.cfg の格納場所

```
<Service Architectのインストールディレクトリ>¥CC¥server¥usrconf¥ejb¥J2EEServer¥usrconf.cfg
```

- usrconf.cfg に追加するパス

```
add.class.path=<Service Architectのインストールディレクトリ>¥CC¥javaee¥1100¥lib¥jaxrs-impl.jar
```

クラスパスの情報を反映させるため，クラスパスを追加したあと，次の手順で HCSC サーバを再起動してください。

1. ［スタート］メニューの［プログラム］から［Cosminexus※］－［テストサーバ停止］を選択します。
   HCSC サーバが停止します。
2. ［スタート］メニューの［プログラム］から［Cosminexus※］－［テストサーバ起動］を選択します。
   HCSC サーバが再起動します。

注※
プログラムフォルダ名を変更している場合，変更したプログラムフォルダから選択してください。

5.2.2 InventoryManagement サービスアダプタの定義

InventoryManagement サービスアダプタについて，機能要件，設定値および設定手順を説明します。この項の操作は，管理者または管理者特権で実行してください。

(1) 機能要件

InventoryManagement サービスアダプタは，RESTful Web サービスである在庫管理サービスを呼び出す HTTP アダプタです。次の条件を満たすように作成します。

(a)　HTTP メソッド

- POST

(b)　データ形式

- JSON

(c)　URL

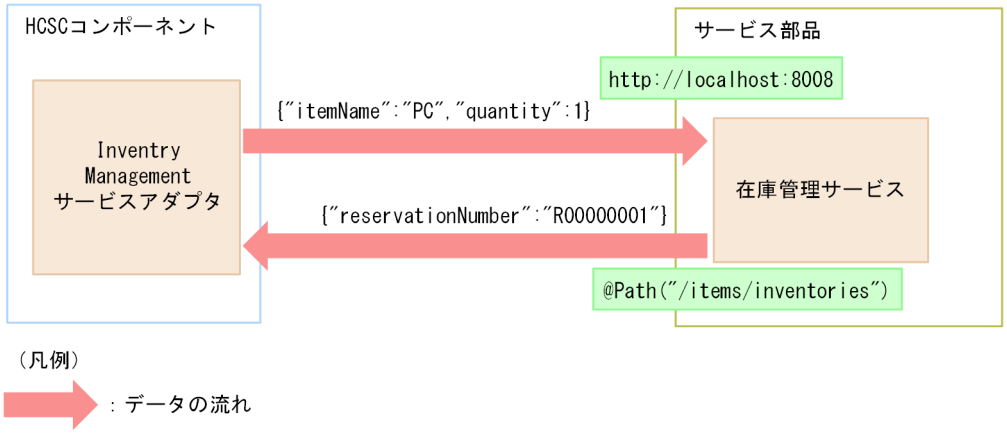
- スキーム：http
- オーソリティ：localhost:8008
- パス：/items/inventories

(d)　インタフェース

	名	変数名	型
入力	商品名	itemName	NCName
	個数	quantity	integer
出力	引当番号	reservationNumber	NCName

これらの要件を満たした InventoryManagement サービスアダプタの構成を次の図に示します。

図 5-3　InventoryManagement サービスアダプタの構成



(2)　アダプタの作成

InventoryManagement サービスアダプタを定義するときに設定する値を次の表に示します。

表 5-1　InventoryManagement サービスアダプタを定義するときに設定する値

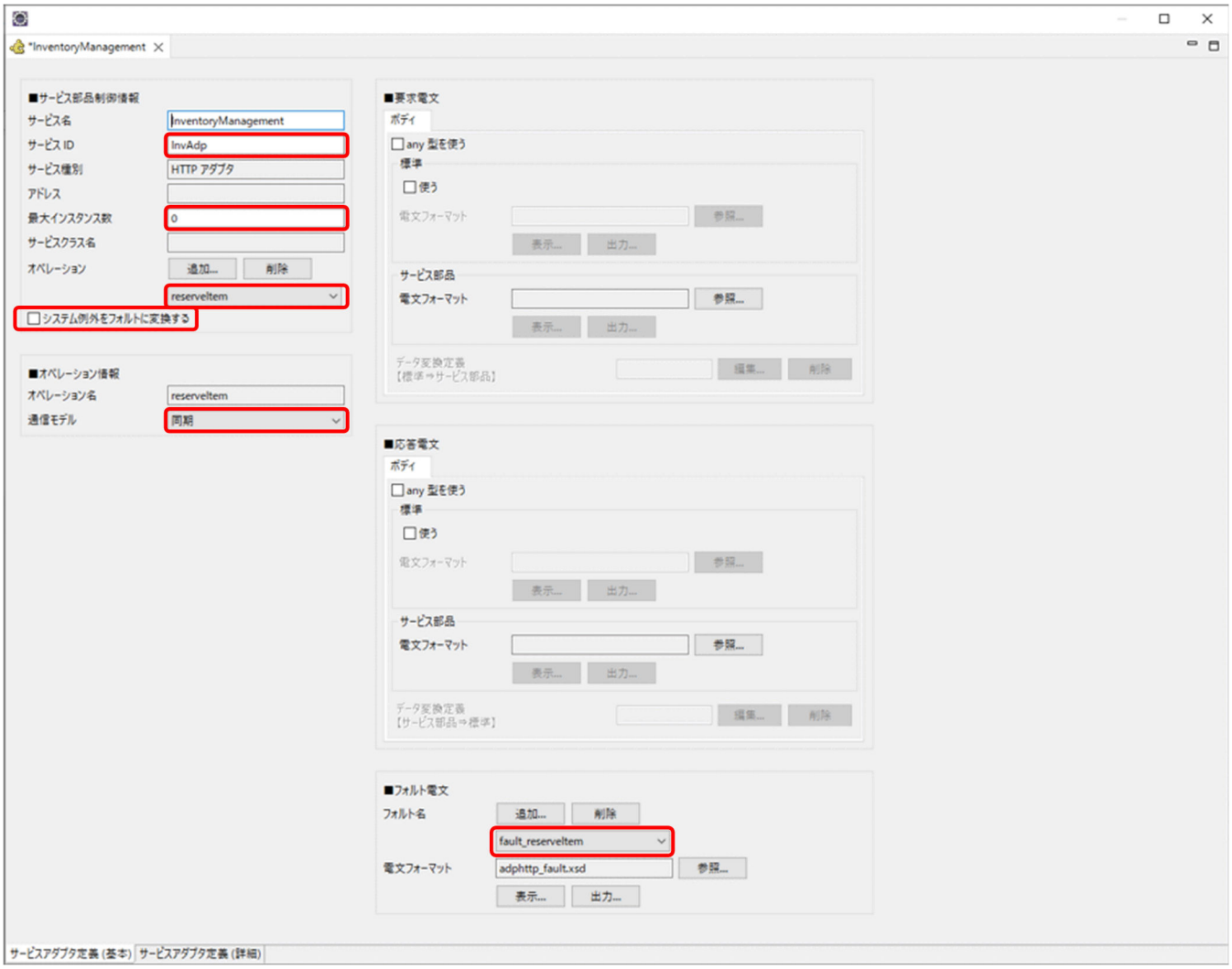
属性名	属性値
サービス種別	HTTP アダプタ

属性名	属性値
サービス名	InventoryManagement
サービス ID	InvAdp
最大インスタンス数	0（初期値）
オペレーション	reserveItem
通信モデル	同期（初期値）
システム例外をフォルトに変換する	チェックしない（初期値）
フォルト名	fault_reserveItem

InventoryManagement サービスアダプタの追加と定義手順を次に示します。

1. Eclipse が終了状態の場合，Eclipse を起動します。
2. ツリービューの［サービス定義一覧］を選択し，右クリックして，［サービスアダプタ追加］を選択します。
   サービスアダプタの種別を設定するダイアログが表示されます。
3. ドロップダウンリストから「HTTP アダプタ」を選択して，［次へ］ボタンをクリックします。
   HTTP アダプタの追加に必要な情報を入力するダイアログが表示されます。
4. サービス名に「InventoryManagement」を入力します。
5. ［終了］ボタンをクリックします。
   サービスアダプタ「InventoryManagement サービスアダプタ」が作成され，サービスアダプタ定義画面が表示されます。
6. サービスアダプタ定義（基本）画面で次のように設定します。
   - 「サービス ID」に「InvAdp」を設定
   - 「最大インスタンス数」に「0」（初期値）を設定
   - 「オペレーション」の「追加」ボタンをクリックし，オペレーション名に「reserveItem」，「フォルト名」に「fault_reserveItem」を設定して，［OK］ボタンをクリック
   - 「システム例外をフォルトに変換する」のチェックを外す（初期値）
   - 「通信モデル」に「同期」（初期値）を設定

   設定後のサービスアダプタ定義（基本）タブの例を次に示します。



オペレーション情報の通信モデルは「同期」から変更しないでください。

7. メニューから［ファイル］－［保管］を選択します。

(3)　電文フォーマットの作成

HTTP アダプタに設定する要求電文と応答電文の電文フォーマット（XML スキーマ）は，使用する JSON ファイルから cscjson2xsd コマンドを実行して作成します。

注意

ここで作成する JSON ファイルは，すべて文字コード UTF-8 で作成してください。

(a)　要求電文

要求電文の XML スキーマの作成について説明します。ここで作成した XML スキーマ「input_Inv.xsd」は，HTTP アダプタの要求電文に設定します。

- コマンドで使用する JSON ファイルの作成

テキストエディタを起動し，次の内容の JSON ファイルを作成します。このファイルの内容は，InventoryManagement サービスアダプタが在庫管理サービスを呼び出す際の電文の一例です。

```
{"itemName":"PC", "quantity":1}
```

ここでは，ファイル名を input_Inv.json とし，この JSON ファイルを任意の場所に格納します。

- cscjson2xsd コマンドの実行例

  この実行例では，入力する JSON ファイルの格納場所を「C:¥Users¥work」，入力する JSON ファイルの名称を「input_Inv.json」，出力する XML スキーマの名称を「input_Inv.xsd」としています。

```
C:¥Users¥work>"%COSMINEXUS_HOME%¥CSCTE¥bin¥cscjson2xsd.bat" -in input_Inv.json -out input
_Inv.xsd
KECT93001-I Execution of the command will now start. (command = cscjson2xsd)
KECT93002-I Execution of the command ended normally. (command = cscjson2xsd)
```

- 作成された XML スキーマ（input_Inv.xsd）

  コマンド実行後，次の内容の XML スキーマ「input_Inv.xsd」が作成されます。

  実際に作成される XML スキーマは，次に示す内容と異なる場合がありますが，スキーマとしての定義内容は同じであるため，問題ありません。

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
  <xs:element name="csc-object">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="itemName"/>
        <xs:element ref="quantity"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="itemName" type="xs:NCName"/>
  <xs:element name="quantity" type="xs:integer"/>
</xs:schema>
```

ここで作成した JSON ファイル，および XML スキーマのサンプルは次のディレクトリに格納されています。

```
<Service Architectのインストールディレクトリ>¥CSCTE¥Samples¥SOAP1.1_1.2mode¥ProductStock_RES
T¥Schema¥InventoryManagementAdapter¥request
```

(b) 応答電文

応答電文の XML スキーマの作成について説明します。ここで作成した XML スキーマ「output_Inv.xsd」は，HTTP アダプタの応答電文に設定します。

- コマンドで使用する JSON ファイルの作成

  テキストエディタを起動し，次の内容の JSON ファイルを作成します。このファイルの内容は，在庫管理サービスが InventoryManagement サービスアダプタに応答する際の電文の一例です。

```
{"reservationNumber": "R00000001"}
```

ここでは，ファイル名を output_Inv.json とし，この JSON ファイルを任意の場所に格納します。

- cscjson2xsd コマンドの実行例

  この実行例では，入力する JSON ファイルの格納場所を「C:¥Users¥work」，入力する JSON ファイルの名称を「output_Inv.json」，出力する XML スキーマの名称を「output_Inv.xsd」としています。

```
C:¥Users¥work>"%COSMINEXUS_HOME%¥CSCTE¥bin¥cscjson2xsd.bat" -in output_Inv.json -out outp
ut_Inv.xsd
KECT93001-I Execution of the command will now start. (command = cscjson2xsd)
KECT93002-I Execution of the command ended normally. (command = cscjson2xsd)
```

- 作成された XML スキーマ（output_Inv.xsd）

  コマンド実行後，次の内容の XML スキーマ「output_Inv.xsd」が作成されます。

  実際に作成される XML スキーマは，次に示す内容と異なる場合がありますが，スキーマとしての定義内容は同じであるため，問題ありません。

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
  <xs:element name="csc-object">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="reservationNumber"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="reservationNumber" type="xs:NCName"/>
</xs:schema>
```

ここで作成した JSON ファイル，および XML スキーマのサンプルは次のディレクトリに格納されています。

```
<Service Architectのインストールディレクトリ>¥CSCTE¥Samples¥SOAP1.1_1.2mode¥ProductStock_RES
T¥Schema¥InventoryManagementAdapter¥response
```

(c) 電文フォーマットの指定

作成した XML スキーマを InventoryManagement サービスアダプタに設定します。

1. ツリービューの［サービス定義一覧］から InventoryManagement サービスアダプタをダブルクリックします。

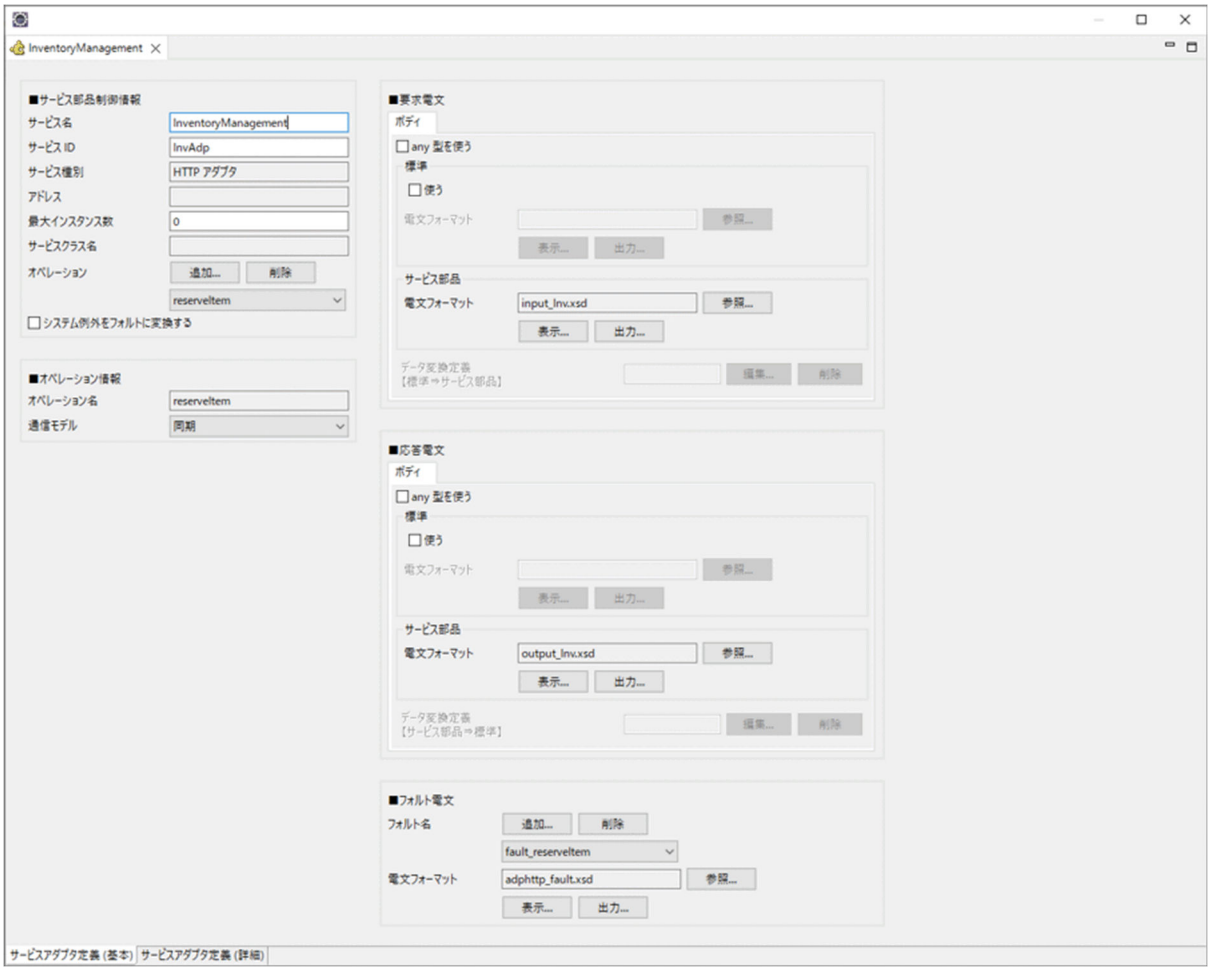
   サービスアダプタ定義（基本）画面が表示されます。

2. ［要求電文］－［サービス部品］－［電文フォーマット］の［参照］ボタンをクリックして，XML スキーマ「input_Inv.xsd」を指定します。

3. ［応答電文］－［サービス部品］－［電文フォーマット］の［参照］ボタンをクリックして，XML スキーマ「output_Inv.xsd」を指定します。

4. メニューから［ファイル］－［保管］を選択します。

設定後のサービスアダプタ定義（基本）タブの例を次に示します。



(4)　定義ファイルの作成

次に示す定義ファイルを作成します。

- HTTP アダプタ実行環境プロパティファイル
  URL など，HTTP アダプタの構成情報を設定します。
- HTTP アダプタ定義ファイル
  HTTP アダプタで JSON-XML 変換機能を利用することを設定します。
  商品手配システムでは，ビジネスプロセスで扱う XML 形式の要求電文を，InventoryManagement サービスアダプタで JSON 形式に変換して在庫管理サービスに送信します。また，在庫管理サービスから応答された JSON 形式の応答電文を XML 形式に変換し，ビジネスプロセスへ応答します。
  JSON-XML 変換機能の詳細については，マニュアル「サービスプラットフォーム 解説」の「2.16.15 HTTP アダプタの JSON-XML 変換機能」を参照してください。

それぞれの定義ファイルについて説明します。

(a)　HTTP アダプタ実行環境プロパティファイル

HTTP アダプタ実行環境プロパティファイルの作成手順を次に示します。この操作は，管理者または管理者特権で実行してください。

1. HTTP アダプタ実行環境プロパティファイルのテンプレートファイルをコピー＆ペーストします。

コピー元のテンプレートファイル

<Service Architect のインストールディレクトリ>¥CSC¥custom-adapter¥HTTP¥config¥template¥serviceid.properties

コピー先のディレクトリ

<Service Architect のインストールディレクトリ>¥CSC¥custom-adapter¥HTTP¥config

2. コピー先のファイル名を，<HTTP アダプタのサービス ID>.properties となるように「InvAdp.properties」へ変更します。

3. ファイル「InvAdp.properties」に次のパラメタを追加します。

追加するパラメタ	設定値	説明
adphttp.request.method	POST	POST メソッドを使用します。
adphttp.request.uri-scheme-authority	http://localhost:8008	スキーム+オーソリティを指定します。
adphttp.request.uri-path	/items/inventories	パス名を指定します。
adphttp.request.header.content-type	application/json	HTTP リクエストヘッダの Content-Type ヘッダのメディアタイプに JSON 形式を指定します。
adphttp.request.part.message.binding	raw	パススルーモードとし，要求電文（ボディ）で指定したデータを HTTP リクエストボディに設定します。

記述例を次に示します。

```
adphttp.request.method=POST
adphttp.request.uri-scheme-authority=http://localhost:8008
adphttp.request.uri-path=/items/inventories
adphttp.request.header.content-type=application/json
adphttp.request.part.message.binding=raw
```

HTTP アダプタ実行環境プロパティファイルは，HTTP アダプタの開始時に実行環境に反映されます。そのため，HTTP アダプタを開始済みで，HTTP アダプタ実行環境プロパティファイルの内容を変更する場合は，いったん HTTP アダプタを停止する必要があります。

(b) HTTP アダプタ定義ファイル

HTTP アダプタ定義ファイルの編集方法を説明します。

1. [サービスアダプタ定義（詳細)] タブでサービスアダプタ定義（詳細）画面に切り替えて，「独自定義ファイル」の「cscadphttp.properties」を選択します。

「cscadphttp.properties」は HTTP アダプタ定義ファイルです。

2. [編集] ボタンをクリックして HTTP アダプタ定義ファイルを開き，次のパラメタを追加します。

追加するパラメタ	設定値	説明
adphttp.request.switchover.json-transfer.mode	true	リクエスト処理時に JSON-XML 変換を使用します。
adphttp.response.switchover.json-transfer.mode	true	レスポンス処理時に JSON-XML 変換を使用します。

記述例を次に示します。

```
adphttp.request.switchover.json-transfer.mode=true
adphttp.response.switchover.json-transfer.mode=true
```

3. メニューから［ファイル］－［すべて保管］を選択します。

5.2.3 DeliveryReceipt サービスアダプタの定義

DeliveryReceipt サービスアダプタについて，機能要件，設定値および設定手順を説明します。この項の操作は，管理者または管理者特権で実行してください。

(1) 機能要件

DeliveryReceipt サービスアダプタは，RESTful Web サービスである配送受付サービスを呼び出す HTTP アダプタです。次の条件を満たすように作成します。

(a) HTTP メソッド

- POST

(b) データ形式

- JSON

(c) URL

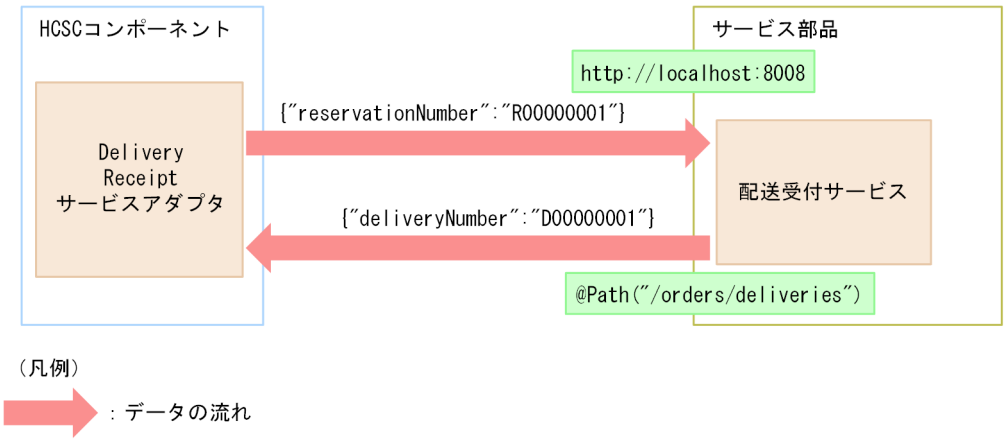
- スキーム：http
- オーソリティ：localhost:8008
- パス：/orders/deliveries

(d) インタフェース

	名	変数名	型
入力	引当番号	reservationNumber	NCName
出力	配送番号	deliveryNumber	NCName

これらの要件を満たした DeliveryReceipt サービスアダプタの構成を次の図に示します。

図 5-4　DeliveryReceipt サービスアダプタの構成



(2)　アダプタの作成

DeliveryReceipt サービスアダプタを定義するときに設定する値を次の表に示します。

表 5-2　DeliveryReceipt サービスアダプタを定義するときに設定する値

属性名	属性値
サービス種別	HTTP アダプタ
サービス名	DeliveryReceipt
サービス ID	DelAdp
最大インスタンス数	0（初期値）
オペレーション	deliverItem
通信モデル	同期（初期値）
システム例外をフォルトに変換する	チェックしない（初期値）
フォルト名	fault_deliverItem

DeliveryReceipt サービスアダプタの追加と定義手順を次に示します。

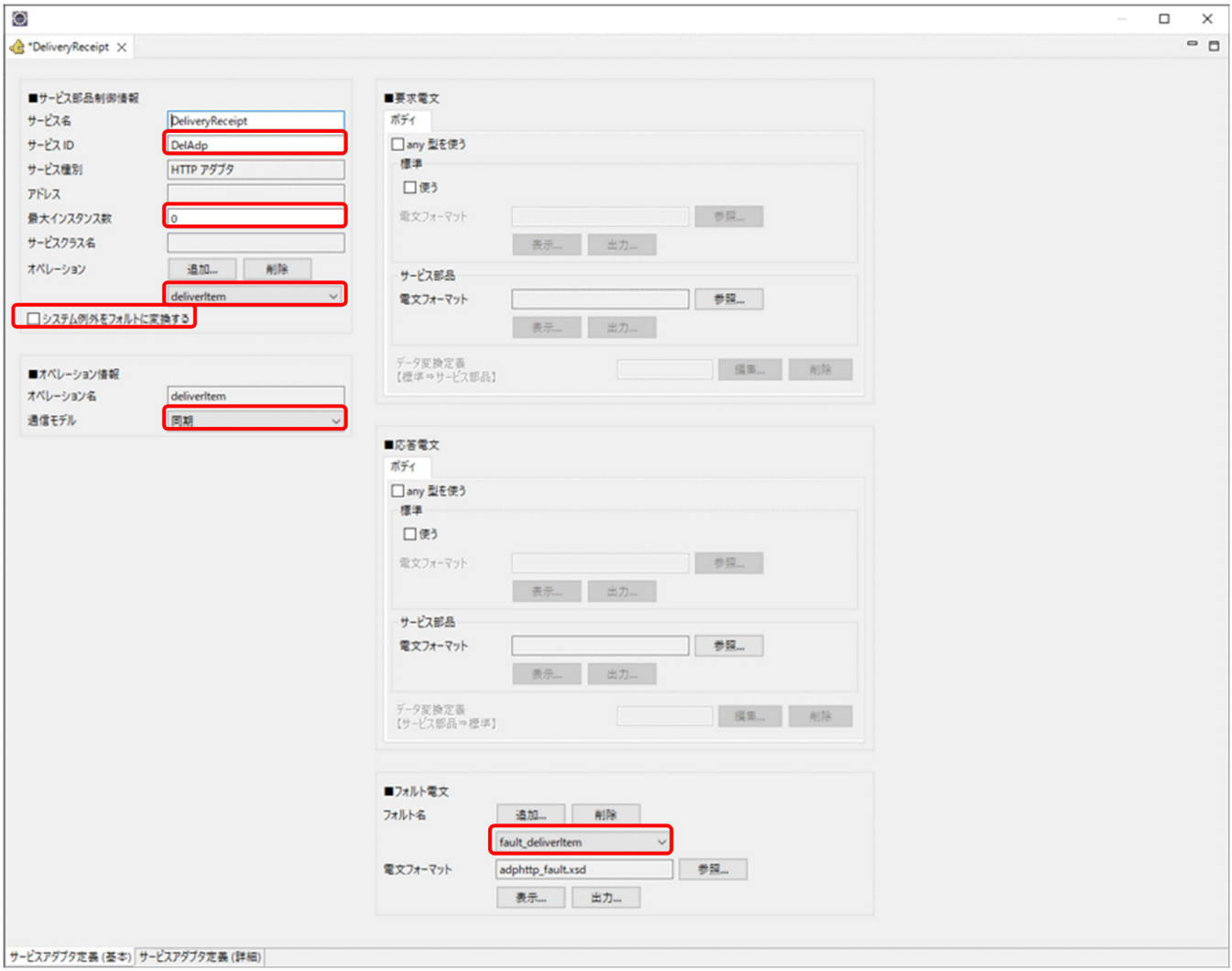
1. ツリービューの［サービス定義一覧］を選択し，右クリックして，［サービスアダプタ追加］を選択します。
   サービスアダプタの種別を設定するダイアログが表示されます。
2. ドロップダウンリストから「HTTP アダプタ」を選択して，［次へ］ボタンをクリックします。
   HTTP アダプタの追加に必要な情報を入力するダイアログが表示されます。
3. サービス名に「DeliveryReceipt」を入力します。
4. ［終了］ボタンをクリックします。

サービスアダプタ「DeliveryReceipt サービスアダプタ」が作成され，サービスアダプタ定義画面が表示されます。

5. サービスアダプタ定義（基本）画面で次のように設定します。

- 「サービス ID」に「DelAdp」を設定
- 「最大インスタンス数」に「0」（初期値）を設定
- 「オペレーション」の「追加」ボタンをクリックし，オペレーション名に「deliverItem」，「フォルト名」に「fault_deliverItem」を設定して，［OK］ボタンをクリック
- 「システム例外をフォルトに変換する」のチェックを外す（初期値）
- 「通信モデル」に「同期」（初期値）を設定

設定後のサービスアダプタ定義（基本）タブの例を次に示します。



オペレーション情報の通信モデルは「同期」から変更しないでください。

6. メニューから［ファイル］－［保管］を選択します。

(3) 電文フォーマットの作成

HTTP アダプタに設定する要求電文と応答電文の電文フォーマット（XML スキーマ）は，使用する JSON から cscjson2xsd コマンドを実行して作成します。

注意

ここで作成する JSON ファイルは，すべて文字コード UTF-8 で作成してください。

(a) 要求電文

要求電文の XML スキーマの作成について説明します。ここで作成した XML スキーマ「input_Del.xsd」は，HTTP アダプタの要求電文に設定します。

- コマンドで使用する JSON ファイルの作成

  テキストエディタを起動し，次の内容の JSON ファイルを作成します。このファイルの内容は，DeliveryReceipt サービスアダプタが配送受付サービスを呼び出す際の電文の一例です。

```
{"reservationNumber": "R00000001"}
```

  ここでは，ファイル名を input_Del.json とし，この JSON ファイルを任意の場所に格納します。

- cscjson2xsd コマンドの実行例

  この実行例では，入力する JSON ファイルの格納場所を「C:¥Users¥work」，入力する JSON ファイルの名称を「input_Del.json」，出力する XML スキーマの名称を「input_Del.xsd」をとしています。

```
C:¥Users¥work>"%COSMINEXUS_HOME%¥CSCTE¥bin¥cscjson2xsd.bat" -in input_Del.json -out input
_Del.xsd
KECT93001-I Execution of the command will now start. (command = cscjson2xsd)
KECT93002-I Execution of the command ended normally. (command = cscjson2xsd)
```

- 作成された XML スキーマ（input_Del.xsd）

  コマンド実行後，次の内容の XML スキーマ「input_Del.xsd」が作成されます。

  実際に作成される XML スキーマは，次に示す内容と異なる場合がありますが，スキーマとしての定義内容は同じであるため，問題ありません。

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
  <xs:element name="csc-object">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="reservationNumber"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="reservationNumber" type="xs:NCName"/>
</xs:schema>
```

ここで作成した JSON ファイル，および XML スキーマのサンプルは次のディレクトリに格納されています。

```
<Service Architectのインストールディレクトリ>¥CSCTE¥Samples¥SOAP1.1_1.2mode¥ProductStock_RES
T¥Schema¥DeliveryReceiptAdapter¥request
```

(b) 応答電文

応答電文の XML スキーマの作成について説明します。ここで作成した XML スキーマ「output_Del.xsd」は，HTTP アダプタの応答電文に設定します。

- コマンドで使用する JSON ファイルの作成

  テキストエディタを起動し，次の内容の JSON ファイルを作成します。このファイルの内容は，配送受付サービスが DeliveryReceipt サービスアダプタに応答する際の電文の一例です。

  ```
  {"deliveryNumber": "D00000001"}
  ```

  ここでは，ファイル名を output_Del.json とし，この JSON ファイルを任意の場所に格納します。

- cscjson2xsd コマンドの実行例

  この実行例では，入力する JSON ファイルの格納場所を「C:¥Users¥work」，入力する JSON ファイルの名称を「output_Del.json」，出力する XML スキーマの名称を「output_Del.xsd」としています。

  ```
  C:¥Users¥work>"%COSMINEXUS_HOME%¥CSCTE¥bin¥cscjson2xsd.bat" -in output_Del.json -out outp
  ut_Del.xsd
  KECT93001-I Execution of the command will now start. (command = cscjson2xsd)
  KECT93002-I Execution of the command ended normally. (command = cscjson2xsd)
  ```

- 作成された XML スキーマ（output_Del.xsd）

  コマンド実行後，次の内容の XML スキーマ「output_Del.xsd」が作成されます。

  実際に作成される XML スキーマは，次に示す内容と異なる場合がありますが，スキーマとしての定義内容は同じであるため，問題ありません。

  ```
  <?xml version="1.0" encoding="UTF-8"?>
  <xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
    <xs:element name="csc-object">
      <xs:complexType>
        <xs:sequence>
          <xs:element ref="deliveryNumber"/>
        </xs:sequence>
      </xs:complexType>
    </xs:element>
    <xs:element name="deliveryNumber" type="xs:NCName"/>
  </xs:schema>
  ```

ここで作成した JSON ファイル，および XML スキーマのサンプルは次のディレクトリに格納されています。

```
<Service Architectのインストールディレクトリ>¥CSCTE¥Samples¥SOAP1.1_1.2mode¥ProductStock_RES
T¥Schema¥DeliveryReceiptAdapter¥response
```

(c) 電文フォーマットの指定

作成した XML スキーマを DeliveryReceipt サービスアダプタに設定します。

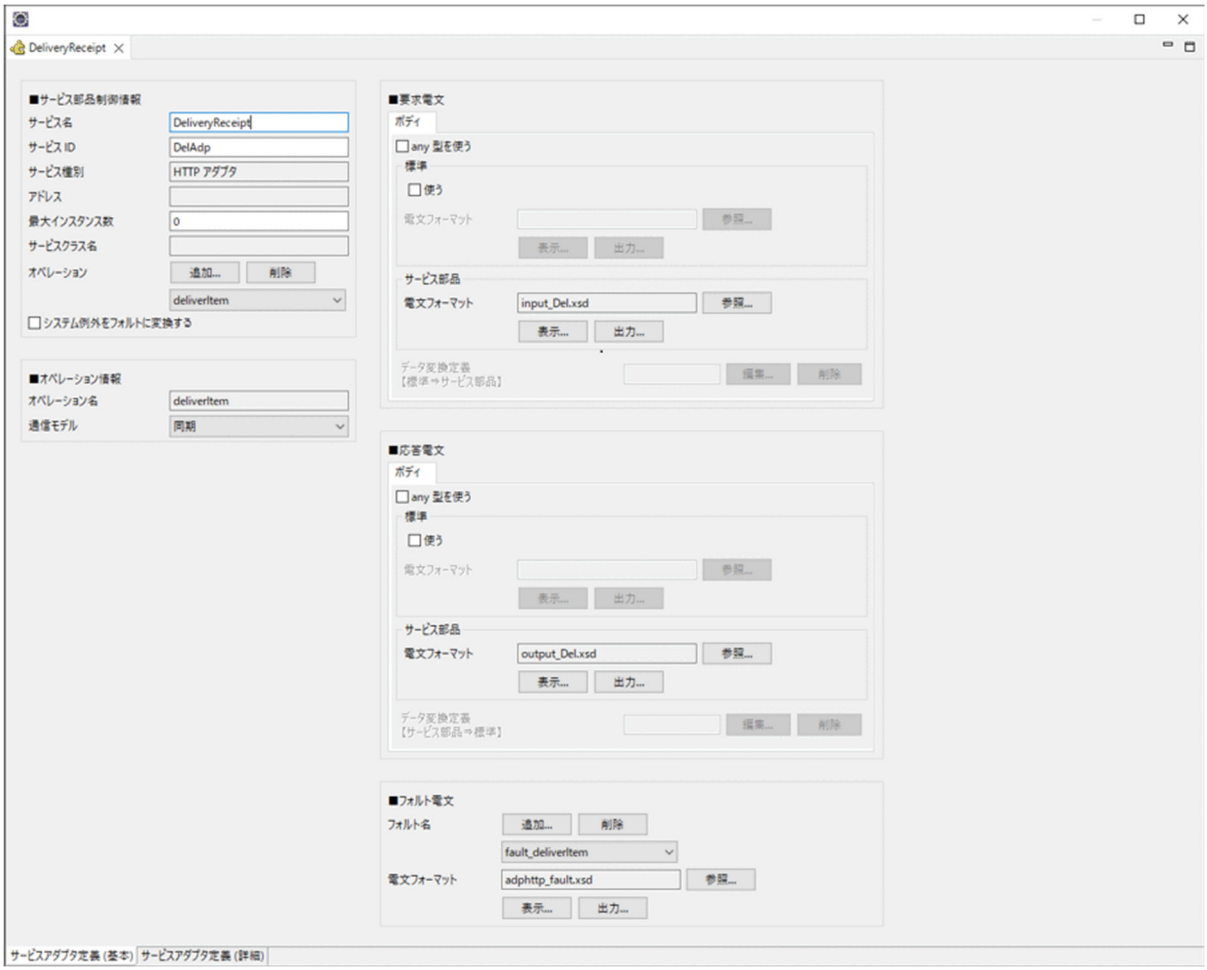
1. ツリービューの［サービス定義一覧］から DeliveryReceipt サービスアダプタをダブルクリックします。
   サービスアダプタ定義（基本）画面が表示されます。

2. ［要求電文］－［サービス部品］－［電文フォーマット］の［参照］ボタンをクリックして，XML スキーマ「input_Del.xsd」を指定します。

3. ［応答電文］－［サービス部品］－［電文フォーマット］の［参照］ボタンをクリックして，XML スキーマ「output_Del.xsd」を指定します。

4. メニューから［ファイル］－［保管］を選択します。

設定後のサービスアダプタ定義（基本）タブの例を次に示します。



(4)　定義ファイルの作成

次に示す定義ファイルを作成します。

- HTTP アダプタ実行環境プロパティファイル
  URL など，HTTP アダプタの構成情報を設定します。
- HTTP アダプタ定義ファイル
  HTTP アダプタで JSON-XML 変換機能を利用することを設定します。

商品手配システムでは，ビジネスプロセスで扱う XML 形式の要求電文を，DeliveryReceipt サービスアダプタで JSON 形式に変換して配送受付サービスに送信します。また，配送受付サービスから応答された JSON 形式の応答電文を XML 形式に変換し，ビジネスプロセスへ応答します。

JSON-XML 変換機能の詳細については，マニュアル「サービスプラットフォーム 解説」の「2.16.15 HTTP アダプタの JSON-XML 変換機能」を参照してください。

それぞれの定義ファイルについて説明します。

(a) HTTP アダプタ実行環境プロパティファイル

HTTP アダプタ実行環境プロパティファイルの作成手順を次に示します。この操作は，管理者または管理者特権で実行してください。

1. HTTP アダプタ実行環境プロパティファイルのテンプレートファイルをコピー＆ペーストします。

コピー元のテンプレートファイル

<Service Architect のインストールディレクトリ>¥CSC¥custom-adapter¥HTTP¥config¥template¥serviceid.properties

コピー先のディレクトリ

<Service Architect のインストールディレクトリ>¥CSC¥custom-adapter¥HTTP¥config

2. コピー先のファイル名を，<HTTP アダプタのサービス ID>.properties となるように「DelAdp.properties」へ変更します。

3. ファイル「DelAdp.properties」に次のパラメタを追加します。

追加するパラメタ	設定値	説明
adphttp.request.method	POST	POST メソッドを使用します。
adphttp.request.uri-scheme-authority	http://localhost:8008	スキーム+オーソリティを指定します。
adphttp.request.uri-path	/orders/deliveries	パス名を指定します。
adphttp.request.header.content-type	application/json	HTTP リクエストヘッダの Content-Type ヘッダのメディアタイプに JSON 形式を指定します。
adphttp.request.part.message.binding	raw	パススルーモードとし，要求電文（ボディ）で指定したデータを HTTP リクエストボディに設定します。

記述例を次に示します。

```
adphttp.request.method=POST
adphttp.request.uri-scheme-authority=http://localhost:8008
adphttp.request.uri-path=/orders/deliveries
adphttp.request.header.content-type=application/json
adphttp.request.part.message.binding=raw
```

HTTP アダプタ実行環境プロパティファイルは，HTTP アダプタの開始時に実行環境に反映されます。そのため，HTTP アダプタを開始済みで，HTTP アダプタ実行環境プロパティファイルの内容を変更する場合は，いったん HTTP アダプタを停止する必要があります。

(b) HTTP アダプタ定義ファイル

HTTP アダプタ定義ファイルの編集方法を説明します。

1. [サービスアダプタ定義（詳細)] タブでサービスアダプタ定義（詳細）画面に切り替えて，「独自定義ファイル」の「cscadphttp.properties」を選択します。
   「cscadphttp.properties」は HTTP アダプタ定義ファイルです。

2. [編集] ボタンをクリックして HTTP アダプタ定義ファイルを開き，次のパラメタを追加します。

追加するパラメタ	設定値	説明
adphttp.request.switchover.json-transfer.mode	true	リクエスト処理時に JSON-XML 変換を使用します。
adphttp.response.switchover.json-transfer.mode	true	レスポンス処理時に JSON-XML 変換を使用します。

   記述例を次に示します。

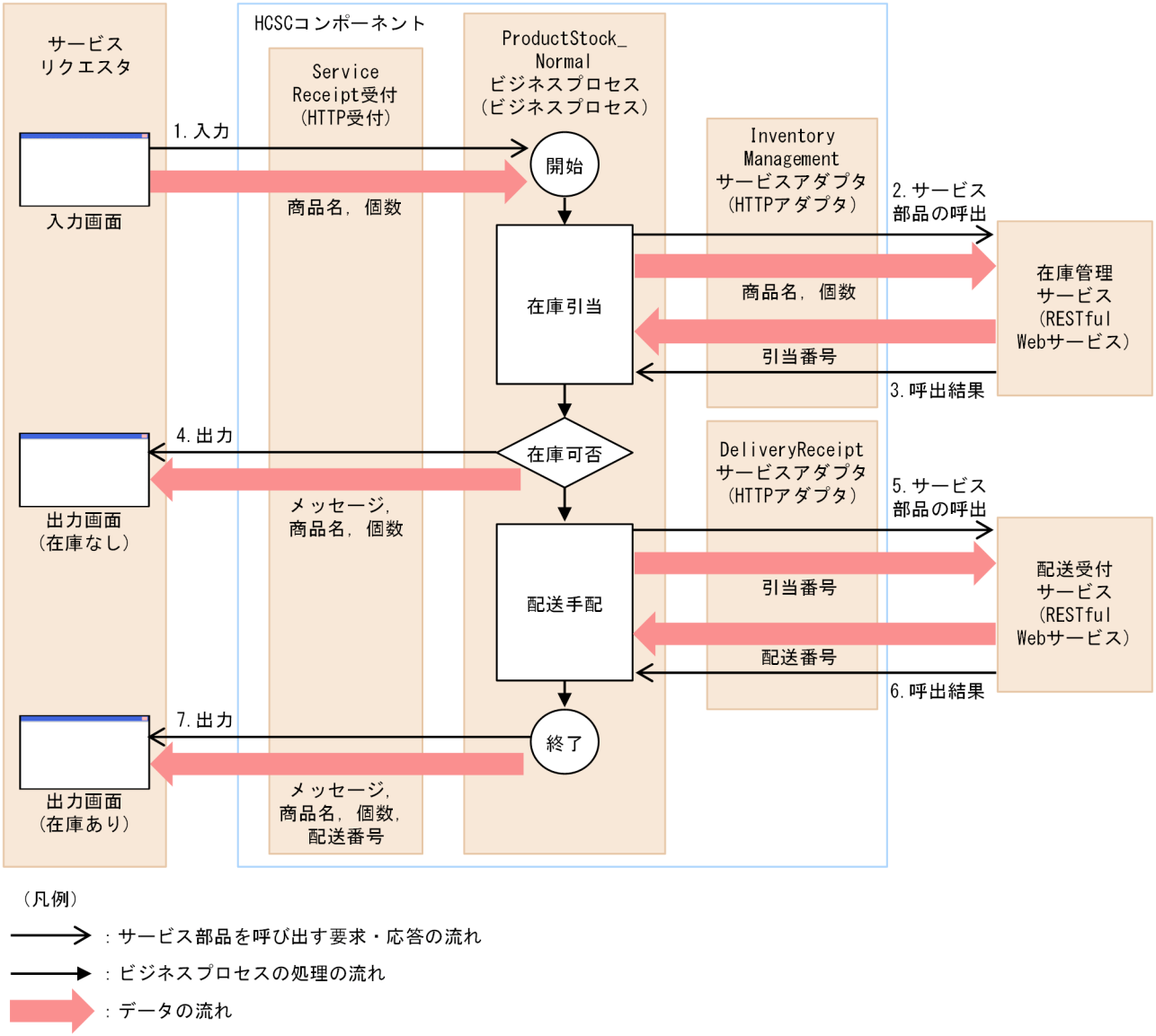
```
adphttp.request.switchover.json-transfer.mode=true
adphttp.response.switchover.json-transfer.mode=true
```

3. メニューから［ファイル］－［すべて保管］を選択します。

5.2.4 ProductStock_Normal ビジネスプロセスの定義

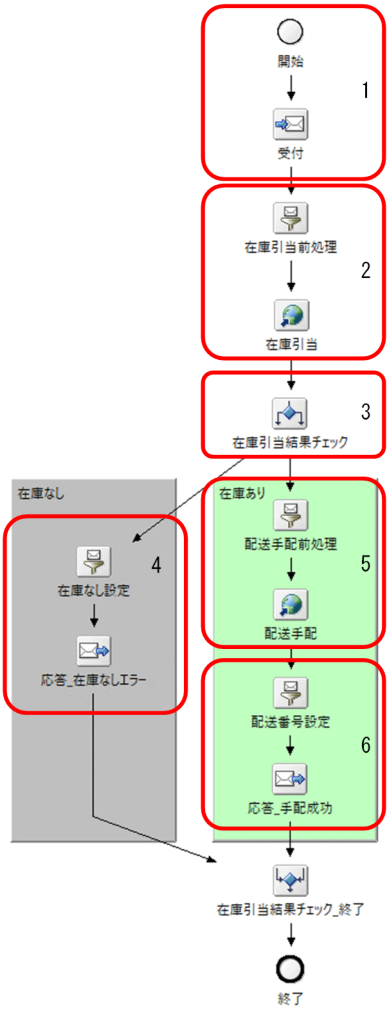
商品手配システム（基本編）の処理の流れを次の図に示します。

図 5-5　商品手配システム（基本編）の処理の流れ



この処理を商品手配システムの ProductStock_Normal ビジネスプロセスで実現すると，次のような処理になります。

図 5-6　ProductStock_Normal ビジネスプロセスの処理



ProductStock_Normal ビジネスプロセスは次の処理を実現します。

1. ProductStock_Normal ビジネスプロセスは，サービスリクエスタから商品名と個数の情報を受信します。
2. InventoryManagement サービスアダプタを介して在庫管理サービスを呼び出し，引当番号を取得します。
   在庫がない場合は「*」を取得します。
3. 在庫可否を確認します。
4. 在庫がない場合は，在庫なしレスポンスをサービスリクエスタに返却します。
5. 在庫がある場合は，DeliveryReceipt サービスアダプタを介して配送受付サービスを呼び出し，配送番号を取得します。
6. サービスリクエスタに，取得した配送番号を返します。

商品手配システムのリクエストには HTTP リクエストを使用します。そのため，ユーザ定義受付（HTTP 受付）をビジネスプロセスに関連づけて定義して，HTTP プロトコルでリクエストを受付できるようにします。

商品手配システムのビジネスプロセスは，次の流れで定義します。

1. 新規ビジネスプロセスを追加します。
   追加方法については，「(1) ビジネスプロセスの追加」を参照してください。
2. ユーザ定義受付を追加します。
   追加方法については，「(2) ユーザ定義受付の追加」を参照してください。
3. ユーザ定義受付の要求電文・応答電文フォーマットを作成します。
   作成方法については，「(3) 電文フォーマットの作成」を参照してください。
4. 変数を設定します。
   設定方法については，「(4) 変数の設定」を参照してください。
5. アクティビティを配置します。
   配置方法については，「(5) アクティビティの配置」を参照してください。
6. アクティビティを定義します。
   定義方法については，「(6) アクティビティの定義」を参照してください。
7. ビジネスプロセスの定義を終了します。

(1) ビジネスプロセスの追加

ProductStock_Normal ビジネスプロセスを追加するときに設定する値を次の表に示します。

表 5-3 ProductStock_Normal ビジネスプロセスを追加するときに設定する値

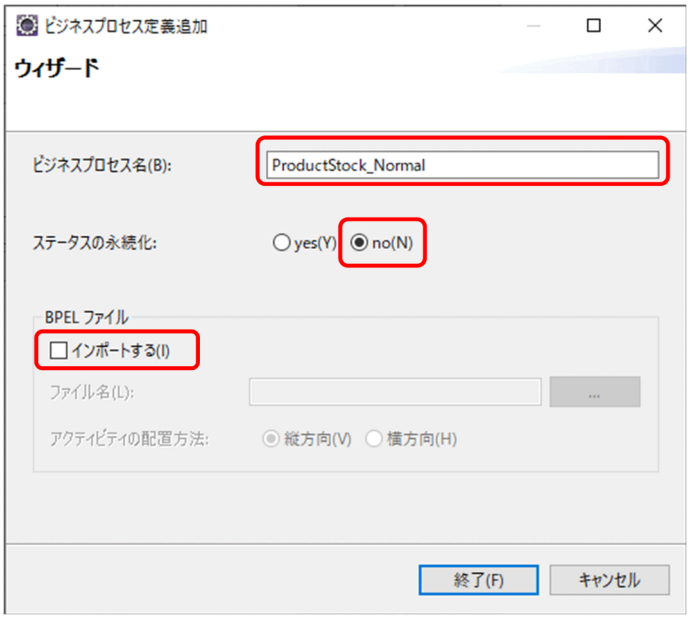
項目名	設定する値	説明
ビジネスプロセス名	ProductStock_Normal	ビジネスプロセスの名称を指定します。
ステータスの永続化	no	データベースに記録を残すかどうかを指定します。記録を残すと，プロセスの進捗状況などを把握できます。この商品手配システムでは，データベースに記録を残さないため，「no」を選択します。
BPEL ファイル	［インポートする］のチェックを外す	上流工程でツールを使って作成した BPEL をインポートする場合は，チェックを入れます。インポートすると，ビジネスプロセスに必要なアクティビティが自動的に表示されます。この商品手配システムでは，インポートしないので，チェックを外します。
サービス ID	ArrBPNor	ビジネスプロセスの ID を指定します。

項目名	設定する値	説明
スコープアクティビティのデフォルトハンドラ	OFF	スコープアクティビティのデフォルトハンドラの設定を「ON」にすると，補償処理が暗黙的に定義されるようになります。この商品手配システムでは，補償処理を定義しないので，「OFF」を選択します。

ProductStock_Normal ビジネスプロセスの追加手順を次に示します。

1. ツリービューの［サービス定義一覧］を選択し，右クリックして，［ビジネスプロセス追加］を選択します。
   ビジネスプロセス定義を追加するためのダイアログが表示されます。

2. ［ビジネスプロセス名］に「ProductStock_Normal」を入力し，［ステータスの永続化］で「no」を選択します。［BPEL ファイル］の［インポートする］はチェックを外したままにしてください。

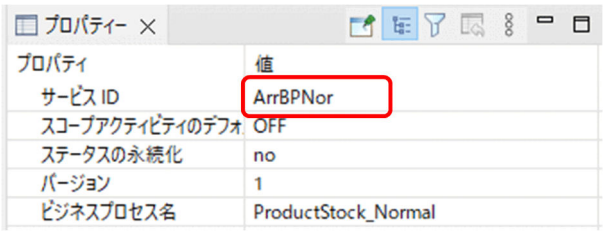


3. ［終了］ボタンをクリックします。
   ビジネスプロセス「ProductStock_Normal」が作成され，ビジネスプロセス定義画面が表示されます。

4. ツリービューで「ProductStock_Normal」を選択します。
   プロパティビューに ProductStock_Normal のプロパティ一覧が表示されます。

5. プロパティビューで，サービス ID の値のセルをクリックします。
   入力できる状態になります。

6. 「ArrBPNor」に変更し，［Enter］キーを押します。



7. 変更してよいかどうかを確認するメッセージが表示されるので，［OK］ボタンをクリックします。

(2)　ユーザ定義受付の追加

ユーザ定義受付では，サービスリクエスタとのインタフェースをあらかじめ決める必要があります。サービスリクエスタと HCSC コンポーネント間は REST で通信するため，ユーザ定義受付は HTTP 受付を設定します。ユーザ定義受付を追加するときに設定する値を次の表に示します。

表 5-4　ユーザ定義受付を追加するときに設定する値

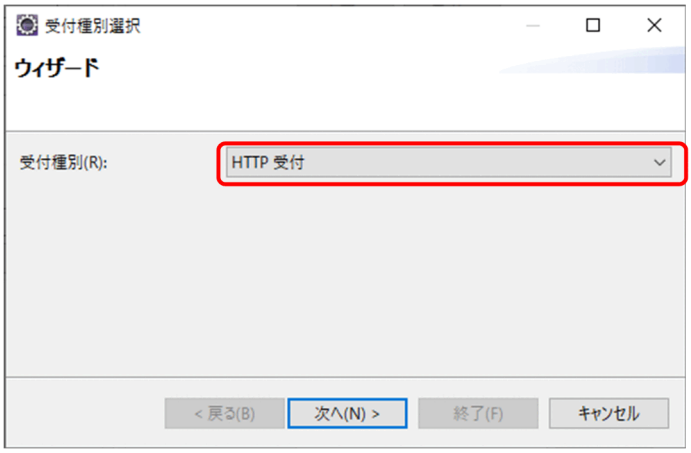
属性名	属性値
受付種別	HTTP 受付
受付名	ServiceReceipt
受付 ID	rcp1
オペレーション	arrangeItem
デフォルトオペレーション名	arrangeItem
通信モデル	同期（初期値）

ProductStock_Normal のユーザ定義受付の追加手順を次に示します。

1. ツリービューのサービス定義一覧からビジネスプロセス名［ProductStock_Normal］を選択し，右クリックして，［ユーザ定義受付（呼出先固定）追加］を選択します。

受付種別を選択するためのダイアログが表示されます。

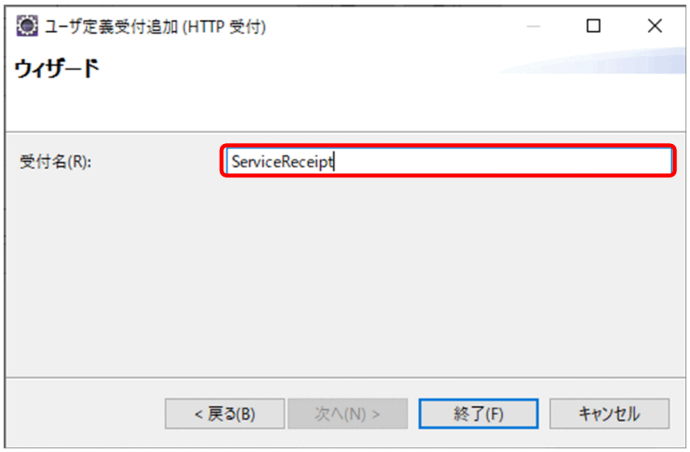
2. [受付種別] のドロップダウンリストから，「HTTP 受付」を選択します。



3. [次へ] ボタンをクリックします。

HTTP 受付を追加するためのダイアログが表示されます。

4. 受付名に「ServiceReceipt」を入力します。



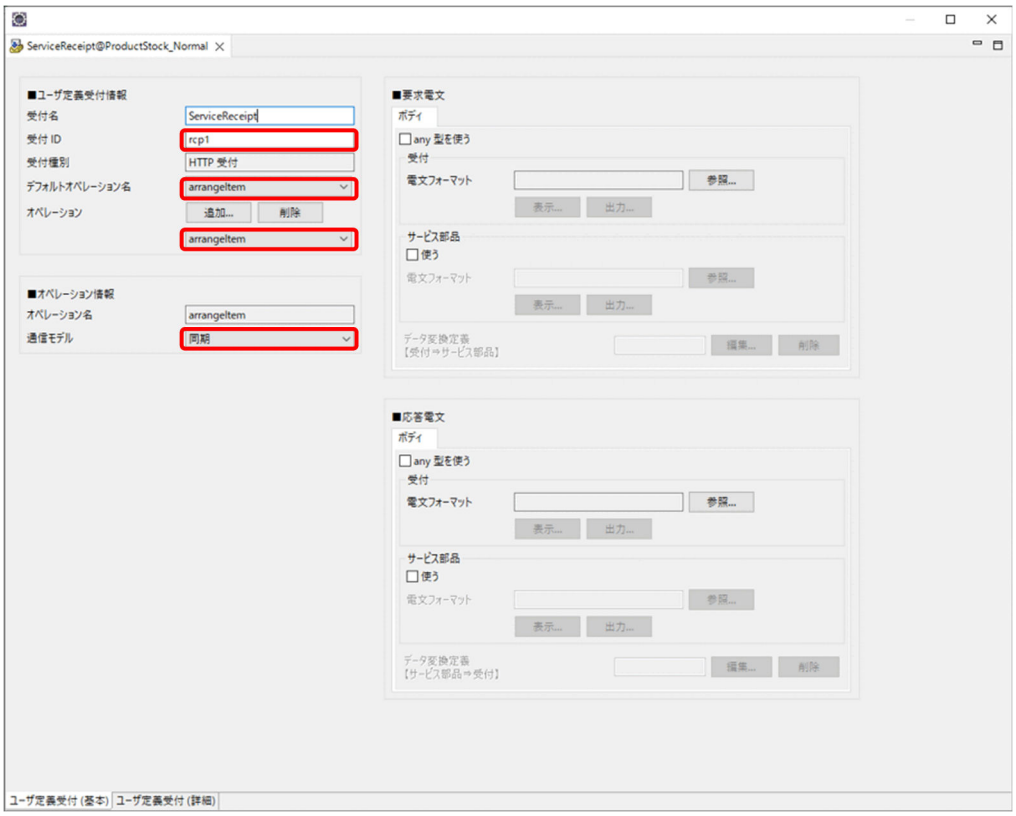
5. [終了] ボタンをクリックします。

HTTP 受付がビジネスプロセスに追加され，ユーザ定義受付定義画面が表示されます。

6. ユーザ定義受付定義（基本）画面で次のように設定します。

- 「受付 ID」に「rcp1」（初期値）を設定
- 「オペレーション」の「追加」ボタンをクリックし，オペレーション名に「arrangeItem」を設定して，[OK] ボタンをクリック
- 「デフォルトオペレーション名」に「arrangeItem」（初期値）を設定
- 「通信モデル」に「同期」（初期値）を設定

設定後のユーザ定義受付（基本）タブの例を次に示します。



オペレーション情報の通信モデルは「同期」から変更しないでください。

7. [ユーザ定義受付（詳細）] タブでユーザ定義受付定義（詳細）画面に切り替えて，「独自定義ファイル」の「cscurecphttp.properties」を選択します。

「cscurecphttp.properties」は HTTP 受付定義ファイルです。

8. [編集] ボタンをクリックして HTTP 受付定義ファイルを開き，次の内容を追加します。

追加するパラメタ	設定値	説明
httprecp.switchover.pass-through.mode	true	パススルーモードを設定します。
httprecp.pass-through.parameter-use	false	HTTP リクエストボディを要求電文として直接ビジネスプロセスに送信します。
httprecp.request.switchover.json-transfer.mode	true	リクエスト処理時に JSON-XML 変換を使用します。
httprecp.response.switchover.json-transfer.mode	true	レスポンス処理時に JSON-XML 変換を使用します。

記述例を次に示します。

```
httprecp.switchover.pass-through.mode=true
httprecp.pass-through.parameter-use=false
httprecp.request.switchover.json-transfer.mode=true
httprecp.response.switchover.json-transfer.mode=true
```

9. メニューから［ファイル］－［すべて保管］を選択します。

(3) 電文フォーマットの作成

HTTP 受付に設定する要求電文と応答電文の電文フォーマット（XML スキーマ）は，使用する JSON ファイルから cscjson2xsd コマンドを実行して作成します。

注意

ここで作成する JSON ファイルは，すべて文字コード UTF-8 で作成してください。

(a) 要求電文

要求電文の XML スキーマの作成について説明します。ここで作成した XML スキーマ「input_Arr.xsd」は，HTTP 受付の要求電文に設定します。

- コマンドで使用する JSON ファイルの作成

  テキストエディタを起動し，次の内容の JSON ファイルを作成します。このファイルの内容は，サービスリクエスタから ServiceReceipt 受付に送られる電文の一例です。

  ```
  {"itemName":"PC", "quantity":1}
  ```

  ここでは，ファイル名を input_Arr.json とし，この JSON ファイルを任意の場所に格納します。

- cscjson2xsd コマンドの実行例

この操作は，管理者または管理者特権で実行してください。

この実行例では，入力する JSON ファイルの格納場所を「C:¥Users¥work」，入力する JSON ファイルの名称を「input_Arr.json」，出力する XML スキーマの名称を「input_Arr.xsd」としています。

```
C:¥Users¥work>"%COSMINEXUS_HOME%¥CSCTE¥bin¥cscjson2xsd.bat" -in input_Arr.json -out input
_Arr.xsd
KECT93001-I Execution of the command will now start. (command = cscjson2xsd)
KECT93002-I Execution of the command ended normally. (command = cscjson2xsd)
```

- 作成された XML スキーマ（input_Arr.xsd）

  コマンド実行後，次の内容の XML スキーマ「input_Arr.xsd」が作成されます。

  実際に作成される XML スキーマは，次に示す内容と異なる場合がありますが，スキーマとしての定義内容は同じであるため，問題ありません。

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
  <xs:element name="csc-object">
    <xs:complexType>
      <xs:sequence>
        <xs:element ref="itemName"/>
        <xs:element ref="quantity"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="itemName" type="xs:NCName"/>
  <xs:element name="quantity" type="xs:integer"/>
</xs:schema>
```

ここで作成した JSON ファイル，および XML スキーマのサンプルは次のディレクトリに格納されています。

```
<Service Architectのインストールディレクトリ>¥CSCTE¥Samples¥SOAP1.1_1.2mode¥ProductStock_RES
T¥Schema¥ServiceReceipt¥request
```

(b) 応答電文

応答電文の XML スキーマの作成について説明します。ここで作成した XML スキーマ「output_ArrNor.xsd」は，HTTP 受付の応答電文に設定します。

- コマンドで使用する JSON ファイルの作成

  テキストエディタを起動し，次の内容の JSON ファイルを作成します。このファイルの内容は，サービスリクエスタが ServiceReceipt 受付から受け取る電文の一例です。

```
{"deliveryNumber": "D00000001", "itemName": "PC", "message": "Product is available for ar
rangement.", "quantity":"1"}
```

  ここでは，ファイル名を output_ArrNor.json とし，この JSON ファイルを任意の場所に格納します。

- cscjson2xsd コマンドの実行例

  この操作は，管理者または管理者特権で実行してください。

この実行例では，入力する JSON ファイルの格納場所を「C:¥Users¥work」，入力する JSON ファイルの名称を「output_ArrNor.json」，出力する XML スキーマの名称を「output_ArrNor.xsd」としています。

```
C:¥Users¥work>"%COSMINEXUS_HOME%¥CSCTE¥bin¥cscjson2xsd.bat" -in output_ArrNor.json -out o
utput_ArrNor.xsd
KECT93001-I Execution of the command will now start. (command = cscjson2xsd)
KECT93002-I Execution of the command ended normally. (command = cscjson2xsd)
```

- 作成された XML スキーマ（output_ArrNor.xsd）

  コマンド実行後，次の内容の XML スキーマ「output_ArrNor.xsd」が作成されます。

  実際に作成される XML スキーマは，次に示す内容と異なる場合がありますが，スキーマとしての定義内容は同じであるため，問題ありません。

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
<xs:element name="csc-object">
  <xs:complexType>
    <xs:sequence>
      <xs:element ref="deliveryNumber"/>
      <xs:element ref="itemName"/>
      <xs:element ref="message"/>
      <xs:element ref="quantity"/>
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="deliveryNumber" type="xs:NCName"/>
<xs:element name="itemName" type="xs:NCName"/>
<xs:element name="message" type="xs:string"/>
<xs:element name="quantity" type="xs:integer"/>
</xs:schema>
```

ここで作成した JSON ファイル，および XML スキーマのサンプルは次のディレクトリに格納されています。

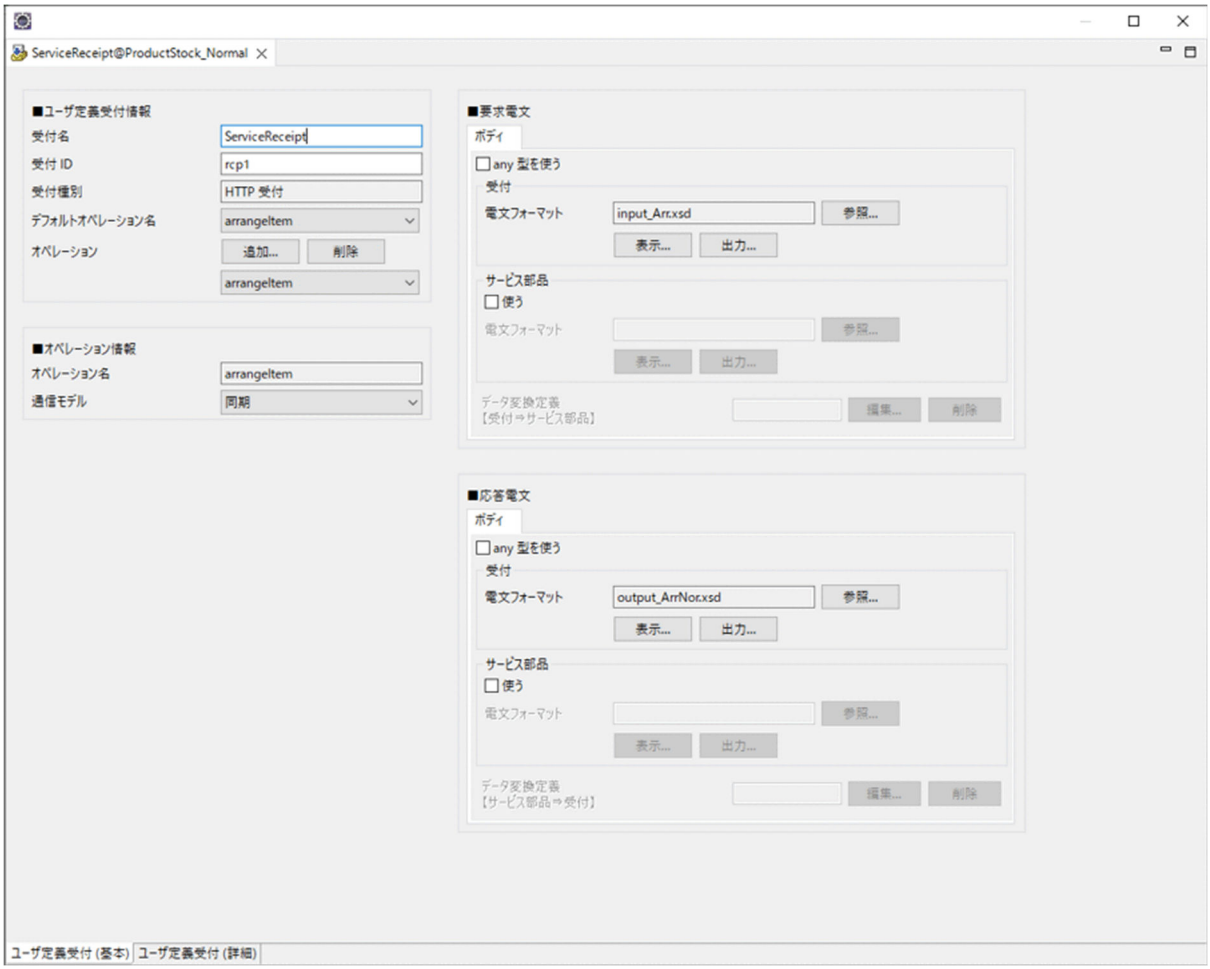
```
<Service Architectのインストールディレクトリ>¥CSCTE¥Samples¥SOAP1.1_1.2mode¥ProductStock_RES
T¥Schema¥ServiceReceipt¥response¥ProductStock_Normal
```

(c) 電文フォーマットの指定

作成した XML スキーマを ServiceReceipt 受付に設定します。

1. [ユーザ定義受付（基本)］タブでユーザ定義受付定義（基本）画面に切り替えます。
2. [要求電文］－［受付］－［電文フォーマット］の［参照］ボタンをクリックして，XML スキーマ「input_Arr.xsd」を指定します。
3. [応答電文］－［受付］－［電文フォーマット］の［参照］ボタンをクリックして，XML スキーマ「output_ArrNor.xsd」を指定します。
4. メニューから［ファイル］－［保管］を選択します。

設定後のユーザ定義受付（基本）タブの例を次に示します。



(4)　変数の設定

ビジネスプロセスでは，アクティビティを定義するときに変数を使用します。そのため，使用する変数をあらかじめ定義しておく必要があります。ProductStock_Normal ビジネスプロセスで使用する変数を次の表に示します。

表 5-5　ProductStock_Normal ビジネスプロセスで使用する変数

変数名	種別	説明
InputData	XML	受付アクティビティで受信するビジネスプロセスの要求電文のボディとして使用します。 「(3) 電文フォーマットの作成」で作成した要求電文フォーマットを取り込み，定義します。
OutputData	XML	応答アクティビティで送信するビジネスプロセスの応答電文のボディとして使用します。 「(3) 電文フォーマットの作成」で作成した応答電文フォーマットを取り込み，定義します。

変数名	種別	説明
InventoryAllocationInputData	XML	サービス呼出アクティビティで在庫管理サービスを呼び出す際の要求電文のボディとして使用します。 「5.2.2 InventoryManagement サービスアダプタの定義」で作成した要求電文フォーマットを取り込み，定義します。
InventoryAllocationOutputData	XML	サービス呼出アクティビティで在庫管理サービスを呼び出す際の応答電文のボディとして使用します。 「5.2.2 InventoryManagement サービスアダプタの定義」で作成した応答電文フォーマットを取り込み，定義します。
DeliveryArrangementInputData	XML	サービス呼出アクティビティで配送受付サービスを呼び出す際の要求電文のボディとして使用します。 「5.2.3 DeliveryReceipt サービスアダプタの定義」で作成した要求電文フォーマットを取り込み，定義します。
DeliveryArrangementOutputData	XML	サービス呼出アクティビティで配送受付サービスを呼び出す際の応答電文のボディとして使用します。 「5.2.3 DeliveryReceipt サービスアダプタの定義」で作成した応答電文フォーマットを取り込み，定義します。

ProductStock_Normal ビジネスプロセスで使用する変数の設定手順を次に示します。

1. ツリービューの［サービス定義一覧］から ProductStock_Normal ビジネスプロセスをダブルクリックします。
   ビジネスプロセス定義画面が表示されます。

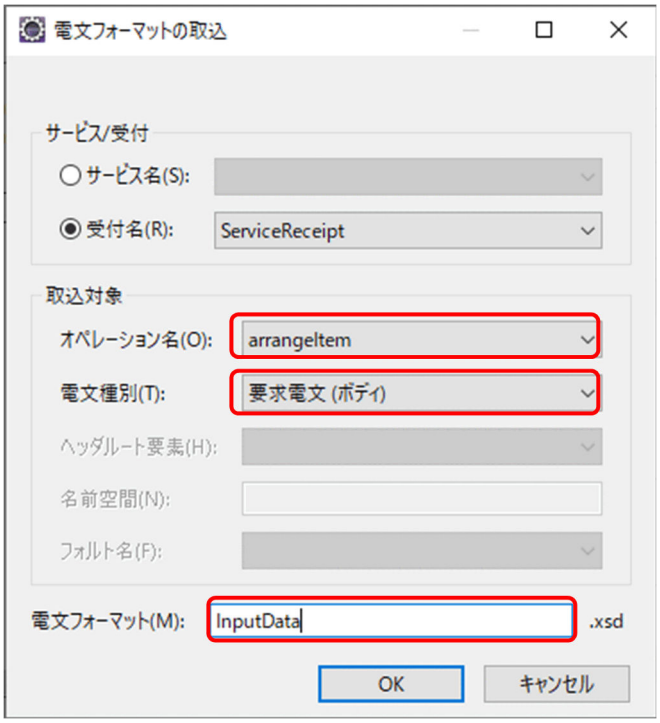
2. ビジネスプロセス定義画面のキャンバス上の［変数・相関セット］アイコンをダブルクリックします。
   ［変数・相関セット一覧］ダイアログが表示されます。

3. ［変数一覧］を選択します。変数名に「InputData」を入力し，種別は，ドロップダウンリストから「XML」を選択します。

4. ［取込］ボタンをクリックします。
   ［電文フォーマットの取込］ダイアログが表示されます。

5. ［受付名］を選択して，ドロップダウンリストから「ServiceReceipt」を選択します。

6. ［オペレーション名］はドロップダウンリストから「arrangeItem」を，［電文種別］はドロップダウンリストから「要求電文（ボディ）」を選択します。［電文フォーマット］に「InputData」を入力します。



7. [OK] ボタンをクリックします。

   [電文フォーマットの取込] ダイアログが閉じます。

8. [変数・相関セット一覧] ダイアログの [追加] ボタンをクリックします。

   変数一覧に変数「InputData」が追加されます。

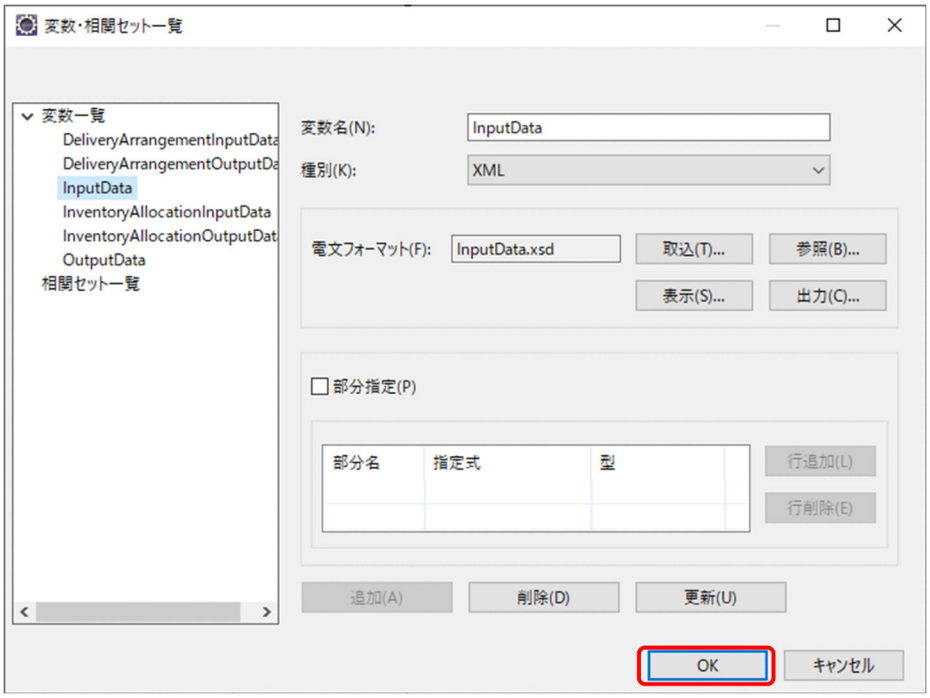
9. 「InputData」以外の変数も，手順 2.～7.と同様の手順で設定します。

   [変数・相関セット一覧] ダイアログで設定する種別はすべて [XML] です。[電文フォーマットの取込] ダイアログでの「InputData」以外の設定値は次のとおりです。

変数名	項目名				
	サービス/受付	サービス/受付の名称	オペレーション名	電文種別	電文フォーマット
OutputData	受付名	ServiceReceipt	arrangeItem	応答電文（ボディ）	OutputData
InventoryAllocationInputData	サービス名	InventoryManagement	reserveItem	要求電文（ボディ）	InventoryAllocationInputData
InventoryAllocationOutputData	サービス名	InventoryManagement	reserveItem	応答電文（ボディ）	InventoryAllocationOutputData
DeliveryArrangementInputData	サービス名	DeliveryReceipt	deliverItem	要求電文（ボディ）	DeliveryArrangementInputData
DeliveryArrangementOutputData	サービス名	DeliveryReceipt	deliverItem	応答電文（ボディ）	DeliveryArrangementOutputData

10. [変数・相関セット一覧] ダイアログの [OK] ボタンをクリックします。

    これで変数が設定できました。



(5)　アクティビティの配置

ProductStock_Normal ビジネスプロセスに必要なアクティビティを次の表に示します。

表 5-6　ProductStock_Normal ビジネスプロセスに必要なアクティビティ

アクティビティ名	アクティビティ種別	説明
受付	受付アクティビティ	サービスリクエスタからの応答を受け付けます。
在庫引当前処理	データ変換アクティビティ	在庫引当時のデータを編集します。
在庫引当	サービス呼出アクティビティ	在庫管理サービスを呼び出します。
在庫引当結果チェック	分岐開始アクティビティ	条件（在庫の有無）による処理をします。
在庫なし設定	データ変換アクティビティ	在庫なしレスポンスを返却できるようにデータを編集します。
応答_在庫なしエラー	応答アクティビティ	処理結果をサービスリクエスタへ返します。
配送手配前処理	データ変換アクティビティ	配送手配時のデータを編集します。
配送手配	サービス呼出アクティビティ	配送受付サービスを呼び出します。
配送番号設定	データ変換アクティビティ	在庫ありレスポンスを返却できるようにデータを編集します。
応答_手配成功	応答アクティビティ	処理結果をサービスリクエスタへ返します。
在庫引当結果チェック_終了	分岐終了アクティビティ	条件（在庫の有無）による処理を終了します。

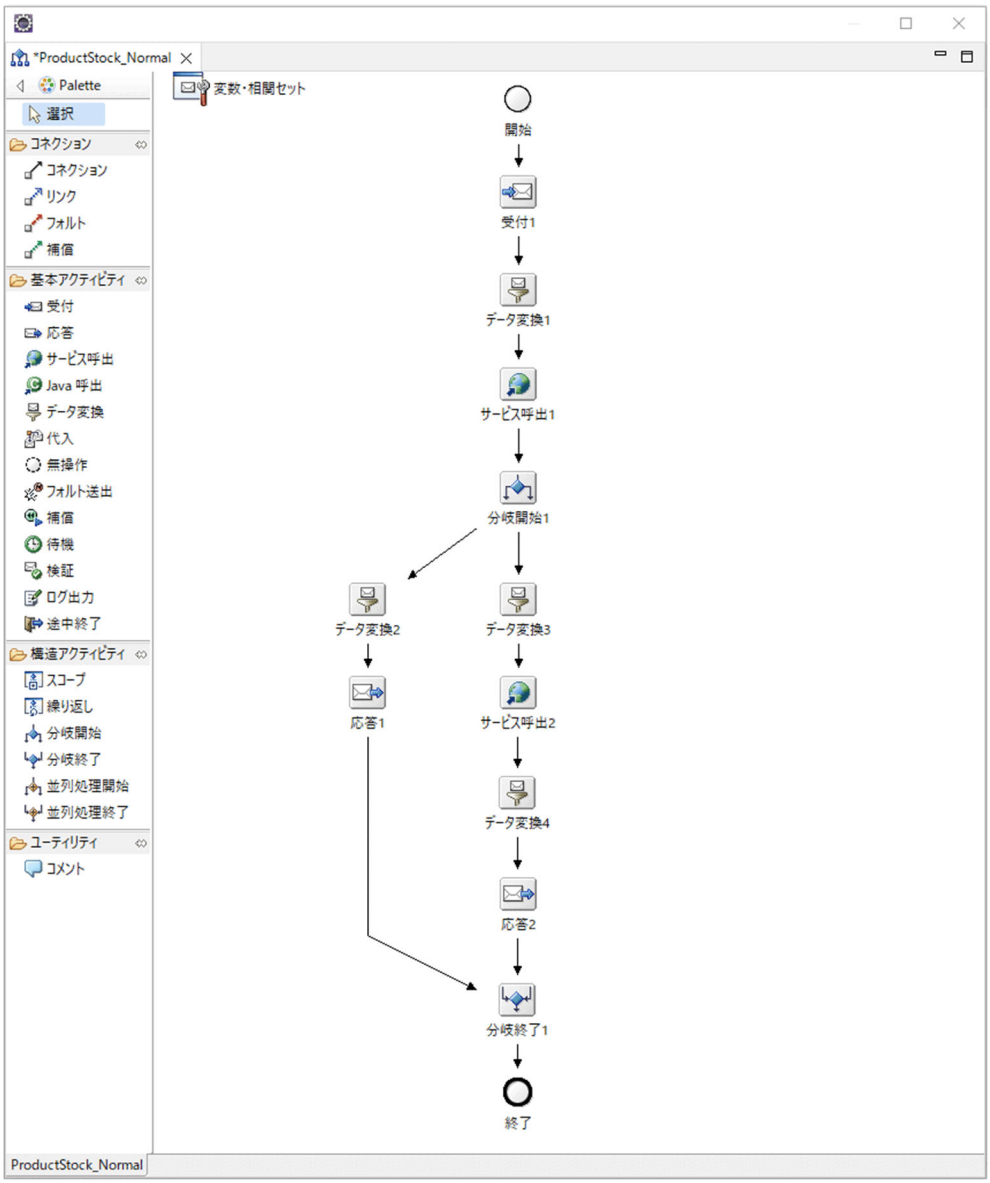
アクティビティの配置手順を次に示します。

1. パレットから［受付］をドラッグし，開始アクティビティの近くにドロップしてください。

開始アクティビティから受付アクティビティに連結されます。

2. 手順 1.と同様に，その他のアクティビティを配置・連結します。

次の図のフローになるよう，アクティビティは上記の表の記載順どおりに配置してください。順番にパレットから連結元のアクティビティの近くにドロップしてください。



ポイント

- 配置後に，分岐終了アクティビティを終了アクティビティの近くに移動することで，分岐終了アクティビティから終了アクティビティに連結できます。
- アクティビティ配置時に連結できなかった場合は，次の手順で連結してください。

1. アクティビティを連結するために，パレットの［コネクション］をクリックします。
2. 連結元である開始アクティビティをクリックします。
3. 連結先である受付アクティビティをクリックします。

4. 受付アクティビティから終了アクティビティまでを手順 1.～手順 3.と同様の手順で 1 つずつ連結します。

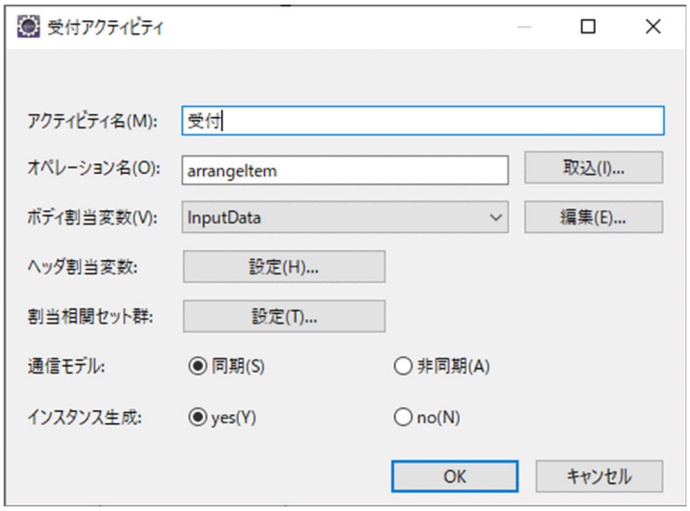
(6) アクティビティの定義

キャンバスへ配置した各アクティビティの内容を定義します。

(a) 受付アクティビティ（アクティビティ名：受付）

1. キャンバスの受付アクティビティ（受付 1）をダブルクリックします。
［受付アクティビティ］ダイアログが表示されます。

2. 次の内容を入力します。



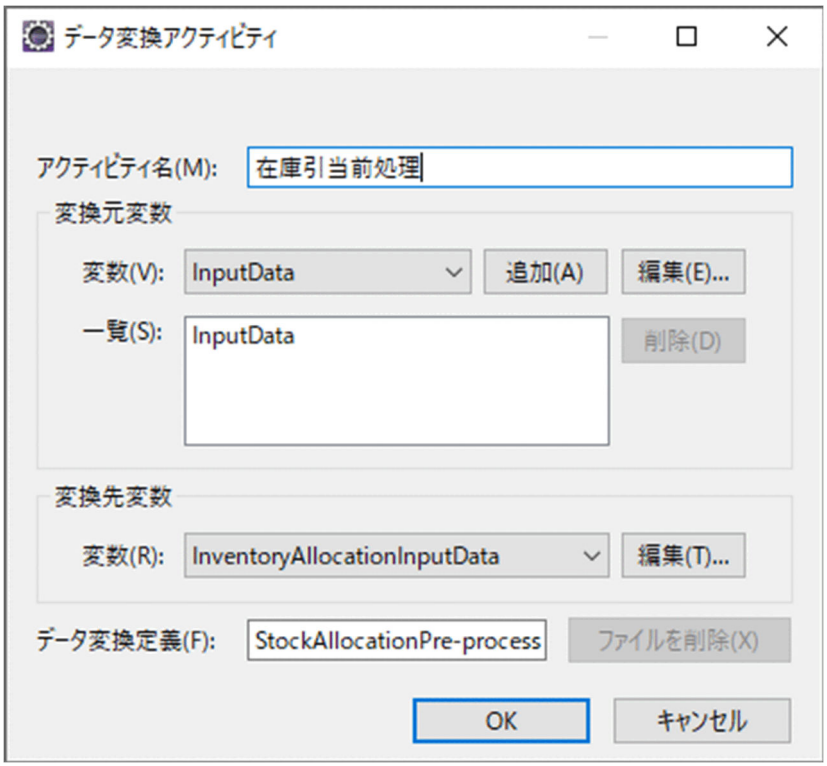
項目名	設定する値	説明
アクティビティ名	受付	アクティビティの名称を入力します。
オペレーション名	arrangeItem	サービスリクエスタから HCSC コンポーネントを呼び出すときに利用するオペレーション（ServiceReceipt 受付）の名称を入力します。
ボディ割当変数	InputData	ビジネスプロセスの要求電文のボディに割り当てる変数をドロップダウンリストから選択します。
ヘッダ割当変数	設定なし	ビジネスプロセスの要求電文のヘッダに変数を割り当てる場合に設定します。この商品手配システムでは使用しないため，設定しません。
割当相関セット群	設定なし	相関セットグループをアクティビティに割り当てる場合に設定します。この商品手配システムでは使用しないため，設定しません。
通信モデル	同期	オペレーションの通信モデルを指定します。ProductStock_Normal は，各サービス部品の呼び出し結果から応答電文フォーマットを作成するため，「同期」を設定します。
インスタンス生成	yes	要求電文を受け付けたときに，プロセスを初期化するかどうかを選択します。この商品手配システムでは，初期化するため「yes」を設定します。

3. [OK] ボタンをクリックします。

(b) データ変換アクティビティ（アクティビティ名：在庫引当前処理）

1. キャンバスのデータ変換アクティビティ（データ変換 1）をダブルクリックします。
   [データ変換アクティビティ] ダイアログが表示されます。

2. 次の内容を入力します。



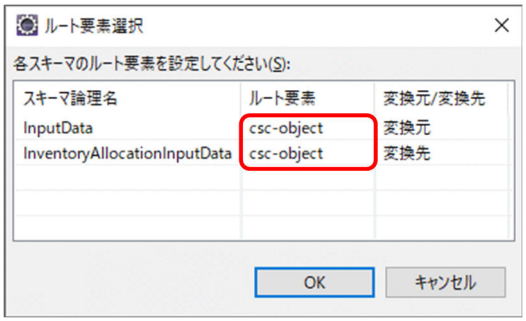
項目名	設定する値	説明
アクティビティ名	在庫引当前処理	アクティビティの名称を入力します。
変数（変換元変数）	InputData	データの変換元になる変数をドロップダウンリストから選択し，[追加] ボタンをクリックします。
変数（変換先変数）	InventoryAllocation InputData	データの変換先になる変数をドロップダウンリストから選択します。
データ変換定義	StockAllocationPre-processing	変数の変換に使うデータ変換定義ファイルの名称を入力します。

3. [OK] ボタンをクリックします。

4. キャンバスのデータ変換アクティビティを右クリックして，[マッピング定義起動] を選択します。
   [ルート要素選択] ダイアログが表示されます。

5. 変換元のスキーマ論理名「InputData」のルート要素をクリックして，ドロップダウンリストから「csc-object」を選択します。

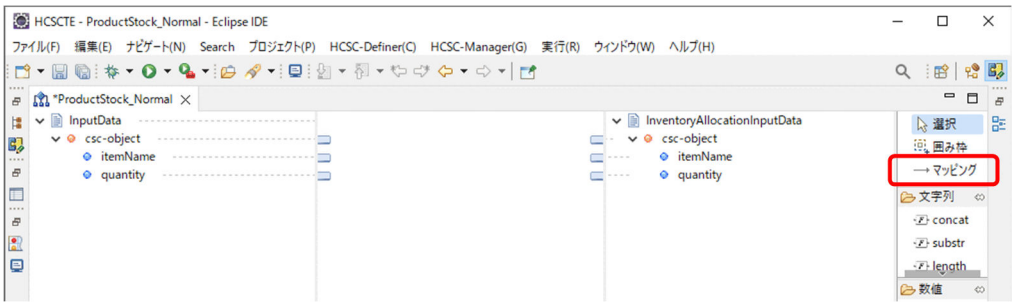
6. 変換先のスキーマ論理名「InventoryAllocationInputData」のルート要素をクリックして，ドロップダウンリストから「csc-object」を選択します。



7. [OK] ボタンをクリックします。

   データ変換定義画面が表示されます。

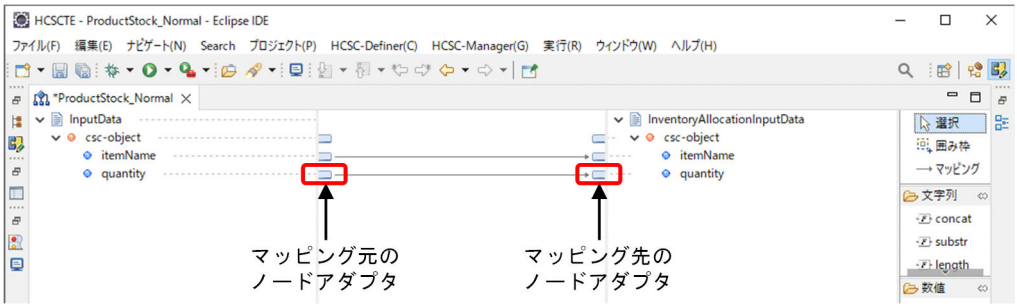
8. データ変換定義画面のパレットから［マッピング］を選択します。



9. マッピング元となる変換元ノードのノードアダプタをクリックします。

10. マッピング先となる変換先ノードのノードアダプタをクリックします。

    マッピング線が設定されます。マッピング元のノードアダプタとマッピング先のノードアダプタとの対応は次のとおりです。



注

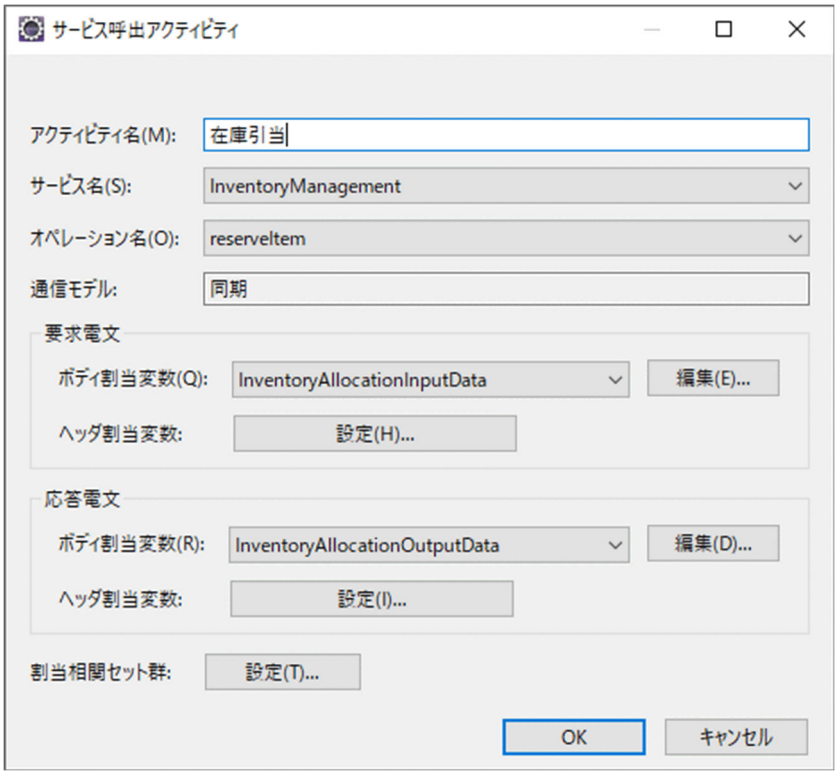
作成した XML スキーマによって変換先スキーマの要素の出現順が一部異なるため，マッピング線の見え方が異なる場合があります。

(c) サービス呼出アクティビティ（アクティビティ名：在庫引当）

1. キャンバスのサービス呼出アクティビティ（サービス呼出 1）をダブルクリックします。

   ［サービス呼出アクティビティ］ダイアログが表示されます。

2. 次の内容を入力します。



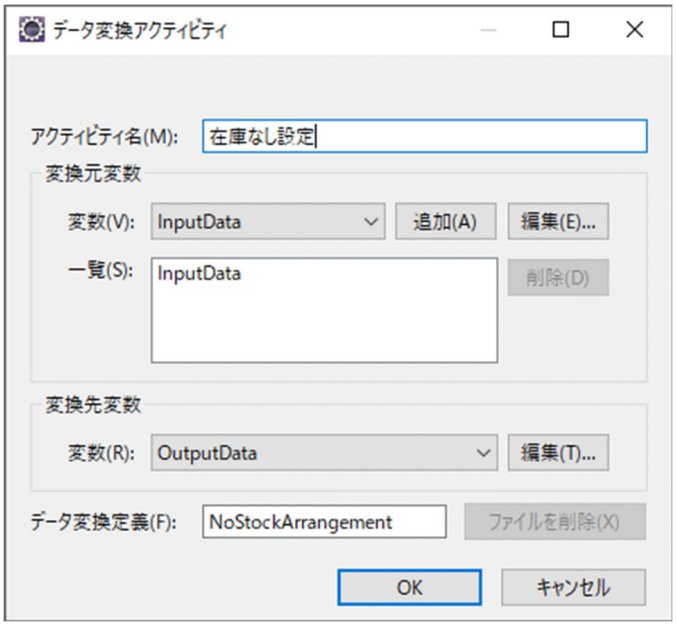
項目名	設定する値	説明
アクティビティ名	在庫引当	アクティビティの名称を入力します。
サービス名	InventoryManagement	サービス部品呼び出し時に使用するサービスアダプタの名称をドロップダウンリストから選択します。
オペレーション名	reserveItem	[サービス名] で指定したサービスアダプタのオペレーションのうち，実際に呼び出すオペレーションの名称を指定します。
通信モデル	同期	[オペレーション名] で指定したオペレーションに設定されている通信モデルが表示されます。
ボディ割当変数（要求電文）	InventoryAllocationInputData	在庫管理サービスを呼び出す要求電文のボディに割り当てる変数をドロップダウンリストから選択します。
ヘッダ割当変数（要求電文）	設定なし	在庫管理サービスを呼び出す要求電文のヘッダに変数を割り当てる場合に設定します。この商品手配システムでは使用しないため，設定しません。
ボディ割当変数（応答電文）	InventoryAllocationOutputData	在庫管理サービスから受け取る応答電文のボディに割り当てる変数をドロップダウンリストから選択します。
ヘッダ割当変数（応答電文）	設定なし	在庫管理サービスから受け取る応答電文のヘッダに変数を割り当てる場合に設定します。この商品手配システムでは使用しないため，設定しません。
割当相関セット群	設定なし	相関セットグループをアクティビティに割り当てる場合に設定します。この商品手配システムでは使用しないため，設定しません。

3. [OK] ボタンをクリックします。

(d)　データ変換アクティビティ（アクティビティ名：在庫なし設定）

1. キャンバスのデータ変換アクティビティ（データ変換 2）をダブルクリックします。
   ［データ変換アクティビティ］ダイアログが表示されます。

2. 次の内容を入力します。



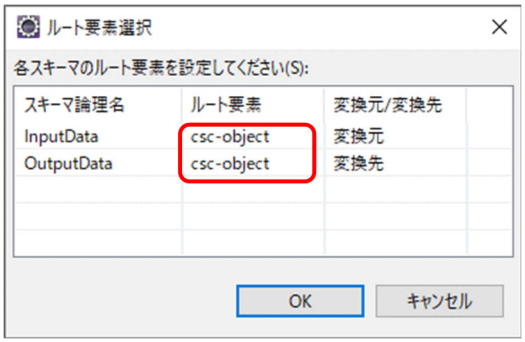
項目名	設定する値	説明
アクティビティ名	在庫なし設定	アクティビティの名称を入力します。
変数（変換元変数）	InputData	データの変換元になる変数をドロップダウンリストから選択し，［追加］ボタンをクリックします。
変数（変換先変数）	OutputData	データの変換先になる変数をドロップダウンリストから選択します。
データ変換定義	NoStockArrangement	変数の変換に使うデータ変換定義ファイルの名称を入力します。

3. ［OK］ボタンをクリックします。

4. キャンバスのデータ変換アクティビティを右クリックして，［マッピング定義起動］を選択します。
   ［ルート要素選択］ダイアログが表示されます。

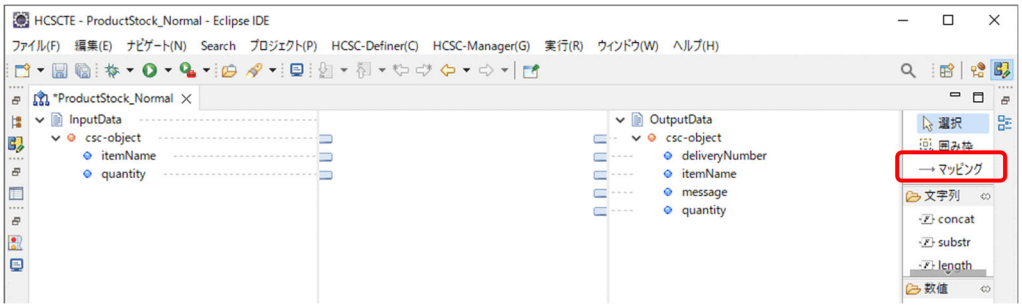
5. 変換元のスキーマ論理名「InputData」のルート要素をクリックして，ドロップダウンリストから「csc-object」を選択します。

6. 変換先のスキーマ論理名「OutputData」のルート要素をクリックして，ドロップダウンリストから「csc-object」を選択します。



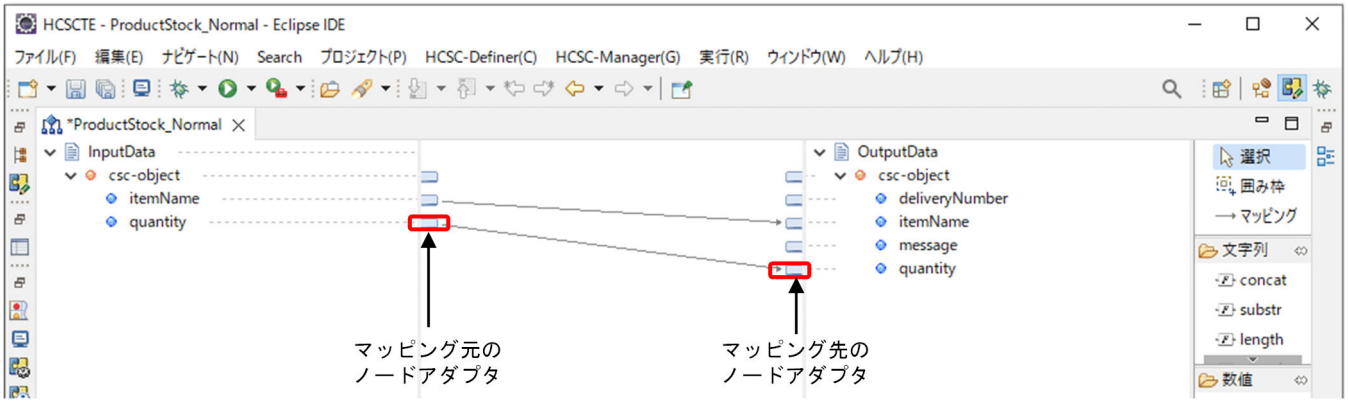
7. [OK] ボタンをクリックします。
   データ変換定義画面が表示されます。

8. データ変換定義画面のパレットから［マッピング］を選択します。



9. マッピング元となる変換元ノードのノードアダプタをクリックします。

10. マッピング先となる変換先ノードのノードアダプタをクリックします。
    マッピング線が設定されます。マッピング元のノードアダプタとマッピング先のノードアダプタとの対応は次のとおりです。



注

作成した XML スキーマによって変換先スキーマの要素の出現順が一部異なるため，マッピング線の見え方が異なる場合があります。

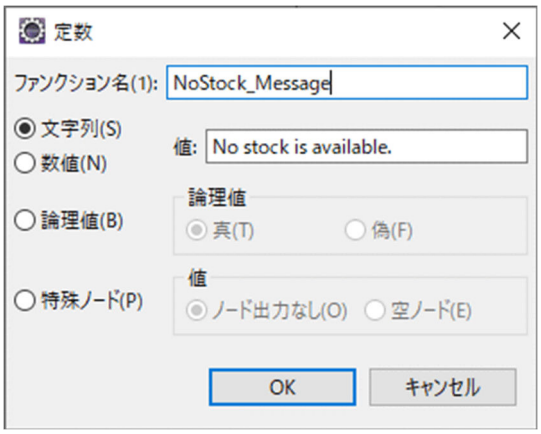
11. データ変換定義画面のパレット［その他］から定数ファンクション（const）を選択します。

12. マッピングビューアで適当な場所をクリックして定数ファンクションを配置します。

13. 定数ファンクションをダブルクリックします。
    ［定数］ダイアログが表示されます。

14. ［定数］ダイアログで次の値を指定します。

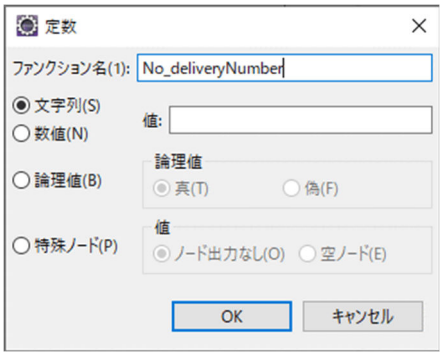
属性名	属性値
ファンクション名	NoStock_Message
文字列	No stock is available.



15. ［OK］ボタンをクリックして［定数］ダイアログを閉じます。

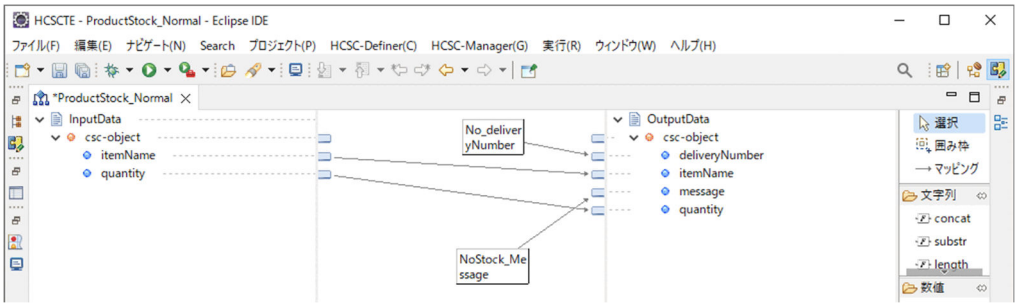
16. 手順 11.～手順 15.と同様の手順で，もう一つ定数ファンクション（const）を作成します。［定数］ダイアログでは次の値を指定します。

属性名	属性値
ファンクション名	No_deliveryNumber
文字列	（何も入力しない）



17. ［OK］ボタンをクリックします。

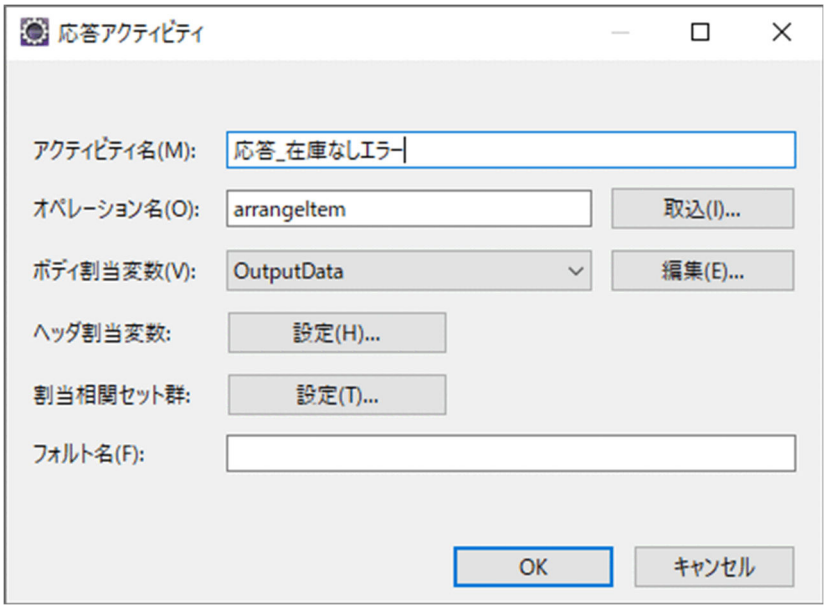
18. 次の図のようになるようにマッピング線を設定します。



(e)　応答アクティビティ（アクティビティ名：応答_在庫なしエラー）

1. キャンバスの応答アクティビティ（応答 1）をダブルクリックします。
   ［応答アクティビティ］ダイアログが表示されます。

2. 次の内容を入力します。



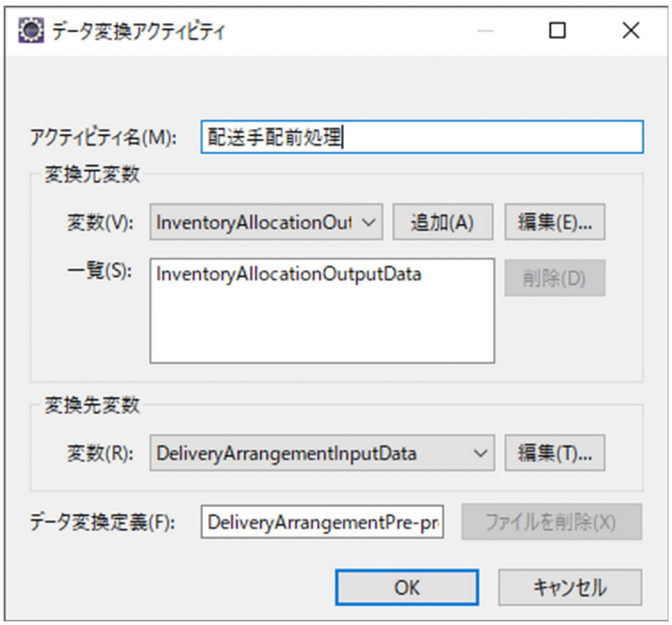
項目名	設定する値	説明
アクティビティ名	応答_在庫なしエラー	アクティビティの名称を入力します。
オペレーション名	arrangeItem	対応する受付アクティビティに指定したオペレーション名を入力します。
ボディ割当変数	OutputData	ビジネスプロセスの応答電文のボディに割り当てる変数をドロップダウンリストから選択します。
ヘッダ割当変数	設定なし	ビジネスプロセスの応答電文のヘッダに変数を割り当てる場合に設定します。この商品手配システムでは使用しないため，設定しません。
割当相関セット群	設定なし	相関セットグループをアクティビティに割り当てる場合に入力します。この商品手配システムでは使用しないため，設定しません。
フォルト名	設定なし	フォルト処理として応答アクティビティを定義して，サービスリクエスタにフォルトが発生したことを示す応答電文を送信する場合のフォルトの名称を入力します。この商品手配システムではフォルト処理を使用しないため，設定しません。

3. [OK] ボタンをクリックします。

(f) データ変換アクティビティ（アクティビティ名：配送手配前処理）

1. キャンバスのデータ変換アクティビティ（データ変換 3）をダブルクリックします。
   [データ変換アクティビティ] ダイアログが表示されます。

2. 次の内容を入力します。



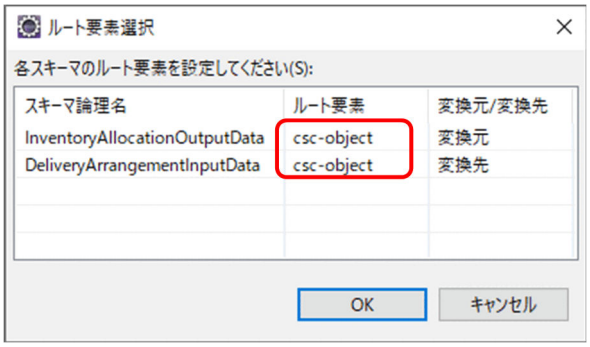
項目名	設定する値	説明
アクティビティ名	配送手配前処理	アクティビティの名称を入力します。
変数（変換元変数）	InventoryAllocationOutputData	データの変換元になる変数をドロップダウンリストから選択し，[追加] ボタンをクリックします。
変数（変換先変数）	DeliveryArrangementInputData	データの変換先になる変数をドロップダウンリストから選択します。
データ変換定義	DeliveryArrangementPre-processing	変数の変換に使うデータ変換定義ファイルの名称を入力します。

3. [OK] ボタンをクリックします。

4. キャンバスのデータ変換アクティビティを右クリックして，[マッピング定義起動] を選択します。
   [ルート要素選択] ダイアログが表示されます。

5. 変換元のスキーマ論理名「InventoryAllocationOutputData」のルート要素をクリックして，ドロップダウンリストから「csc-object」を選択します。

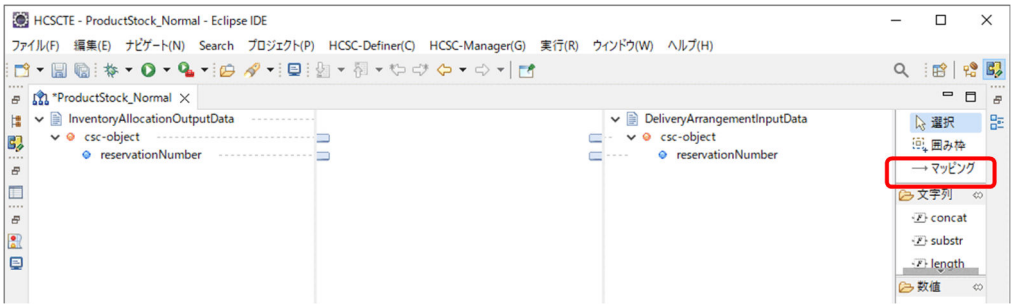
6. 変換先のスキーマ論理名「DeliveryArrangementInputData」のルート要素をクリックして，ドロップダウンリストから「csc-object」を選択します。



7. [OK] ボタンをクリックします。

   データ変換定義画面が表示されます。

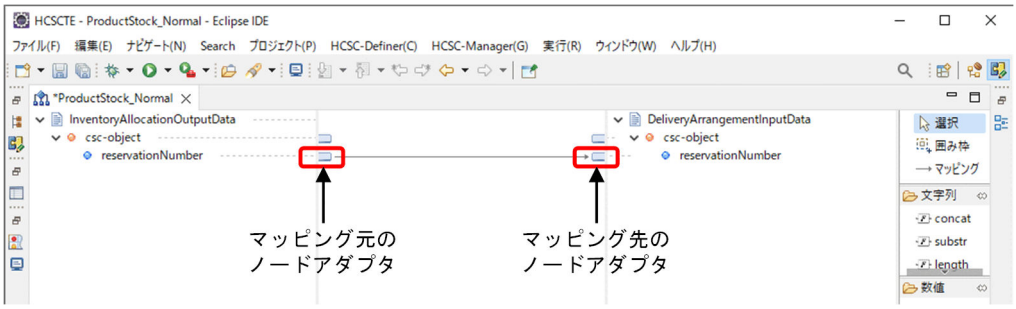
8. データ変換定義画面のパレットから［マッピング］を選択します。



9. マッピング元となる変換元ノードのノードアダプタをクリックします。

10. マッピング先となる変換先ノードのノードアダプタをクリックします。

    マッピング線が設定されます。マッピング元のノードアダプタとマッピング先のノードアダプタとの対応は次のとおりです。



注

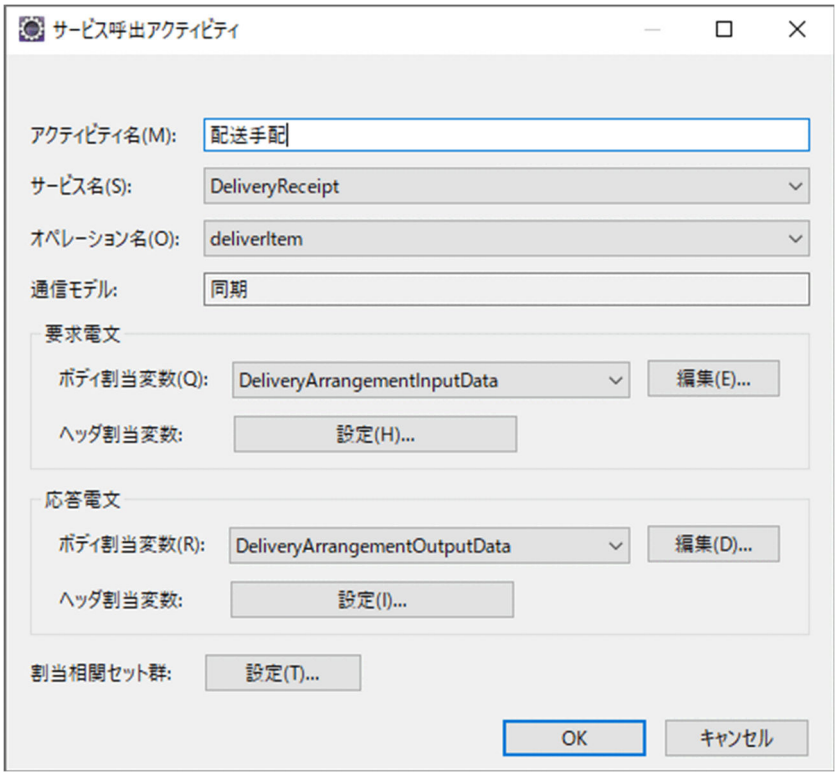
作成した XML スキーマによって変換先スキーマの要素の出現順が一部異なるため，マッピング線の見え方が異なる場合があります。

(g) サービス呼出アクティビティ（アクティビティ名：配送手配）

1. キャンバスのサービス呼出アクティビティ（サービス呼出 2）をダブルクリックします。

   ［サービス呼出アクティビティ］ダイアログが表示されます。

2. 次の内容を入力します。



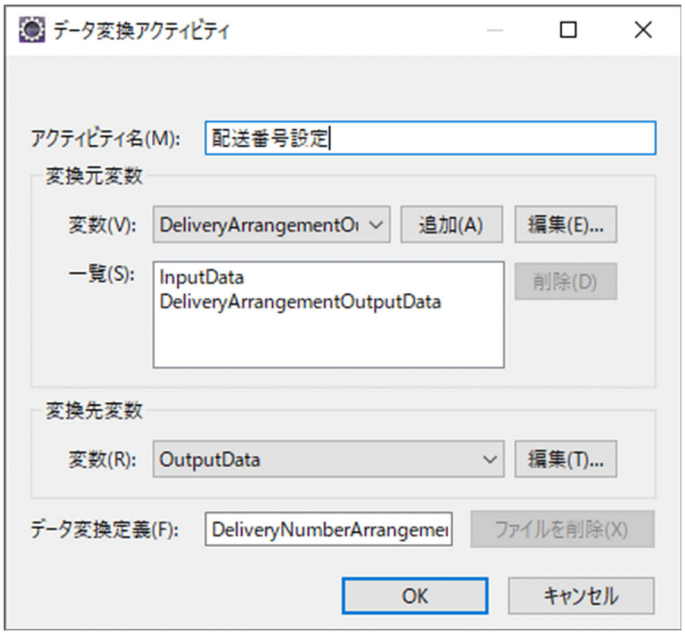
項目名	設定する値	説明
アクティビティ名	配送手配	アクティビティの名称を入力します。
サービス名	DeliveryReceipt	サービス部品呼び出し時に使用するサービスアダプタの名称をドロップダウンリストから選択します。
オペレーション名	deliverItem	[サービス名] で指定したサービスアダプタのオペレーションのうち，実際に呼び出すオペレーションの名称を指定します。
通信モデル	同期	[オペレーション名] で指定したオペレーションに設定されている通信モデルが表示されます。
ボディ割当変数（要求電文）	DeliveryArrangementInputData	配送手配サービスを呼び出す要求電文のボディに割り当てる変数をドロップダウンリストから選択します。
ヘッダ割当変数（要求電文）	設定なし	配送手配サービスを呼び出す要求電文のヘッダに変数を割り当てる場合に設定します。この商品手配システムでは使用しないため，設定しません。
ボディ割当変数（応答電文）	DeliveryArrangementOutputData	配送手配サービスから受け取る応答電文のボディに割り当てる変数をドロップダウンリストから選択します。
ヘッダ割当変数（応答電文）	設定なし	配送手配サービスから受け取る応答電文のヘッダに変数を割り当てる場合に設定します。この商品手配システムでは使用しないため，設定しません。
割当相関セット群	設定なし	相関セットグループをアクティビティに割り当てる場合に設定します。この商品手配システムでは使用しないため，設定しません。

3. [OK] ボタンをクリックします。

(h)　データ変換アクティビティ（アクティビティ名：配送番号設定）

1. キャンバスのデータ変換アクティビティ（データ変換 4）をダブルクリックします。
［データ変換アクティビティ］ダイアログが表示されます。

2. 次の内容を入力します。



項目名	設定する値	説明
アクティビティ名	配送番号設定	アクティビティの名称を入力します。
変数（変換元変数）※	InputData DeliveryArrangementOutputData	データの変換元になる変数をドロップダウンリストから選択し，［追加］ボタンをクリックします。
変数（変換先変数）	OutputData	データの変換先になる変数をドロップダウンリストから選択します。
データ変換定義	DeliveryNumberArrangement	変数の変換に使うデータ変換定義ファイルの名称を入力します。

注※　変換元変数は 2 つ設定が必要です。

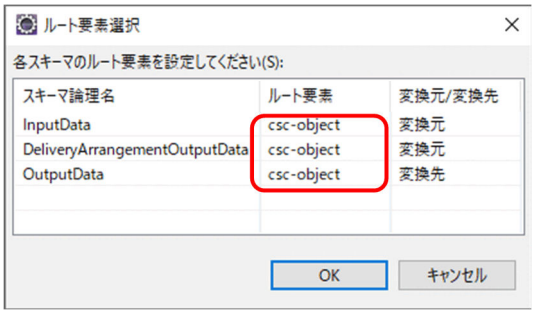
3. ［OK］ボタンをクリックします。

4. キャンバスのデータ変換アクティビティを右クリックして，［マッピング定義起動］を選択します。
［ルート要素選択］ダイアログが表示されます。

5. 変換元のスキーマ論理名「InputData」のルート要素をクリックして，ドロップダウンリストから「csc-object」を選択します。

6. 変換元のスキーマ論理名「DeliveryArrangementOutputData」のルート要素をクリックして，ドロップダウンリストから「csc-object」を選択します。

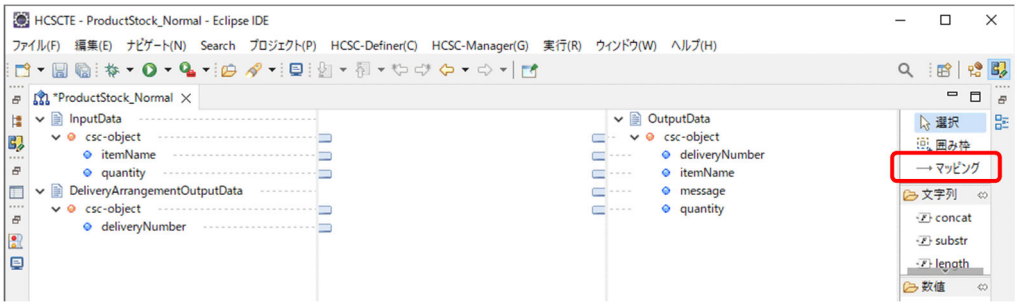
7. 変換先のスキーマ論理名「OutputData」のルート要素をクリックして，ドロップダウンリストから「csc-object」を選択します。



8. [OK] ボタンをクリックします。

   データ変換定義画面が表示されます。

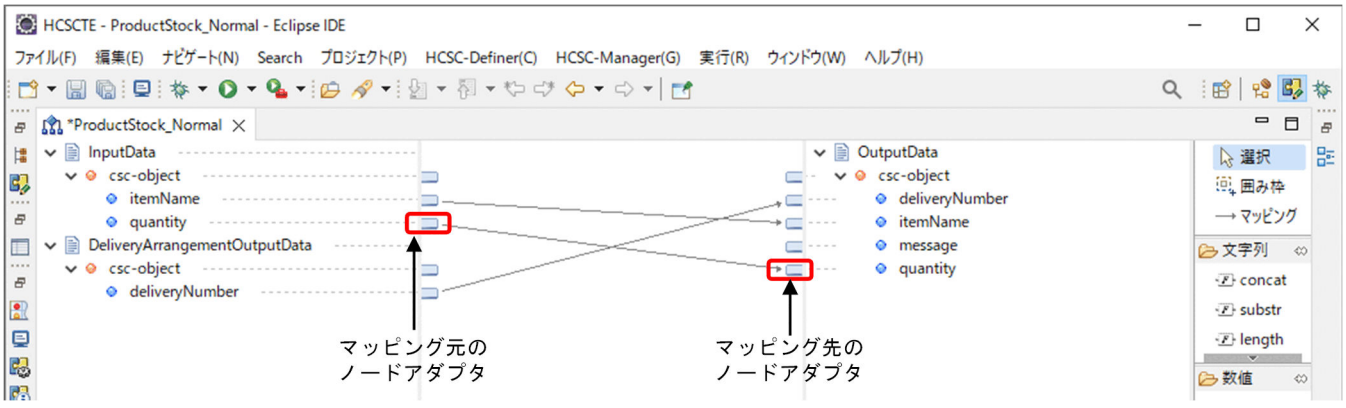
9. データ変換定義画面のパレットから［マッピング］を選択します。



10. マッピング元となる変換元ノードのノードアダプタをクリックします。

11. マッピング先となる変換先ノードのノードアダプタをクリックします。

    マッピング線が設定されます。マッピング元のノードアダプタとマッピング先のノードアダプタとの対応は次のとおりです。



注

作成した XML スキーマによって変換先スキーマの要素の出現順が一部異なるため，マッピング線の見え方が異なる場合があります。

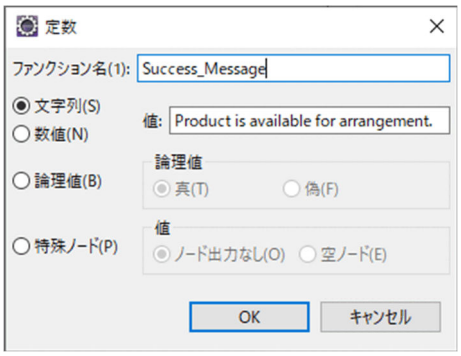
12. データ変換定義画面のパレット［その他］から定数ファンクション（const）を選択します。

13. マッピングビューアで適当な場所をクリックして定数ファンクションを配置します。

14. 定数ファンクションをダブルクリックします。
    ［定数］ダイアログが表示されます。

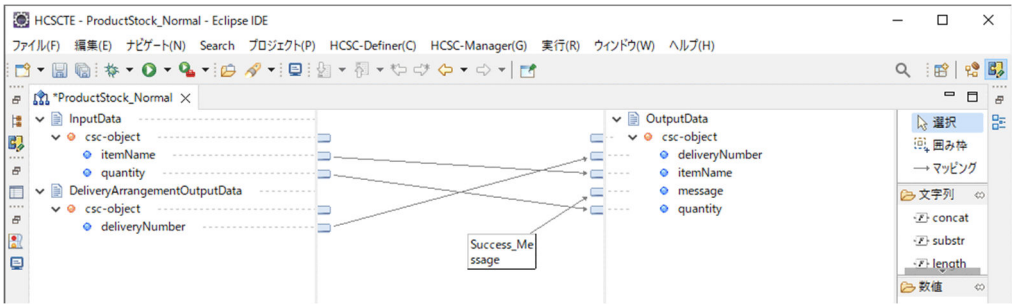
15. ［定数］ダイアログで次の値を指定します。

属性名	属性値
ファンクション名	Success_Message
文字列	Product is available for arrangement.



16. ［OK］ボタンをクリックします。

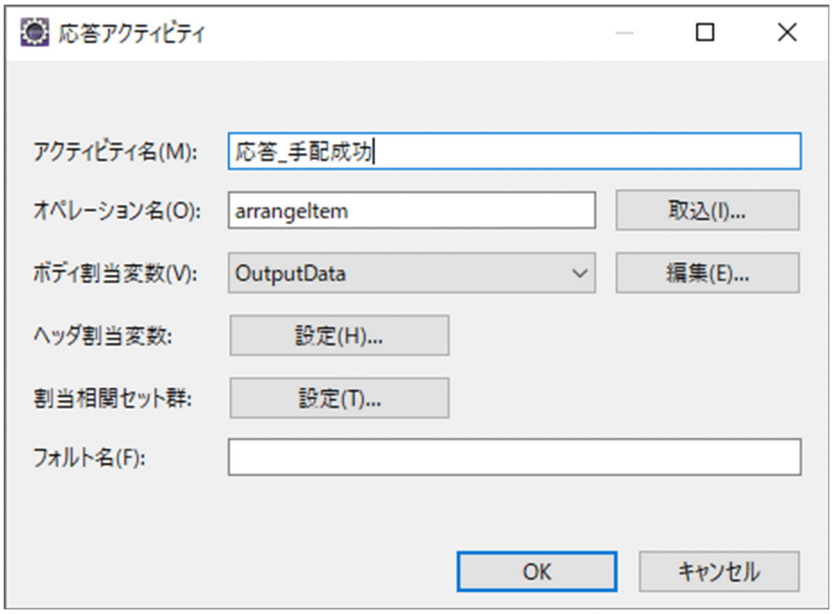
17. 次の図のようになるようにマッピング線を設定します。



(i)　応答アクティビティ（アクティビティ名：応答_手配成功）

1. キャンバスの応答アクティビティ（応答 2）をダブルクリックします。
   ［応答アクティビティ］ダイアログが表示されます。

2. 次の内容を入力します。



項目名	設定する値	説明
アクティビティ名	応答_手配成功	アクティビティの名称を入力します。
オペレーション名	arrangeItem	対応する受付アクティビティに指定したオペレーション名を入力します。
ボディ割当変数	OutputData	ビジネスプロセスの応答電文のボディに割り当てる変数をドロップダウンリストから選択します。
ヘッダ割当変数	設定なし	ビジネスプロセスの応答電文のヘッダに変数を割り当てる場合に設定します。この商品手配システムでは使用しないため，設定しません。
割当相関セット群	設定なし	相関セットグループをアクティビティに割り当てる場合に入力します。この商品手配システムでは使用しないため，設定しません。
フォルト名	設定なし	フォルト処理として応答アクティビティを定義して，サービスリクエスタにフォルトが発生したことを示す応答電文を送信する場合のフォルトの名称を入力します。この商品手配システムではフォルト処理を使用しないため，設定しません。

3. [OK] ボタンをクリックします。

(j) 分岐開始アクティビティ（アクティビティ名：在庫引当結果チェック）

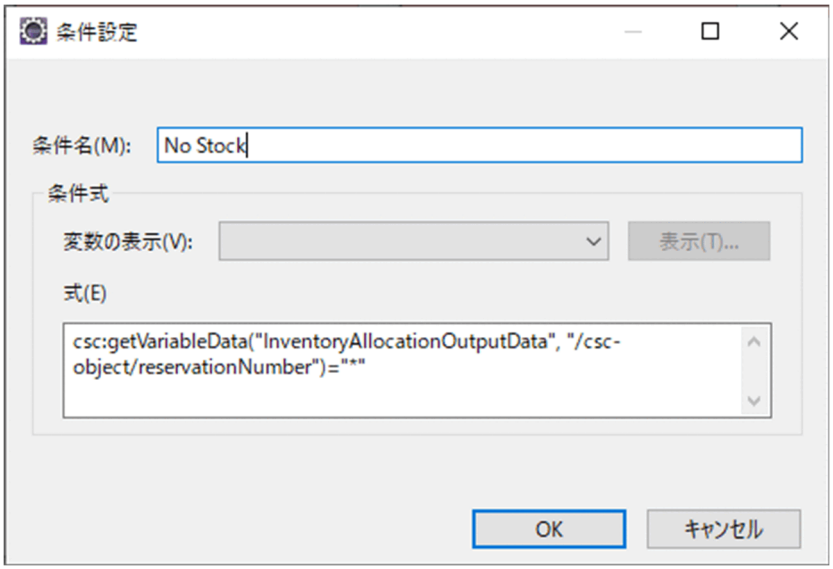
1. キャンバスの分岐開始アクティビティ（分岐開始 1）をダブルクリックします。
   ［分岐アクティビティ］ダイアログが表示されます。

2. アクティビティ名に「在庫引当結果チェック」を入力します。

3. ［遷移先］が［在庫なし設定］の行をクリックし，［上へ］ボタンをクリックして，いちばん上の行に移動します。

4. いちばん上の行を選択し，［条件設定］ボタンをクリックします。
   ［条件設定］ダイアログが表示されます。

5. 次の内容を入力します。



項目名	設定する値	説明
条件名	No Stock	在庫がない場合の条件の名称を指定します。
変数の表示	なし	このドロップダウンリストは，条件に変数を利用した式を設定する場合に一時的に使用するものです。ここでは何も設定しません。
式※	csc:getVariableData("InventoryAllocationOutputData", "/csc-object/reservationNumber")="*"	在庫がない場合の条件式を XPath 式で指定します。

注※

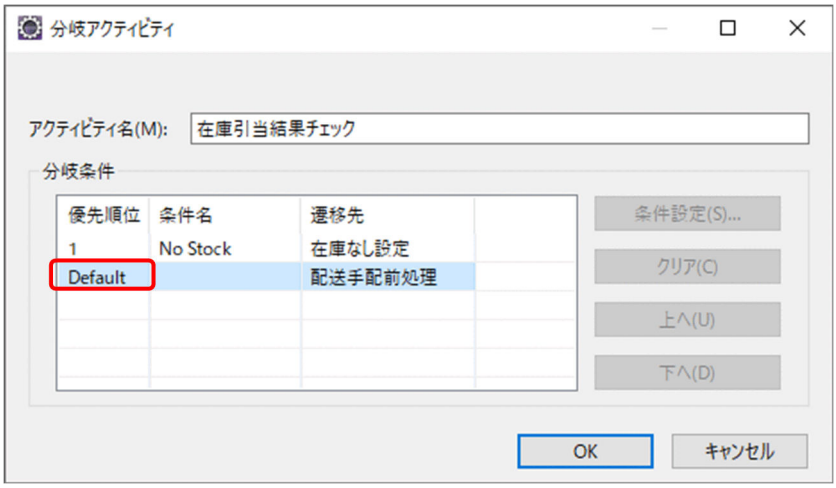
「式」の「設定する値」はテキストボックスの幅で折り返して表示されます。

6. [OK] ボタンをクリックします。

［分岐アクティビティ］ダイアログに戻ります。

7. [遷移先] が［配送手配前処理］の［優先順位］をクリックし，ドロップダウンリストから「Default」を選択します。

「Default」を選択すると，条件の設定は不要になります。

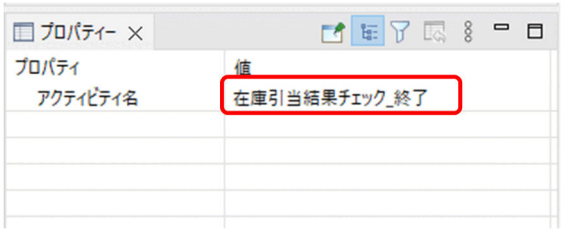


8. [OK] ボタンをクリックします。

(k) 分岐終了アクティビティ（アクティビティ名：在庫引当結果チェック_終了）

1. キャンバスの分岐終了アクティビティ（分岐終了 1）をクリックします。
   プロパティビューに，分岐終了アクティビティの内容が表示されます。

2. アクティビティ名に，「在庫引当結果チェック_終了」を入力します。



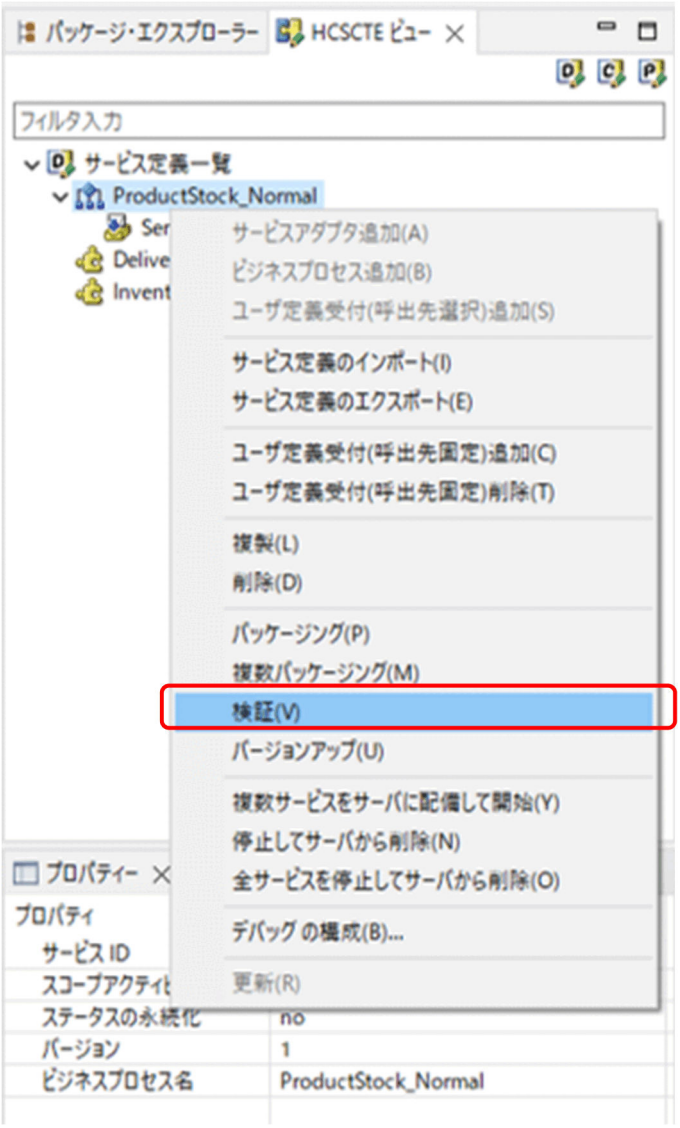
3. すべてのアクティビティを定義したら，メニューから［ファイル］－［保管］を選択して，ビジネスプロセスの定義を終了します。

5.2.5 コンポーネントの検証とパッケージング

作成したコンポーネントは定義した内容が正しいかどうか検証してからパッケージングします。

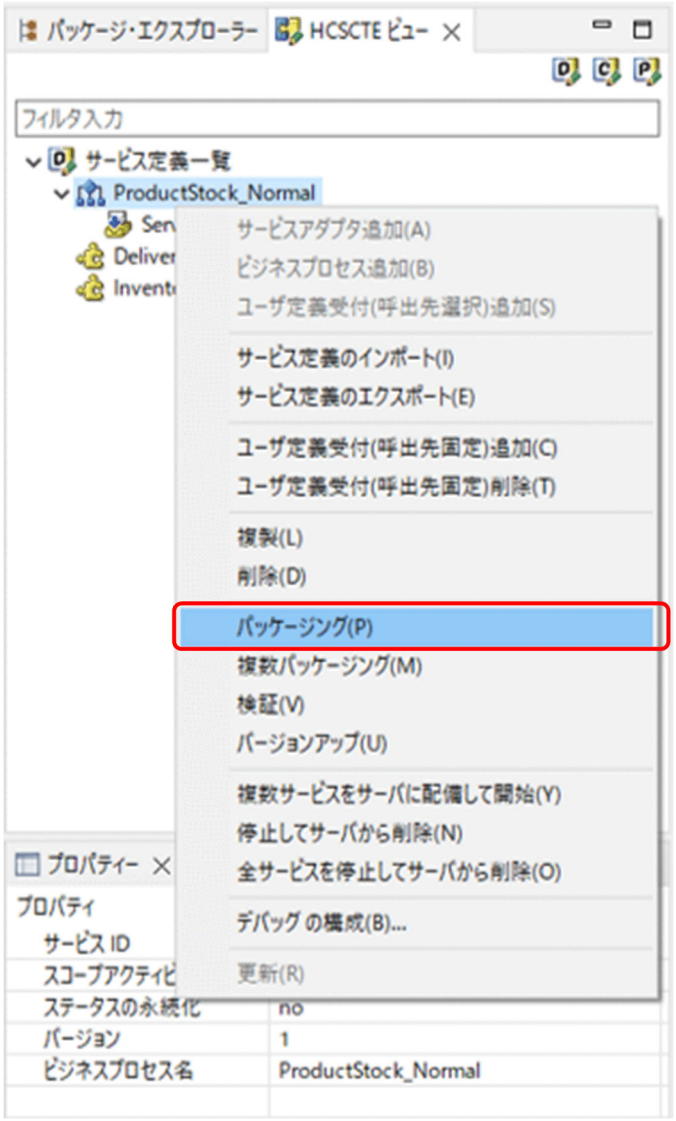
ProductStock_Normal ビジネスプロセスの検証とパッケージングの手順を次に示します。

1. ツリービューの［ProductStock_Normal］を選択し，右クリックして，［検証］を選択します。



検証結果がコンソールビューに表示されます。エラーが発生した場合は，メッセージに従って修正します。

2. ツリービューの［ProductStock_Normal］を選択し，右クリックして，［パッケージング］を選択します。



パッケージングの処理が開始されます。処理終了後，結果を知らせるメッセージが表示されます。

3. 次のどちらかの操作を実行します。

   パッケージングが成功した場合は，［OK］ボタンをクリックします。

   パッケージングが失敗した場合は，メッセージに従って対処し，パッケージングを再実行します。

4. ツリービューの［DeliveryReceipt］サービスアダプタ，および［InventoryManagement］サービスアダプタに対して，1～3 の手順に従って，検証・パッケージングしてください。

5.2.6　HCSC コンポーネントの配備と開始

HCSC コンポーネントをテスト環境に配備して，開始します。

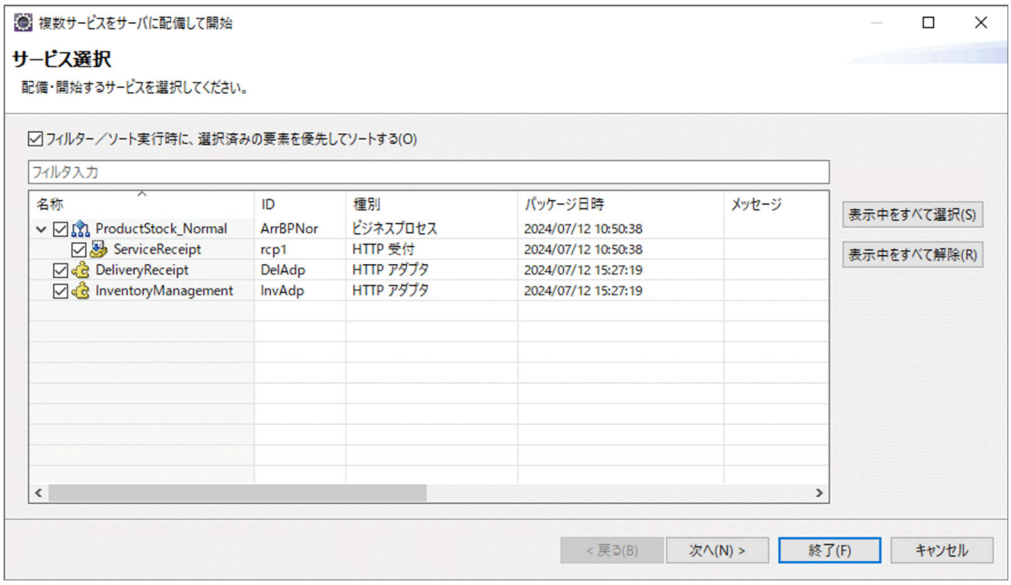
1. ツリービューのサービス定義一覧で，サービス定義一覧または HCSC コンポーネントを選択します。

2. 選択したサービス定義一覧または HCSC コンポーネントを右クリックして，［複数サービスをサーバに配備して開始］を選択します。

複数サービスをサーバに配備して開始ウィザードが表示されます。

3. サービス選択ページで配備および起動するサービスを選択します。

ここでは，すべてのサービスを選択してください。



4. [終了] ボタンをクリックします。
   処理中であることを知らせるメッセージが表示されたあと，結果を知らせるメッセージが表示されます。
   ログインしていない場合は，アカウント認証画面が表示されます。手順 5 を実施してください。

5. [ユーザ ID] に「admin」を，[パスワード] に「admin」を入力し，[OK] ボタンをクリックします。

6. [OK] ボタンをクリックします。
   これで配備と開始が完了しました。

5.2.7 HCSC コンポーネントの配備内容の確認

HCSC サーバに，HCSC コンポーネントが配備されていることと，起動していることを確認します。

1. メニューから［ウィンドウ］－［ビューの表示］－［その他］を選択します。
   ［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] ダイアログで［HCSC-Manager］－［HCSC-Manager ビュー］を選択し，［開く］ボタンをクリックします。
   ［HCSC-Manager］ビューが表示されます。

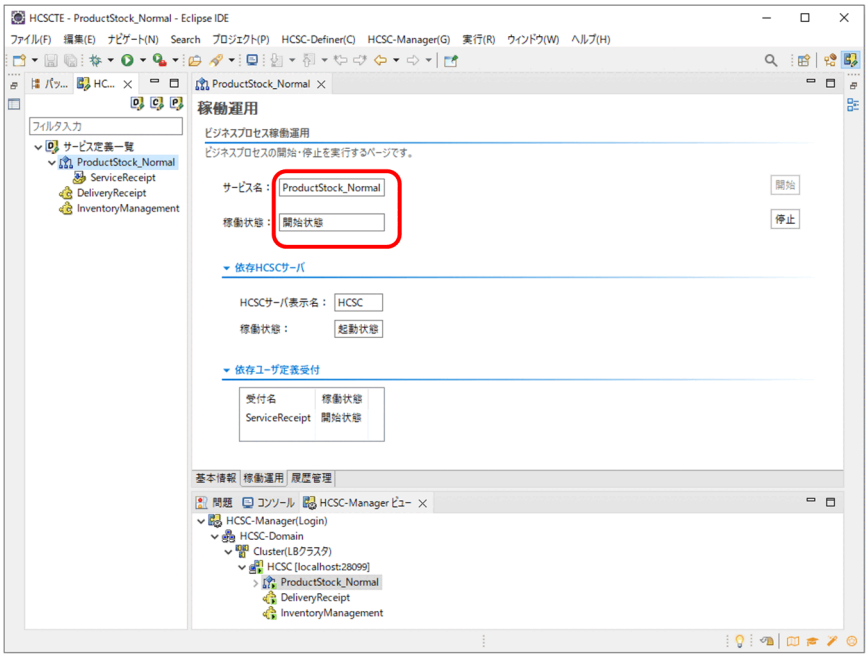
3. [HCSC-Manager] ビューの［HCSC-Manager(Logout)］を右クリックし，［ログイン］を選択します。
   ログイン画面が表示されます。

4. [管理ユーザー ID] に「admin」を，[パスワード] に「admin」を入力して，[OK] ボタンをクリックします。
   HCSC-Manager へのログインが完了します。

5. HCSC コンポーネントが起動していることを確認します。

［HCSC-Manager］ビューの［HCSC-Manager(Login)］－［HCSC-Domain］－［Cluster(LB クラスタ)］－［HCSC[localhost:28099]］－［ProductStock_Normal］をダブルクリックして［基本情報］ページを表示させます。［稼働運用］タブをクリックして［稼働運用］ページを開きます。［稼働状態］が［開始状態］になっているかどうかを確認します。

上記と同じ手順で［DeliveryReceipt］アダプタと［InventoryManagement］アダプタの稼働状態を確認します。



6. **［HCSC-Manager］ビューの［HCSC-Manager(Login)］を右クリックし，［ログアウト］を選択します。**

   HCSC-Manager からのログアウトを確認するメッセージが表示されます。

7. **［OK］ボタンをクリックします。**

   HCSC-Manager からのログアウトが完了し，［HCSC-Manager(Logout)］と表示されます。

これで，ビジネスプロセス・サービスアダプタの準備ができました。

5.3　商品手配システム（基本編）を実行する

商品手配システムのサンプルプログラムが提供するサービス部品を利用することで，開発した商品手配システムの動作を確認できます。

以降の項で，これらの実行方法について説明します。

注意事項

サンプルプログラムで提供するサービス部品を利用しない場合は，サービス部品を作成して確認することもできます。サービス部品の作成方法はマニュアル「アプリケーションサーバ アプリケーション開発ガイド」を参照してください。

5.3.1　開発した商品手配システム（基本編）を実行する準備

商品手配システムを実行する前に，次の作業が実施済みであることを確認してください。

1. サーバが起動していることを確認します。

   次の 2 つのサーバが起動していることを確認してください。

   HCSC サーバ

   起動方法は「3.4.2　テスト環境の起動」で説明しています。

   RESTful Web サービス用 Web システム

   起動方法は「4.2　サービス部品用 Web システムの構築」で説明しています。

2. RESTful Web サービスが開始していることを確認します。

   開始方法は「4.3　RESTful Web サービスの構築」で説明しています。

3. HCSC コンポーネントの定義と，HCSC サーバへの配備が完了していることを確認します。

   次の点を確認します。

   - HTTP アダプタを動作させるためのクラスパスが追加済であること
     参照先：「5.2.1　HTTP アダプタを動作させるためのクラスパスの設定」
   - InventoryManagement サービスアダプタの定義と，HCSC サーバへの配備が完了していること
     参照先：「5.2.2　InventoryManagement サービスアダプタの定義」
   - DeliveryReceipt サービスアダプタの定義と，HCSC サーバへの配備が完了していること
     参照先：「5.2.3　DeliveryReceipt サービスアダプタの定義」
   - ProductStock_Normal ビジネスプロセスの定義と，HCSC サーバへの配備が完了していること
     参照先：「5.2.4　ProductStock_Normal ビジネスプロセスの定義」

5.3.2　商品手配システム（基本編）の実行

商品手配システム（基本編）を実行する方法について説明します。

(1)　実行方法

サービスリクエスタから HTTP リクエストを送信します。

(a)　HTTP リクエストの内容

HTTP リクエストの内容を次に示します。

項目	内容
HTTP メソッド	POST
データ形式	JSON
URL※1	http://localhost:80/rcp1/arrangeItem
入力インタフェース※2	"{¥"itemName¥":¥"$itemName¥", ¥"quantity¥":$quantity}"

注※1

HTTP 受付での URL の指定方法の詳細については，マニュアル「サービスプラットフォーム 解説」の「2.15.4　HTTP 受付のリクエスト処理」を参照してください。

注※2

$itemName で入力できる商品は，次の 4 パターンだけです。これ以外を入力すると，在庫なしレスポンスを返却します。

- PC
- TV
- Camera
- Smartphone

$quantity には自然数だけ入力できます。在庫数は各商品 10 個ずつです。在庫がない場合，在庫なしレスポンスを返却します。

(b)　コマンド実行例

ここでは，サービスリクエスタには curl コマンドを使用します。コマンドプロンプトを起動し，次のコマンドを入力して実行してください。

```
curl http://localhost:80/rcp1/arrangeItem -X POST -H ”Content-Type: application/json” -d ”{¥”itemName¥”:¥”PC¥”, ¥”quantity¥”:1}”
```

(2) 実行結果の確認

コマンドの実行後，実行結果が表示されます。

表示される内容を次に説明します。

(a) 正常に応答した場合（手配が完了した場合）

在庫ありレスポンスとして，手配が完了したことを知らせるメッセージ，商品名，個数，および配送番号が表示されます。

- 出力例

```
{”deliveryNumber”: ”D00000001”, ”itemName”: ”PC”, ”message”: ”Product is available for ar
rangement.”, ”quantity”:”1”}
```

(b) 正常に応答した場合（在庫がない場合または商品リスト以外の商品を入力した場合）

在庫なしレスポンスとして，在庫がないことを知らせるメッセージ，商品名，および個数が表示されます。

- 出力例

```
{”deliveryNumber”:””,”itemName”:”PC”,”message”:”No stock is available.”,”quantity”:”1”}
```

(c) ビジネスプロセス（基本編）の実行時に例外が発生した場合

次の内容が表示されます。

- 出力例（在庫管理サービスを停止した状態で実行）

```
<error-message>KDEC80421-E Processing will now be canceled because an error occurred duri
ng custom reception framework processing. (reception name = ServiceReceipt, reception ID
= rcp1, operation name = arrangeItem, RootApInfo = 127.0.0.1/26656/0x0000000000002a5d, er
ror message = A service call failed. business-process-definition-name = ProductStock_Norm
al, business-process-definition-version = 1, process-instance-ID = J2EEServer_0127001_Pro
ductStock_Normal_1713749012284_200599499_58, invoke-activity-name = 在庫引当, service-nam
e = InventoryManagement, details = A service call was interrupted because a custom servic
e access error was detected. (adapter name = InvAdp, HCSCCommonID = CSC_HCSC_2024-04-22_1
0:23:32.260_2, ServiceRequestID = MSG_HCSC_SyncBP_2024-04-22_10:23:41.320_4, error messag
e = An HTTP adapter error occurred. (adapter name = InvAdp, operation name = reserveItem,
 exception information = jp.co.Hitachi.soft.csc.cstmadp.http.rt.exception.CSAHTAException
: KDEC81054-E An error status code was received from the service. (adapter name = InvAdp,
 operation name = reserveItem, status code = 404)), code = KDEC81499-E) ErrorCode=KDEC030
05-E ErrorCode=KDEC20023-E, error code = KDEC80421-E)</error-message>
```

ここで示した結果になるかどうかテストするために，在庫管理サービスを一時的に停止させたい場合は，「5.3.3　商品在庫数の初期化方法」を参照して在庫管理サービスを停止させてください。

5.3.3　商品在庫数の初期化方法

商品手配システムでの各商品の総在庫数は 10 個です。手配が完了すると，その数だけ在庫が減少します。

在庫がなくなった場合は，サービスをリデプロイするか，RESTful Web サービス配備用 Web システムを再起動する必要があります。これによって在庫数が 10 個に戻ります。

サービスのリデプロイおよび RESTful Web サービス配備用 Web システムの再起動方法を次に説明します。

(1)　サービスのリデプロイ

サービスの停止と再起動について，在庫管理サービスと配送受付サービスに分けて説明します。この操作は，管理者または管理者特権で実行してください。

(a)　在庫管理サービスの停止と再起動

在庫管理サービスの停止と再起動について説明します。

- 在庫管理サービスの停止

  次のコマンドを実行します。

```
”%COSMINEXUS_HOME%¥CC¥admin¥bin¥cjstopapp” jaxrsserver -nameserver corbaname::localhost:9
01 -name Sample_application_jaxrs_inventorymanagement
```

- 停止した在庫管理サービスの再起動

  次のコマンドを実行します。

```
”%COSMINEXUS_HOME%¥CC¥admin¥bin¥cjstartapp” jaxrsserver -nameserver corbaname::localhost:
901 -name Sample_application_jaxrs_inventorymanagement
```

(b)　配送受付サービスの停止と再起動

配送受付サービスの停止と再起動について説明します。

- 配送受付サービスの停止

  次のコマンドを実行します。

```
”%COSMINEXUS_HOME%¥CC¥admin¥bin¥cjstopapp” jaxrsserver -nameserver corbaname::localhost:9
01 -name Sample_application_jaxrs_delivery
```

- 停止した配送受付サービスの再起動

  次のコマンドを実行します。

```
”%COSMINEXUS_HOME%¥CC¥admin¥bin¥cjstartapp” jaxrsserver -nameserver corbaname::localhost:
901 -name Sample_application_jaxrs_delivery
```

(2) RESTful Web サービス配備用 Web システムの再起動

RESTful Web サービス配備用 Web システムを停止させ，その後再起動します。この操作は，管理者または管理者特権で実行してください。

1. RESTful Web サービス配備用 Web システムを停止します。

   Web システムを停止するコマンドは次のとおりです。

```
”%COSMINEXUS_HOME%¥manager¥bin¥cmx_stop_target” -m localhost -u admin -p admin -s RESTful
WebServiceSystem -mode ALL
```

2. RESTful Web サービス配備用 Web システムを再起動します。

   Web システムを再起動するコマンドは次のとおりです。

```
”%COSMINEXUS_HOME%¥manager¥bin¥cmx_start_target” -m localhost -u admin -p admin -s RESTfu
lWebServiceSystem -mode ALL
```

5.4　ProductStock_Normal ビジネスプロセスのデバッグ

「5.2.4　ProductStock_Normal ビジネスプロセスの定義」で定義したビジネスプロセスをデバッグして，ビジネスプロセスの処理の流れを確認してみましょう。

「5.4.1　商品手配システムが提供するサービス部品を利用するデバッグ」では，商品手配システムが提供するサービス部品を利用するデバッグについて説明します。サービス部品を使用せずに，サービスのエミュレーションの機能を使ってデバッグする場合は，「5.4.2　サービスのエミュレーション機能を利用する場合」を参照してください。

5.4.1　商品手配システムが提供するサービス部品を利用するデバッグ

商品手配システムが提供するサービス部品を利用するデバッグの方法について説明します。

商品手配システムのビジネスプロセスのデバッグを実行する前に，次の作業が実施済みであることを確認してください。

1. **サーバが起動していることを確認します。**
   次の 2 つのサーバが起動していることを確認してください。

   HCSC サーバ
   　起動方法は「3.4.2　テスト環境の起動」で説明しています。

   RESTful Web サービス用 Web システム
   　起動方法は「4.2　サービス部品用 Web システムの構築」で説明しています。

2. **RESTful Web サービスが開始していることを確認します。**
   開始方法は「4.3　RESTful Web サービスの構築」で説明しています。

3. **HCSC コンポーネントの定義と，HCSC サーバへの配備が完了していることを確認します。**
   次の点を確認します。
   - HTTP アダプタを動作させるためのクラスパスが追加済であること
     参照先：「5.2.1　HTTP アダプタを動作させるためのクラスパスの設定」
   - InventoryManagement サービスアダプタの定義と，HCSC サーバへの配備が完了していること
     参照先：「5.2.2　InventoryManagement サービスアダプタの定義」
   - DeliveryReceipt サービスアダプタの定義と，HCSC サーバへの配備が完了していること
     参照先：「5.2.3　DeliveryReceipt サービスアダプタの定義」
   - ProductStock_Normal ビジネスプロセスの定義が完了していること
     参照先：「5.2.4　ProductStock_Normal ビジネスプロセスの定義」

ビジネスプロセスのデバッグの流れを次に示します。

1. アクティビティにブレークポイントを設定します。
2. デバッガを起動します。
3. サービスリクエスタからリクエストを送信し，ビジネスプロセスの処理を開始します。
4. アクティビティごとにデバッグを実行します。
5. ビジネスプロセスのデバッグを終了します。

(1) ブレークポイントの設定

ビジネスプロセスの処理を中断するアクティビティにブレークポイントを設定します。

ここでは，サービス呼出アクティビティ（在庫引当）にブレークポイントを設定します。

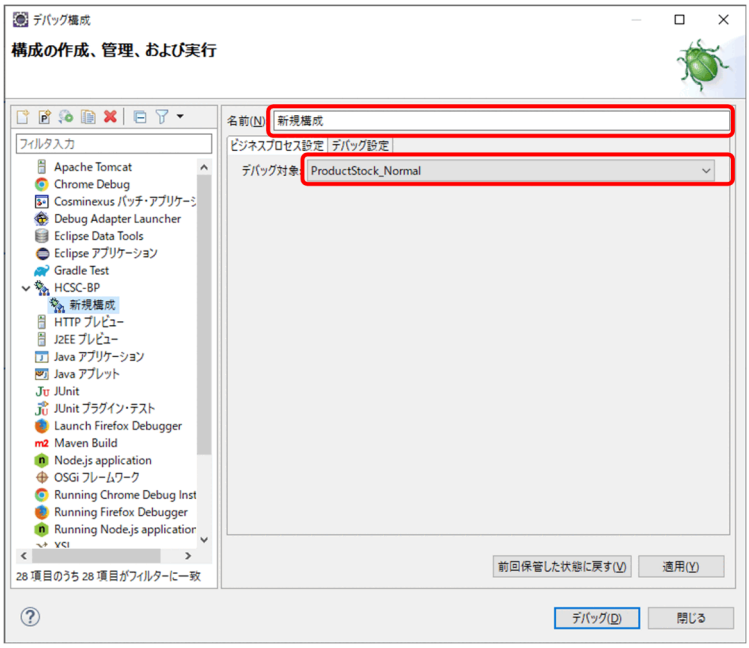
ブレークポイントを設定する手順を次に示します。

1. ビジネスプロセス定義画面のキャンバスでサービス呼出アクティビティ（在庫引当）を右クリックし，［ブレークポイントの追加］を選択します。
   アクティビティにブレークポイントが追加されます。次の図のように，アクティビティの横にブレークポイントを表すチェックが付きます。

(2) デバッガの起動

デバッガを起動する手順を次に示します。

1. Eclipse のメニューから［実行］－［デバッグの構成］を選択します。
   ［デバッグ構成］ダイアログが表示されます。
2. ［デバッグ構成］ダイアログのメニューから［HCSC-BP］を右クリックし，［新規］を選択します。
3. ［名前］に任意の名称を入力し，［デバッグ対象］は，ドロップダウンリストから「ProductStock_Normal」を選択します。



4. [デバッグ] ボタンをクリックします。
   アカウント認証画面が表示されます。

5. [ユーザ ID] に「admin」，[パスワード] に「admin」を入力して，[OK] ボタンをクリックします。
   処理中であることを知らせるメッセージが表示されます。2 回目以降にデバッガを起動するときは，アカウント認証画面は表示されないで，すぐに処理が開始されます。

6. [OK] ボタンをクリックします。
   デバッガが起動されます。

(3) リクエストの送信

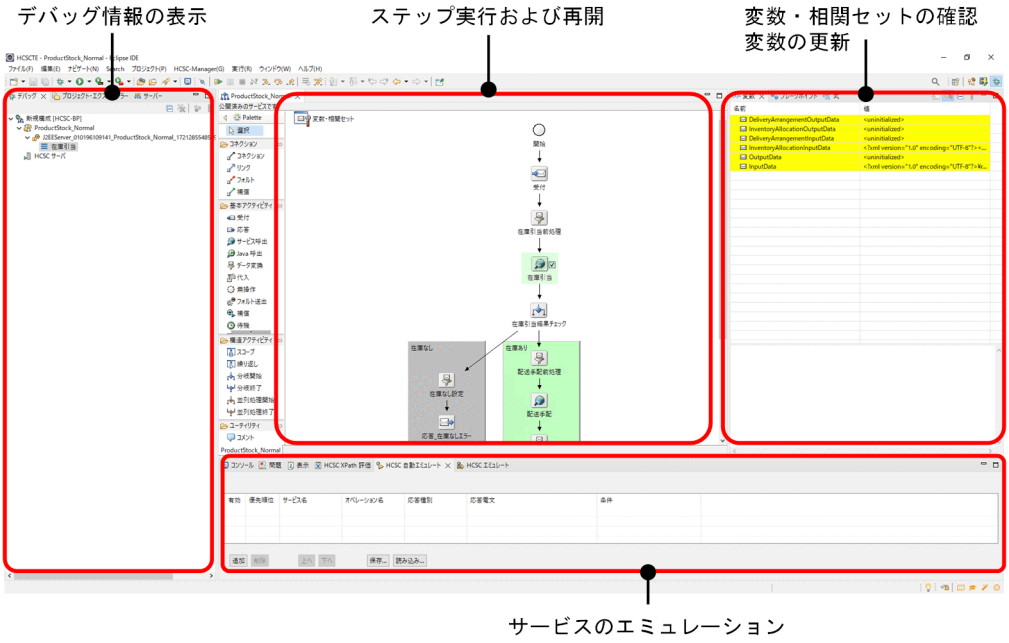
サービスリクエスタからリクエストを送信して，ビジネスプロセスの処理を開始します。

リクエストの送信手順については，「5.3.2 商品手配システム（基本編）の実行」を参照してください。

(4) ビジネスプロセスのデバッグの実行

ビジネスプロセスの処理が中断している状態では，次に示す画面でデバッグを実行できます。

図 5-7　ビジネスプロセスのデバッグ画面



ビジネスプロセスのデバッグで実行できる項目を次の表に示します。

表 5-7　ビジネスプロセスのデバッグで実行できる項目

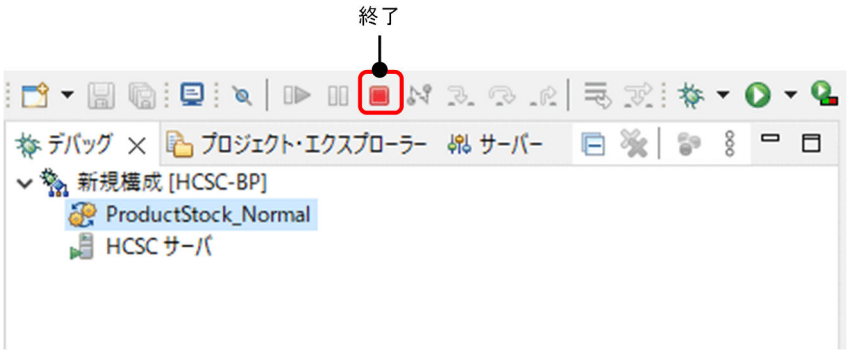
項目	説明
変数・相関セットの確認	ビジネスプロセスで現在使用している変数，および相関セットを確認できます。
変数の更新	ビジネスプロセスで現在使用している変数の値を更新できます。
ステップ実行および再開	[デバッグ] ビューでは次の操作を実行できます。これらの操作によって，アクティビティごとにデバッグを実行してください。 • ステップイン（ ） 次のアクティビティに進みます。分岐開始アクティビティで実行した場合は，分岐先で最初に処理されるアクティビティに進みます。 • ステップオーバー（ ） 分岐開始アクティビティで実行した場合は，分岐終了アクティビティまでをまとめて処理します。分岐開始アクティビティ以外で実行した場合は，ステップインと同じ動作になります。 • ステップリターン（ ） スコープアクティビティおよび繰り返しアクティビティの内部のアクティビティで実行した場合は，所属するアクティビティの次のアクティビティまで処理します。 • 再開（ ） 次にブレークポイントを設定しているアクティビティまで，ビジネスプロセスの処理を再開します。
サービスのエミュレーション	作成済みの応答電文を利用することで，サービスをエミュレーションできます。 サービスをエミュレーションする手順については，「5.4.2　サービスのエミュレーション機能を利用する場合」を参照してください。

(5)　ビジネスプロセスのデバッグの終了

ビジネスプロセスの処理の流れを確認したら，ビジネスプロセスのデバッグを終了します。

ビジネスプロセスのデバッグを終了する手順を次に示します。

1. [デバッグ] ビューで「ProductStock_Normal」を選択し，[終了] アイコンをクリックします。



ビジネスプロセスのデバッグが終了します。

5.4.2　サービスのエミュレーション機能を利用する場合

サービスのエミュレーション機能では，あらかじめサービスの応答電文を作成しておき，HCSC エミュレートビューに設定することで，実際のサービスを呼び出す代わりとすることができます。この機能によって，サービス呼出アクティビティに設定したサービス部品がなくても，ビジネスプロセスのデバッグ処理を進められます。

> **参考**
>
> cscgenbinary コマンドでバイナリ電文を作成し，応答電文として使用できます。cscgenbinary コマンドについては，マニュアル「サービスプラットフォーム リファレンス」の「cscgenbinary (バイナリフォーマット定義ファイルからバイナリ電文の生成)」を参照してください。

サービスのエミュレーションは，ビジネスプロセスの処理が中断しているときに設定できます。ここでは，[5.4.1(3)　リクエストの送信]」までの手順を完了し，サービス呼出アクティビティ（在庫引当）でビジネスプロセスの処理が中断しているものとして，「在庫引当」サービスをエミュレーションする手順について説明します。

「在庫引当」サービスをエミュレーションする手順を次に示します。

1. XML ファイルを作成します。
2. HCSC エミュレートビューを表示します。
3. 作成した XML ファイルを利用してサービスのエミュレーションを実行します。

サービスのエミュレーションを実行する前に，デバッガを起動して，サービスリクエスタからリクエストを送信しておく必要があります。

(1) XML ファイルの作成

サービスの応答として利用する XML ファイルを作成します。

XML ファイルを作成する手順を次に示します。

1. Eclipse のメニューから，［ファイル］－［新規］－［その他］を選択し，表示されるダイアログで［一般］－［ファイル］を選択します。
   ［新規ファイル］ダイアログが表示されます。
2. ［新規ファイル］ダイアログで XML ファイルを保管するディレクトリを選択します。
3. ［ファイル名］に任意のファイル名（拡張子：.xml）を入力します。
4. ［終了］ボタンをクリックします。
   XML エディタが表示されます。
5. XML エディタの［ソース］タブをクリックし，次のように入力します（△は半角の空白です）。

```
<?xml△version=”1.0”△encoding=”UTF-8”?>
<csc-object><reservationNumber>R00000001</reservationNumber></csc-object>
```

6. メニューから［ファイル］－［保管］を選択します。
   XML ファイルが保存されます。

(2) HCSC エミュレートビューの表示

サービスのエミュレーションは HCSC エミュレートビューで実行できます。

HCSC エミュレートビューを表示する手順を次に示します。

1. Eclipse のメニューから，［ウィンドウ］－［ビューの表示］－［その他］を選択します。
   ［ビューの表示］ダイアログが表示されます。
2. ［デバッグ］－［HCSC エミュレート］を選択し，［開く］ボタンをクリックします。
   HCSC エミュレートビューが表示されます。

(3) サービスのエミュレーションの実行

作成した XML ファイルを利用して，HCSC エミュレートビューでサービスのエミュレーションを実行します。

サービスのエミュレーションの実行手順を次に示します。

1. [デバッグ] ビューで在庫引当（サービス呼出アクティビティ）を選択します。

2. HCSC エミュレートビューの［追加］ボタンをクリックします。
   HCSC エミュレートビューのテーブルに行が追加されます。

3. 応答種別のセルを選択し，通常応答を選択します。

4. 応答電文のセルを選択し，［...］ボタンをクリックします。
   ［ファイル選択］ダイアログが表示されます。

5. 作成した XML ファイルを選択し，［OK］ボタンをクリックします。

6. [ステップ実行] ボタンをクリックします。
   サービス呼出アクティビティ（在庫引当）からの応答がエミュレーションされます。
   ビジネスプロセスの処理が分岐開始アクティビティ（在庫引当結果チェック）に進みます。

以降のアクティビティへ進むには，［デバッグ］ビューを操作してください。詳細は，「5.4.1(4)　ビジネスプロセスのデバッグの実行」を参照してください。

5.5　商品手配システムの開発（フォルトハンドリング編）

「5.2.4　ProductStock_Normal ビジネスプロセスの定義」では，正常系だけを開発しました。この項では，異常系のハンドリングを実施した商品手配システムを開発します。

商品手配システム（フォルトハンドリング編）では，次に示す４つのコンポーネントを開発します。

- InventoryManagement サービスアダプタ
- DeliveryReceipt サービスアダプタ
- ProductStock_FaultHandling ビジネスプロセス
- ServiceReceipt 受付

InventoryManagement サービスアダプタと DeliveryReceipt サービスアダプタは，商品手配システム（基本編）で開発したものを再利用して修正する形で開発します。

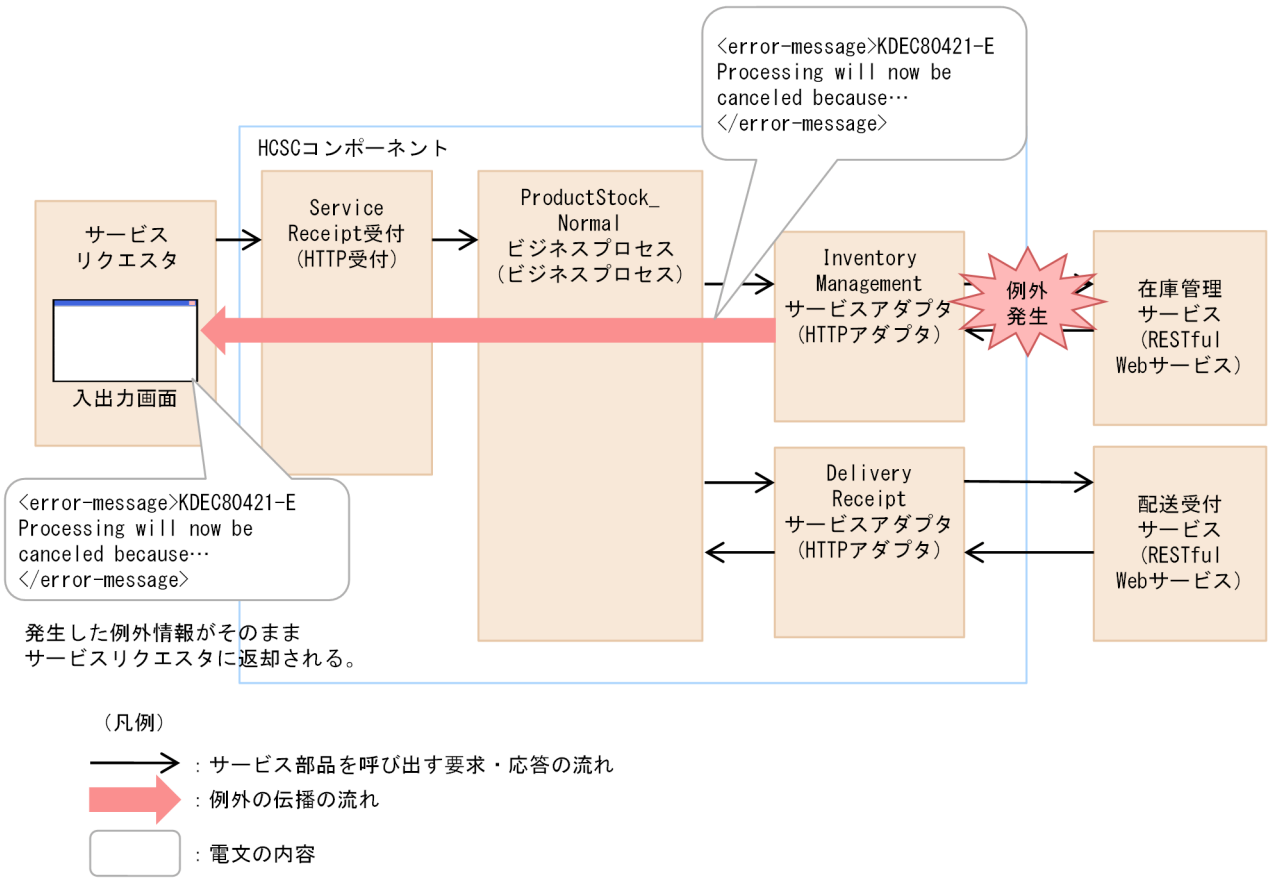
ProductStock_FaultHandling ビジネスプロセスと ServiceReceipt 受付は，ProductStock_Normal ビジネスプロセスを複製したものに追加・修正する形で開発します。

5.5.1　概要

商品手配システム（基本編）では，システムの正常系の処理だけを実装しました。そのため，商品手配システムで例外が発生しなければ，商品手配システムは正しく処理を実施した結果をサービスリクエスタに返却できます。しかし，通信が失敗するなどの例外が発生した場合は，意図した結果を返却できません。

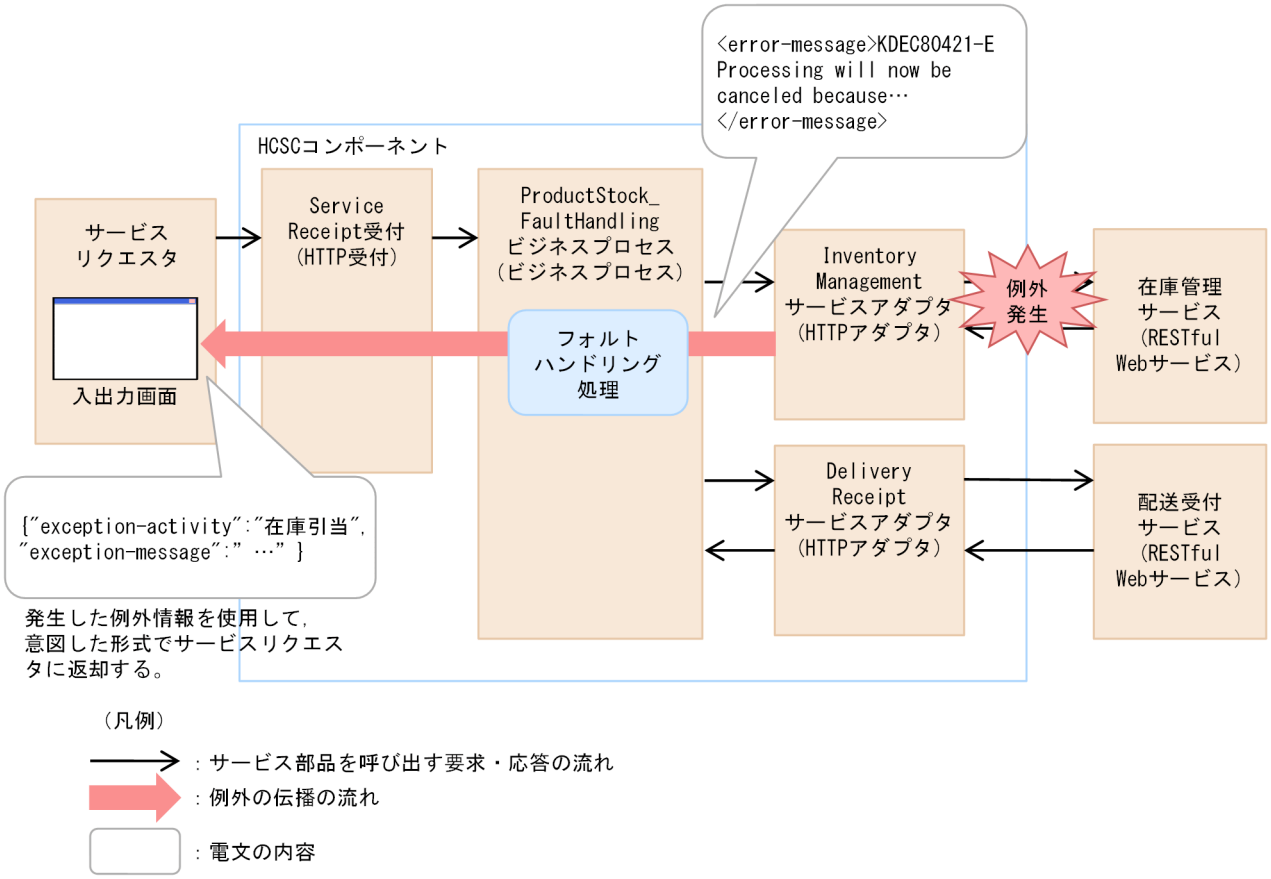
例えば，次の図のように，InventoryManagement サービスアダプタと在庫管理サービスの間の通信が失敗した場合，発生した例外情報をサービスリクエスタにそのまま返却することになります。

図 5-8　例外情報の返却例（商品手配システム（基本編））



商品手配システム（フォルトハンドリング編）では，商品手配システムで発生した例外をハンドリングし，サービスリクエスタに意図した結果を返却できるようにします。例えば，InventoryManagement アダプタと在庫管理サービスの間の通信が失敗した場合は，次の図のようにビジネスプロセスで例外情報を加工して，意図した形式のレスポンスをサービスリクエスタに返却するようにします。

図 5-9　例外情報の返却例（商品手配システム（フォルトハンドリング編））



5.5.2　例外ハンドリングの仕様

商品手配システム（フォルトハンドリング編）では，次の 3 つの例外をハンドリングします。

1. サービスリクエスタからの JSON 電文の内容が不正である例外（入力値検証例外）
2. InventoryManagement アダプタと在庫管理サービスの間の通信が失敗した例外（在庫引当例外）
3. DeliveryReceipt アダプタと配送受付サービスの間の通信が失敗した例外（配送手配例外）

それぞれの例外について次に示します。

(1)　サービスリクエスタからの JSON 電文の内容が不正である例外（入力値検証例外）

サービスリクエスタからの JSON 電文の内容を検証します。内容が不正な場合，その原因と対処方法を含んだメッセージを作成し，サービスリクエスタに返却します。

検証する内容を次に示します。

項目	検証内容
itemName（商品名）	値が代入されているか 入力された商品名は商品リスト※に存在するか
quantity（個数）	値が代入されているか 1～10 の範囲内の整数か

注※

商品リストに含まれる商品は次の 4 つです。

・PC

・TV

・Camera

・Smartphone

JSON 電文の内容が不正の場合，サービスリクエスタには次のレスポンスを返却します。

- レスポンス（ボディ）

  次に示す形式のレスポンス（ボディ）を返却します。

名	変数名	型
例外が発生したアクティビティ名	exception-activity	string
例外メッセージ	exception-message	string

レスポンス（ボディ）の出力例を次に示します。

```
{"exception-activity": "入力値検証", "exception-message": "quantityを整数で入力してください"}
```

- レスポンス（ヘッダ）

  ステータスコード 400（クライアントエラー）を返却します。

(2) InventoryManagement アダプタと在庫管理サービスの間の通信が失敗した例外（在庫引当例外）

InventoryManagement アダプタと在庫管理サービスの間の通信が失敗した場合，例外情報を含んだメッセージを作成し，サービスリクエスタに返却します。

返却するレスポンスの形式を次に示します。

- レスポンス（ボディ）

  次に示す形式のレスポンス（ボディ）を返却します。

名	変数名	型
例外が発生したアクティビティ名	exception-activity	string
例外メッセージ	exception-message	string

レスポンス（ボディ）の出力例を次に示します。

```
{"exception-activity":"在庫引当","exception-message":"jp.co.Hitachi.soft.csc.msg.adapter.protocol.CSCMsgCustomAccessException: A service call was interrupted because a custom service access error was detected. (adapter name = InvAdp, HCSCCommonID = CSC_HCSC_2024-06-25_17:27:38.029_6, ServiceRequestID = MSG_HCSC_SyncBP_2024-06-25_17:27:38.045_14, error message = An HTTP adapter error occurred. (adapter name = InvAdp, operation name = reserveItem, exception information = jp.co.Hitachi.soft.csc.cstmadp.http.rt.exception.CSAHTAException: KDEC81054-E An error status code was received from the service. (adapter name = InvAdp, operation name = reserveItem, status code = 404)), code = KDEC81499-E) ErrorCode=KDEC03005-E"}
```

- レスポンス（ヘッダ）

  ステータスコード 500（サーバエラー）を返却します。

(3) DeliveryReceipt アダプタと配送受付サービスの間の通信が失敗した例外（配送手配例外）

DeliveryReceipt アダプタと配送受付サービスの間の通信が失敗した場合，例外情報を含んだメッセージを作成し，サービスリクエスタに返却します。

返却するレスポンスの形式を次に示します。

- レスポンス（ボディ）

  次に示す形式のレスポンス（ボディ）を返却します。

名	変数名	型
例外が発生したアクティビティ名	exception-activity	string
例外メッセージ	exception-message	string

レスポンス（ボディ）の出力例を次に示します。

```
{"exception-activity":"配送手配","exception-message":"jp.co.Hitachi.soft.csc.msg.adapter.protocol.CSCMsgCustomAccessException: A service call was interrupted because a custom service access error was detected. (adapter name = DelAdp, HCSCCommonID = CSC_HCSC_2024-06-25_17:33:39.366_8, ServiceRequestID = MSG_HCSC_SyncBP_2024-06-25_17:33:39.625_18, error message = An HTTP adapter error occurred. (adapter name = DelAdp, operation name = deliverItem, exception information = jp.co.Hitachi.soft.csc.cstmadp.http.rt.exception.CSAHTAException: KDEC81054-E An error status code was received from the service. (adapter name = DelAdp, operation name = deliverItem, status code = 404)), code = KDEC81499-E) ErrorCode=KDEC03005-E"}
```

- レスポンス（ヘッダ）

  ステータスコード 500（サーバエラー）を返却します。

5.5.3 例外ハンドリングの実装の流れ

商品手配システム（フォルトハンドリング編）での例外ハンドリングの実装の流れを次に示します。

1. **サービス部品との通信で発生した例外をサービスアダプタ内でフォルトに変換できるようにサービスアダプタを設定します。**

   在庫管理サービス，配送受付サービスとの通信で発生した例外をハンドリングするため，商品手配システム（基本編）で開発した InventoryManagement サービスアダプタと DeliveryReceipt サービスアダプタの設定を変更します。

   詳細については，次の説明を参照してください。

   InventoryManagement サービスアダプタ

   5.5.4　InventoryManagement サービスアダプタの定義

   DeliveryReceipt サービスアダプタ

   5.5.5　DeliveryReceipt サービスアダプタの定義

   なお，例外とフォルトには次の違いがあります。

   - 例外：プログラム上のエラー
   - フォルト：プログラム上のエラーをビジネスプロセスで扱えるフォルト電文に置き換えたもの

   例外をフォルトに変換することで，例外情報をフォルト電文としてビジネスプロセス内で検知できます。

2. **意図したレスポンスを返却できるように，ProductStock_FaultHandling ビジネスプロセスでフォルトハンドリングを実装します。**

   ProductStock_FaultHandling ビジネスプロセスは，複製した ProductStock_Normal ビジネスプロセスに追加・修正する形で開発します。

   「5.5.2　例外ハンドリングの仕様」で説明した 3 つの例外ハンドリングの仕様を満たすように，ProductStock_FaultHandling ビジネスプロセスにアクティビティを定義して実装します。

5.5.4　InventoryManagement サービスアダプタの定義

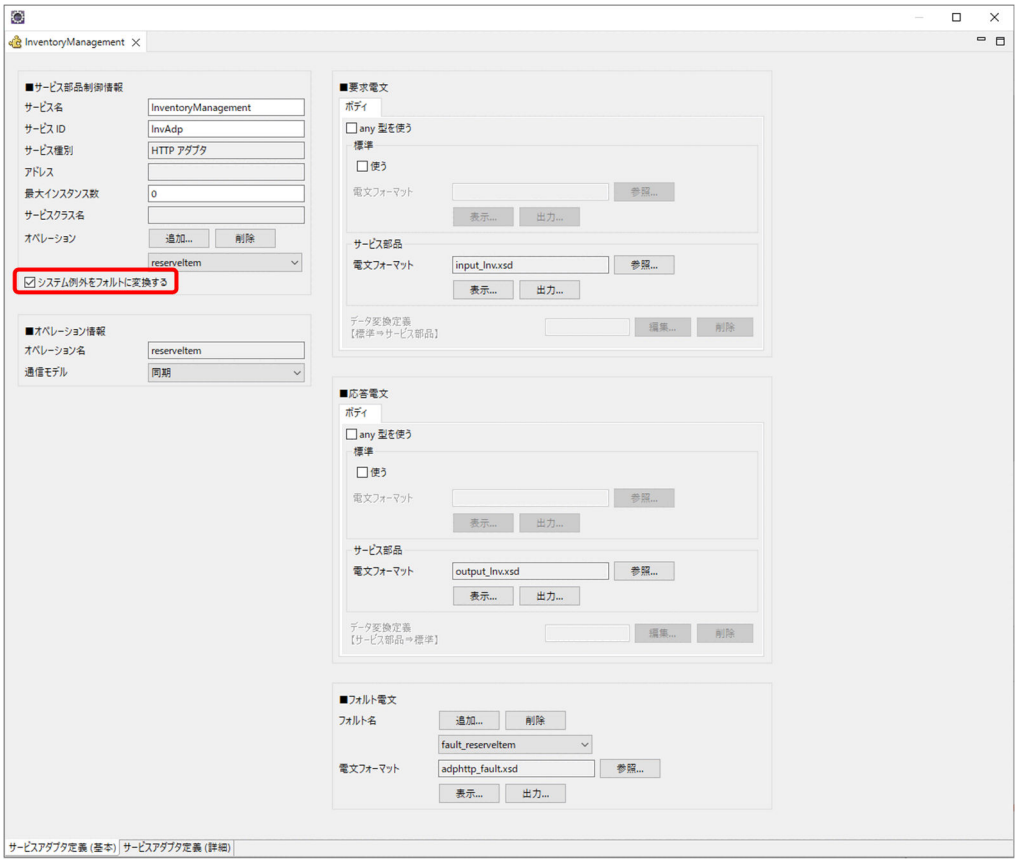
商品手配システム（フォルトハンドリング編）では，InventoryManagement サービスアダプタは，商品手配システム（基本編）で作成したものを再利用して開発します。

開発手順を次に示します。

1. **ツリービューの［サービス定義一覧］から InventoryManagement サービスアダプタをダブルクリックします。**

   サービスアダプタ定義（基本）画面が表示されます。

2. **［システム例外をフォルトに変換する］のチェックボックスにチェックを入れます。**



3. メニューから［ファイル］－［保管］を選択します。

4. 非公開化をするかを確認するメッセージが表示されるので，［OK］ボタンをクリックします。

これ以外の定義は，商品手配システム（基本編）で作成した InventoryManagement サービスアダプタと同じです。定義内容および設定方法については，「5.2.2　InventoryManagement サービスアダプタの定義」を参照してください。

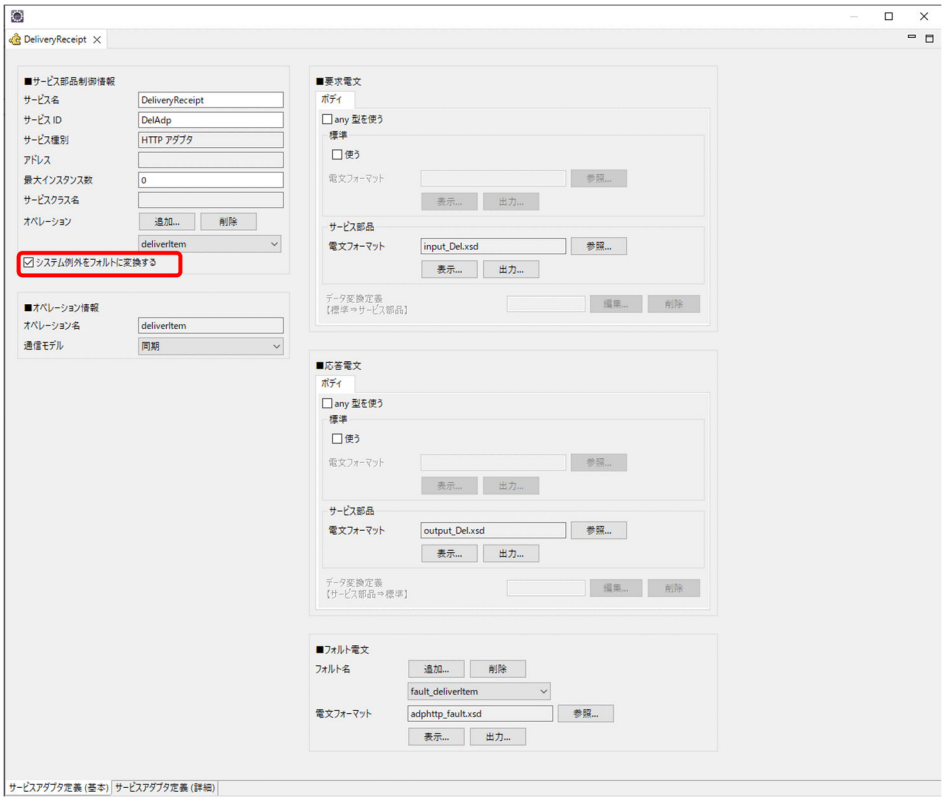
5.5.5　DeliveryReceipt サービスアダプタの定義

商品手配システム（フォルトハンドリング編）では，DeliveryReceipt サービスアダプタは，商品手配システム（基本編）で作成したものを再利用して開発します。

開発手順を次に示します。

1. ツリービューの［サービス定義一覧］から DeliveryReceipt サービスアダプタをダブルクリックします。
   サービスアダプタ定義（基本）画面が表示されます。

2. ［システム例外をフォルトに変換する］のチェックボックスにチェックを入れます。



3. メニューから［ファイル］－［保管］を選択します。

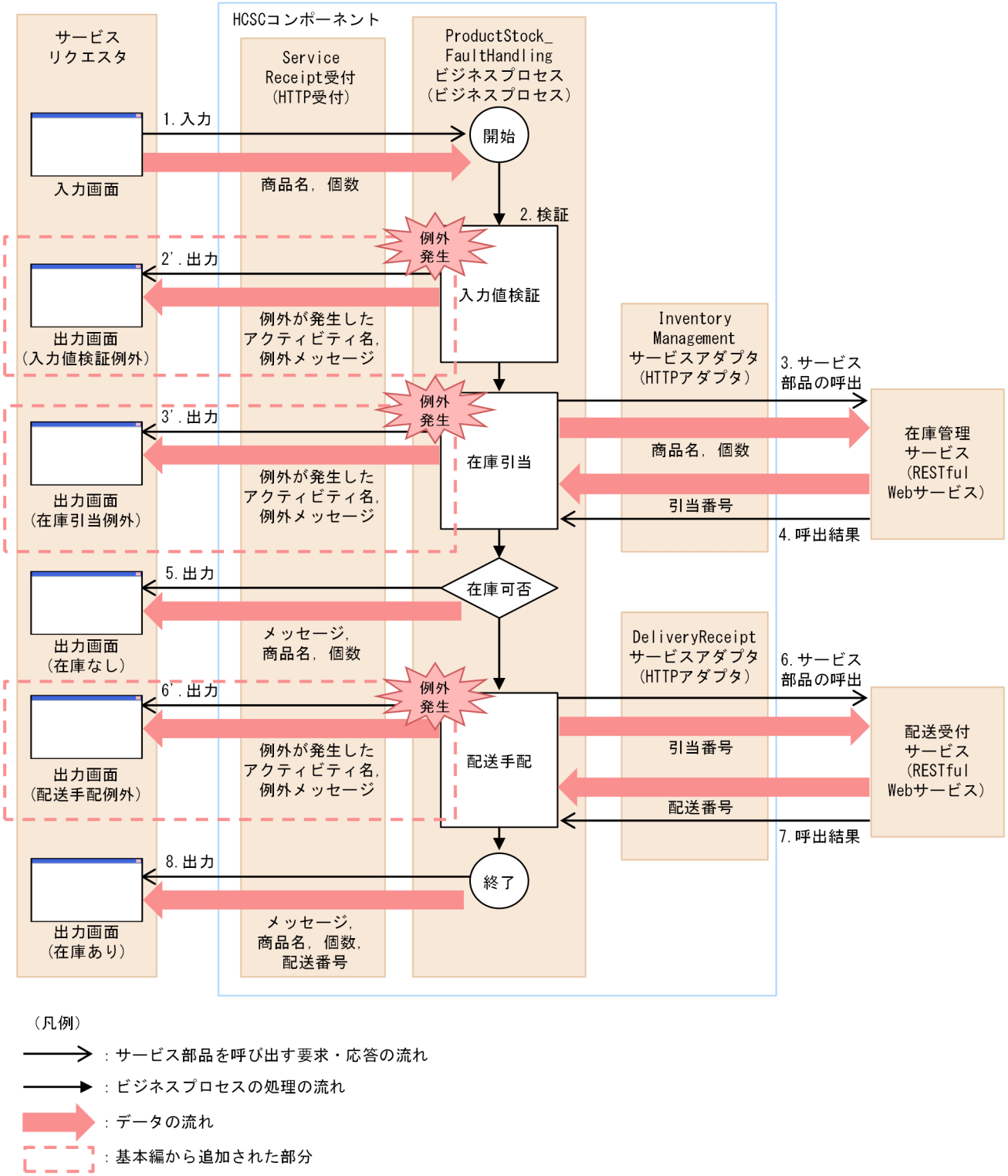
4. 非公開化をするかを確認するメッセージが表示されるので，［OK］ボタンをクリックします。

これ以外の定義は，商品手配システム（基本編）で作成した DeliveryReceipt サービスアダプタと同じです。定義内容および設定方法については，「5.2.3　DeliveryReceipt サービスアダプタの定義」を参照してください。

5.5.6　ProductStock_FaultHandling ビジネスプロセスの定義

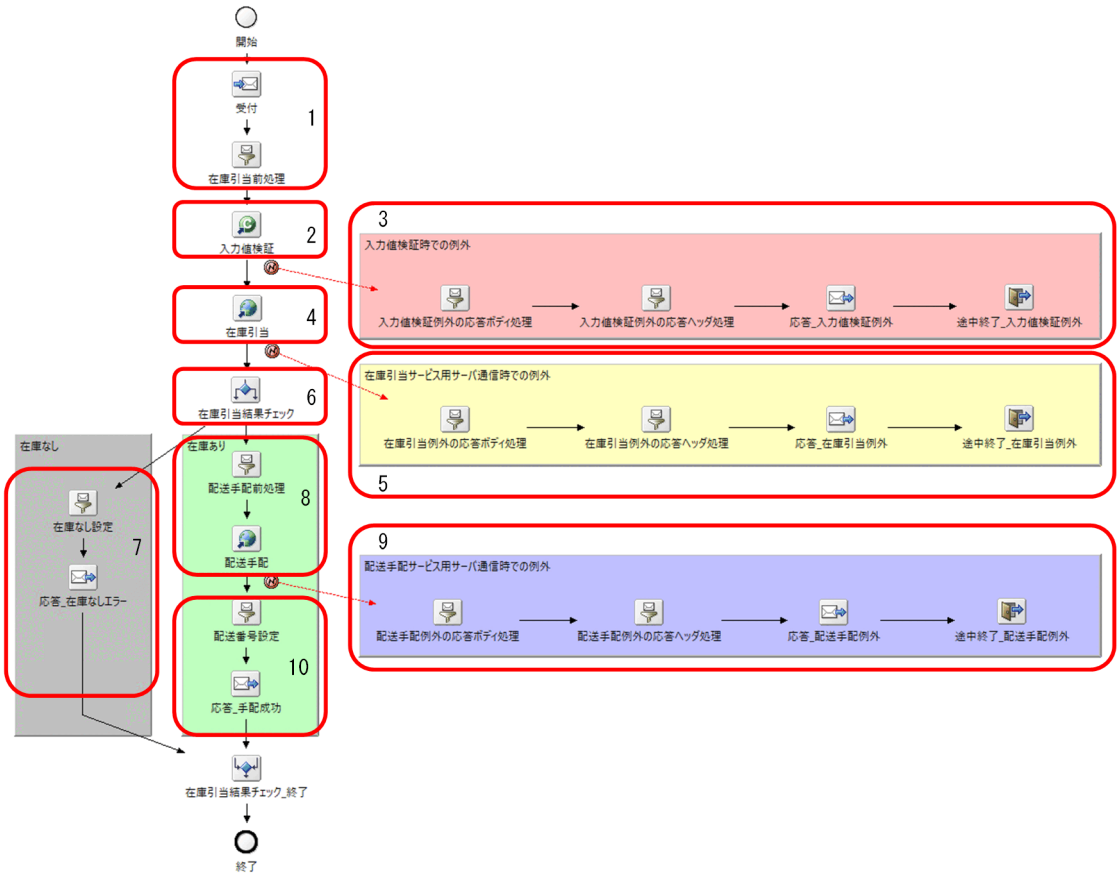
商品手配システム（フォルトハンドリング編）の処理の流れを次の図に示します。

図 5-10　商品手配システム（フォルトハンドリング編）の処理の流れ



上記の処理を ProductStock_FaultHandling ビジネスプロセスで実現すると，次の図のような処理になります。

図 5-11　ProductStock_FaultHandling ビジネスプロセスの処理



ProductStock_FaultHandling ビジネスプロセスは次の処理を実現します。

1. ProductStock_FaultHandling ビジネスプロセスは，サービスリクエスタから商品名と個数の情報を受信します。
2. 入力値検証を実施します。
3. 入力値検証で例外が発生した場合は，「5.5.2(1)　サービスリクエスタからの JSON 電文の内容が不正である例外（入力値検証例外）」で示す例外のレスポンスをサービスリクエスタに返却します。
4. InventoryManagement サービスアダプタを介して在庫管理サービスを呼び出します。
5. 在庫管理サービスとの通信時に例外が発生した場合は，「5.5.2(2)　InventoryManagement アダプタと在庫管理サービスの間の通信が失敗した例外（在庫引当例外）」で示す例外のレスポンスをサービスリクエスタに返却します。
6. 在庫可否を確認します。
7. 在庫がない場合は，在庫なしレスポンスをサービスリクエスタに返却します。
8. 在庫がある場合は，DeliveryReceipt サービスアダプタを介して配送受付サービスを呼び出し，配送番号を取得します。

9. 配送受付サービスとの通信時に例外が発生した場合は，「5.5.2(3)　DeliveryReceipt アダプタと配送受付サービスの間の通信が失敗した例外（配送手配例外）」で示す例外のレスポンスをサービスリクエスタに返却します。

10. サービスリクエスタに取得した配送番号を返却します。

また，商品手配システム（基本編）と同様にリクエストは HTTP リクエストとし，ユーザ定義受付（HTTP 受付）をビジネスプロセスに関連づけて定義します。

商品手配システム（フォルトハンドリング編）のビジネスプロセスは，次の流れで定義します。

1. ProductStock_Normal を複製し，ProductStock_FaultHandling を作成します。
   修正方法については，「(1)　ビジネスプロセスの複製」を参照してください。

2. ユーザ定義受付を修正します。
   修正方法については，「(2)　ユーザ定義受付の確認」を参照してください。

3. ユーザ定義受付の要求電文・応答電文フォーマットを作成します。
   作成方法については，「(3)　電文フォーマットの作成」を参照してください。

4. 変数を追加・修正します。
   修正方法については，「(4)　変数の設定」を参照してください。

5. 追加する処理に応じてアクティビティを配置・定義します。
   配置方法については，「(5)　アクティビティの配置」を参照してください。
   定義方法については，「(6)　アクティビティの定義」を参照してください。

6. アクティビティを修正します。
   変数を修正したため，次のアクティビティも修正します。
   アクティビティ名：在庫なし設定
   配置と定義については，「(6)(n)　データ変換アクティビティ（アクティビティ名：在庫なし設定）」を参照してください。
   アクティビティ名：配送番号設定
   配置と定義については，「(6)(v)　データ変換アクティビティ（アクティビティ名：配送番号設定）」を参照してください。

7. フォルト処理を定義します。
   定義方法については，「(7)　フォルト処理の定義」を参照してください。

8. ビジネスプロセスの定義を終了します。

(1)　ビジネスプロセスの複製

ProductStock_Normal ビジネスプロセスを複製し，ProductStock_FaultHandling ビジネスプロセスを追加するときに設定する値を次の表に示します。

表 5-8　ProductStock_FaultHandling ビジネスプロセスを追加するときに設定する値

項目名	設定する値	説明
ビジネスプロセス名	ProductStock_FaultHandling	ビジネスプロセスの名称を指定します。
ステータスの永続化	no	データベースに記録を残すかどうかを指定します。記録を残すと，プロセスの進捗状況などを把握できます。この商品手配システムでは，データベースに記録を残さないため，「no」を選択します。
サービス ID	ArrBPFau	ビジネスプロセスの ID を指定します。
スコープアクティビティのデフォルトハンドラ	OFF	スコープアクティビティのデフォルトハンドラの設定を「ON」にすると，補償処理が暗黙的に定義されるようになります。この商品手配システムでは，補償処理を定義しないので，「OFF」を選択します。

ProductStock_FaultHandling ビジネスプロセスの追加手順を次に示します。

1. ツリービューの［サービス定義一覧］で，ProductStock_Normal ビジネスプロセスを選択して右クリックします。
   サービス一覧のポップアップメニューが表示されます。

2. ポップアップメニューから［複製］を選択します。
   複製が完了したメッセージが表示されます。

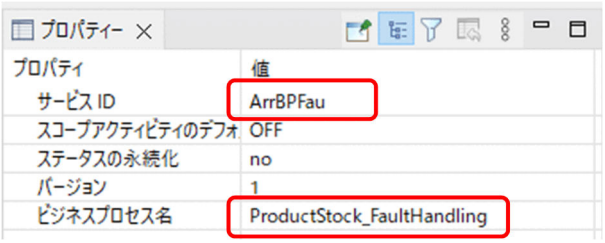
3. ツリービューで「Copy1_ProductStock_Normal」を選択します。
   プロパティビューに Copy1_ProductStock_Normal のプロパティ一覧が表示されます。

4. プロパティビューで，ビジネスプロセス名の値のセルをクリックします。
   入力できる状態になります。

5. 「ProductStock_FaultHandling」に変更し，［Enter］キーを押します。

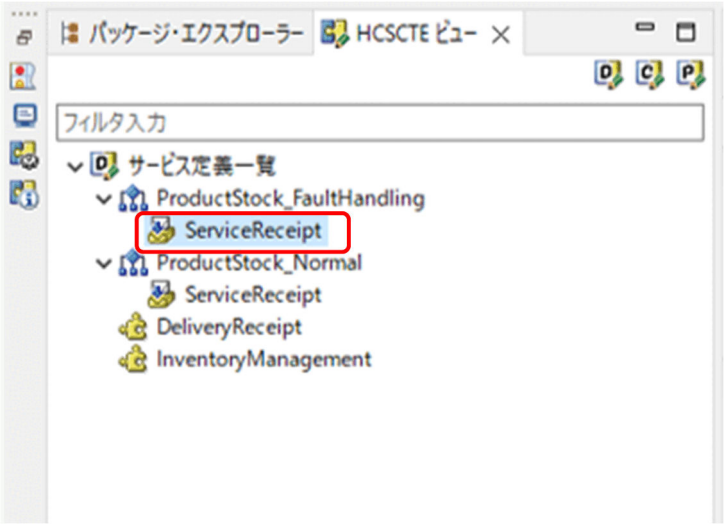
6. 変更してよいかどうかを確認するメッセージが表示されるので，［OK］ボタンをクリックします。

7. 同様にして，サービス ID の値を「ArrBPFau」に変更します。



(2)　ユーザ定義受付の確認

「(1) ビジネスプロセスの複製」では，ProductStock_Normal ビジネスプロセスを複製して ProductStock_FaultHandling ビジネスプロセスを作成しました。ビジネスプロセスを複製すると，そのビジネスプロセスに含まれるユーザ定義受付も複製されます。そのため，ProductStock_FaultHandling ビジネスプロセスには，ProductStock_Normal ビジネスプロセスに含まれていたユーザ定義受付（ServiceReceipt）と同じ内容の受付が含まれています。



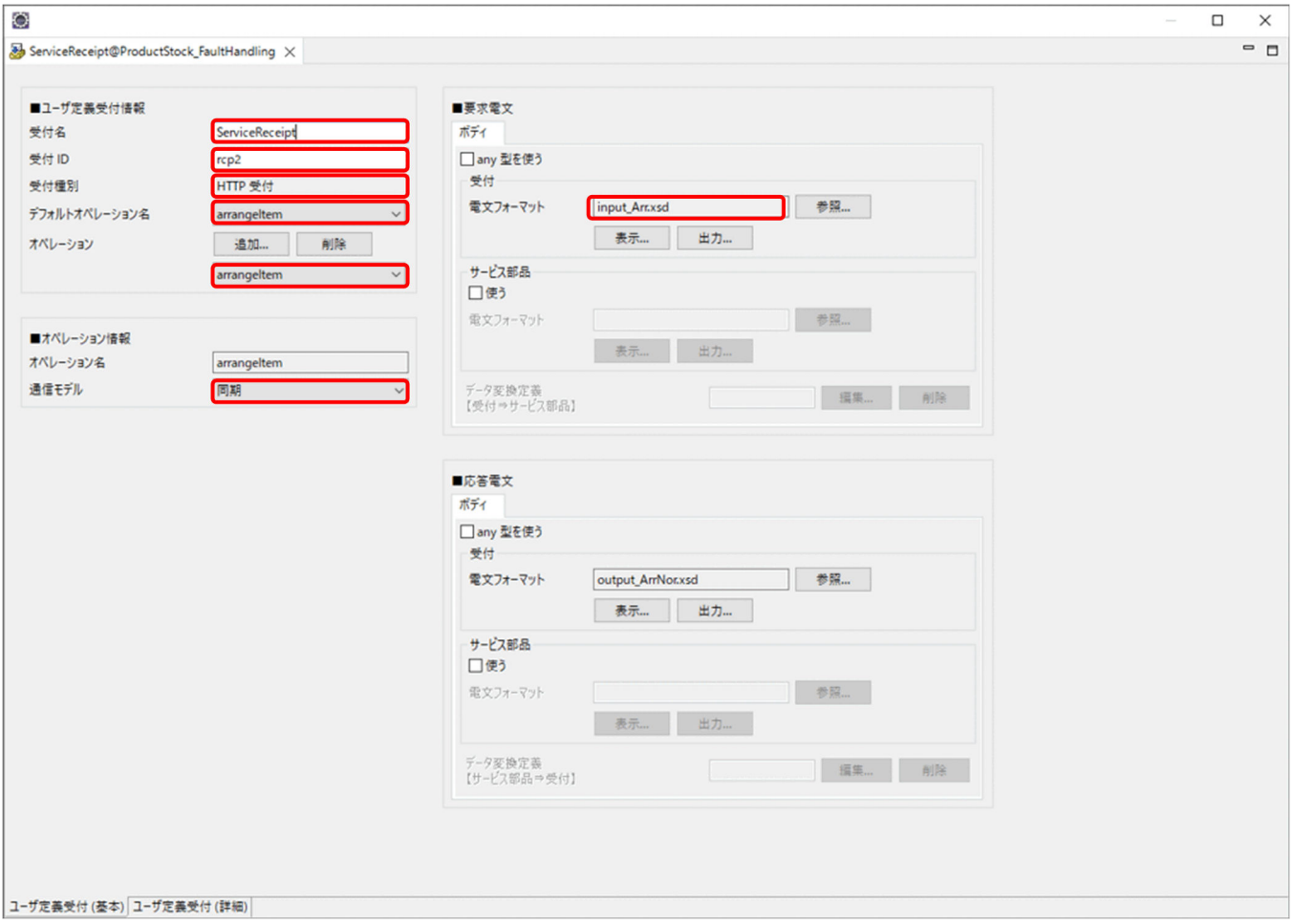
商品手配システム（フォルトハンドリング編）では，ProductStock_FaultHandling ビジネスプロセスに含まれるユーザ定義受付（ServiceReceipt）を一部変更して開発します。変更個所は，応答電文の電文フォーマットだけです。応答電文の電文フォーマット以外の項目は，既存の設定をそのまま利用します。

ここでは，応答電文の電文フォーマットを除き，設定されている値が正しいかどうかを確認します。手順を次に示します。

1. ツリービューの［サービス定義一覧］から，ProductStock_FaultHandling ビジネスプロセス配下の「ServiceReceipt」をダブルクリックします。
   ユーザ定義受付定義（基本）画面が表示されます。

2. 次の値が設定されていることを確認します。

属性名	属性値
受付種別	HTTP 受付
受付名	ServiceReceipt
受付 ID	rcp2
オペレーション	arrangeItem
デフォルトオペレーション名	arrangeItem
通信モデル	同期（初期値）
［要求電文］－［受付］－［電文フォーマット］	input_Arr.xsd



3. [ユーザ定義受付（詳細）] タブでユーザ定義受付定義（詳細）画面に切り替えて，[独自定義ファイル] の「cscurecphttp.properties」を選択します。

4. [編集] ボタンをクリックして HTTP 受付定義ファイルを開き，次の内容が記述されていることを確認します。

確認するパラメタ	設定値	説明
httprecp.switchover.pass-through.mode	true	パススルーモードを設定します。
httprecp.pass-through.parameter-use	false	HTTP リクエストボディを要求電文として直接ビジネスプロセスに送信します。
httprecp.request.switchover.json-transfer.mode	true	リクエスト処理時に JSON-XML 変換を使用します。
httprecp.response.switchover.json-transfer.mode	true	レスポンス処理時に JSON-XML 変換を使用します。

記述例を次に示します。

```
httprecp.switchover.pass-through.mode=true
httprecp.pass-through.parameter-use=false
httprecp.request.switchover.json-transfer.mode=true
httprecp.response.switchover.json-transfer.mode=true
```

(3) 電文フォーマットの作成

ServiceReceipt 受付の応答電文の電文フォーマットを作成し，受付に反映させます。

作成する電文フォーマット（XML スキーマ）は，使用する JSON ファイルから cscjson2xsd コマンドを実行して作成します。

注意

ここで作成する JSON ファイルは，すべて文字コード UTF-8 で作成してください。

(a) 応答電文

応答電文の XML スキーマの作成について説明します。ここで作成した XML スキーマ「output_ArrFau.xsd」は，HTTP 受付の応答電文に設定します。

- cscjson2xsd コマンドで使用する JSON ファイル

  4 つの JSON ファイルを使用します。テキストエディタを起動し，次の 4 つの JSON ファイルをそれぞれ作成します。

  ■output_ArrNor.json（在庫ありレスポンス）

  JSON ファイルの内容を次に示します。この内容は，「5.2.4(3)(b) 応答電文」で作成した JSON ファイルの内容と同じです。

```
{"deliveryNumber": "D00000001", "itemName": "PC", "message": "Product is available for arrangement.", "quantity": 1}
```

  ■output_JavaExc.json（入力値検証例外）

  JSON ファイルの内容を次に示します。この内容は，「5.5.2(1)サービスリクエスタからの JSON 電文の内容が不正である例外（入力値検証例外）」で作成したレスポンス（ボディ）の内容と同じです。

```
{"exception-activity": "入力値検証", "exception-message": "quantityを整数で入力してください"}
```

  ■output_InventoryExc.json（在庫引当例外）

  JSON ファイルの内容を次に示します。この内容は，「5.5.2(2) InventoryManagement アダプタと在庫管理サービスの間の通信が失敗した例外（在庫引当例外）」で作成したレスポンス（ボディ）の内容と同じです。

```
{"exception-activity":"在庫引当","exception-message":"jp.co.Hitachi.soft.csc.msg.adapter.protocol.CSCMsgCustomAccessException: A service call was interrupted because a custom service access error was detected. (adapter name = InvAdp, HCSCCommonID = CSC_HCSC_2024-06-25_17:27:38.029_6, ServiceRequestID = MSG_HCSC_SyncBP_2024-06-25_17:27:38.045_14, error message = An HTTP adapter error occurred. (adapter name = InvAdp, operation name = reserveItem, exception information = jp.co.Hitachi.soft.csc.cstmadp.http.rt.exception.CSAHTAException: KDEC81054-E An error status code was received from the service. (adapter name = InvAdp, operation name = reserveItem, status code = 404)), code = KDEC81499-E) ErrorCode=KDEC03005-E"}
```

  ■output_DeliveryExc.json（配送手配例外）

JSON ファイルの内容を次に示します。この内容は，「5.5.2(3) DeliveryReceipt アダプタと配送受付サービスの間の通信が失敗した例外（配送手配例外）」で作成したレスポンス（ボディ）の内容と同じです。

```
{"exception-activity":"配送手配","exception-message":"jp.co.Hitachi.soft.csc.msg.adapter.
protocol.CSCMsgCustomAccessException: A service call was interrupted because a custom ser
vice access error was detected. (adapter name = DelAdp, HCSCCommonID = CSC_HCSC_2024-06-2
5_17:33:39.366_8, ServiceRequestID = MSG_HCSC_SyncBP_2024-06-25_17:33:39.625_18, error me
ssage = An HTTP adapter error occurred. (adapter name = DelAdp, operation name = deliverI
tem, exception information = jp.co.Hitachi.soft.csc.cstmadp.http.rt.exception.CSAHTAExcep
tion: KDEC81054-E An error status code was received from the service. (adapter name = Del
Adp, operation name = deliverItem, status code = 404)), code = KDEC81499-E) ErrorCode=KDE
C03005-E"}
```

- 作成した JSON ファイルのフォルダ構成

  ここでは，作成した JSON ファイルを次のフォルダに格納するものとします。

```
C:¥Users¥work¥response_FaultHandling
 └──output_ArrNor.json
    output_DeliveryExc.json
    output_InventoryExc.json
    output_JavaExc.json
```

- cscjson2xsd コマンドの実行例

  この操作は，管理者または管理者特権で実行してください。

  この実行例では，出力する XML スキーマとして「output_ArrFau.xsd」を指定しています。

```
C:¥Users¥work¥response_FaultHandling>"%COSMINEXUS_HOME%¥CSCTE¥bin¥cscjson2xsd.bat" -in ./
 -out output_ArrFau.xsd
KECT93001-I Execution of the command will now start. (command = cscjson2xsd)
KECT93002-I Execution of the command ended normally. (command = cscjson2xsd)
```

- 作成された XML スキーマ（output_ArrFau.xsd）

  コマンド実行後，次の内容の XML スキーマ「output_ArrFau.xsd」が作成されます。

  実際に作成される XML スキーマは，次に示す内容と異なる場合がありますが，スキーマとしての定義内容は同じであるため，問題ありません。

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
  <xs:element name="csc-object">
    <xs:complexType>
      <xs:choice>
        <xs:sequence>
          <xs:element ref="exception-activity"/>
          <xs:element ref="exception-message"/>
        </xs:sequence>
        <xs:sequence>
          <xs:element ref="deliveryNumber"/>
          <xs:element ref="itemName"/>
          <xs:element ref="message"/>
          <xs:element ref="quantity"/>
        </xs:sequence>
      </xs:choice>
    </xs:complexType>
```

```
  </xs:element>
  <xs:element name="exception-activity" type="xs:NCName"/>
  <xs:element name="exception-message" type="xs:string"/>
  <xs:element name="deliveryNumber" type="xs:NCName"/>
  <xs:element name="itemName" type="xs:NCName"/>
  <xs:element name="message" type="xs:string"/>
  <xs:element name="quantity" type="xs:integer"/>
</xs:schema>
```

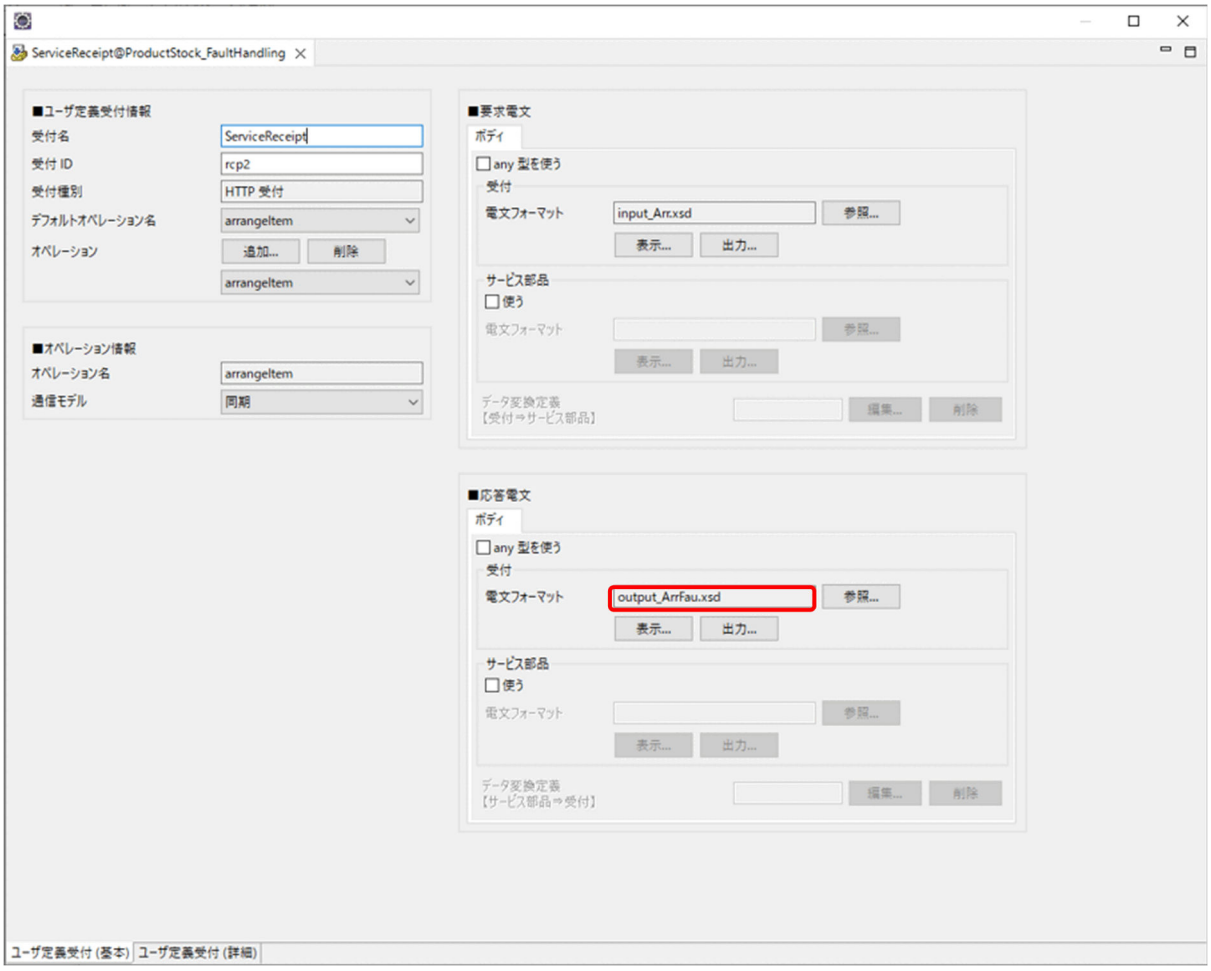
ここで作成したJSONファイル，およびXMLスキーマのサンプルは次のディレクトリに格納されています。

```
<Service Architectのインストールディレクトリ>¥CSCTE¥Samples¥SOAP1.1_1.2mode¥ProductStock_RES
T¥Schema¥ServiceReceipt¥response¥ProductStock_FaultHandling
```

(b) 電文フォーマットの指定

作成したXMLスキーマをServiceReceipt受付に設定します。手順を次に示します。

1. [ユーザ定義受付（基本）] タブでユーザ定義受付定義（基本）画面に切り替えます。
2. [応答電文] － [受付] － [電文フォーマット] の [参照] ボタンをクリックして，XMLスキーマ「output_ArrFau.xsd」を指定します。



3. メニューから［ファイル］－［保管］を選択します。

これで ServiceReceipt 受付の作成が完了しました。

(4) 変数の設定

ビジネスプロセスでは，アクティビティを定義するときに変数を使用します。そのため，使用する変数をあらかじめ定義しておく必要があります。ProductStock_FaultHandling ビジネスプロセスで使用する変数を次の表に示します。

表 5-9 ProductStock_FaultHandling ビジネスプロセスで使用する変数

変数名	種別	説明
InputData	XML	受付アクティビティで受信するビジネスプロセスの要求電文のボディとして使用します。 ServiceReceipt 受付の要求電文フォーマットを取り込み，定義します。
OutputData	XML	応答アクティビティで送信するビジネスプロセスの応答電文のボディとして使用します。 「(3) 電文フォーマットの作成」で作成した ServiceReceipt 受付の応答電文フォーマットを取り込み，定義します。
InventoryAllocationInputData	XML	サービス呼出アクティビティで在庫管理サービスを呼び出す際の要求電文のボディとして使用します。 InventoryManagement サービスアダプタの要求電文フォーマットを取り込み，定義します。
InventoryAllocationOutputData	XML	サービス呼出アクティビティで在庫管理サービスを呼び出す際の応答電文のボディとして使用します。 InventoryManagement サービスアダプタの応答電文フォーマットを取り込み，定義します。
DeliveryArrangementInputData	XML	サービス呼出アクティビティで配送受付サービスを呼び出す際の要求電文のボディとして使用します。 DeliveryReceipt サービスアダプタの要求電文フォーマットを取り込み，定義します。
DeliveryArrangementOutputData	XML	サービス呼出アクティビティで配送受付サービスを呼び出す際の応答電文のボディとして使用します。 DeliveryReceipt サービスアダプタの応答電文フォーマットを取り込み，定義します。
FaultHeader	XML	例外発生時に返却する応答電文のヘッダとして使用します。 Service Architect で用意されている HTTP アダプタ応答電文用スキーマファイルを取り込み，定義します。
FaultJava	XML	Java 呼出アクティビティで例外が発生した際のフォルト電文のボディとして使用します。 Service Architect で用意されている Java 呼出アクティビティのフォルト処理用スキーマファイルを取り込み，定義します。
FaultInvoke	XML	サービス呼出アクティビティで例外が発生した際のフォルト電文のボディとして使用します。

変数名	種別	説明
FaultInvoke	XML	Service Architect で用意されている，サービスアダプタで発生した例外をフォルト電文に変換する設定にした場合に使用するスキーマファイルを取り込み，定義します。

これらの変数は，次の変数を除いて ProductStock_Normal ビジネスプロセスで設定済みです。

- FaultHeader
- FaultJava
- FaultInvoke
- OutputData（ServiceReceipt 受付の応答電文を変更したため，OutputData も更新する必要があります）

ここでは，これら４つの変数を設定します。設定手順を次に示します。

1. ツリービューの［サービス定義一覧］から ProductStock_FaultHandling ビジネスプロセスをダブルクリックします。
   ビジネスプロセス定義画面が表示されます。
2. ビジネスプロセス定義画面のキャンバス上の［変数・相関セット］アイコンをダブルクリックします。
   ［変数・相関セット一覧］ダイアログが表示されます。
3. ［変数一覧］を選択します。変数名に「FaultHeader」を入力し，種別は，ドロップダウンリストから「XML」を選択します。
4. ［参照］ボタンをクリックして次のファイルを取り込みます。
   <Service Architect のインストールディレクトリ>¥CSC¥custom-reception¥http¥schema¥urecp_http_header_response.xsd
5. ［変数・相関セット一覧］ダイアログの［追加］ボタンをクリックします。
   変数一覧に変数「FaultHeader」が追加されます。
6. 「FaultJava」，「FaultInvoke」も，手順 2.～4.と同様の手順で設定します。
   ［変数・相関セット一覧］ダイアログで設定する種別はすべて［XML］です。それぞれの変数で取り込むファイルを次に示します。

   FaultJava
   <Service Architect のインストールディレクトリ>¥CSC¥schema¥fault¥cscinvokejavafault.xsd

   FaultInvoke
   <Service Architect のインストールディレクトリ>¥CSC¥system¥msg¥cscfault.xsd
7. 「OutputData」を更新するため，［変数一覧］から変数名「OutputData」を選択します。
8. ［取込］ボタンをクリックします。

［電文フォーマットの取込］ダイアログが表示されます。

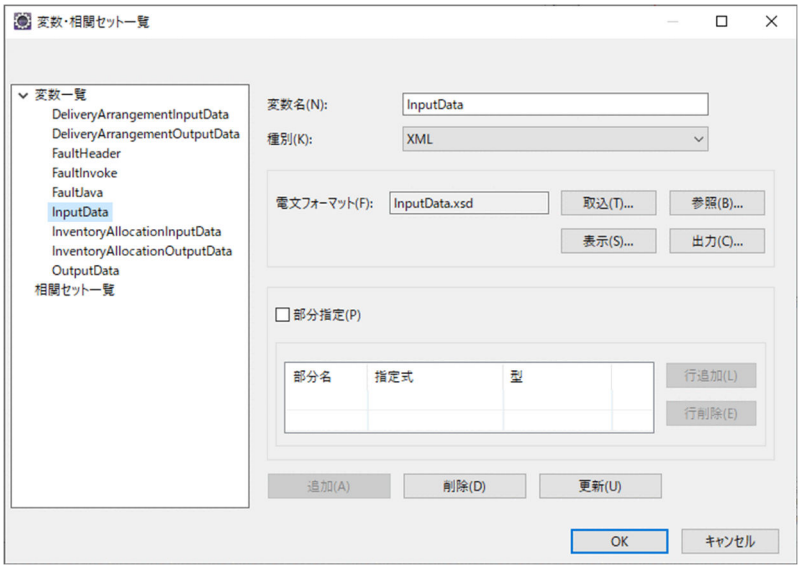
9. ［受付名］を選択して，ドロップダウンリストから「ServiceReceipt」を選択します。

10. ［オペレーション名］はドロップダウンリストから「arrangeItem」を，［電文種別］はドロップダウンリストから「応答電文（ボディ）」を選択します。［電文フォーマット］に「OutputData」を入力します。

11. ［OK］ボタンをクリックします。
［電文フォーマットの取込］ダイアログが閉じます。

12. ［変数・相関セット一覧］ダイアログの［更新］ボタンをクリックします。
変数「OutputData」が更新されます。

13. ［変数・相関セット一覧］ダイアログの［OK］ボタンをクリックします。
これで変数が設定できました。



(5)　アクティビティの配置

(a)　ProductStock_FaultHandling ビジネスプロセスに必要なアクティビティ

ProductStock_FaultHandling ビジネスプロセスに必要なアクティビティを次の表に示します。

表 5-10　ProductStock_FaultHandling ビジネスプロセスに必要なアクティビティ

アクティビティ名	アクティビティ種別	説明
受付	受付アクティビティ	サービスリクエスタからの応答を受け付けます。
在庫引当前処理	データ変換アクティビティ	在庫引当時のデータを編集します。
入力値検証	Java 呼出アクティビティ	入力値を検証します。

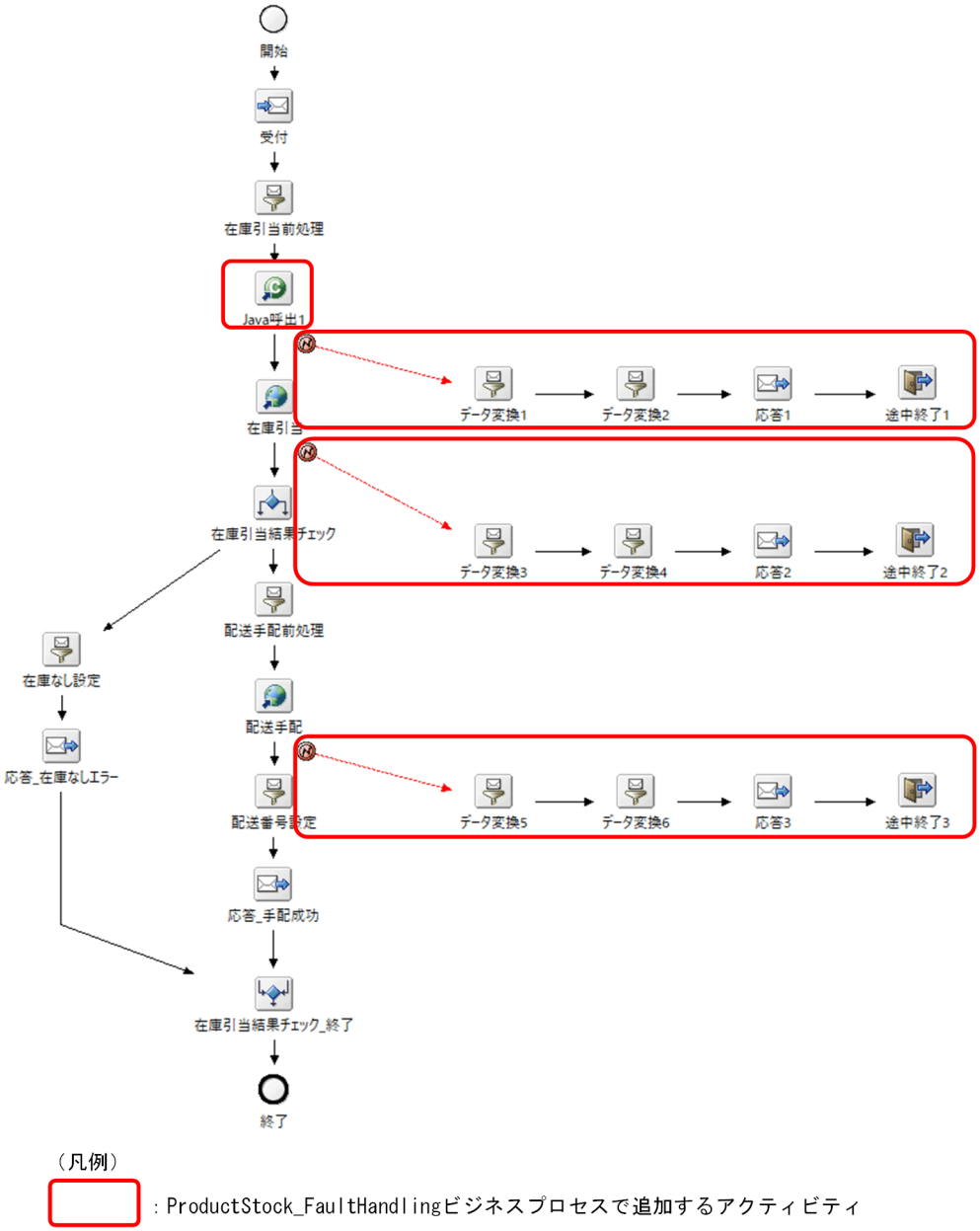
アクティビティ名	アクティビティ種別	説明
入力値検証例外の応答ボディ処理	データ変換アクティビティ	入力値検証例外用レスポンスのボディのデータを編集します。
入力値検証例外の応答ヘッダ処理	データ変換アクティビティ	入力値検証例外用レスポンスのヘッダのデータを編集します。
応答_入力値検証例外	応答アクティビティ	処理結果をサービスリクエスタへ返します。
途中終了_入力値検証例外	途中終了アクティビティ	入力値検証で例外が発生した場合，途中終了します。
在庫引当	サービス呼出アクティビティ	在庫管理サービスを呼び出します。
在庫引当例外の応答ボディ処理	データ変換アクティビティ	在庫引当例外用レスポンスのボディのデータを編集します。
在庫引当例外の応答ヘッダ処理	データ変換アクティビティ	在庫引当例外用レスポンスのヘッダのデータを編集します。
応答_在庫引当例外	応答アクティビティ	処理結果をサービスリクエスタへ返します。
途中終了_在庫引当例外	途中終了アクティビティ	在庫引当で例外が発生した場合，途中終了します。
在庫引当結果チェック	分岐開始アクティビティ	条件（在庫の有無）による処理をします。
在庫なし設定	データ変換アクティビティ	在庫なしレスポンスを返却できるようにデータを編集します。
応答_在庫なしエラー	応答アクティビティ	処理結果をサービスリクエスタへ返します。
配送手配前処理	データ変換アクティビティ	配送手配時のデータを編集します。
配送手配	サービス呼出アクティビティ	配送受付サービスを呼び出します。
配送手配例外の応答ボディ処理	データ変換アクティビティ	配送手配例外用レスポンスのボディのデータを編集します。
配送手配例外の応答ヘッダ処理	データ変換アクティビティ	配送手配例外用レスポンスのヘッダのデータを編集します。
応答_配送手配例外	応答アクティビティ	処理結果をサービスリクエスタへ返します。
途中終了_配送手配例外	途中終了アクティビティ	配送手配で例外が発生した場合，途中終了します。
配送番号設定	データ変換アクティビティ	在庫ありレスポンスを返却できるようにデータを編集します
応答_手配成功	応答アクティビティ	処理結果をサービスリクエスタへ返します。
在庫引当結果チェック_終了	分岐終了アクティビティ	条件（在庫の有無）による処理を終了します。

(b)　アクティビティの配置方法

次の図のフローになるよう，アクティビティを配置します。アクティビティは表 5-10 に記載している順番どおりに，パレットから連結元のアクティビティの近くにドロップしてください。

なお，次に示すアクティビティは［パレット］－［コネクション］－［フォルト］を選択して，連結元アクティビティから連結先アクティビティを連結してください。

連結元アクティビティ	連結先アクティビティ
入力値検証（Java 呼出 1）	入力値検証例外の応答ボディ処理（データ変換 1）
在庫引当	在庫引当例外の応答ボディ処理（データ変換 3）
配送手配	配送手配例外の応答ボディ処理（データ変換 5）



（凡例）

：ProductStock_FaultHandlingビジネスプロセスで追加するアクティビティ

ポイント

- 配置後に，分岐終了アクティビティを終了アクティビティの近くに移動することで，分岐終了アクティビティから終了アクティビティに連結できます。

- アクティビティ配置時に連結できなかった場合は，次の手順で連結してください。

1. アクティビティを連結するために，パレットの［コネクション］をクリックします。
2. 連結元である開始アクティビティをクリックします。
3. 連結先である受付アクティビティをクリックします。
4. 受付アクティビティから終了アクティビティまでを手順 1.～手順 3.と同様の手順で 1 つずつ連結します。

(6) アクティビティの定義

キャンバスへ配置した各アクティビティの内容を定義します。

ProductStock_Nomal ビジネスプロセスで定義済のアクティビティは定義内容が引き継がれています。ここでは，主に ProductStock_FaultHandling ビジネスプロセスで新しく追加するアクティビティの定義について説明します。

(a) 受付アクティビティ（アクティビティ名：受付）

ProductStock_Nomal ビジネスプロセスで定義済のアクティビティです。次のように定義されていることを確認します。

項目名	設定値
アクティビティ名	受付
オペレーション名	arrangeItem
ボディ割当変数	InputData
ヘッダ割当変数	設定なし
割当相関セット群	設定なし
通信モデル	同期
インスタンス生成	yes

各設定項目の詳細については，「5.2.4(6)(a) 受付アクティビティ（アクティビティ名：受付）」を参照してください。

(b) データ変換アクティビティ（アクティビティ名：在庫引当前処理）

ProductStock_Nomal ビジネスプロセスで定義済のアクティビティです。次のように定義されていることを確認します。

項目名	設定値
アクティビティ名	在庫引当前処理

項目名	設定値
変数（変換元変数）	InputData
変数（変換先変数）	InventoryAllocationInputData
データ変換定義	StockAllocationPre-processing

また，データ変換のマッピング定義を確認します。

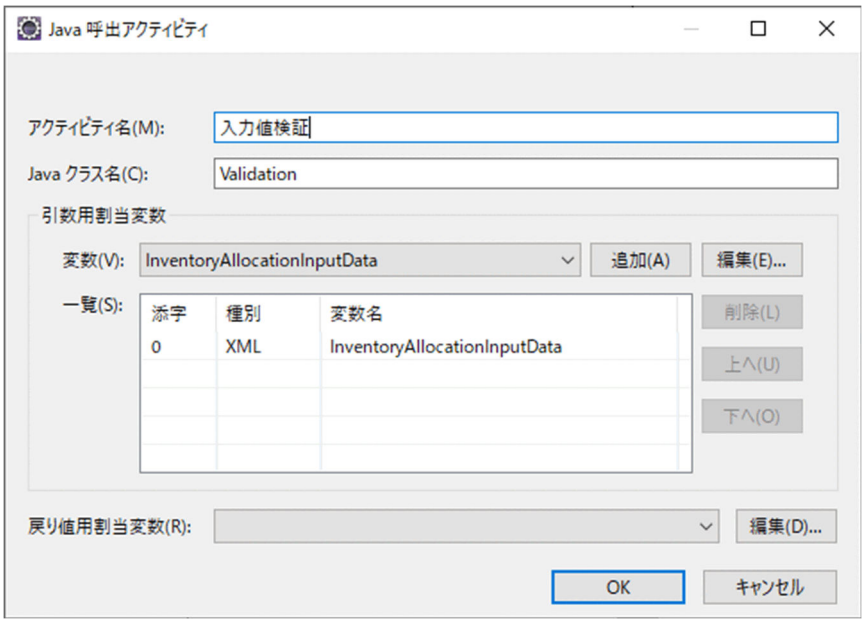
各設定項目の詳細については，「5.2.4(6)(b)　データ変換アクティビティ（アクティビティ名：在庫引当前処理）」を参照してください。

(c)　Java 呼出アクティビティ（アクティビティ名：入力値検証）

ProductStock_FaultHandling ビジネスプロセスで新しく追加したアクティビティです。次の手順で定義します。

1. キャンバスの Java 呼出アクティビティ（Java 呼出 1）をダブルクリックします。
   [Java 呼出アクティビティ] ダイアログが表示されます。

2. 次の内容を入力します。



項目名	設定する値	説明
アクティビティ名	入力値検証	アクティビティの名称を入力します。
Java クラス名	Validation	Java クラス名を入力します。
引数用割当変数	添字：0 種別：XML 変数名：InventoryAllocationInputData	[Java クラス名] で指定した Java クラスを呼び出すときの引数に割り当てる変数を指定します。
戻り値用割当変数	なし	[Java クラス名] で指定した Java クラスを呼び出すときの戻り値に割り当てる変数を選択し

項目名	設定する値	説明
戻り値用割当変数	なし	ます。この商品手配システムでは使用しないため，設定しません。

3. [OK] ボタンをクリックします。

4. キャンバスの Java 呼出アクティビティをダブルクリックするか，または右クリックして［Java エディタ起動］を選択します。

   Java クラス Validation を作成するかを確認するメッセージが表示されるので，「はい」をクリックします。

   Eclipse の Java エディタが起動されます。Java エディタには，［Java 呼出アクティビティ］ダイアログで指定したクラスのソースコードが表示されます。

5. Java クラスのソースコードを Java エディタで編集します。

   次のディレクトリに格納されている Java ソースファイル（Validation.java）をテキストエディタで開き，Java ソースコードをすべてコピーして，Java エディタで開いた Validation.java にペーストします。これにより，ディレクトリに格納されている Validation.java の内容を Java 呼出アクティビティのソースコードに反映させます。

   ```
   <Service Architectのインストールディレクトリ>¥CSCTE¥Samples¥SOAP1.1_1.2mode¥ProductStock_REST¥Repository¥Validation.java
   ```

6. 編集したソースコードを保存して，Java エディタを終了します。

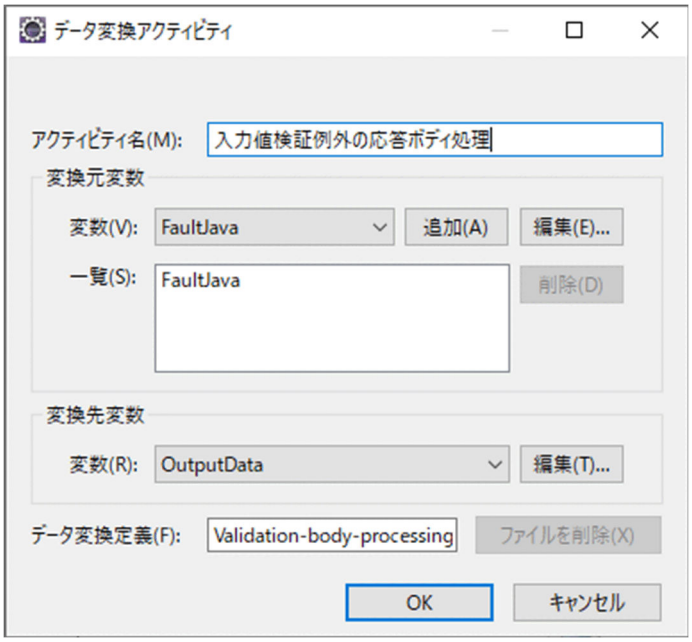
(d) データ変換アクティビティ（アクティビティ名：入力値検証例外の応答ボディ処理）

ProductStock_FaultHandling ビジネスプロセスで新しく追加したアクティビティです。次の手順で定義します。

1. キャンバスのデータ変換アクティビティ（データ変換 1）をダブルクリックします。

   ［データ変換アクティビティ］ダイアログが表示されます。

2. 次の内容を入力します。



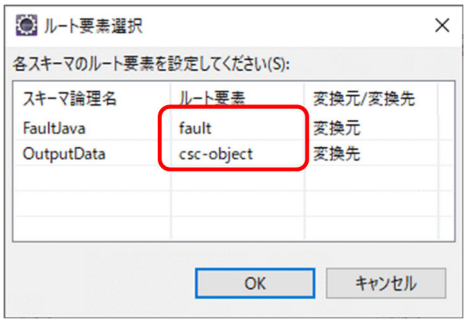
項目名	設定する値	説明
アクティビティ名	入力値検証例外の応答ボディ処理	アクティビティの名称を入力します。
変数（変換元変数）	FaultJava	データの変換元になる変数をドロップダウンリストから選択し，［追加］ボタンをクリックします。
変数（変換先変数）	OutputData	データの変換先になる変数をドロップダウンリストから選択します。
データ変換定義	Validation-body-processing	変数の変換に使うデータ変換定義ファイルの名称を入力します。

3. ［OK］ボタンをクリックします。

4. キャンバスのデータ変換アクティビティを右クリックして，［マッピング定義起動］を選択します。
   ［ルート要素選択］ダイアログが表示されます。

5. 変換元のスキーマ論理名「FaultJava」のルート要素をクリックして，ドロップダウンリストから「fault」を選択します。

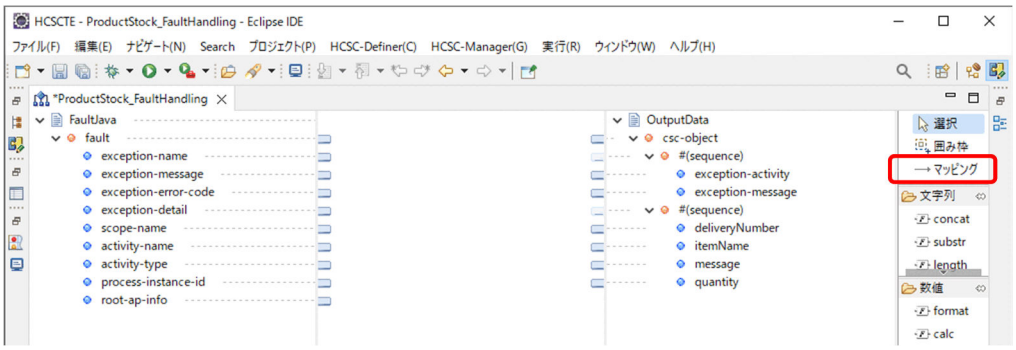
6. 変換先のスキーマ論理名「OutputData」のルート要素をクリックして，ドロップダウンリストから「csc-object」を選択します。



7. [OK] ボタンをクリックします。

   データ変換定義画面が表示されます。

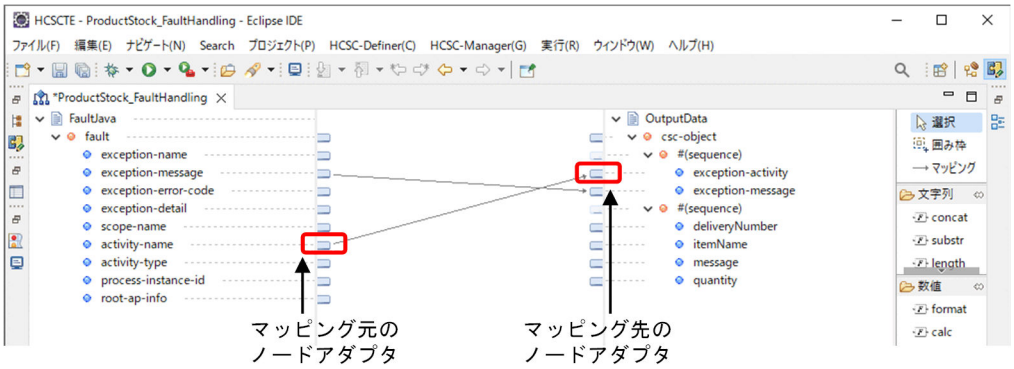
8. データ変換定義画面のパレットから［マッピング］を選択します。



9. マッピング元となる変換元ノードのノードアダプタをクリックします。

10. マッピング先となる変換先ノードのノードアダプタをクリックします。

    マッピング線が設定されます。マッピング元のノードアダプタとマッピング先のノードアダプタとの対応は次のとおりです。



    注

    作成した XML スキーマによって変換先スキーマの要素の出現順が一部異なるため，マッピング線の見え方が異なる場合があります。

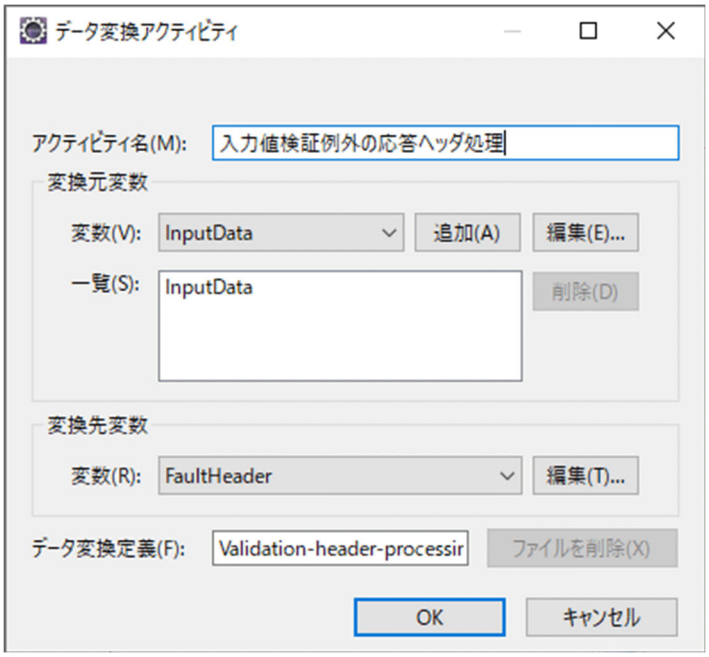
(e)　データ変換アクティビティ（アクティビティ名：入力値検証例外の応答ヘッダ処理）

ProductStock_FaultHandling ビジネスプロセスで新しく追加したアクティビティです。次の手順で定義します。

1. キャンバスのデータ変換アクティビティ（データ変換 2）をダブルクリックします。

   ［データ変換アクティビティ］ダイアログが表示されます。

2. 次の内容を入力します。



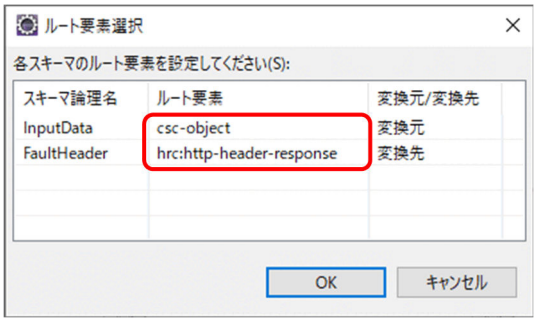
項目名	設定する値	説明
アクティビティ名	入力値検証例外の応答ヘッダ処理	アクティビティの名称を入力します。
変数（変換元変数）	InputData	データの変換元になる変数をドロップダウンリストから選択し，［追加］ボタンをクリックします。
変数（変換先変数）	FaultHeader	データの変換先になる変数をドロップダウンリストから選択します。
データ変換定義	Validation-header-processing	変数の変換に使うデータ変換定義ファイルの名称を入力します。

3. ［OK］ボタンをクリックします。

4. キャンバスのデータ変換アクティビティを右クリックして，［マッピング定義起動］を選択します。
   ［ルート要素選択］ダイアログが表示されます。

5. 変換元のスキーマ論理名「InputData」のルート要素をクリックして，ドロップダウンリストから「csc-object」を選択します。

6. 変換先のスキーマ論理名「FaultHeader」のルート要素をクリックして，ドロップダウンリストから「hrc:http-header-response」を選択します。



7. [OK] ボタンをクリックします。
   データ変換定義画面が表示されます。

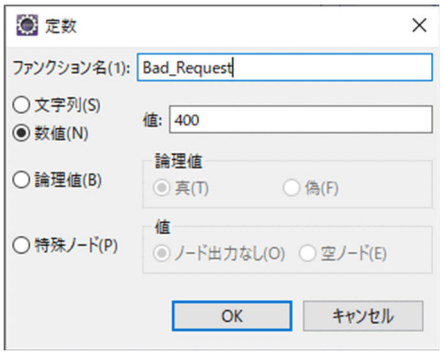
8. データ変換定義画面のパレット［その他］から定数ファンクション（const）を選択します。

9. マッピングビューアで適当な場所をクリックして定数ファンクションを配置します。

10. 定数ファンクションをダブルクリックします。
    ［定数］ダイアログが表示されます。

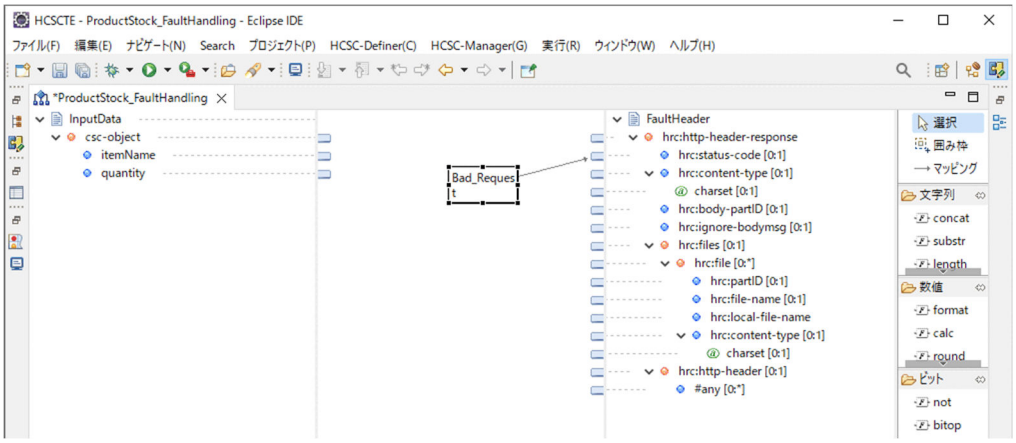
11. [定数] ダイアログで次の値を指定します。

属性名	属性値
ファンクション名	Bad_Request
数値	400



12. [OK] ボタンをクリックします。

13. 次の図のようになるようにマッピング線を設定します。



注

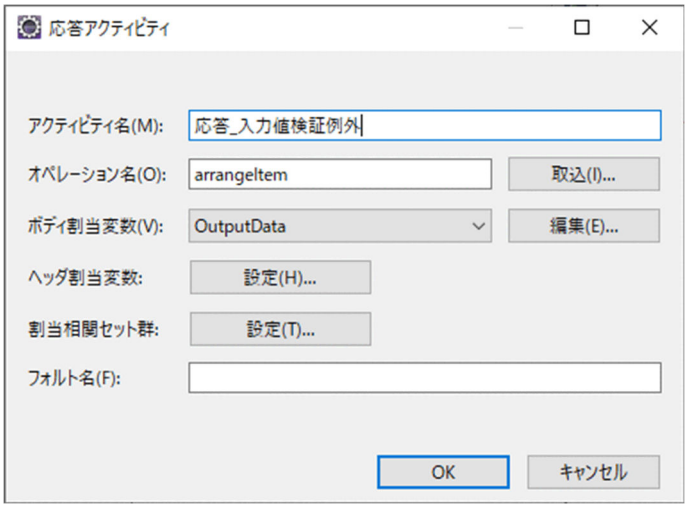
作成した XML スキーマによって変換先スキーマの要素の出現順が一部異なるため，マッピング線の見え方が異なる場合があります。

(f)　応答アクティビティ（アクティビティ名：応答_入力値検証例外）

ProductStock_FaultHandling ビジネスプロセスで新しく追加したアクティビティです。次の手順で定義します。

1. キャンバスの応答アクティビティ（応答 1）をダブルクリックします。
   ［応答アクティビティ］ダイアログが表示されます。

2. 次の内容を入力します。



項目名	設定する値	説明
アクティビティ名	応答_入力値検証例外	アクティビティの名称を入力します。
オペレーション名	arrangeItem	対応する受付アクティビティに指定したオペレーション名を入力します。
ボディ割当変数	OutputData	ビジネスプロセスの応答電文のボディに割り当てる変数をドロップダウンリストから選択します。
ヘッダ割当変数	割当変数：FaultHeader ルート要素：http-header-response 名前空間： http://www.hitachi.co.jp/soft/xml/cosminexus/csc/reception/http/response	ビジネスプロセスの応答電文のヘッダに変数を割り当てる場合に設定します。 ヘッダ割当変数の［設定］ボタンをクリックし，［ヘッダ割当変数］ダイアログで［追加］ボタンをクリックして，［割当変数］［ルート要素］［名前空間］を設定してください。
割当相関セット群	設定なし	相関セットグループをアクティビティに割り当てる場合に入力します。この商品手配システムでは使用しないため，設定しません。
フォルト名	設定なし	フォルト処理として応答アクティビティを定義して，サービスリクエスタにフォルトが発生したことを示す応答電文を送信する場合のフォルトの名称を入力します。この商品手配システムではフォルト処理を使用しないため，設定しません。

3. ［OK］ボタンをクリックします。

(g)　途中終了アクティビティ（アクティビティ名：途中終了_入力値検証例外）

ProductStock_FaultHandling ビジネスプロセスで新しく追加したアクティビティです。次の手順で定義します。

1. キャンバスの途中終了アクティビティ（途中終了 1）をダブルクリックします。
   ［途中終了］ダイアログが表示されます。

2. ［途中終了アクティビティ］ダイアログにアクティビティ名を入力します。
   アクティビティ名「途中終了_入力値検証例外」を入力します。

3. ［OK］ボタンをクリックします。

(h)　サービス呼出アクティビティ（アクティビティ名：在庫引当）

ProductStock_Nomal ビジネスプロセスで定義済のアクティビティです。次のように定義されていることを確認します。

項目名	設定する値
アクティビティ名	在庫引当
サービス名	InventoryManagement
オペレーション名	reserveItem
通信モデル	同期
ボディ割当変数（要求電文）	InventoryAllocationInputData
ヘッダ割当変数（要求電文）	設定なし
ボディ割当変数（応答電文）	InventoryAllocationOutputData
ヘッダ割当変数（応答電文）	設定なし
割当相関セット群	設定なし

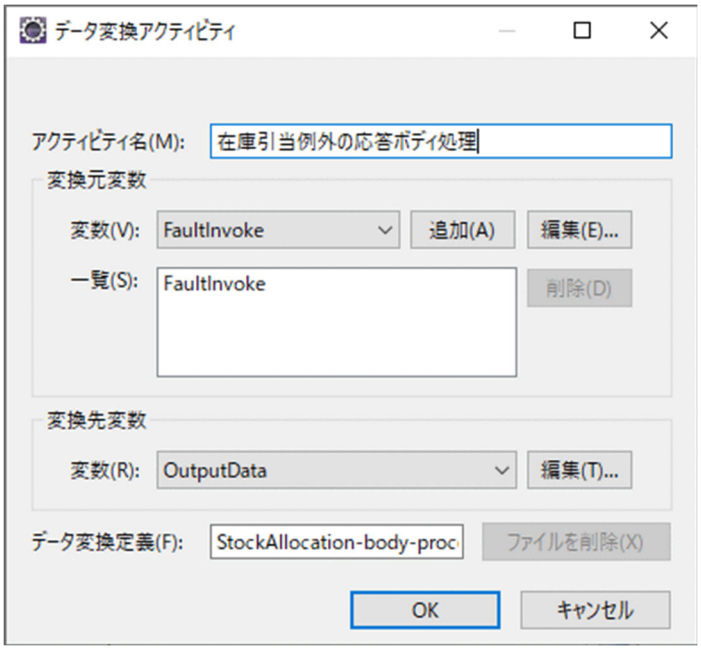
各設定項目の詳細については，「5.2.4(6)(c)　サービス呼出アクティビティ（アクティビティ名：在庫引当）」を参照してください。

(i)　データ変換アクティビティ（アクティビティ名：在庫引当例外の応答ボディ処理）

ProductStock_FaultHandling ビジネスプロセスで新しく追加したアクティビティです。次の手順で定義します。

1. キャンバスのデータ変換アクティビティ（データ変換 3）をダブルクリックします。
   ［データ変換アクティビティ］ダイアログが表示されます。

2. 次の内容を入力します。



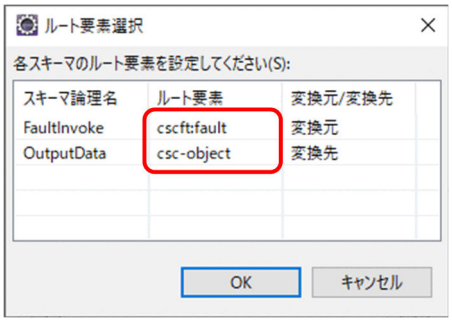
項目名	設定する値	説明
アクティビティ名	在庫引当例外の応答ボディ処理	アクティビティの名称を入力します。
変数（変換元変数）	FaultInvoke	データの変換元になる変数をドロップダウンリストから選択し，［追加］ボタンをクリックします。
変数（変換先変数）	OutputData	データの変換先になる変数をドロップダウンリストから選択します。
データ変換定義	StockAllocation-body-processing	変数の変換に使うデータ変換定義ファイルの名称を入力します。

3. ［OK］ボタンをクリックします。

4. キャンバスのデータ変換アクティビティを右クリックして，［マッピング定義起動］を選択します。
   ［ルート要素選択］ダイアログが表示されます。

5. 変換元のスキーマ論理名「FaultInvoke」のルート要素をクリックして，ドロップダウンリストから「cscft:fault」を選択します。

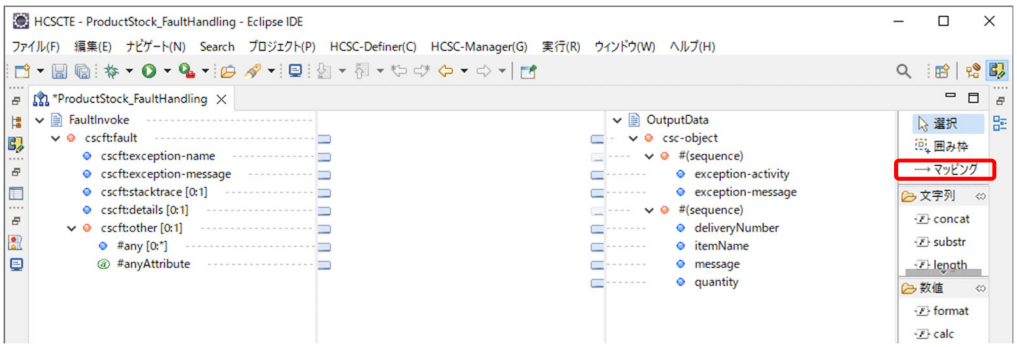
6. 変換先のスキーマ論理名「OutputData」のルート要素をクリックして，ドロップダウンリストから「csc-object」を選択します。



7. [OK] ボタンをクリックします。

   データ変換定義画面が表示されます。

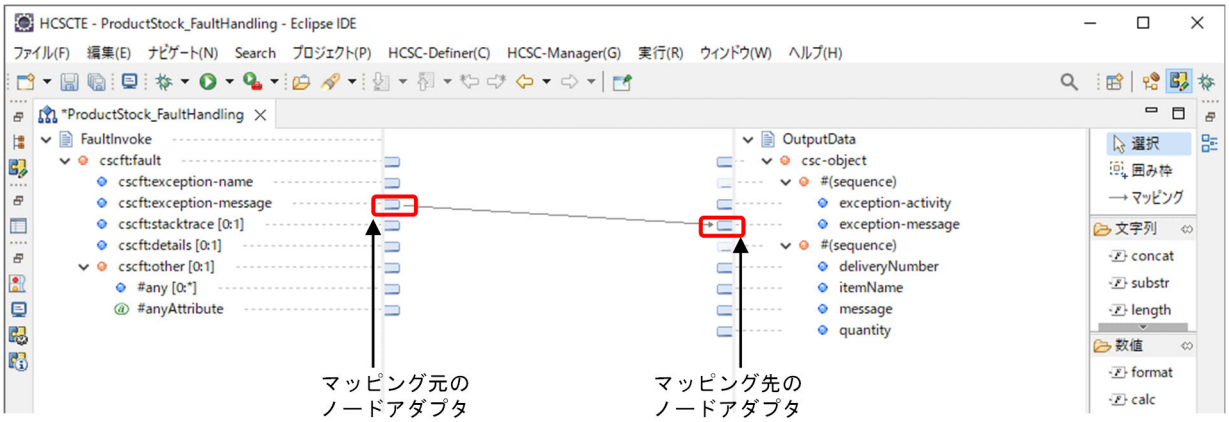
8. データ変換定義画面のパレットから［マッピング］を選択します。



9. マッピング元となる変換元ノードのノードアダプタをクリックします。

10. マッピング先となる変換先ノードのノードアダプタをクリックします。

    マッピング線が設定されます。マッピング元のノードアダプタとマッピング先のノードアダプタとの対応は次のとおりです。



    注

    作成した XML スキーマによって変換先スキーマの要素の出現順が一部異なるため，マッピング線の見え方が異なる場合があります。

11. データ変換定義画面のパレット［その他］から定数ファンクション（const）を選択します。

12. マッピングビューアで適当な場所をクリックして定数ファンクションを配置します。

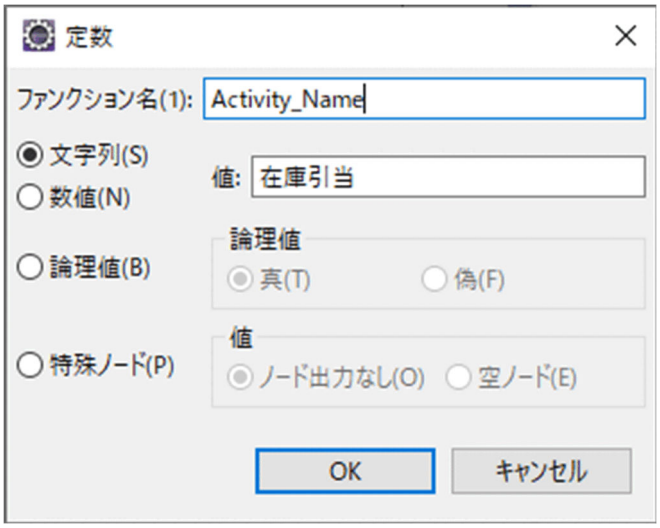
13. 定数ファンクションをダブルクリックします。

    ［定数］ダイアログが表示されます。

14. ［定数］ダイアログで次の値を指定します。

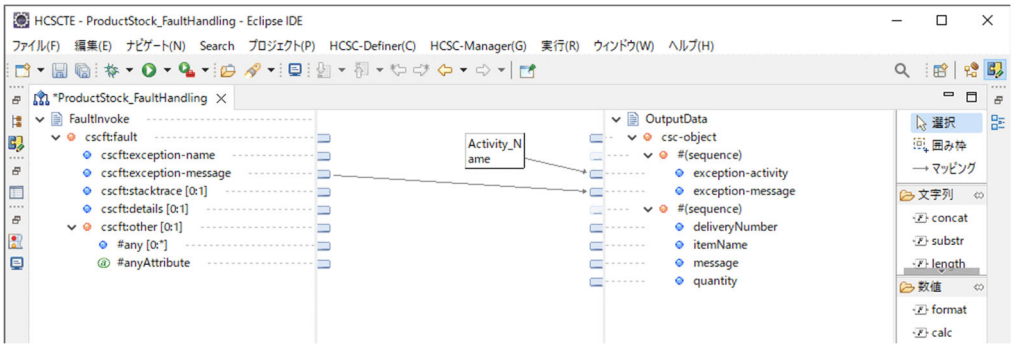
属性名	属性値
ファンクション名	Activity_Name

属性名	属性値
文字列	在庫引当



15. [OK] ボタンをクリックします。

16. 次の図のようになるようにマッピング線を設定します。

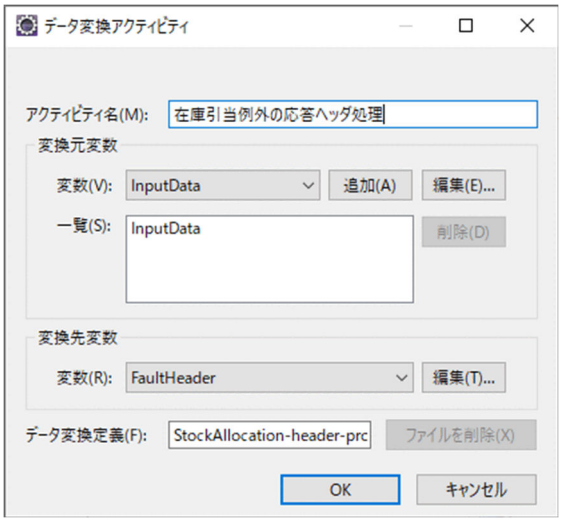


(j) データ変換アクティビティ（アクティビティ名：在庫引当例外の応答ヘッダ処理）

ProductStock_FaultHandling ビジネスプロセスで新しく追加したアクティビティです。次の手順で定義します。

1. キャンバスのデータ変換アクティビティ（データ変換 4）をダブルクリックします。
   [データ変換アクティビティ] ダイアログが表示されます。

2. 次の内容を入力します。



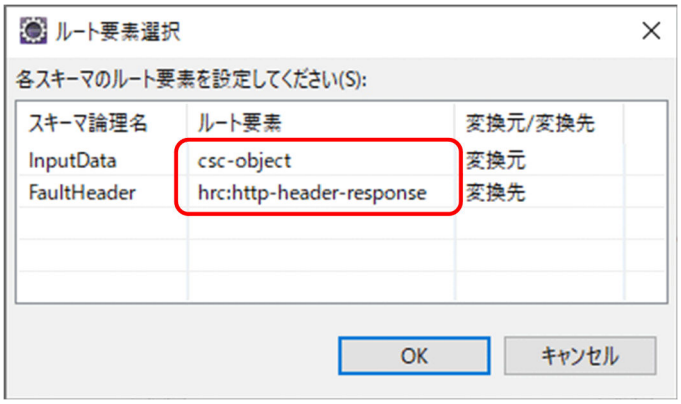
項目名	設定する値	説明
アクティビティ名	在庫引当例外の応答ヘッダ処理	アクティビティの名称を入力します。
変数（変換元変数）	InputData	データの変換元になる変数をドロップダウンリストから選択し，［追加］ボタンをクリックします。
変数（変換先変数）	FaultHeader	データの変換先になる変数をドロップダウンリストから選択します。
データ変換定義	StockAllocation-header-processing	変数の変換に使うデータ変換定義ファイルの名称を入力します。

3. ［OK］ボタンをクリックします。

4. キャンバスのデータ変換アクティビティを右クリックして，［マッピング定義起動］を選択します。
   ［ルート要素選択］ダイアログが表示されます。

5. 変換元のスキーマ論理名「InputData」のルート要素をクリックして，ドロップダウンリストから「csc-object」を選択します。

6. 変換先のスキーマ論理名「FaultHeader」のルート要素をクリックして，ドロップダウンリストから「hrc:http-header-response」を選択します。



7. ［OK］ボタンをクリックします。

データ変換定義画面が表示されます。

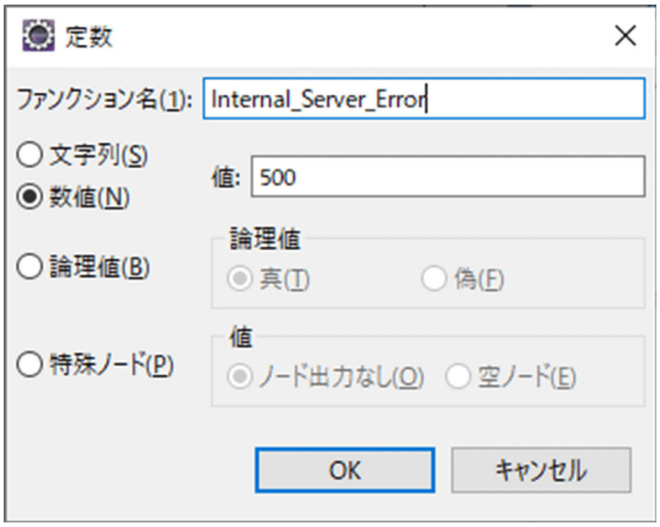
8. データ変換定義画面のパレット［その他］から定数ファンクション（const）を選択します。

9. マッピングビューアで適当な場所をクリックして定数ファンクションを配置します。

10. 定数ファンクションをダブルクリックします。
［定数］ダイアログが表示されます。

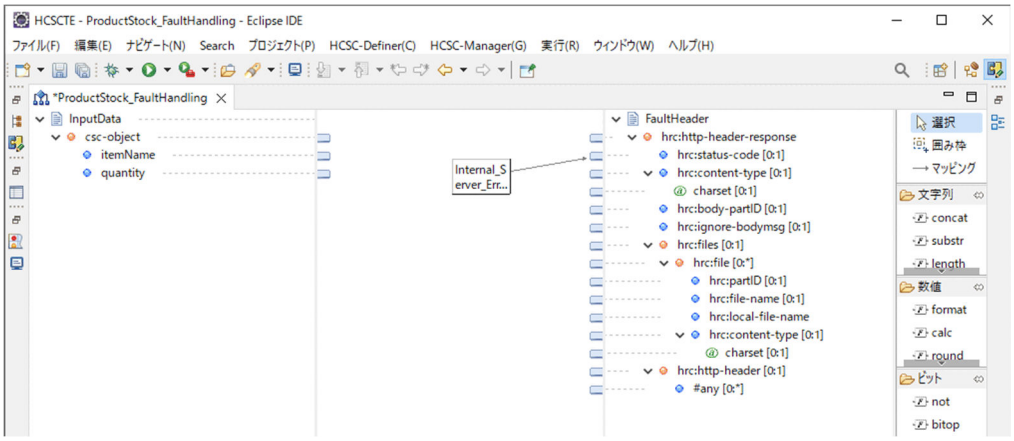
11. ［定数］ダイアログで次の値を指定します。

属性名	属性値
ファンクション名	Internal_Server_Error
数値	500



12. ［OK］ボタンをクリックします。

13. 次の図のようになるようにマッピング線を設定します。



注

作成した XML スキーマによって変換先スキーマの要素の出現順が一部異なるため，マッピング線の見え方が異なる場合があります。

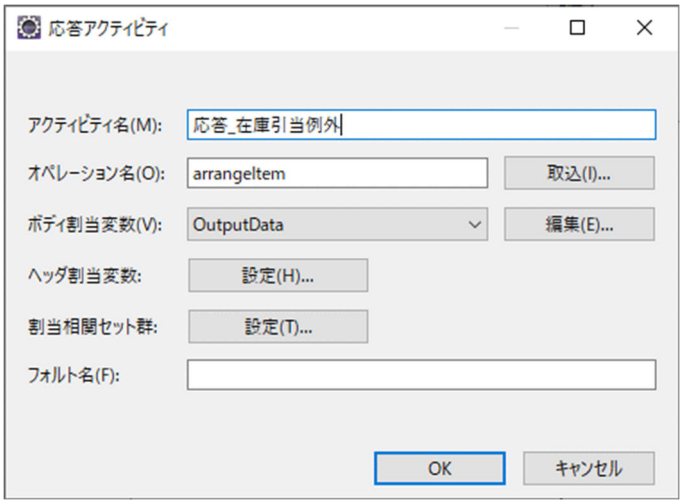
(k) 応答アクティビティ（アクティビティ名：応答_在庫引当例外）

ProductStock_FaultHandling ビジネスプロセスで新しく追加したアクティビティです。次の手順で定義します。

1. キャンバスの応答アクティビティ（応答 2）をダブルクリックします。

［応答アクティビティ］ダイアログが表示されます。

2. 次の内容を入力します。



項目名	設定する値	説明
アクティビティ名	応答_在庫引当例外	アクティビティの名称を入力します。
オペレーション名	arrangeItem	対応する受付アクティビティに指定したオペレーション名を入力します。
ボディ割当変数	OutputData	ビジネスプロセスの応答電文のボディに割り当てる変数をドロップダウンリストから選択します。
ヘッダ割当変数	割当変数：FaultHeader ルート要素：http-header-response 名前空間： http://www.hitachi.co.jp/soft/xml/cosminexus/csc/reception/http/response	ビジネスプロセスの応答電文のヘッダに変数を割り当てる場合に設定します。 ヘッダ割当変数の［設定］ボタンをクリックし，［ヘッダ割当変数］ダイアログで［追加］ボタンをクリックして，［割当変数］［ルート要素］［名前空間］を設定してください。
割当相関セット群	設定なし	相関セットグループをアクティビティに割り当てる場合に入力します。この商品手配システムでは使用しないため，設定しません。
フォルト名	設定なし	フォルト処理として応答アクティビティを定義して，サービスリクエスタにフォルトが発生したことを示す

項目名	設定する値	説明
フォルト名	設定なし	応答電文を送信する場合のフォルトの名称を入力します。この商品手配システムではフォルト処理を使用しないため，設定しません。

3. [OK] ボタンをクリックします。

(l) 途中終了アクティビティ（アクティビティ名：途中終了_在庫引当例外）

ProductStock_FaultHandling ビジネスプロセスで新しく追加したアクティビティです。次の手順で定義します。

1. キャンバスの途中終了アクティビティ（途中終了 2）をダブルクリックします。
   ［途中終了アクティビティ］ダイアログが表示されます。
2. [途中終了アクティビティ] ダイアログにアクティビティ名を入力します。
   アクティビティ名「途中終了_在庫引当例外」を入力します。
3. [OK] ボタンをクリックします。

(m) 分岐開始アクティビティ（アクティビティ名：在庫引当結果チェック）

ProductStock_Nomal ビジネスプロセスで定義済のアクティビティです。［条件設定］ダイアログで次のように定義されていることを確認します。

項目名	設定する値	説明
条件名	No Stock	在庫がない場合の条件の名称を指定します。
変数の表示	なし	このドロップダウンリストは，条件に変数を利用した式を設定する場合に一時的に使用するものです。ここでは何も設定しません。
式※	csc:getVariableData("InventoryAllocationOutputData", "/csc-object/reservationNumber")="*"	在庫がない場合の条件式を XPath 式で指定します。

注※
「式」の「設定する値」はテキストボックスの幅で折り返して表示されます。

また，［分岐アクティビティ］ダイアログで，［遷移先］が［配送手配前処理］の［優先順位］が「Default」であることを確認します。

各設定項目の詳細については，「5.2.4(6)(j)　分岐開始アクティビティ（アクティビティ名：在庫引当結果チェック）」を参照してください。

(n)　データ変換アクティビティ（アクティビティ名：在庫なし設定）

ProductStock_Nomal ビジネスプロセスで定義済のアクティビティです。ProductStock_FaultHandling ビジネスプロセスでは，変換先変数の OutputData の電文フォーマットが変更されているため，データ変換のマッピングを再定義します。

1. キャンバスのデータ変換アクティビティ（在庫なし設定）を右クリックして，［設定］を選択します。

   ［データ変換アクティビティ］ダイアログが表示されます。

2. 次のように定義されていることを確認します。

項目名	設定する値
アクティビティ名	在庫なし設定
変数（変換元変数）	InputData
変数（変換先変数）	OutputData
データ変換定義	NoStockArrangement

   各設定項目の詳細については，「5.2.4(6)(d)　データ変換アクティビティ（アクティビティ名：在庫なし設定）」を参照してください。

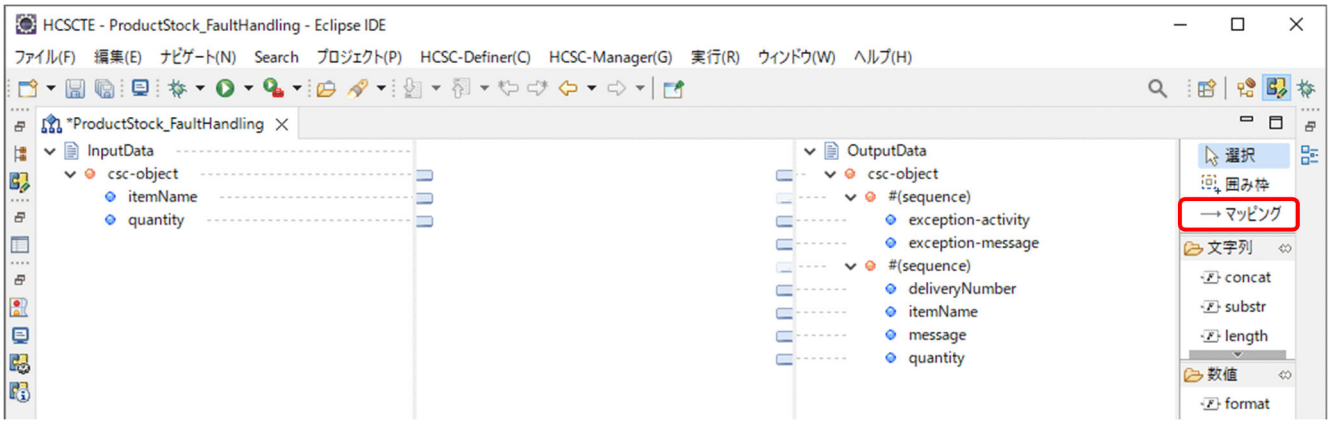
3. ［OK］ボタンをクリックします。

4. キャンバスのデータ変換アクティビティを右クリックして，［マッピング定義起動］を選択します。

   電文フォーマットの更新を反映させるかどうかを確認するメッセージが表示されるので，［OK］ボタンをクリックします。

   データ変換定義画面が表示されます。

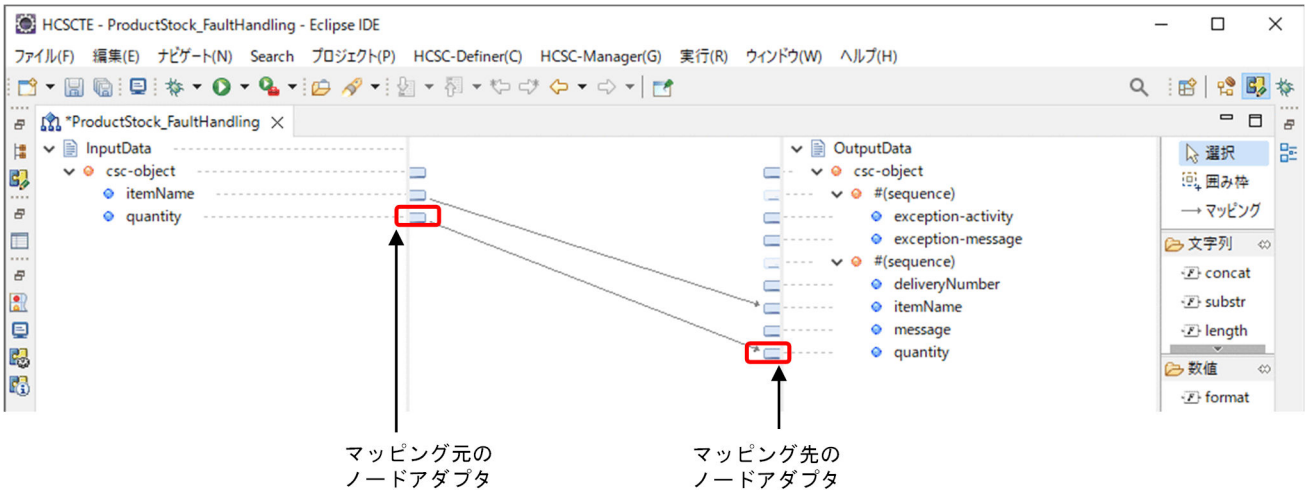
5. データ変換定義画面のパレットから［マッピング］を選択します。



6. マッピング元となる変換元ノードのノードアダプタをクリックします。

7. マッピング先となる変換先ノードのノードアダプタをクリックします。

   マッピング線が設定されます。マッピング元のノードアダプタとマッピング先のノードアダプタとの対応は次のとおりです。



注

作成した XML スキーマによって変換先スキーマの要素の出現順が一部異なるため，マッピング線の見え方が異なる場合があります。

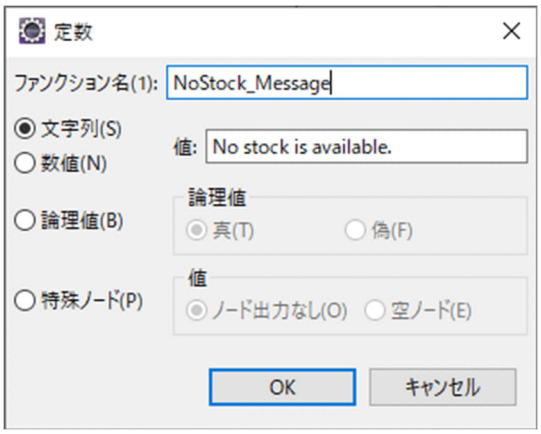
8. データ変換定義画面のパレット［その他］から定数ファンクション（const）を選択します。

9. マッピングビューアで適当な場所をクリックして定数ファンクションを配置します。

10. 定数ファンクションをダブルクリックします。
［定数］ダイアログが表示されます。

11. ［定数］ダイアログで次の値を指定します。

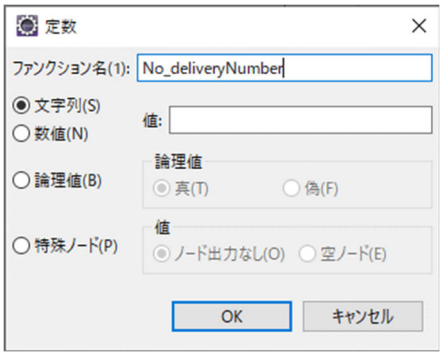
属性名	属性値
ファンクション名	NoStock_Message
文字列	No stock is available.



12. ［OK］ボタンをクリックして［定数］ダイアログを閉じます。

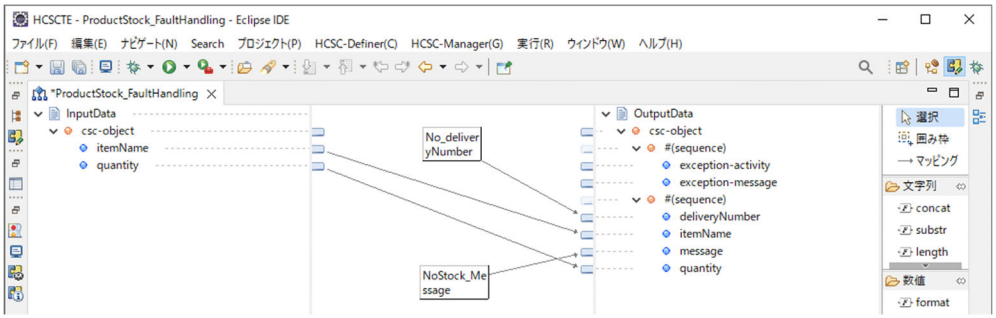
13. 手順 8.～手順 12.と同様の手順で，もう一つ定数ファンクション（const）を作成します。［定数］ダイアログでは次の値を指定します。

属性名	属性値
ファンクション名	No_deliveryNumber
文字列	（何も入力しない）



14. [OK] ボタンをクリックします。

15. 次の図のようになるようにマッピング線を設定します。



(o) 応答アクティビティ（アクティビティ名：応答_在庫なしエラー）

ProductStock_Nomal ビジネスプロセスで定義済のアクティビティです。次のように定義されていることを確認します。

項目名	設定する値
アクティビティ名	応答_在庫なしエラー
オペレーション名	arrangeItem
ボディ割当変数	OutputData
ヘッダ割当変数	設定なし
割当相関セット群	設定なし
フォルト名	設定なし

各設定項目の詳細については，「5.2.4(6)(e)　応答アクティビティ（アクティビティ名：応答_在庫なしエラー）」を参照してください。

(p)　データ変換アクティビティ（アクティビティ名：配送手配前処理）

ProductStock_Nomal ビジネスプロセスで定義済のアクティビティです。次のように定義されていることを確認します。

項目名	設定する値
アクティビティ名	配送手配前処理
変数（変換元変数）	InventoryAllocationOutputData
変数（変換先変数）	DeliveryArrangementInputData
データ変換定義	DeliveryArrangementPre-processing

また，データ変換のマッピング定義を確認します。

各設定項目の詳細については，「5.2.4(6)(f)　データ変換アクティビティ（アクティビティ名：配送手配前処理）」を参照してください。

(q)　サービス呼出アクティビティ（アクティビティ名：配送手配）

ProductStock_Nomal ビジネスプロセスで定義済のアクティビティです。次のように定義されていることを確認します。

項目名	設定する値
アクティビティ名	配送手配
サービス名	DeliveryReceipt
オペレーション名	deliverItem
通信モデル	同期
ボディ割当変数（要求電文）	DeliveryArrangementInputData
ヘッダ割当変数（要求電文）	設定なし
ボディ割当変数（応答電文）	DeliveryArrangementOutputData
ヘッダ割当変数（応答電文）	設定なし
割当相関セット群	設定なし

各設定項目の詳細については，「5.2.4(6)(g)　サービス呼出アクティビティ（アクティビティ名：配送手配）」を参照してください。

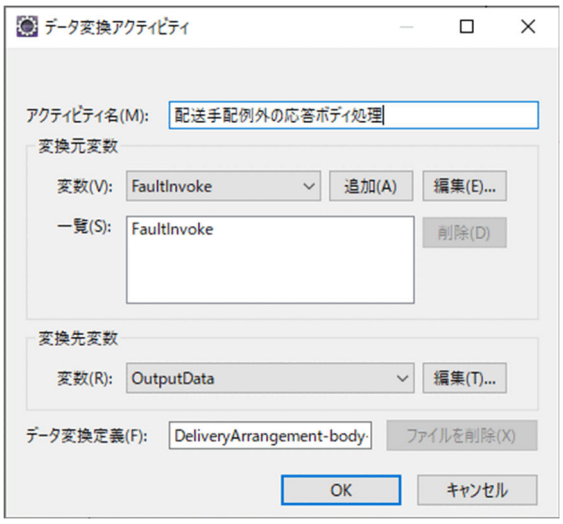
(r)　データ変換アクティビティ（アクティビティ名：配送手配例外の応答ボディ処理）

ProductStock_FaultHandling ビジネスプロセスで新しく追加したアクティビティです。次の手順で定義します。

1. キャンバスのデータ変換アクティビティ（データ変換 5）をダブルクリックします。

［データ変換アクティビティ］ダイアログが表示されます。

2. 次の内容を入力します。



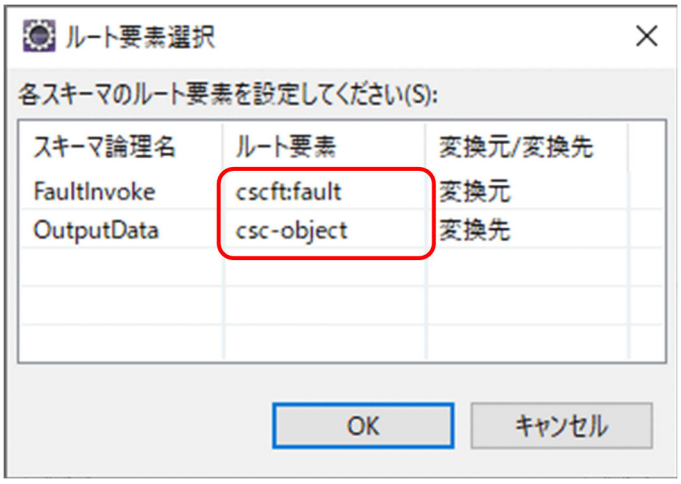
項目名	設定する値	説明
アクティビティ名	配送手配例外の応答ボディ処理	アクティビティの名称を入力します。
変数（変換元変数）	FaultInvoke	データの変換元になる変数をドロップダウンリストから選択し，［追加］ボタンをクリックします。
変数（変換先変数）	OutputData	データの変換先になる変数をドロップダウンリストから選択します。
データ変換定義	DeliveryArrangement-body-processing	変数の変換に使うデータ変換定義ファイルの名称を入力します。

3. ［OK］ボタンをクリックします。

4. キャンバスのデータ変換アクティビティを右クリックして，［マッピング定義起動］を選択します。
［ルート要素選択］ダイアログが表示されます。

5. 変換元のスキーマ論理名「FaultInvoke」のルート要素をクリックして，ドロップダウンリストから「cscft:fault」を選択します。

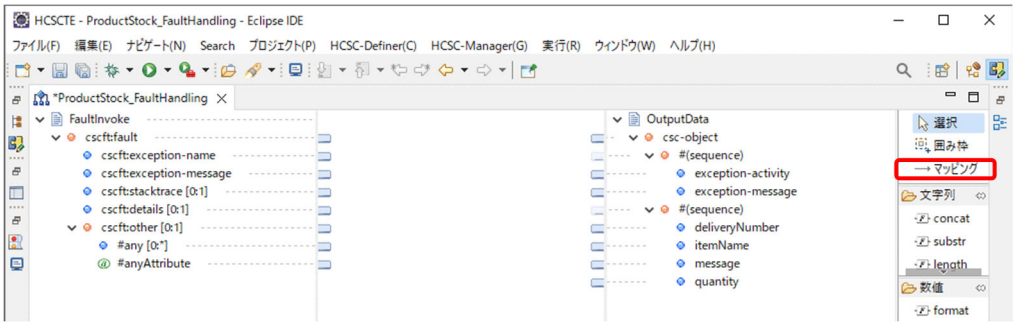
6. 変換先のスキーマ論理名「OutputData」のルート要素をクリックして，ドロップダウンリストから「csc-object」を選択します。



7. [OK] ボタンをクリックします。

   データ変換定義画面が表示されます。

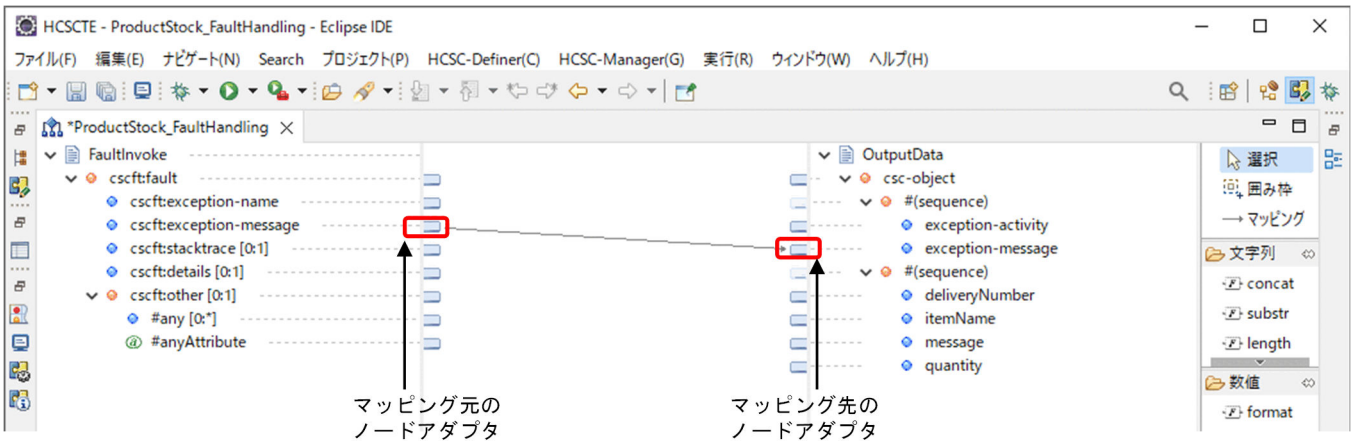
8. データ変換定義画面のパレットから［マッピング］を選択します。



9. マッピング元となる変換元ノードのノードアダプタをクリックします。

10. マッピング先となる変換先ノードのノードアダプタをクリックします。

    マッピング線が設定されます。マッピング元のノードアダプタとマッピング先のノードアダプタとの対応は次のとおりです。



注

作成した XML スキーマによって変換先スキーマの要素の出現順が一部異なるため，マッピング線の見え方が異なる場合があります。

11. データ変換定義画面のパレット［その他］から定数ファンクション（const）を選択します。

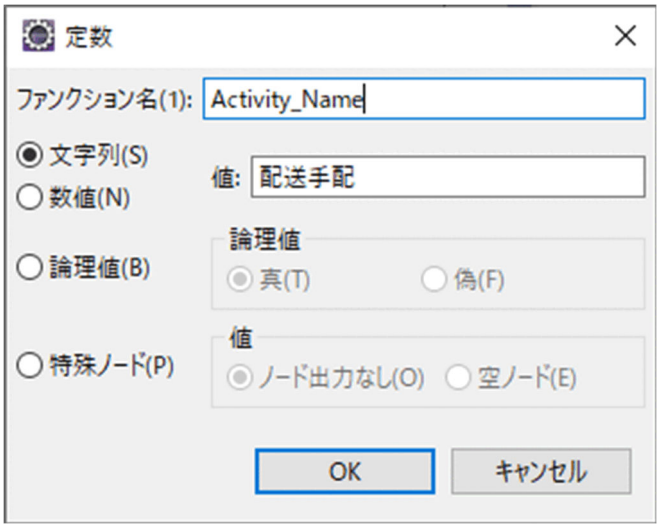
12. マッピングビューアで適当な場所をクリックして定数ファンクションを配置します。

13. 定数ファンクションをダブルクリックします。

［定数］ダイアログが表示されます。

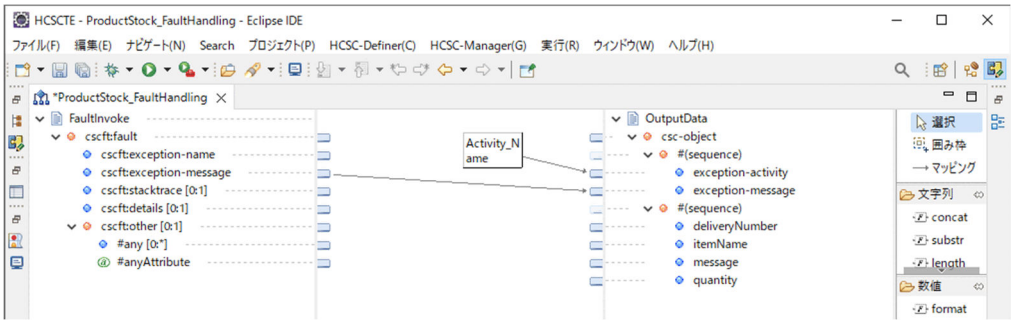
14. ［定数］ダイアログで次の値を指定します。

属性名	属性値
ファンクション名	Activity_Name
文字列	配送手配



15. ［OK］ボタンをクリックします。

16. 次の図のようになるようにマッピング線を設定します。

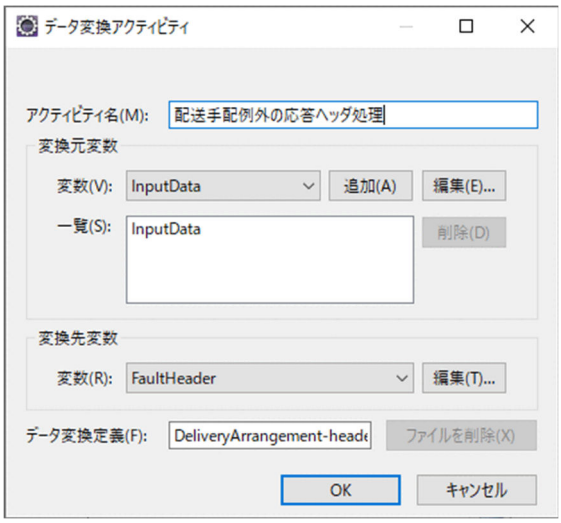


(s)　データ変換アクティビティ（アクティビティ名：配送手配例外の応答ヘッダ処理）

ProductStock_FaultHandling ビジネスプロセスで新しく追加したアクティビティです。次の手順で定義します。

1. キャンバスのデータ変換アクティビティ（データ変換 6）をダブルクリックします。
   ［データ変換アクティビティ］ダイアログが表示されます。

2. 次の内容を入力します。



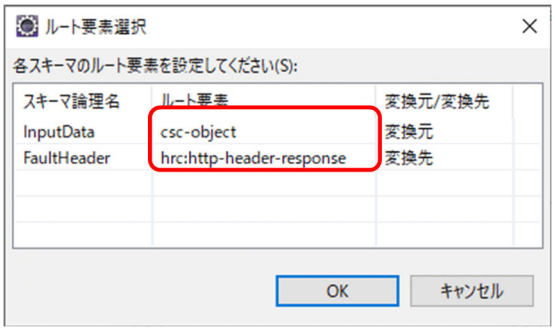
項目名	設定する値	説明
アクティビティ名	配送手配例外の応答ヘッダ処理	アクティビティの名称を入力します。
変数（変換元変数）	InputData	データの変換元になる変数をドロップダウンリストから選択し，［追加］ボタンをクリックします。
変数（変換先変数）	FaultHeader	データの変換先になる変数をドロップダウンリストから選択します。
データ変換定義	DeliveryArrangement-header-processing	変数の変換に使うデータ変換定義ファイルの名称を入力します。

3. ［OK］ボタンをクリックします。

4. キャンバスのデータ変換アクティビティを右クリックして，［マッピング定義起動］を選択します。
   ［ルート要素選択］ダイアログが表示されます。

5. 変換元のスキーマ論理名「InputData」のルート要素をクリックして，ドロップダウンリストから「csc-object」を選択します。

6. 変換先のスキーマ論理名「FaultHeader」のルート要素をクリックして，ドロップダウンリストから「hrc:http-header-response」を選択します。



7. [OK] ボタンをクリックします。
   データ変換定義画面が表示されます。

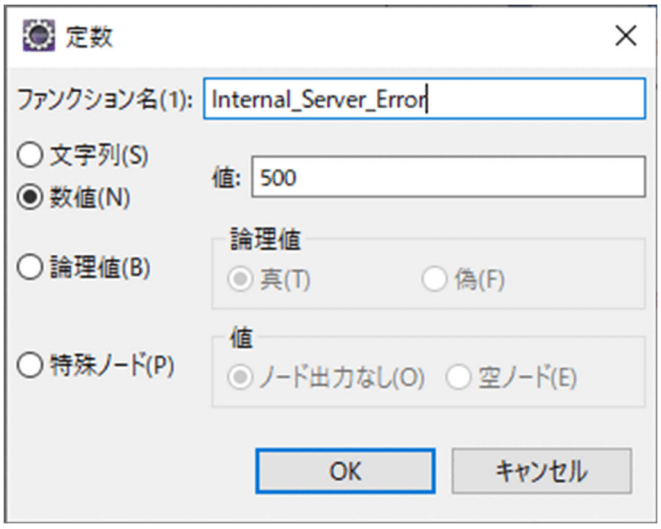
8. データ変換定義画面のパレット［その他］から定数ファンクション（const）を選択します。

9. マッピングビューアで適当な場所をクリックして定数ファンクションを配置します。

10. 定数ファンクションをダブルクリックします。
    ［定数］ダイアログが表示されます。

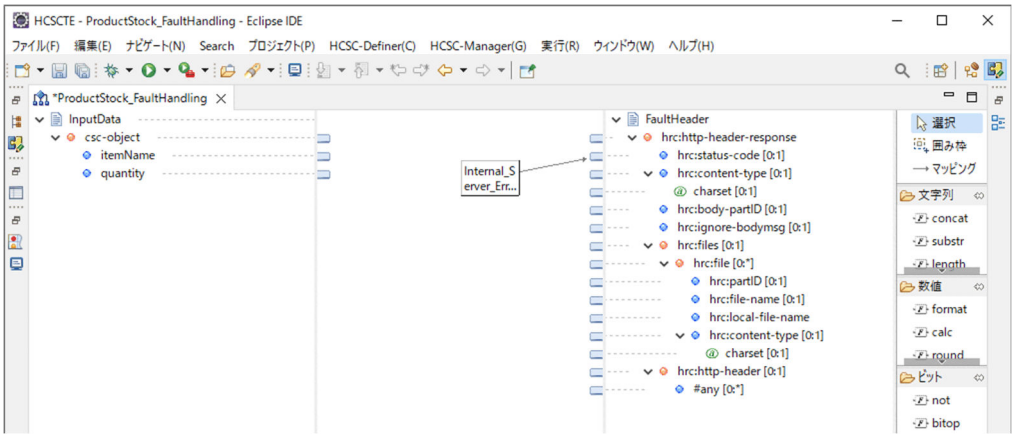
11. [定数] ダイアログで次の値を指定します。

属性名	属性値
ファンクション名	Internal_Server_Error
数値	500



12. [OK] ボタンをクリックします。

13. 次の図のようになるようにマッピング線を設定します。



注

作成した XML スキーマによって変換先スキーマの要素の出現順が一部異なるため，マッピング線の見え方が異なる場合があります。

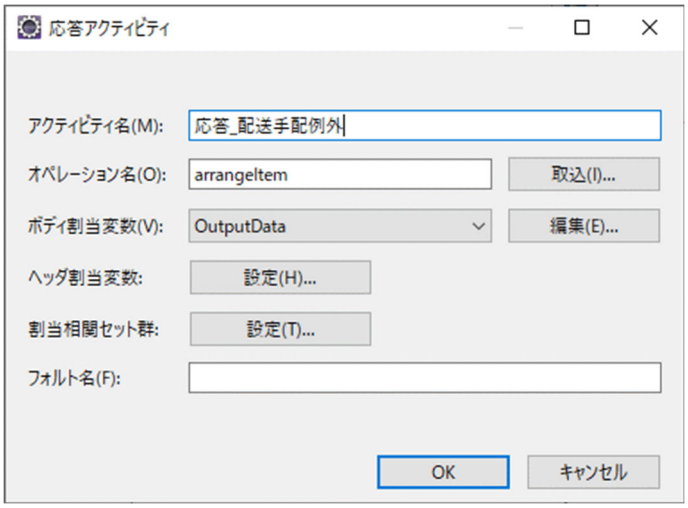
(t)　応答アクティビティ（アクティビティ名：応答_配送手配例外）

ProductStock_FaultHandling ビジネスプロセスで新しく追加したアクティビティです。次の手順で定義します。

1. キャンバスの応答アクティビティ（応答 3）をダブルクリックします。

［応答アクティビティ］ダイアログが表示されます。

2. 次の内容を入力します。



項目名	設定する値	説明
アクティビティ名	応答_配送手配例外	アクティビティの名称を入力します。
オペレーション名	arrangeItem	対応する受付アクティビティに指定したオペレーション名を入力します。
ボディ割当変数	OutputData	ビジネスプロセスの応答電文のボディに割り当てる変数をドロップダウンリストから選択します。

項目名	設定する値	説明
ヘッダ割当変数	割当変数：FaultHeader ルート要素：http-header-response 名前空間： http://www.hitachi.co.jp/soft/xml/cosminexus/csc/reception/http/response	ビジネスプロセスの応答電文のヘッダに変数を割り当てる場合に設定します。 ヘッダ割当変数の［設定］ボタンをクリックし，［ヘッダ割当変数］ダイアログで［追加］ボタンをクリックして，［割当変数］［ルート要素］［名前空間］を設定してください。
割当相関セット群	設定なし	相関セットグループをアクティビティに割り当てる場合に入力します。この商品手配システムでは使用しないため，設定しません。
フォルト名	設定なし	フォルト処理として応答アクティビティを定義して，サービスリクエスタにフォルトが発生したことを示す応答電文を送信する場合のフォルトの名称を入力します。この商品手配システムではフォルト処理を使用しないため，設定しません。

3. ［OK］ボタンをクリックします。

(u) 途中終了アクティビティ（アクティビティ名：途中終了_配送手配例外）

ProductStock_FaultHandling ビジネスプロセスで新しく追加したアクティビティです。次の手順で定義します。

1. キャンバスの途中終了アクティビティ（途中終了 3）をダブルクリックします。
   ［途中終了アクティビティ］ダイアログが表示されます。
2. ［途中終了アクティビティ］ダイアログにアクティビティ名を入力します。
   アクティビティ名「途中終了_配送手配例外」を入力します。
3. ［OK］ボタンをクリックします。

(v) データ変換アクティビティ（アクティビティ名：配送番号設定）

ProductStock_Nomal ビジネスプロセスで定義済のアクティビティです。ProductStock_FaultHandling ビジネスプロセスでは，変換先変数の OutputData の電文フォーマットが変更されているため，データ変換のマッピングを再定義します。

1. キャンバスのデータ変換アクティビティ（配送番号設定）を右クリックして，［設定］を選択します。
   ［データ変換アクティビティ］ダイアログが表示されます。
2. 次のように定義されていることを確認します。

項目名	設定する値
アクティビティ名	配送番号設定
変数（変換元変数）※	InputData

項目名	設定する値
変数（変換元変数）※	DeliveryArrangementOutputData
変数（変換先変数）	OutputData
データ変換定義	DeliveryNumberArrangement

注※　変換元変数は 2 つ設定が必要です。

各設定項目の詳細については，「5.2.4(6)(h)　データ変換アクティビティ（アクティビティ名：配送番号設定）」を参照してください。

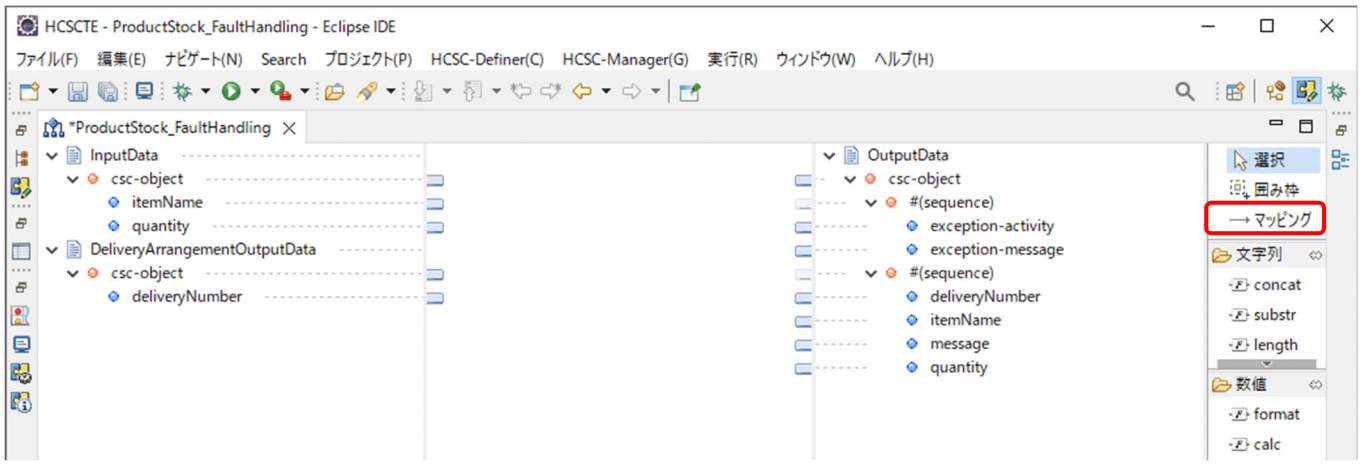
3. [OK] ボタンをクリックします。

4. キャンバスのデータ変換アクティビティを右クリックして，［マッピング定義起動］を選択します。

電文フォーマットの更新を反映させるかどうかを確認するメッセージが表示されるので，［OK］ボタンをクリックします。

データ変換定義画面が表示されます。

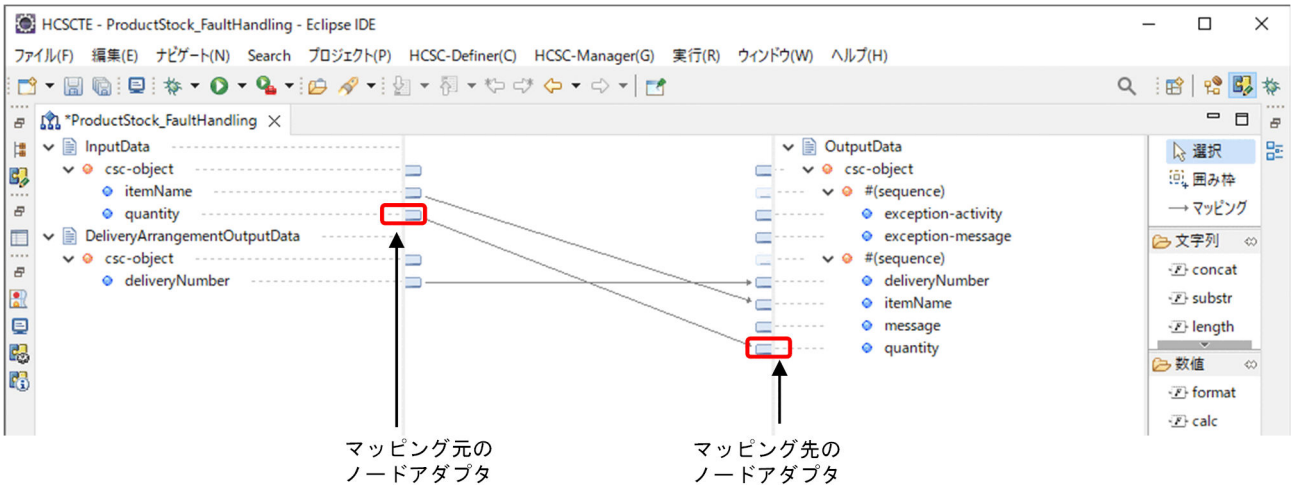
5. データ変換定義画面のパレットから［マッピング］を選択します。



6. マッピング元となる変換元ノードのノードアダプタをクリックします。

7. マッピング先となる変換先ノードのノードアダプタをクリックします。

マッピング線が設定されます。マッピング元のノードアダプタとマッピング先のノードアダプタとの対応は次のとおりです。



注

作成した XML スキーマによって変換先スキーマの要素の出現順が一部異なるため，マッピング線の見え方が異なる場合があります。

8. データ変換定義画面のパレット［その他］から定数ファンクション（const）を選択します。

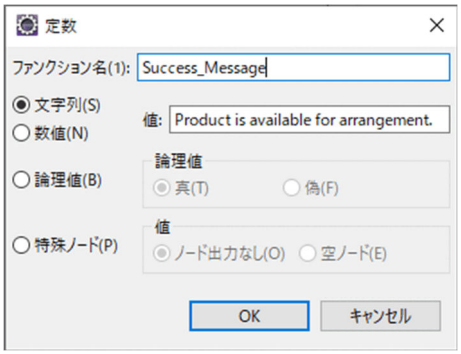
9. マッピングビューアで適当な場所をクリックして定数ファンクションを配置します。

10. 定数ファンクションをダブルクリックします。

［定数］ダイアログが表示されます。

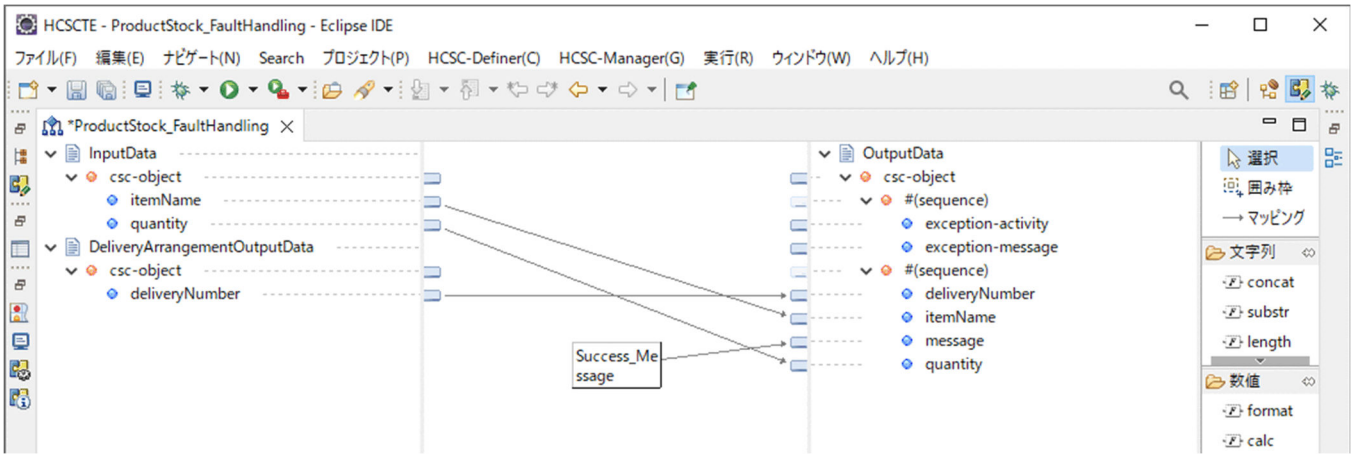
11. ［定数］ダイアログで次の値を指定します。

属性名	属性値
ファンクション名	Success_Message
文字列	Product is available for arrangement.



12. ［OK］ボタンをクリックします。

13. 次の図のようになるようにマッピング線を設定します。



(w)　応答アクティビティ（アクティビティ名：応答_手配成功）

ProductStock_Nomal ビジネスプロセスで定義済のアクティビティです。次のように定義されていることを確認します。

項目名	設定する値
アクティビティ名	応答_手配成功
オペレーション名	arrangeItem
ボディ割当変数	OutputData
ヘッダ割当変数	設定なし
割当相関セット群	設定なし
フォルト名	設定なし

各設定項目の詳細については，「5.2.4(6)(i)　応答アクティビティ（アクティビティ名：応答_手配成功）」を参照してください。

(7)　フォルト処理の定義

フォルトハンドリングを実現するために，フォルトコネクション（赤い破線矢印）に対してフォルト処理を定義します。

(a)　Java 呼出アクティビティ（アクティビティ名：入力値検証）を連結元とするフォルトコネクションの定義

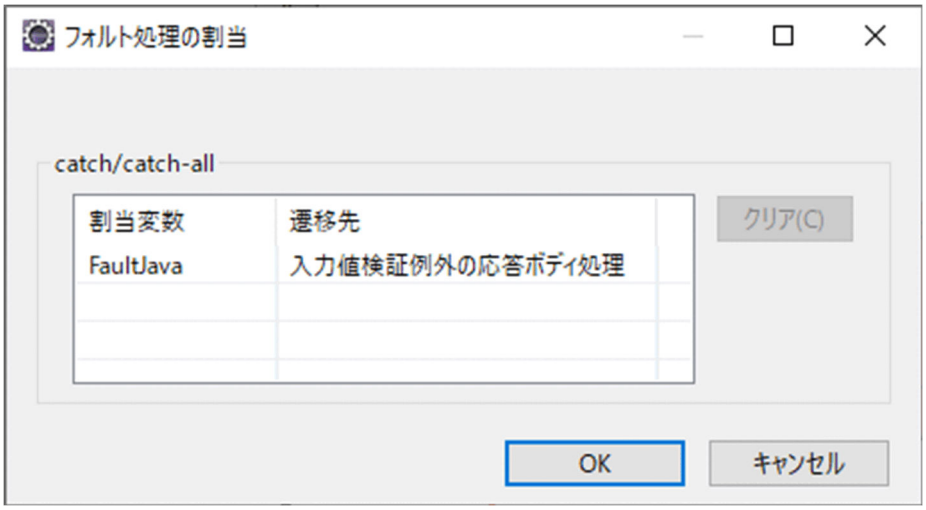
1. キャンバスの Java 呼出アクティビティ（入力値検証）を連結元とするフォルトコネクションの連結線をダブルクリックします。

   ［フォルト処理の割当］ダイアログが表示されます。

2. フォルト処理を定義します。

   ドロップダウンリストで次の内容を割り当ててください。

属性名	属性値	指定内容
割当変数	FaultJava	フォルトの対象となる変数を指定します。
遷移先	入力値検証例外の応答ボディ処理	フォルトが発生した場合に処理する連結先のアクティビティを指定します。



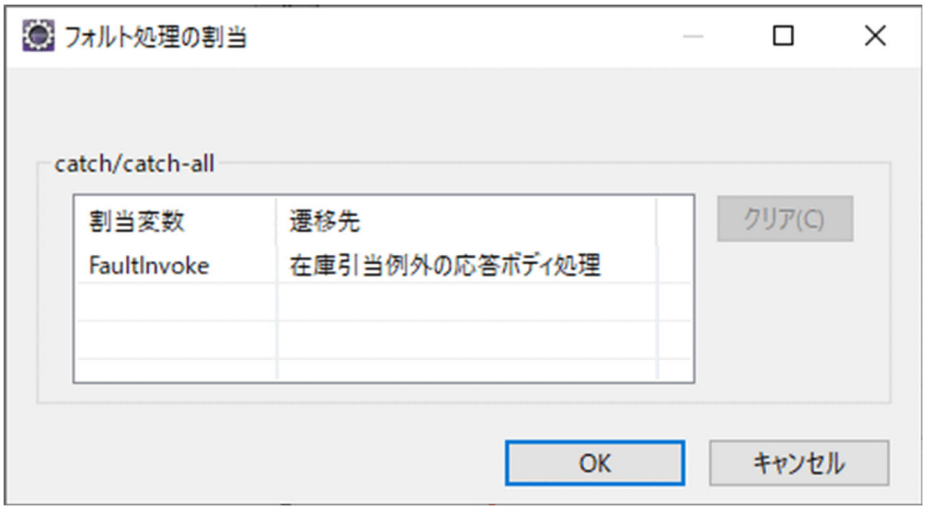
3. [OK] ボタンをクリックします。

(b) サービス呼出アクティビティ（アクティビティ名：在庫引当）を連結元とするフォルトコネクションの定義

1. キャンバスのサービス呼出アクティビティ（在庫引当）を連結元とするフォルトコネクションの連結線をダブルクリックします。
   [フォルト処理の割当] ダイアログが表示されます。

2. フォルト処理を定義します。
   ドロップダウンリストで次の内容を割り当ててください。

属性名	属性値	指定内容
割当変数	FaultInvoke	フォルトの対象となる変数を指定します。
遷移先	在庫引当例外の応答ボディ処理	フォルトが発生した場合に処理する連結先のアクティビティを指定します。



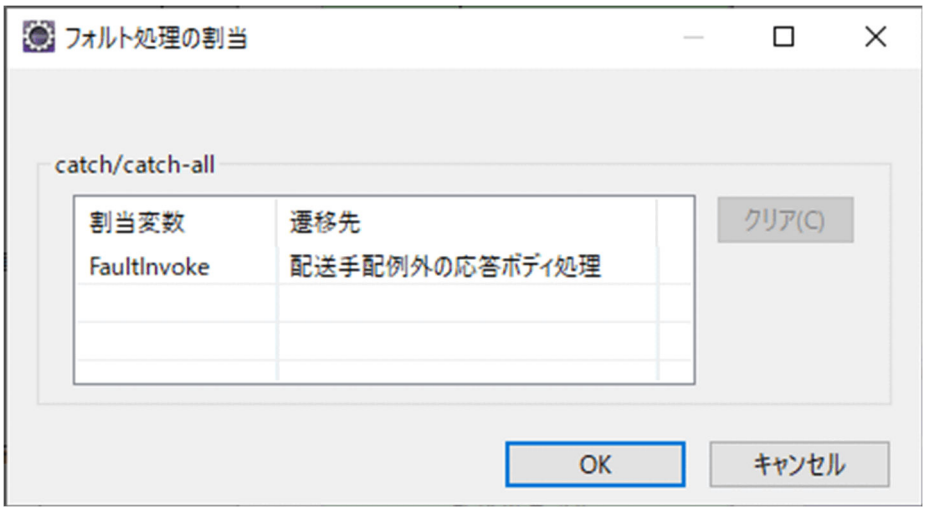
3. [OK] ボタンをクリックします。

(c) サービス呼出アクティビティ（アクティビティ名：配送手配）を連結元とするフォルトコネクションの定義

1. キャンバスのサービス呼出アクティビティ（配送手配）を連結元とするフォルトコネクションの連結線をダブルクリックします。
   [フォルト処理の割当] ダイアログが表示されます。

2. フォルト処理を定義します。
   ドロップダウンリストで次の内容を割り当ててください。

属性名	属性値	指定内容
割当変数	FaultInvoke	フォルトの対象となる変数を指定します。
遷移先	配送手配例外の応答ボディ処理	フォルトが発生した場合に処理する連結先のアクティビティを指定します。



3. [OK] ボタンをクリックします。

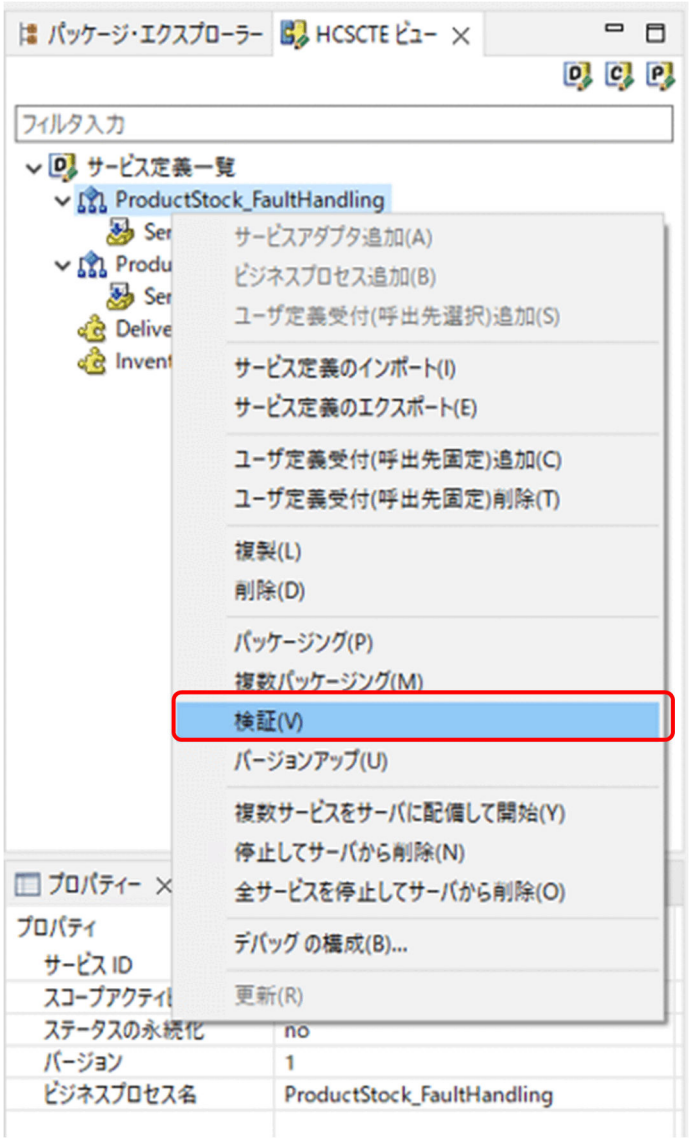
4. すべての定義が完了したら，メニューから［ファイル］－［保管］を選択して，ビジネスプロセスの定義を終了します。

5.5.7　コンポーネントの検証とパッケージング

作成したコンポーネントは定義した内容が正しいかどうか検証してからパッケージングします。

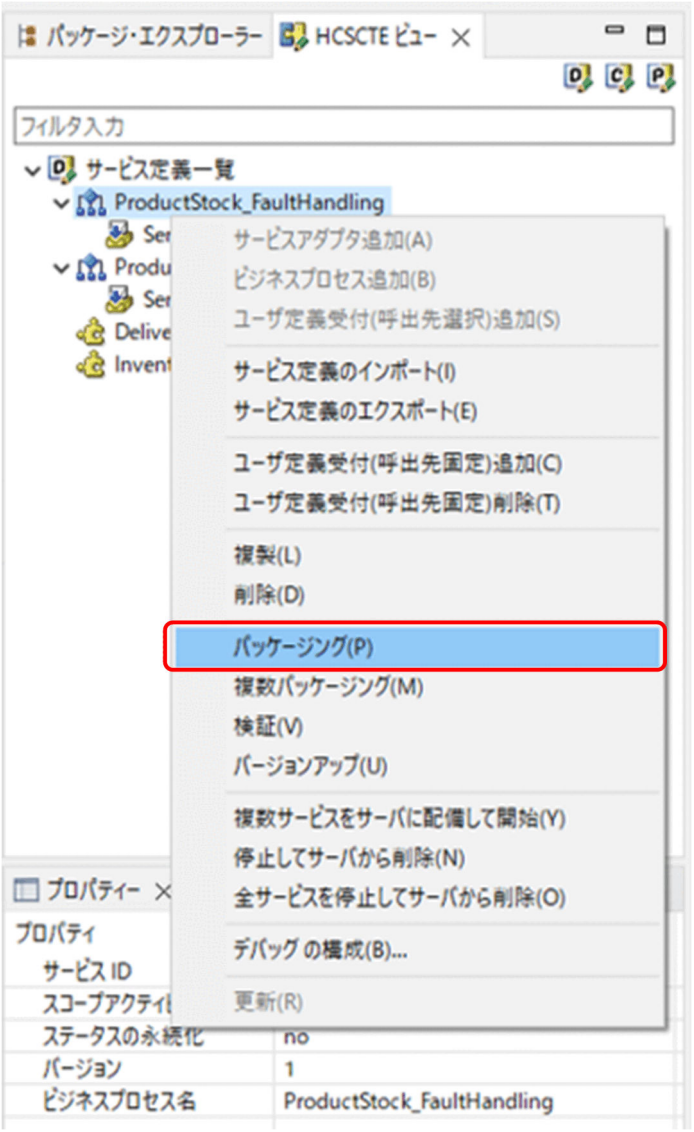
ProductStock_FaultHandling ビジネスプロセスの検証とパッケージングの手順を次に示します。

1. ツリービューの［ProductStock_FaultHandling］を選択し，右クリックして，［検証］を選択します。



検証結果がコンソールビューに表示されます。エラーが発生した場合は，メッセージに従って修正します。

2. ツリービューの［ProductStock_FaultHandling］を選択し，右クリックして，［パッケージング］を選択します。



パッケージングの処理が開始されます。処理終了後，結果を知らせるメッセージが表示されます。

3. 次のどちらかの操作を実行します。

   パッケージングが成功した場合は，［OK］ボタンをクリックします。

   パッケージングが失敗した場合は，メッセージに従って対処し，パッケージングを再実行します。

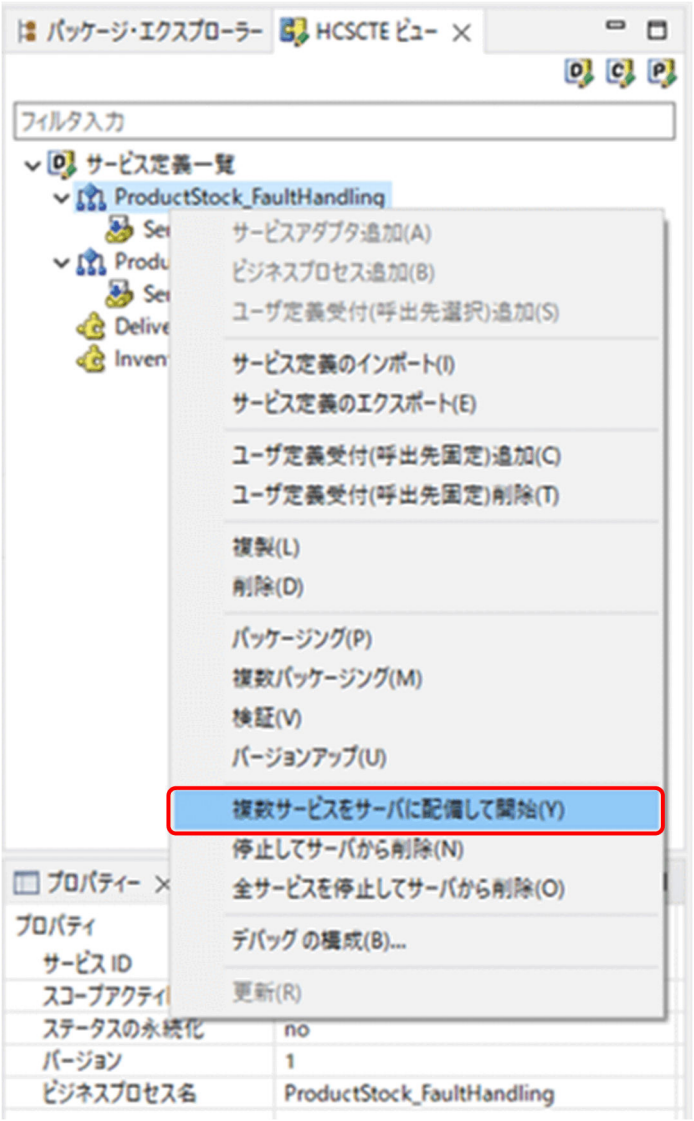
4. ツリービューの［DeliveryReceipt］サービスアダプタ，および［InventoryManagement］サービスアダプタに対して，手順 1.～手順 3.に従って，検証・パッケージングしてください。

5.5.8 HCSC コンポーネントの配備と開始

HCSC コンポーネントをテスト環境に配備して，開始します。

1. ツリービューのサービス定義一覧で，サービス定義一覧または HCSC コンポーネントを選択します。

2. 選択したサービス定義一覧または HCSC コンポーネントを右クリックして，［複数サービスをサーバに配備して開始］を選択します。

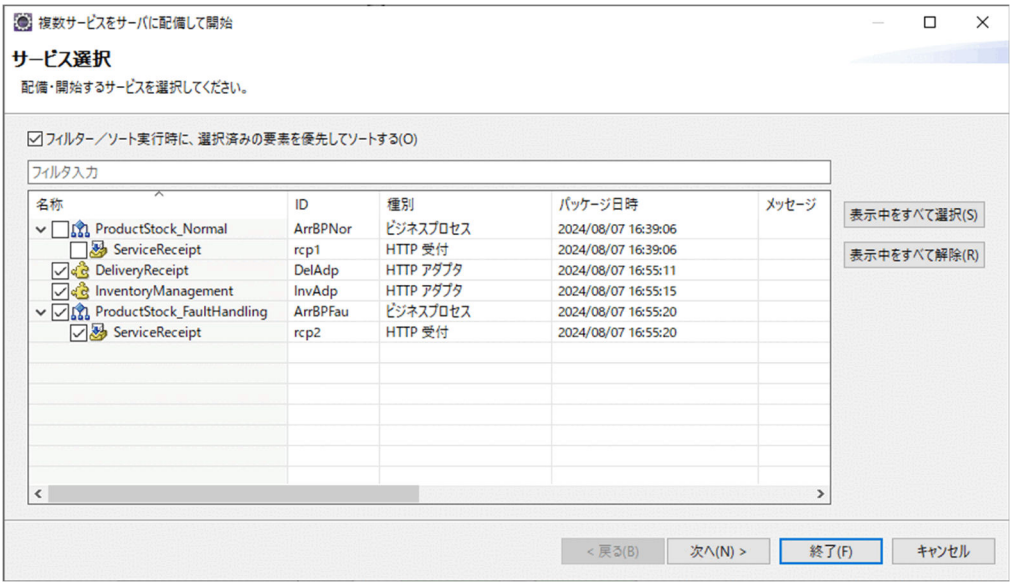


複数サービスをサーバに配備して開始ウィザードが表示されます。

3. サービス選択ページで配備および起動するサービスを選択します。

ここでは，次のサービスを選択してください。

- ProductStock_FaultHandling ビジネスプロセス
- ServiceReceipt 受付
- InventoryManagement サービスアダプタ
- DeliveryReceipt サービスアダプタ



4. [終了] ボタンをクリックします。
   処理中であることを知らせるメッセージが表示されたあと，結果を知らせるメッセージが表示されます。
   ログインしていない場合は，アカウント認証画面が表示されます。手順 5 を実施してください。

5. [ユーザ ID] に「admin」を，[パスワード] に「admin」を入力し，[OK] ボタンをクリックします。

6. [OK] ボタンをクリックします。
   これで配備と開始が完了しました。

5.5.9 HCSC コンポーネントの配備内容の確認

HCSC サーバに，HCSC コンポーネントが配備されていることと，起動していることを確認します。

1. メニューから［ウィンドウ］－［ビューの表示］－［その他］を選択します。
   ［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] ダイアログで［HCSC-Manager］－［HCSC-Manager ビュー］を選択し，［開く］ボタンをクリックします。
   ［HCSC-Manager］ビューが表示されます。

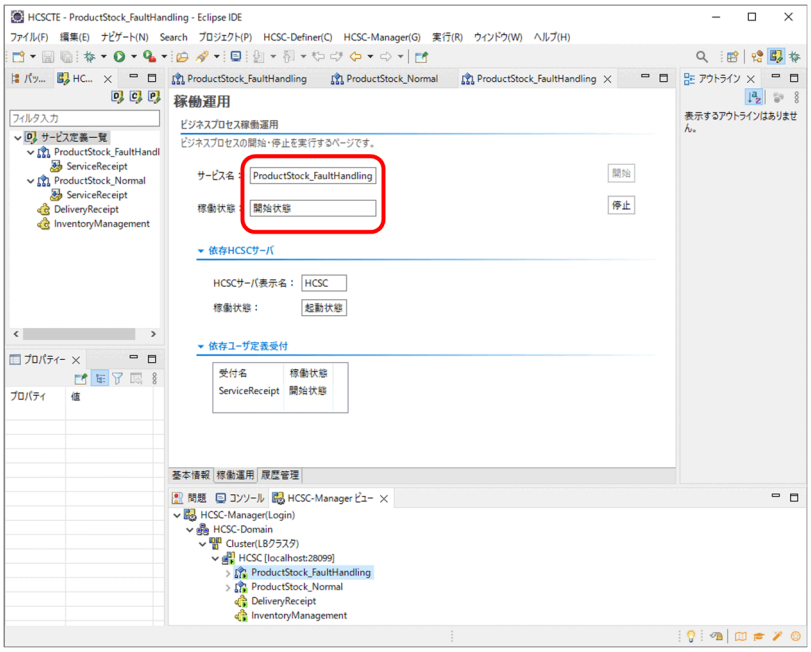
3. [HCSC-Manager] ビューの［HCSC-Manager(Logout)］を右クリックし，［ログイン］を選択します。
   ログイン画面が表示されます。

4. [管理ユーザー ID] に「admin」を，[パスワード] に「admin」を入力して，[OK] ボタンをクリックします。
   HCSC-Manager へのログインが完了します。

5. HCSC コンポーネントが起動していることを確認します。

［HCSC-Manager］ビューの［HCSC-Manager(Login)］－［HCSC-Domain］－［Cluster(LB クラスタ)］－［HCSC[localhost:28099]］－［ProductStock_FaultHandling］をダブルクリックして［基本情報］ページを表示させます。［稼働運用］タブをクリックして［稼働運用］ページを開きます。［稼働状態］が［開始状態］になっているかどうかを確認します。

上記と同じ手順で［DeliveryReceipt］アダプタと［InventoryManagement］アダプタの稼働状態を確認します。



6. **［HCSC-Manager］ビューの［HCSC-Manager(Login)］を右クリックし，［ログアウト］を選択します。**

   HCSC-Manager からのログアウトを確認するメッセージが表示されます。

7. **［OK］ボタンをクリックします。**

   HCSC-Manager からのログアウトが完了し，［HCSC-Manager(Logout)］と表示されます。

これで，ビジネスプロセス・サービスアダプタの準備ができました。

5.6　商品手配システム（フォルトハンドリング編）を実行する

商品手配システムが提供するサービス部品を利用することで，開発した商品手配システムの動作を確認できます。

以降の項で，これらの実行方法について説明します。

5.6.1　開発した商品手配システム（フォルトハンドリング編）を実行する準備

商品手配システムを実行する前に，次の作業が実施済みであることを確認してください。

1. サーバが起動していることを確認します。

   次の 2 つのサーバが起動していることを確認してください。

   HCSC サーバ

   起動方法は「3.4.2　テスト環境の起動」で説明しています。

   RESTful Web サービス用 Web システム

   起動方法は「4.2　サービス部品用 Web システムの構築」で説明しています。

2. RESTful Web サービスが開始していることを確認します。

   開始方法は「4.3　RESTful Web サービスの構築」で説明しています。

3. HCSC コンポーネントの定義と，HCSC サーバへの配備が完了していることを確認します。

   次の点を確認します。

   - HTTP アダプタを動作させるためのクラスパスが追加済であること
     参照先：「5.2.1　HTTP アダプタを動作させるためのクラスパスの設定」
   - InventoryManagement サービスアダプタの定義と，HCSC サーバへの配備が完了していること
     参照先：「5.5.4　InventoryManagement サービスアダプタの定義」
   - DeliveryReceipt サービスアダプタの定義と，HCSC サーバへの配備が完了していること
     参照先：「5.5.5　DeliveryReceipt サービスアダプタの定義」
   - ProductStock_FaultHandling ビジネスプロセスの定義と，HCSC サーバへの配備が完了していること
     参照先：「5.5.6　ProductStock_FaultHandling ビジネスプロセスの定義」

5.6.2　商品手配システム（フォルトハンドリング編）の実行

商品手配システム（フォルトハンドリング編）を実行する方法について説明します。

(1) 実行方法

サービスリクエスタから HTTP リクエストを送信します。

(a) HTTP リクエストの内容

HTTP リクエストの内容を次に示します。

項目	内容
HTTP メソッド	POST
データ形式	JSON
URL※1	http://localhost:80/rcp2/arrangeItem
入力インタフェース※2	"{¥"itemName¥":¥"$itemName¥", ¥"quantity¥":$quantity}"

注※1

HTTP 受付での URL の指定方法の詳細については，マニュアル「サービスプラットフォーム 解説」の「2.15.4　HTTP 受付のリクエスト処理」を参照してください。

注※2

$itemName で入力できる商品は，次の 4 パターンだけです。これ以外を入力すると，入力値検証例外が発生します。

- PC
- TV
- Camera
- Smartphone

$quantity には 1～10 の整数だけ入力できます。1～10 の整数以外を入力すると，入力値検証例外が発生します。在庫数は各商品 10 個ずつです。在庫がない場合，在庫なしレスポンスを返却します。

(b) コマンド実行例

ここでは，サービスリクエスタには curl コマンドを使用します。コマンドプロンプトを起動し，次のコマンドを入力して実行してください。

```
curl http://localhost:80/rcp2/arrangeItem -X POST -H "Content-Type: application/json" -d "{¥"itemName¥":¥"PC¥", ¥"quantity¥":1}"
```

(2) 実行結果の確認

コマンドの実行後，実行結果を示す値が返されます。また，実行結果を表すメッセージと処理内容が表示されます。

表示される内容を次に説明します。

(a) 正常に応答した場合（手配が完了した場合）

在庫ありレスポンスとして，手配が完了したことを知らせるメッセージ，商品名，個数，および配送番号が表示されます。

- 出力例

```
{”deliveryNumber”: ”D00000001”, ”itemName”: ”PC”, ”message”: ”Product is available for arrangement.”, ”quantity”:”1”}
```

(b) 正常に応答した場合（在庫がない場合）

在庫なしレスポンスとして，在庫がないことを知らせるメッセージ，商品名，および個数が表示されます。

- 出力例

```
{”deliveryNumber”:””,”itemName”:”PC”,”message”:”No stock is available.”,”quantity”:”1”}
```

(c) ビジネスプロセス（フォルトハンドリング編）の実行時に例外が発生した場合

ビジネスプロセス（フォルトハンドリング編）の実行時に例外が発生した場合，例外が発生したアクティビティ名および例外メッセージが表示されます。発生する例外の種類を次に示します。

- 入力値検証例外
- 在庫引当例外
- 配送手配例外

これらの例外を発生させる例を次に示します。

- 入力値検証例外

  例外を発生させるための条件を次に示します。

  - $itemName に値が入力されていない
  - $itemName の入力値が商品リストに存在しない
  - $quantity に値が入力されていない
  - $quantity が整数ではない
  - $quantity が 1～10 以外の整数である

  例外を発生させる例として，$quantity に整数以外を入力した場合を次に示します。

  次の curl コマンドをコマンドプロンプトで実行します。

```
curl http://localhost:80/rcp2/arrangeItem -X POST -H ”Content-Type: application/json” -d ”{¥”itemName¥”:¥”PC¥”, ¥”quantity¥”:¥”a¥”}”
```

  出力値を次に示します。

```
{”exception-activity”: ”入力値検証”, ”exception-message”: ”quantityを整数で入力してください”}
```

- 在庫引当例外

  例外を発生させる例として，在庫管理サービスを停止させた状態で，ProductStock_FaultHandling ビジネスプロセス（フォルトハンドリング編）を実行した場合を次に示します。

  まず，在庫管理サービスを停止させます。停止方法については「5.3.3(1)　サービスのリデプロイ」の在庫管理サービスの停止方法を参照してください。

  その後，次の curl コマンドをコマンドプロンプトで実行します。

```
curl http://localhost:80/rcp2/arrangeItem -X POST -H "Content-Type: application/json" -d
"{¥"itemName¥":¥"PC¥", ¥"quantity¥":1}"
```

  出力値を次に示します。

```
{"exception-activity":"在庫引当","exception-message":"jp.co.Hitachi.soft.csc.msg.adapter.
protocol.CSCMsgCustomAccessException: A service call was interrupted because a custom ser
vice access error was detected. (adapter name = InvAdp, HCSCCommonID = CSC_HCSC_2024-06-2
5_17:27:38.029_6, ServiceRequestID = MSG_HCSC_SyncBP_2024-06-25_17:27:38.045_14, error me
ssage = An HTTP adapter error occurred. (adapter name = InvAdp, operation name = reserveI
tem, exception information = jp.co.Hitachi.soft.csc.cstmadp.http.rt.exception.CSAHTAExcep
tion: KDEC81054-E An error status code was received from the service. (adapter name = Inv
Adp, operation name = reserveItem, status code = 404)), code = KDEC81499-E) ErrorCode=KDE
C03005-E"}
```

- 配送手配例外

  例外を発生させる例として，配送受付サービスを停止させた状態で，ProductStock_FaultHandling ビジネスプロセス（フォルトハンドリング編）を実行した場合を次に示します。

  まず，配送受付サービスを停止させます。停止方法については「5.3.3(1)　サービスのリデプロイ」の配送受付サービスの停止方法を参照してください。

  その後，次の curl コマンドをコマンドプロンプトで実行します。

```
curl http://localhost:80/rcp2/arrangeItem -X POST -H "Content-Type: application/json" -d
"{¥"itemName¥":¥"PC¥", ¥"quantity¥":1}"
```

  出力値を次に示します。

```
{"exception-activity":"配送手配","exception-message":"jp.co.Hitachi.soft.csc.msg.adapter.
protocol.CSCMsgCustomAccessException: A service call was interrupted because a custom ser
vice access error was detected. (adapter name = DelAdp, HCSCCommonID = CSC_HCSC_2024-06-2
5_17:33:39.366_8, ServiceRequestID = MSG_HCSC_SyncBP_2024-06-25_17:33:39.625_18, error me
ssage = An HTTP adapter error occurred. (adapter name = DelAdp, operation name = deliverI
tem, exception information = jp.co.Hitachi.soft.csc.cstmadp.http.rt.exception.CSAHTAExcep
tion: KDEC81054-E An error status code was received from the service. (adapter name = Del
Adp, operation name = deliverItem, status code = 404)), code = KDEC81499-E) ErrorCode=KDE
C03005-E"}
```

5.6.3　商品在庫数の初期化方法

商品手配システムでの各商品の総在庫数は 10 個です。手配が完了すると，その数だけ在庫が減少します。

在庫がなくなった場合は，サービスをリデプロイするか，Web システムを再起動する必要があります。これによって在庫数が 10 個に戻ります。

サービスのリデプロイおよび Web システムの再起動方法は基本編と同様です。詳細については，「5.3.3 商品在庫数の初期化方法」を参照してください。

5.7　ProductStock_FaultHandling ビジネスプロセスのデバッグ

ビジネスプロセスのデバッグ方法は，基本編と同様です。デバッグ方法の詳細については，「5.4 ProductStock_Normal ビジネスプロセスのデバッグ」を参照してください。

Eclipse でデバッグ対象を選択する際，「ProductStock_FaultHandling」を選択してください。

6 商品手配システムの環境を削除する

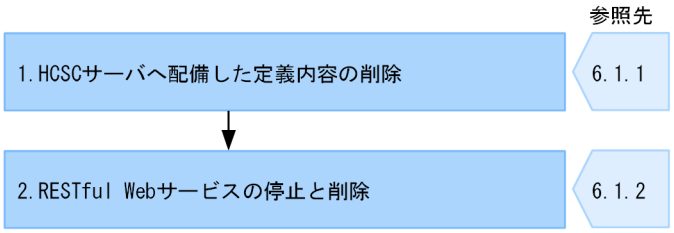
この章では，商品手配システムの環境の削除手順について説明します。

6.1　サービスの削除

商品手配システムの削除手順について説明します。

削除するまでの流れを次の図に示します。

図 6-1　商品手配システムの削除の流れ



それぞれの作業の概要を説明します。

1. HCSC サーバへ配備した定義内容の削除

   「5.2.6　HCSC コンポーネントの配備と開始」または，「5.5.8　HCSC コンポーネントの配備と開始」で HCSC サーバへ配備した定義内容を削除します。詳細については，「6.1.1　HCSC サーバへ配備した定義内容の削除」を参照してください。

2. RESTful Web サービスの停止と削除

   「4.　サービス部品を構築する」で構築した在庫管理サービスと配送受付サービスの両方を停止および削除します。詳細については，「6.1.2　RESTful Web サービスの停止と削除」を参照してください。

以降の項では，この流れに沿って商品手配システムを削除する手順を説明します。

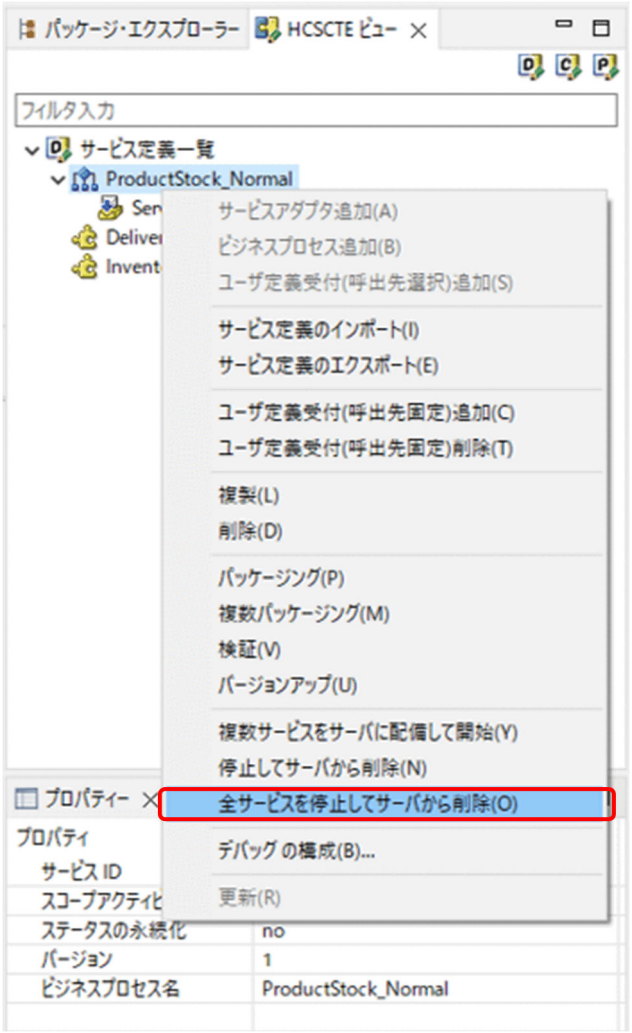
6.1.1　HCSC サーバへ配備した定義内容の削除

ここでは，HCSC サーバ上に配備した HCSC コンポーネントの削除方法と定義情報初期化方法を説明します。

定義情報初期化とはリポジトリ初期化のことを指します。リポジトリに含まれる情報については，マニュアル「サービスプラットフォーム システム構築・運用ガイド」の「4.1.1　リポジトリの情報」を参照してください。

1. HCSCTE ツリービューのサービス定義一覧で，サービス定義一覧または HCSC コンポーネントを選択します。

2. 選択したサービス定義一覧または HCSC コンポーネントを右クリックして，［全サービスを停止してサーバから削除］を選択します。



アカウント認証画面が表示されます。

3. [ユーザ ID] に「admin」を，[パスワード] に「admin」を入力し，[OK] ボタンをクリックします。

   処理中であることを知らせるメッセージが表示されたあと，結果を知らせるメッセージが表示されます。これで HCSC サーバへ配備した HCSC コンポーネントの削除が完了しました。

4. HCSCTE プロジェクトから，定義を完全に削除するには，[HCSC-Definer] － [定義情報管理] － [定義情報初期化] を選択します。

   初期化を確認するダイアログが表示されるので，[はい] ボタンをクリックします。これで定義が完全に削除されました。

6.1.2 RESTful Web サービスの停止と削除

在庫管理サービスと配送受付サービスの両方を停止および削除します。この操作は，管理者または管理者特権で実行してください。

(1)　在庫管理サービスの停止・削除

(a)　サービスの停止

次のコマンドでサービスを停止します。

```
”%COSMINEXUS_HOME%¥CC¥admin¥bin¥cjstopapp” jaxrsserver -nameserver corbaname::localhost:901
-name Sample_application_jaxrs_inventorymanagement
```

(b)　サービスの削除

次のコマンドでサービスを削除します。

```
”%COSMINEXUS_HOME%¥CC¥admin¥bin¥cjdeleteapp” jaxrsserver -nameserver corbaname::localhost:90
1 -name Sample_application_jaxrs_inventorymanagement
```

(2)　配送受付サービスの停止・削除

(a)　サービスの停止

次のコマンドでサービスを停止します。

```
”%COSMINEXUS_HOME%¥CC¥admin¥bin¥cjstopapp” jaxrsserver -nameserver corbaname::localhost:901
-name Sample_application_jaxrs_delivery
```

(b)　サービスの削除

次のコマンドでサービスを削除します。

```
”%COSMINEXUS_HOME%¥CC¥admin¥bin¥cjdeleteapp” jaxrsserver -nameserver corbaname::localhost:90
1 -name Sample_application_jaxrs_delivery
```

6.2　環境の停止

商品手配システムの環境を停止するには，テスト環境を停止してから，RESTful Web サービス配備用 Web システムを停止します。

以降の項では，この流れに沿って環境を停止する手順を説明します。

6.2.1　テスト環境の停止

テスト環境を停止する手順を次に示します。この操作は，管理者または管理者特権で実行してください。

1. ［スタート］メニューの［プログラム］から［Cosminexus※］－［テストサーバ停止］を選択して，テスト環境の Performance Tracer，J2EE サーバ，および，標準受付とユーザ定義受付を含む HCSC サーバを停止します。

   注※

   プログラムフォルダ名を変更している場合は，変更したプログラムフォルダから選択してください。

6.2.2　サービス部品用 Web システムの停止

RESTful Web サービス配備用 Web システムの RESTfulWebServiceSystem を停止します。この操作は，管理者または管理者特権で実行してください。

1. 次のコマンドを実行し，RESTful Web サービス配備用 Web システム（RESTfulWebServiceSystem）を停止します。

```
”%COSMINEXUS_HOME%¥manager¥bin¥cmx_stop_target” -m localhost -u admin -p admin -s RESTful WebServiceSystem -mode ALL
```

6.3　アンセットアップとアンインストール

アンセットアップとアンインストールは，次の順で実施します。

1. サービス部品用 Web システムのアンセットアップ
2. テスト環境のアンセットアップ
3. Eclipse のアンセットアップ
4. Service Architect のアンインストール

ここでは，それぞれの手順について説明します。

6.3.1　サービス部品用 Web システムのアンセットアップ

サービス部品用 Web システムのアンセットアップとして，RESTful Web サービス配備用 Web システムの削除，および J2EE サーバの削除を実施します。手順を次に示します。この操作は，管理者または管理者特権で実行してください。

1. 次のコマンドを実行して，RESTful Web サービス配備用 Web システム（RESTfulWebServiceSystem）を削除します。

```
"%COSMINEXUS_HOME%¥manager¥bin¥cmx_delete_system" -m localhost -u admin -p admin -s RESTf
ulWebServiceSystem
```

2. RESTful Web サービス配備用 Web システム（RESTfulWebServiceSystem）内に構築した J2EE サーバ jaxrsserver を削除します。

   cmx_delete_system コマンドで RESTfulWebServiceSystem を削除しても RESTfulWebServiceSystem を構築した localhost に J2EE サーバが残るため，次のコマンドを実行して J2EE サーバを削除します。

```
"%COSMINEXUS_HOME%¥CC¥server¥bin¥cjsetup" -d jaxrsserver
```

6.3.2　テスト環境のアンセットアップ

テスト環境をアンセットアップするには，HCSC 簡易セットアップ機能を使います。テスト環境のアンセットアップ方法を次に示します。この操作は，管理者または管理者特権で実行してください。

1. テスト環境を起動している場合は，テスト環境を停止します。また，Eclipse を起動している場合は，Eclipse を終了します。

2. [スタート] メニューの［プログラム］から［Cosminexus※］－［テスト環境セットアップ］を選択します。

   HCSC 簡易セットアップの画面の［メイン］ページが表示されます。

注※

プログラムフォルダ名を変更している場合は，変更したプログラムフォルダから選択してください。

3. [アンセットアップ] ボタンをクリックします。

テスト環境のアンセットアップが開始され，しばらくすると完了します。

6.3.3 Eclipse のアンセットアップ

Eclipse セットアップ機能で構築した環境は，Eclipse セットアップ機能でアンセットアップします。

Eclipse セットアップ機能を使用したアンセットアップを実施した場合に削除される項目を次に示します。

表 6-1 Eclipse セットアップ機能のアンセットアップで削除される項目

項目	削除の有無
Eclipse Platform	削除される※1
Eclipse の configuration フォルダ (ユーザごとのフォルダ)	削除されない※2
Eclipse のワークスペース	削除されない
Eclipse のショートカット	削除される

注※1

ユーザが作成したファイルも含め，対象となるフォルダ内にあるすべてのデータが削除されます。

注※2

configuration フォルダおよびワークスペースは，Eclipse が生成します。このため，Eclipse セットアップ機能のアンセットアップを実施しても削除されません。削除したい場合は手動で削除してください。手動での削除は，「(2) 手動でアンセットアップする場合」を参照してください。

なお，Eclipse セットアップ機能で構築した環境をアンセットアップする前に，Service Architect をアンインストールおよび再インストールした場合は，構築した環境を手動でアンセットアップしてください。手動でのアンセットアップは，「(2) 手動でアンセットアップする場合」を参照してください。

注意事項

アンセットアップ実行前に，Eclipse を終了してください。Eclipse 起動中にアンセットアップを実行すると，Eclipse のインストールディレクトリが削除されません。なお，アンセットアップ実行後に Eclipse のインストールディレクトリが残っていた場合は，<Eclipse のインストールディレクトリ>¥eclipse 以下のディレクトリおよびファイルを手動で削除してください。

(1) Eclipse セットアップ機能を使用する場合

Eclipse セットアップ機能を使用したアンセットアップの手順を次に示します。

1. [スタート] メニューから，[プログラム] − [Cosminexus※] − [Eclipse アンセットアップ] を選択します。

   Eclipse セットアップ機能が起動して，[アンセットアップ - Eclipse セットアップ] ダイアログの [アンセットアップの確認] ページが表示されます。

   注※

   プログラムフォルダ名を変更している場合は，変更したプログラムフォルダから選択してください。

2. [アンセットアップの内容] エリアに表示された内容を確認して，[実行] ボタンをクリックします。

   [進行状況] ページが表示されます。

   アンセットアップが終了すると，[アンセットアップの完了] ページが表示されます。

3. [終了] ボタンをクリックします。

   [アンセットアップ - Eclipse セットアップ] ダイアログが閉じます。

注意事項

アンセットアップを実行すると，セットアップ実行時にデスクトップに追加した，eclipse.exe へのショートカットが削除されますが，画面には表示されたままで，削除されていないように見える場合があります。この場合には，デスクトップを最新の情報に更新すると，ショートカットの削除が画面に反映されます。

(2) 手動でアンセットアップする場合

Eclipse セットアップ機能で構築した環境をアンセットアップする前に，Service Architect をアンインストールおよび再インストールした場合は，構築した環境を手動でアンセットアップしてください。手動でアンセットアップする手順を次に示します。

1. Eclipse のショートカットを削除します。

   次に示すファイルを管理者が削除します。

```
C:¥Users¥Public¥Desktop¥Eclipse.lnk
```

2. Eclipse の configuration フォルダを削除します。

フォルダの場所を次に示します。

```
C:¥Users¥（全ユーザ分）¥ADP
```

3. Eclipse のフォルダを削除します。

デフォルトのフォルダの場所を次に示します。

```
<Service Architectのインストールディレクトリ>¥ADP¥IDE¥eclipse
```

6.3.4 Service Architect のアンインストール

Service Architect をアンインストールします。アンインストールには，Administrator 権限または管理者特権が必要です。

なお，Service Architect をアンインストールする前に，必ず Eclipse をアンセットアップしてください。

1. Windows の［スタート］メニューから，［プログラム］－［Cosminexus※］－［uCosminexus Service Architect アンインストール］を選択します。

アンインストールを確認するダイアログが表示されます。

注※

プログラムフォルダ名を変更している場合は，変更したプログラムフォルダから選択してください。

2. ［はい］ボタン，または［いいえ］ボタンをクリックします。

［はい］ボタンをクリックした場合

アンインストールが開始され， Service Architect の構成ソフトウェアがすべて削除されます。

［いいえ］ボタンをクリックした場合

アンインストールする構成ソフトウェアを選択するダイアログが表示されます。アンインストールする構成ソフトウェアを選択して，［次へ］ボタンをクリックすると，アンインストールが開始され，選択した構成ソフトウェアが削除されます。

付録

付録 A　サービス部品のサンプルプログラムのファイル構成

商品手配システムのサービス部品のサンプルプログラムは，次の場所にあります。

<Service Architectのインストールディレクトリ>¥CSCTE¥Samples¥SOAP1.1_1.2mode¥ProductStock_REST

商品手配システムのサービス部品のサンプルプログラムのファイル構成を次に示します。

```
ProductStock_REST
 ├Repository
 │ └ProductStock_Normal.zip
 │  ProductStock_FaultHandling.zip
 │  Validation.java
 ├Server
 │ └jaxrsserver_def.xml
 ├Schema
 │ ├ServiceReceipt
 │ │ ├request
 │ │ │ └input_Arr.xsd
 │ │ │   input_Arr.json
 │ │ └response
 │ │   ├ProductStock_Normal
 │ │   │ └output_ArrNor.xsd
 │ │   │   output_ArrNor.json
 │ │   └ProductStock_FaultHandling
 │ │     └output_ArrFau.xsd
 │ │       output_ArrNor.json
 │ │       output_DeliveryExc.json
 │ │       output_InventoryExc.json
 │ │       output_JavaExc.json
 │ ├InventoryManagementAdapter
 │ │ ├request
 │ │ │ └input_Inv.xsd
 │ │ │   input_Inv.json
 │ │ └response
 │ │   └output_Inv.xsd
 │ │     output_Inv.json
 │ └DeliveryReceiptAdapter
 │   ├request
 │   │ └input_Del.xsd
 │   │   input_Del.json
 │   └response
 │     └output_Del.xsd
 │       output_Del.json
 └Service
   ├InventoryManagement
   │ ├META-INF
   │ │ └application.xml
   │ ├WEB-INF
   │ │ ├web.xml
   │ │ └classes
   │ │   └com
   │ │     └sample
   │ │       └resources
   │ │         └Resource.class
```

```
    │ │           ReserveItem.class
    │ │           ReserveItemResponse.class
    │ ├src
    │ │ └com
    │ │   └sample
    │ │     └resources
    │ │       └Resource.java
    │ │         ReserveItem.java
    │ │         ReserveItemResponse.java
    │ └inventorymanagement.ear
    │   inventorymanagement.war
    └DeliveryReceipt
      ├META-INF
      │ └application.xml
      ├WEB-INF
      │ ├web.xml
      │ └classes
      │   └com
      │     └sample
      │       └resources
      │         └Resource.class
      │           DeliverItem.class
      │           DeliverItemResponse.class
      ├src
      │ └com
      │   └sample
      │     └resources
      │       └Resource.java
      │         DeliverItem.java
      │         DeliverItemResponse.java
      └delivery.ear
       delivery.war
```

付録 B　RESTful Web サービスのサービス構成とファイル構成

付録 B.1　在庫管理サービス

在庫管理サービスの構成を次に示します。

表 B-1　在庫管理サービスの構成

項番	属性名	属性値
1	デプロイする J2EE サーバの名称	jaxrsserver
2	NIO HTTP サーバのポート番号	8008
3	ネーミングサーバの URL	corbaname::localhost:901
4	コンテキストルート	items
5	ルートリソースクラスのコンテキストパス	inventories
6	パッケージ名	com.sample.resources
7	ルートリソースクラス	com.sample.resources.Resource
8	WAR ファイル名	inventorymanagement.war
9	EAR ファイル名	inventorymanagement.ear
10	サービス名	Sample_application_jaxrs_inventorymanagement

在庫管理サービスのファイル構成を次に示します。

```
InventoryManagement
 ├META-INF
 │ └application.xml
 ├WEB-INF
 │ ├web.xml
 │ └classes
 │   └com
 │     └sample
 │       └resources
 │         └Resource.class
 │           ReserveItem.class
 │           ReserveItemResponse.class
 ├src
 │ └com
 │   └sample
 │     └resources
 │       └Resource.java
 │         ReserveItem.java
 │         ReserveItemResponse.java
 └inventorymanagement.ear
  inventorymanagement.war
```

付録 B.2　配送受付サービス

配送受付サービスの構成を次に示します。

表 B-2　配送受付サービスの構成

項番	属性名	属性値
1	デプロイする J2EE サーバの名称	jaxrsserver
2	NIO HTTP サーバのポート番号	8008
3	ネーミングサーバの URL	corbaname::localhost:901
4	コンテキストルート	orders
5	ルートリソースクラスのコンテキストパス	deliveries
6	パッケージ名	com.sample.resources
7	ルートリソースクラス	com.sample.resources.Resource
8	WAR ファイル名	delivery.war
9	EAR ファイル名	delivery.ear
10	サービス名	Sample_application_jaxrs_delivery

配送受付サービスのファイル構成を次に示します。

```
DeliveryReceipt
 ├META-INF
 │ └application.xml
 ├WEB-INF
 │ ├web.xml
 │ └classes
 │   └com
 │     └sample
 │       └resources
 │         └Resource.class
 │           DeliverItem.class
 │           DeliverItemResponse.class
 ├src
 │ └com
 │   └sample
 │     └resources
 │       └Resource.java
 │         DeliverItem.java
 │         DeliverItemResponse.java
 └delivery.ear
  delivery.war
```

付録 C　HCSC コンポーネントのサンプルプログラムインポート

商品手配システムの HCSC コンポーネントは，サンプルプログラムとしても提供しています。HCSC コンポーネントは，リポジトリ情報（ZIP ファイル）として提供されます。HCSC コンポーネントのインポート方法について説明します。

1. Eclipse のメニューから，［HCSC-Definer］－［定義情報管理］－［全定義情報インポート］を選択します。

   リポジトリの上書きを確認するダイアログが表示されます。

2. ［はい］ボタンをクリックします。

   リポジトリ情報の ZIP ファイルを選択する［リポジトリインポート］ダイアログが表示されます。使用するサンプルプログラムの ZIP ファイルを選択してください。

   基本編：

   <Service Architectのインストールディレクトリ>¥CSCTE¥Samples¥SOAP1.1_1.2mode¥ProductStock_REST¥Repository¥ProductStock_Normal.zip

   フォルトハンドリング編：

   <Service Architectのインストールディレクトリ>¥CSCTE¥Samples¥SOAP1.1_1.2mode¥ProductStock_REST¥Repository¥ProductStock_FaultHandling.zip

3. ［開く］ボタンをクリックします。

   インポート対象の定義情報を選択する［リポジトリインポート］ダイアログが表示されます。

4. ［サービス定義］のチェックボックスをチェックして，［OK］ボタンをクリックします。

付録 D　Eclipse セットアップ機能実行時の情報の採取

Eclipse セットアップ機能を実行したときに設定した内容（セットアップ内容，設定変更内容，アンセットアップ内容）は，**セットアップログ**としてファイルに保存されます。

セットアップログの確認方法，およびセットアップログで確認できる内容を説明します。

- **セットアップログの確認方法**

  Windows の［スタート］メニューから，［プログラム］－［Cosminexus※］－［Eclipse セットアップログ］を選択すると，セットアップログが表示されます。

  注※

  プログラムフォルダ名を変更している場合は，変更したプログラムフォルダから選択してください。

- **セットアップログで確認できる内容**

  Eclipse 環境のセットアップログでは，次の内容を確認できます。

  - Eclipse 環境をセットアップしたときの設定内容
  - アンセットアップした Eclipse 環境の設定内容

  これらの情報は，実行された順番に 1 つのファイルに格納されます。このため，最新の情報は，ファイルの最後に格納されています。なお，エラーまたは処理の中断が発生した場合でも，これらの情報は格納されます。

  なお，Eclipse セットアップ機能実行時のエラーメッセージは，Eclipse セットアップ機能のコンソールに出力されます。出力されたエラーメッセージについては，マニュアル「アプリケーションサーバ メッセージ(構築／運用／開発用)」を参照して対処してください。

付録 E　用語解説

マニュアルで使用する用語について

マニュアル「アプリケーションサーバ & BPM/ESB 基盤 用語解説」を参照してください。

索引