uCosminexus Service Platform

Basic Development Guide

3020-3-Y43-40(E)

■ Relevant program products

See the description for relevant program products in the preface of the manual "*Application Server Overview*".

■ Export restrictions

If you export this product, please check all restrictions (for example, Japan's Foreign Exchange and Foreign Trade Law, and USA export control laws and regulations), and carry out all required procedures.

If you require more information or clarification, please contact your Hitachi sales representative.

■ Trademarks

■ Microsoft product screen shots

■ Microsoft product name abbreviations

This manual uses the following abbreviations for Microsoft product names.

Full name or meaning	Abbreviation		
Microsoft(R) Windows(R) Internet Explorer(R) 6	Internet Explorer 6		
Microsoft(R) Windows(R) Internet Explorer(R) 7	Internet Explorer 7		
Microsoft(R) Windows(R) Internet Explorer(R) 8	Internet Explorer 8		
Microsoft(R) Cluster Service	Microsoft Cluster Service		
Microsoft(R) Windows(R) 7 Enterprise	Windows 7		Windows
Microsoft(R) Windows(R) 7 Professional			
Microsoft(R) Windows(R) 7 Ultimate			
Microsoft(R) Windows Server(R) 2003, Enterprise Edition Operating System (x86)	Windows Server 2003 Enterprise Edition	Windows Server 2003	
Microsoft(R) Windows Server(R) 2003, Standard Edition Operating System (x86)	Windows Server 2003 Standard Edition		
Microsoft(R) Windows Server(R) 2003 R2, Enterprise Edition Operating System (x86)	Windows Server 2003 R2 Enterprise Edition	Windows Server 2003 R2	

Full name or meaning	Abbreviation		
Microsoft(R) Windows Server(R) 2003 R2, Standard Edition Operating System (x86)	Windows Server 2003 R2 Standard Edition	Windows Server 2003 R2	Windows
Microsoft(R) Windows Server(R) 2003, Enterprise x64 Edition Operating System	Windows Server 2003 Enterprise x64 Edition	Windows Server 2003 (x64)	
Microsoft(R) Windows Server(R) 2003, Standard x64 Edition Operating System	Windows Server 2003 Standard x64 Edition		
Microsoft(R) Windows Server(R) 2003 R2, Enterprise x64 Edition Operating System	Windows Server 2003 R2 Enterprise x64 Edition	Windows Server 2003 R2 (x64)	
Microsoft(R) Windows Server(R) 2003 R2, Standard x64 Edition Operating System	Windows Server 2003 R2 Standard x64 Edition		
Microsoft(R) Windows Server(R) 2008 Enterprise 32-bit	Windows Server 2008 x86		
Microsoft(R) Windows Server(R) 2008 Standard 32-bit			
Microsoft(R) Windows Server(R) 2008 Enterprise x64 Edition	Windows Server 2008 x64		
Microsoft(R) Windows Server(R) 2008 Standard x64 Edition			
Microsoft(R) Windows Server(R) 2008 R2 Enterprise	Windows Server 2008 R2		
Microsoft(R) Windows Server(R) 2008 R2 Standard			
Microsoft(R) Windows Vista(R) Business	Windows Vista Business	Windows Vista	
Microsoft(R) Windows Vista(R) Enterprise	Windows Vista Enterprise		
Microsoft(R) Windows Vista(R) Ultimate	Windows Vista Ultimate		
Microsoft(R) Windows(R) XP Professional Operating System	Windows XP		
Process Modeler 5 for Microsoft(R) Visio Professional Edition	Process Modeler 5 for Microsoft Visio Professional Edition		

■ Restrictions

Information in this document is subject to change without notice and does not represent a commitment on the part of Hitachi. The software described in this manual is furnished according to a license agreement with Hitachi. The license agreement contains all of the terms and conditions governing your use of the software and documentation, including all warranty rights, limitations of liability, and disclaimers of warranty.

Material contained in this document may describe Hitachi products not available or features not available in your country.

No part of this material may be reproduced in any form or by any means without permission in writing from the publisher.

Printed in Japan.

■ Issued

Oct. 2015: 3020-3-Y43-40(E)

■ Copyright

Preface

For details on the prerequisites before reading this manual, see the description in the introduction for the manual "*Application Server Overview*".

Contents

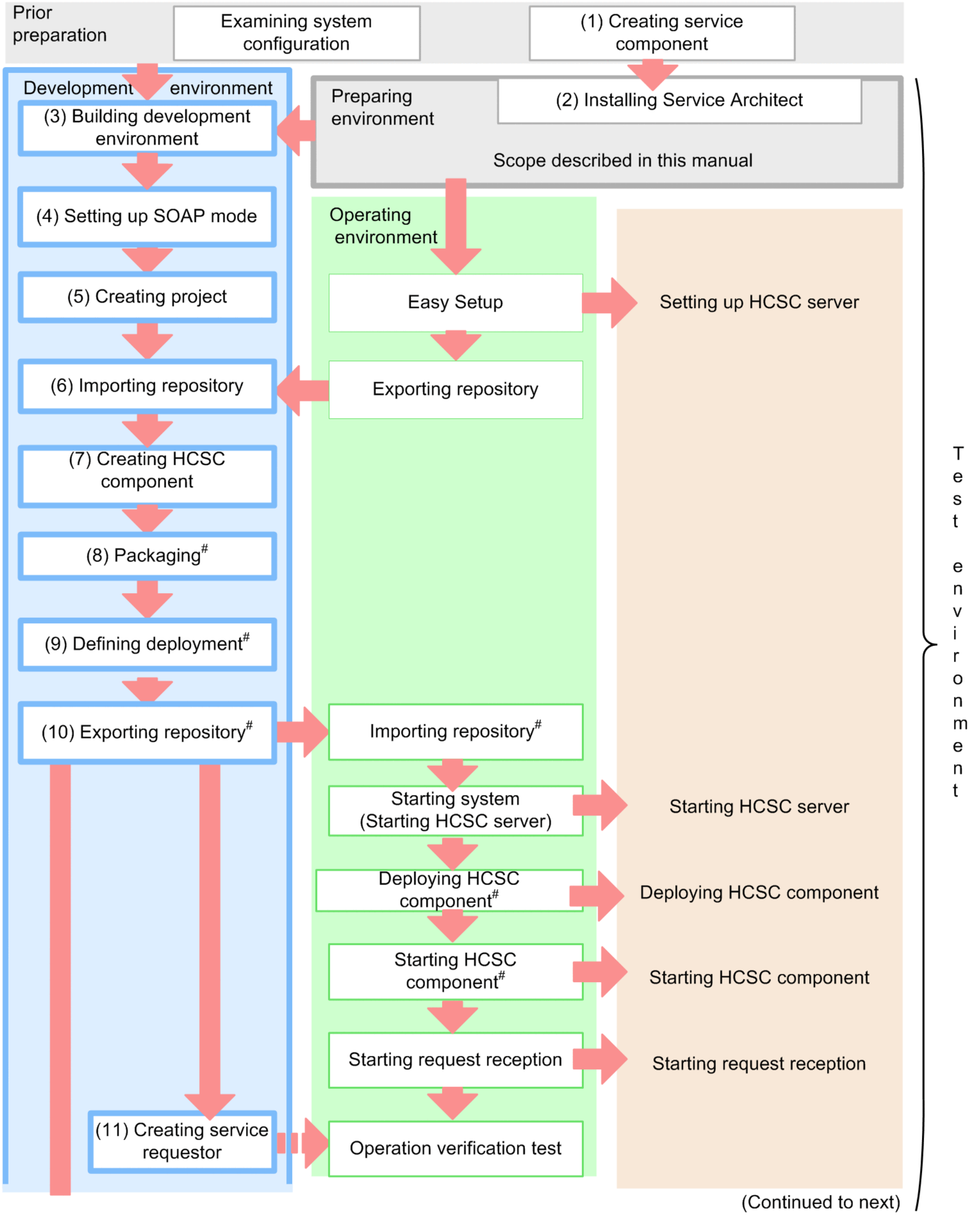
1 Overview of System Development Based on SOA

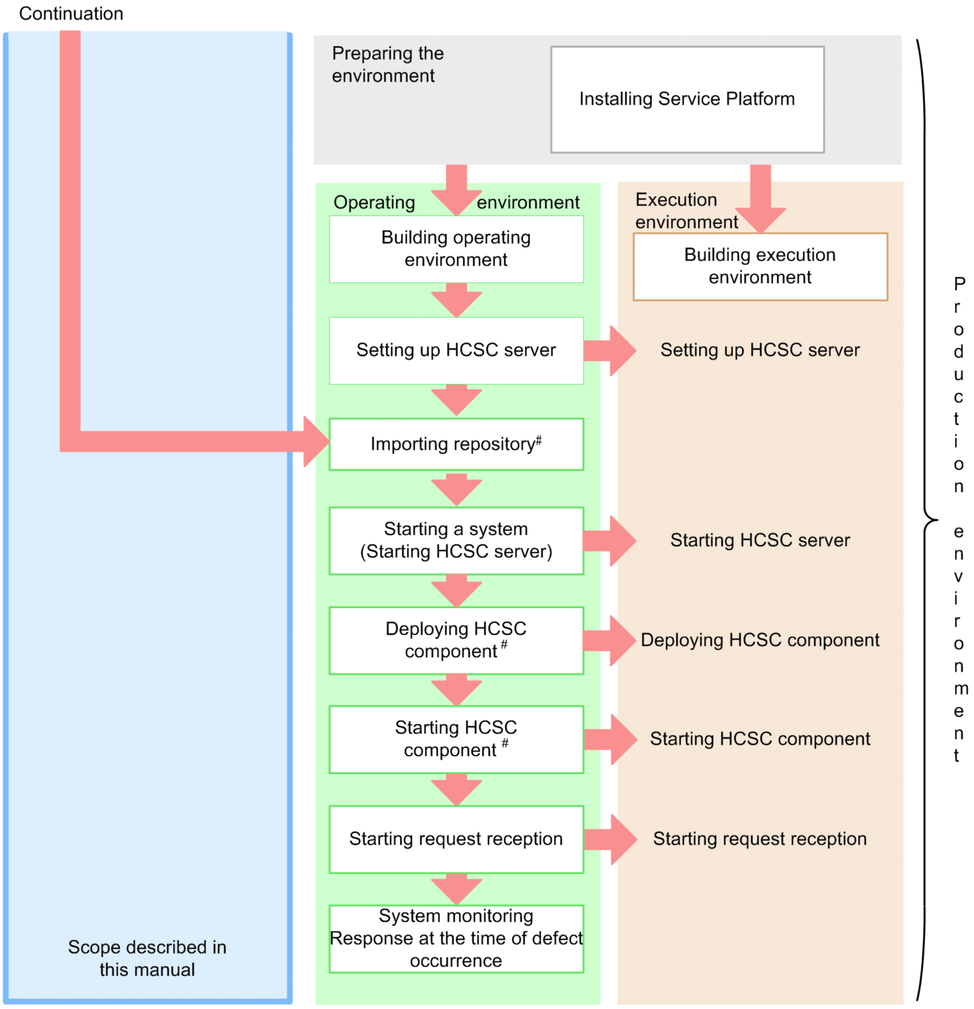
This chapter provides an overview of how a system can be developed based on SOA.

1.1 Flow from Development up to Actual Operation

The following figure shows the workflow from system development using a Cosminexus Service Platform to the actual application.

Figure 1–1: Flow from development to actual application




Continuation
Scope described in this manual
Preparing the environment
Installing Service Platform
Operating environment
Building operating environment
Setting up HCSC server
Importing repository#
Starting a system (Starting HCSC server)
Deploying HCSC component#
Starting HCSC component#
Starting request reception
System monitoring Response at the time of defect occurrence
Execution environment
Building execution environment
Setting up HCSC server
Starting HCSC server
Deploying HCSC component
Starting HCSC component
Starting request reception
Production environment

Note#

In the development environment, you can also perform these operations in a batch; however, you perform the batch execution when developing a system or during the unit testing and the integration testing. For details, see "7.5 Batch execution of processes for deploying HCSC components on the HCSC Server and then starting".

This manual explains the workflow and tasks to be performed in the development environment shown in the figure. For details about the tasks to be performed in the operating environment and execution environment, see "Service platform System Setup and Operation Guide".

The following subsections provide overviews of the individual tasks shown in the figure.

(1) Creating a service component

Create the service components to be used on the Cosminexus Service Platform. The interface information needs to be defined in a service component. For details on the types of available services, see "2.6 Types of Available Service Components and Their Application Scopes".

(2) Installing Service Architect

Install Service Architect on the machine to be used as development environment.

For details on installing, see "2.1.2 Installation".

(3) Setting up the development environment

Set up the development environment by procuring the archive file of Eclipse.

For details, see Step 2 and 3 in "2.1 Setup for Using the Development Environment".

(4) Setting up the SOAP mode

Set up the SOAP mode to be used in the system. The SOAP mode includes the SOAP1.1 mode and the SOAP1.1/1.2 combined mode.

For details about the overview and setup methods of SOAP modes, see "2.2.2 SOAP mode to be used".

(5) Creating a project

Create an HCSCTE project with HCSCTE embedded Eclipse. Create a project and set up properties before developing HCSC components.

For details on creating a project, see "3.1.1 Creating a Project" and for details on setting up properties, see " 3.1.2 Setting up Properties".

(6) Importing a repository

Import a repository that is set up in the operating environment with easy setup, into the development environment.

Import a repository set up with easy setup in operating environment, to the development environment. Import the repository that was set up with easy setup in operating environment, to the development environment.

For details on how to import a repository, see "3.2.3 Importing a Repository".

(7) Creating HCSC components

Create HCSC components such as the service adapter for invoking the already created service components, business process for invoking multiple service components and user-defined reception for receiving the execution request from the service adapter.

Also, you must create a message format, which is the format of the message for invoking service components, before defining the respective HCSC components.

(a) Creating a message format

Create a format (message format) of a message for invoking service components. The method of creating message format differs depending on the type of service component to be invoked and the type of message to be used.

For creating a message format, see "4. Creating Message Formats". For the screen to be used in creating a message format (binary format definition file), see "1.2.1 Binary Format Definition Window" in "Service Platform Reference Guide". For the dialogs to be used when creating a message format (binary format definition file), see "1.3 Dialog Boxes Related to Binary Format Definition" in "Service Platform Reference Guide".

You must use a command, to create message format for using DB adapter (service adapter for operating database). For details on the command, see "csamkxmls (Creating XML format definition file for DB adapter)" in "Service Platform Reference Guide".

(b) Defining adapters

Create adapters for invoking service components (including database manipulation) and define their details. The following two types of adapters are available: service adapters for invoking service components and database adapters for manipulating databases.

Create a service adapter for invoking service components (including database operations) and define the details.

For details on defining the service adapter, see "3. Defining Adapters" in "Service Platform Reception and Adapter Definition Guide". For the details on screens and dialogs to be used when defining the service adapter, see "1.2.2 Service Adapter Definition Window" in "Service Platform Reference Guide".

(c) Defining business processes

Create business processes and define their details.

For details on defining a business process, see "5. Defining Business Processes".

Also, for the details on screens and dialogs to be used when defining a business process, see "1.2.3 Business Process Definition window" in "Service Platform Reference Guide".

(d) Creating a data transformation definition

Crate data transformation definition when message format of the request message for invoking service component differs than the message format of the service component to be invoked.

For the details on creating the data transformation definition, see " 6. Defining Data Transformation".

Also, for the details of screens and dialogs used when creating data transformation definition, see "1.2.5 Data Transformation Definition Window" in "Service Platform Reference Guide".

(e) Defining a user-defined reception

Define user-defined reception, when you want to define any format as an interface that receives the service component execution request and returns the response. Note that the business process is the only HCSC component that can receive request with user-defined reception.

For details on defining the user-defined reception, see "2. Defining User-Defined Reception" in "Service Platform Reception and Adapter Definition Guide".

Also, for details on screens and dialogs to be used when defining the user-defined reception, see "1.2.6 User-Defined Reception Definition Window" in "Service Platform Reference Guide".

(8) Packaging

Enable packaging to the EAR file and deployment from operating environment to the execution environment for the created HCSC component.

For details on packaging of HCSC component, see "7. Packaging HCSC Components and Defining Deployment".

(9) Deployment definition

Define how to deploy the packaged HCSC components from the operating environment to the execution environment (create a deployment definition), and update the system configuration definition.

For details on the method of deployment definition, see "7. Packaging HCSC Components and Defining Deployment".

(10) Exporting the repository

Export repository for which creation, packaging and deployment definition of HCSC component has been executed, to the operating environment.

For details on method of exporting repository, see "3.2.2 Exporting a Repository".

(11) Creating a service requester

Create a service requester for receiving a service component execution request from a task owner in the execution environment and sending the execution request to the HCSC component.

For details on creating a service requester, see " 8. Creating Service Requesters".

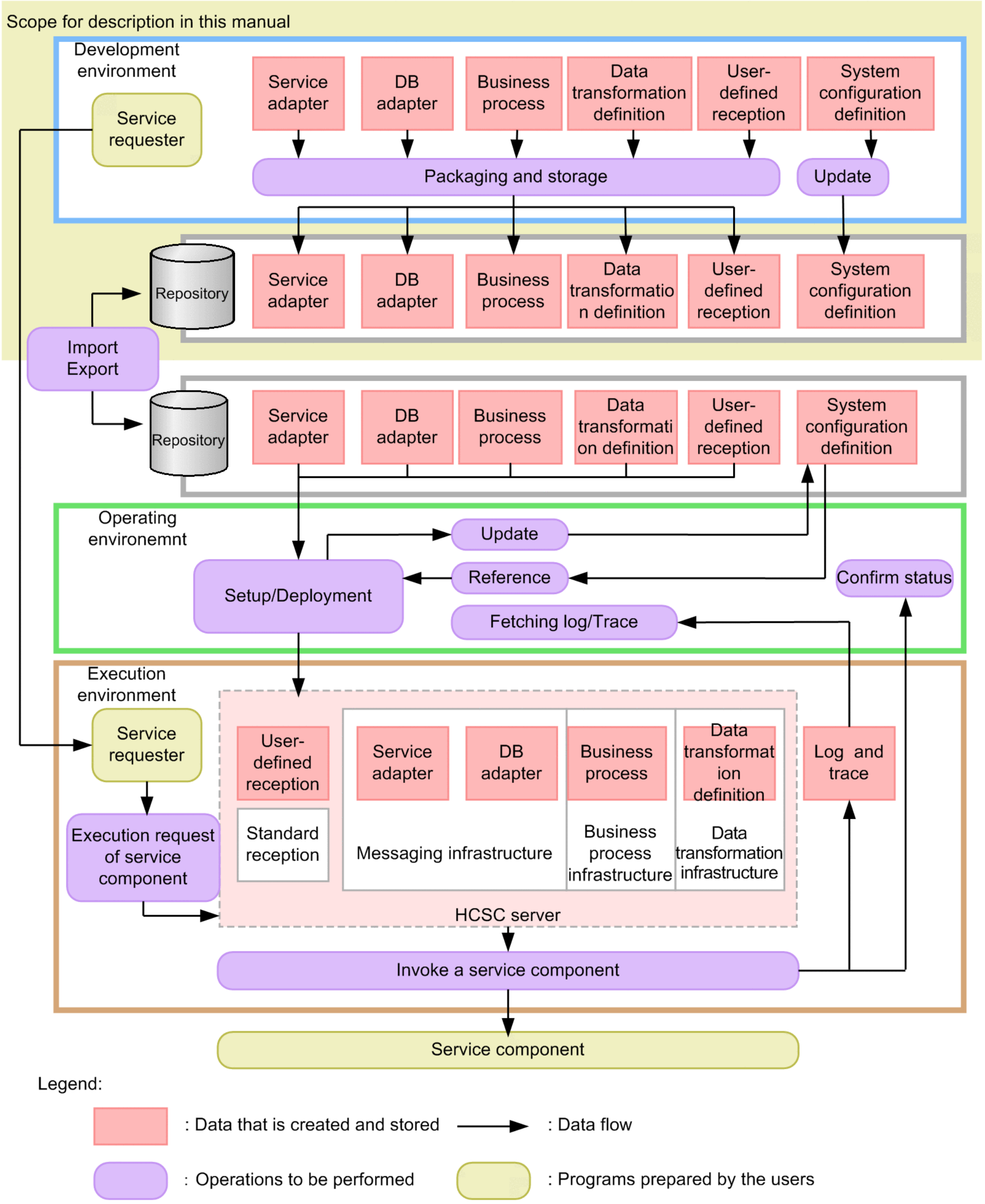
1.2 Relationship Between the Overall System and the Development Environment

A Cosminexus Service Platform consists of a development environment, an operating environment, and an execution environment, which are interrelated. The positioning of the development environment is explained below.

(1) Positioning in terms of operation

The following figure shows how the development environment is positioned in the context of the overall system operations.

Figure 1–2: Positioning in terms of operation



In the development environment, the newly created HCSC components and deployment definitions are stored in a repository, and data is transferred to the operating environment. Data that is created in the development environment and stored in the repository is deployed from the operating environment to the execution environment.

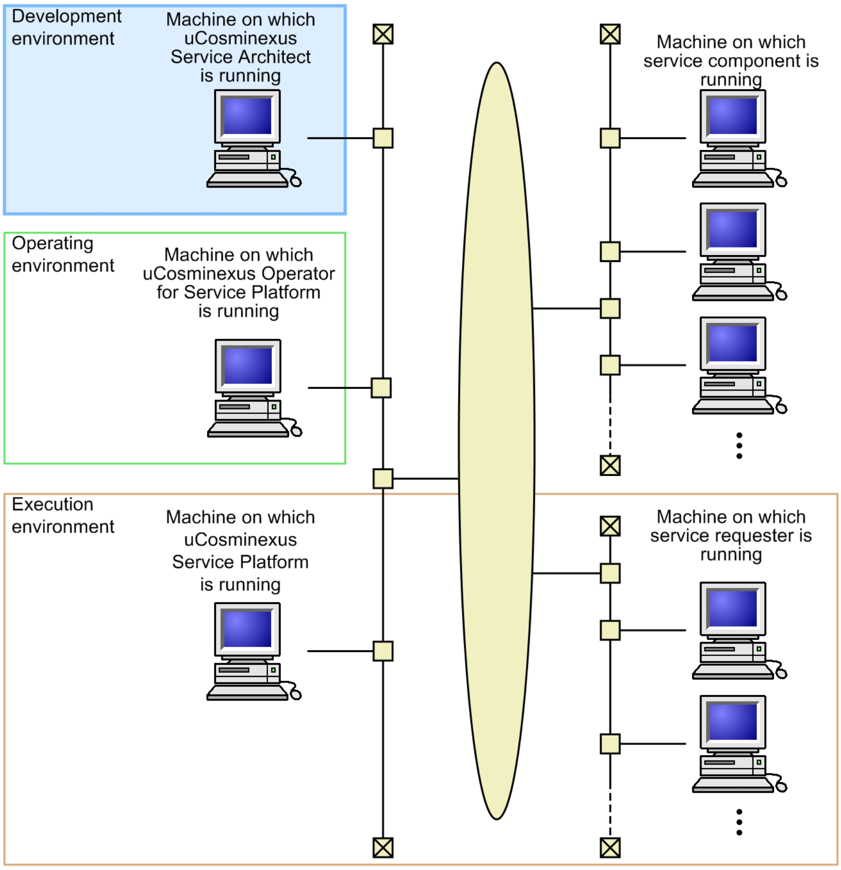
To create a deployment definition, you need the information on the HCSC server that has been set up in the operating environment. You need to use the repository to obtain this information from the operating environment. For details about how to use a repository to exchange information with the operating environment, see "3.2 Managing a Repository".

(2) Positioning within network

The development environment, the operating environment, and the execution environment are linked together using a network, such as the Internet and an intranet, to configure a system.

The following figure shows the positioning of the development environment within a network in the overall system.

Figure 1–3: Positioning within network


Development environment
Machine on which uCosminexus Service Architect is running
Operating environment
Machine on which uCosminexus Operator for Service Platform is running
Execution environment
Machine on which uCosminexus Service Platform is running
Machine on which service component is running
Machine on which service requester is running

The above mentioned example describes a case wherein the Operating environment and the execution environment are built on different machines by using uCosminexus Operator for Service Platform. You can also build the Operating environment and the Execution environment on one of the machines on which uCosminexus Service Platform is running. You can also build the development environment .on another network.

2 Before Developing a System

This chapter explains the tasks you need to perform and the items you need to know prior to developing a Cosminexus Service Platform system.

2.1 Setup for Using the Development Environment

The setup procedures for using the development environment of a Cosminexus Service Platform are described below.

1. Install uCosminexus Service Architect.

   For details, see "*2.1.2 Installation*".

2. Download an archive file of Eclipse and set up Eclipse.

   For details, see *2.4 Set up using the Eclipse setup functionality* and *2.5 Eclipse settings* in the *Application Server Application Development Guide*.

3. Build the test environment as and when required.

   When using the HCSC Easy Setup functionality

   For details, see "*2.4 Easy Setup of the Test Environment*".

   When the HCSC Easy Setup functionality is not used

   See the manual *Cosminexus Service Platform System Setup and Operation Guide*, and build the operating and execution environments.

   Also, for using an embedded database in the test environment, see "*2.1.4 Creating embedded database*".

4. Set up the environment.

   For details, see "*2.1.3 Environment Setup*".

   ! Important note

   - In a test environment, you can deploy the applications (service requester and service components) to be used for testing in the HCSC server (J2EE server).

     However, the following names are reserved, and therefore, cannot be used as the names of applications to be deployed:

     - Names beginning with `CSC`

     - Service ID of the business process and service adapter

     - Reception ID of user-defined reception
   - Make sure you use the HCSC Easy Setup functionality to debug a business process in the test environment.
   - When starting an instance of Eclipse in which HCSCTE plug-ins are embedded, make sure that the following argument is specified in the `-vmargs` option in the `eclipse.ini` file:

     `-Djava.endorsed.dirs=`*service-platform-installation-directory*`\jaxp\lib`
   - Hitachi does not provide support for Eclipse, except for the Eclipse plug-in functions provided by Application Server and Service Platform. Users are expected to investigate how to use Eclipse and take action in response to any errors displayed by Eclipse.

2.1.1 Prerequisites

The following are the prerequisites for using the development environment of Service Platform.

Users that use the development environment

Users that use the development environment must belong to the `Administrators` or `Power Users` group.

Software programs that must be installed and set up

Before using the development environment, make sure that the following software programs have been installed and set up:

- Eclipse
- WSDL4J 1.5.1#

  #: This software is required when SOAP1.1 mode is used. For details about SOAP1.1 mode, see *2.2.2 SOAP mode to be used*.

2.1.2 Installation

Use the installer to install Service Architect. Install the software according to the instructions given by the installer. Note that the installation work must be performed by a user with Administrator privileges. For details, see *2.2.2 Installing new Application Server (In Windows)* in the *Application Server System Setup and Operation Guide*. Note that the product name *Application Server* in the installation procedure must be replaced with *Service Platform*.

To specify the installation destination, specify a path name of up to 50 alphanumeric characters.

In the window for selecting the installation type, select **Standard**.

After installing Service Architect, restart the system.

Tip

When upgrading from a previous version, you can migrate the repository information used in the previous environment to a format that enables the repository information to be used in the upgraded version.

For details about the migration method, see *Appendix A. Migrating from an Earlier Version*.

2.1.3 Environment Setup

To set up the environment in which HCSCTE is embedded in Eclipse, perform the following tasks:

- Setting up the SOAP mode
- Creating an HCSCTE project
- Setting up properties (setting up repositories and validation)

For details about the environment setup, see "*2.2.3 SOAP mode settings*" and "*3. Managing Project and Managing Repository*".

Reference note

For displaying a message format and in the case of message format definition file with multi-byte characters, perform the following operations and change the encoding to UTF-8:

1. Select **Window**, **Settings**, **General**, **Workspace**, and then **Encode text file**.
2. Set **Others (UTF-8)**.

2.1.4 Creating embedded database

Service Architect includes embedded database that can be used as test environment to test the developed HCSC components.

If you use embedded database to set up the HCSC server for testing in the development environment, you need not arrange or set up a separate DB server.

! Important note

- You need to set up embedded database explained hereafter, when you want to set up the test environment by using the HCSC easy setup function.

  For details on HCSC easy setup function, see "*2.4 Easy Setup of the Test Environment*".
- Create HCSC server and embedded database for testing, on the same machine having development environment. You cannot use embedded database, by setting up DB server on another machine.

  Embedded database is based on HiRDB. When using other database (Oracle), set up DB server on another machine.
- When you have set up the embedded database, with HCSC easy setup function, you cannot connect to the embedded database remotely from an external machine. When you want to operate embedded database by using HiRDB SQL Executer, start HiRDB SQL Executer on the test environment machine and specify "localhost" in the host name of [CONNECT] dialog.

The following points describe the settings required for using the embedded database.

(1) Settings on DB server

The embedded database is installed at the stage of installing Service Architect. Therefore, you need not install embedded database separately.

You must execute the following tasks for using the embedded database.

- Installing HiRDB SQL Executer and setting up environment variables
- Setting up DB server and configuring the environment
- Setting up users and defining schemas
- Preparing RDAREAs
- Restarting the embedded database

Each of the above-listed tasks is explained as follows:

(a) Installing HiRDB SQL Executer and setting up environment variables

Install HiRDB SQL Executer included in the embedded database.

For details on how to install HiRDB SQL Executer when creating the embedded database, see "3.2 Installing HiRDB SQL Executer" in the "Application Server Application Development Guide".

Once the installation is complete, add/setup the following environment variable PATH.

```
<Service platform installation directory>\DB\BIN;
<Service platform installation directory>\DB\CLIENT\UTL;
```

(b) Setting up DB server and configuring the environment

Use Eclipse for setting up DB server and configuring environment of the embedded database.

For the method to set up DB server and configure the environment, see "Appendix B.3 Creating embedded database" in "Application Server Application Development Guide".

In DB server setup settings screen, select [Large (2GB)] as the size of the created database.

(c) Setting up users and defining schemas

Define the user and schema that will use the database.

For the method to set up the user and define schema of the embedded database, see "Appendix B.3 Creating the embedded database" in "Application Server Application Development Guide".

(d) Preparing RDAREAs

Create, add or expanded RDAREAs as and when required, to store the management information table of HCSC Messaging. Also, when you create, add or expanded RDAREAs, check whether it is processed properly.

Procedure for creating, adding or expanding RDAREAs is as follows:

1. If the connection information of HiRDB server which is different than the connection information of the embedded database, is set up in the OS system variables (PDHOST, PDNAMEPORT, PDUSER), delete that information with set command.
2. From [Start] menu of Windows, select [Cosminexus]-[Database console].

   Environment variables that operate the embedded database are set.
3. Run the following command, depending on the operations.
   - pdfmkfs command (for creating RDAREAs)
   - pdmod command (for adding and expanding RDAREAs)
   - pddbls command (for displaying the status of RDAREAs)

   For the above-mentioned commands, see the "HiRDB Command Reference".
4. When you add or expand RDAREAs, add the pdbuffer operand of DB definition file (HiRDB system definition file) of embedded database and expand the global buffer of the embedded database, as and when needed.

   DB definition file of the embedded database is under the DB definition file directory specified at the time of setting up the DB server.

(e) Restarting the embedded database

Stop and restart the embedded database.

For the method to stop and restart the embedded database, see "6.2 Starting and stopping the embedded database" in "Application Server Application Development Guide".

(2) Settings in DB client (HCSC server)

When using the embedded database, you must perform following operations, as the settings of DB client (HCSC server).

- Setting up environment variables of DB client
- Setting up the environment variables group
- Setting up the concurrent connections count of embedded database
- Setting up user limitation and preparing RDAREAs for Reliable Messaging

Each operation is explained as follows:

(a) Setting up environment variables of DB client

Set up the environment variables `PDXAMODE` and `PDTXACANUM`.

(b) Setting up environment variables group

Register the environment variables group with "Client environment variable registration tool" in the following path.

<*Service platform installation directory*>`\DB\CLIENT\UTL`

For the details on registering the environment variables group with the client environment variable registration tool, see the "HiRDB UAP Development Guide".

Specify the registered environment variable group name with `XA_OPEN` character string of the DB Connector property. For `XA_OPEN` character string (`XAOpenString`) of DB Connector properties, see "<4.2.2 Defining DB Connector properties" in "Application Server Application Setup Guide".

(c) Setting up the concurrent connections count of embedded database

Increase the concurrent connections count of embedded database, as and when required. Change the operand of the DB definition file of embedded database. DB definition file exists in directory to be set up through the settings screen of DB server setup, at the time of implementing work of "*(1)(b) Setting up DB server and configuring the environment*".

Change the following operands in the DB definition file.

- `pd_max_users` (maximum concurrent connections count)
- `pd_max_server_process` (maximum number of concurrently started server processes)
- `pd_process_count` (number of resident processes)

Also, set the value of environment variable `PDTXACANUM` to the value greater than the specified value of `pd_max_users`.

(d) Setting up user limitation and preparing RDAREAs for Reliable Messaging

Set up user limitation and prepare RDAREAs for Reliable Messaging, as and when required.

The method to set up user limitations is same as the method given in "*(1)(c) Setting up users and defining schemas*".

The method to prepare RD area is same as the method given in "*(1)(d) Preparing RDAREAs*".

2.1.5 Uninstalling

This section describes the method to uninstall Service Architect. For details, see "3.3 Unsetting up the system environment and uninstalling Application Server" in "Application Server System Setup and Operation Guide". Read the product name "Application Server" in the uninstallation procedure, as "Service Platform".

User with Administrator privileges must perform uninstallation.

Unset up the Eclipse environment, before implementing uninstallation. For unset up procedure, see "2.8.2 Unsetting up Eclipse environment" in "Application Server Application Development Guide".

Implement uninstallation after stopping the components of execution environment such as J2EE server, Management Server, PRF. When using embedded database, first stop the embedded database and then implement uninstallation.

Uninstallation procedure is as follows:

1. Execute either of the following operations.
   - From [Start] menu of Windows, select [Cosminexus] - [Uninstall uCosminexus Service Architect].
   - Select [uCosminexus Service Architect] from the [Add or Remove Programs] of [Control Panel].

   Dialog for confirming uninstallation of Service Architect is displayed.
2. Click the [Yes] or [No] button.

   When you click the [Yes] button

   Uninstallation starts and all the configuration software of Service Architect are deleted.

   When you click the [No] button

   Dialog for selecting configuration software to be uninstalled is displayed. When you select the configuration software to be uninstalled and click the [Next] button, uninstallation starts and the selected configuration software is deleted.

2.2 Selecting the configuration format and the SOAP modes

Before you develop a system on the Cosminexus Service Platform, you must clearly specify the functionality and SOAP modes that will be used in the system.

This section describes the functionality used in the system and the usage of the database and Cosminexus RM. This section also describes the standard specifications for the Web Services corresponding to the SOAP modes used.

2.2.1 Usage existence of database and Reliable Messaging

Usage existence of database and Reliable Messaging differ depending the functions of service platform to be used and operations to be performed. Following table shows the usage of database and Reliable Messaging for each function to be used.

Table 2–1: Usage of database and Reliable Messaging for each function to be used

Function to be used		Database	Reliable Messaging
Following synchronous reception is to be used • Standard reception (Web service) • Standard reception (SessionBean) • SOAP reception • TP1/RPC reception • FTP reception • HTTP reception • Message Queue reception		N	N
Following asynchronous reception is to be used • Standard reception (MDB(WS-R)) • Standard reception (MDB(DB queue)) • Message Queue reception		N	N
Following synchronous service adapters are to be used • SOAP adapter • SessionBean adapter • TP1 adapter • File adapter • Object Access adapter • Message Queue adapter • FTP adapter • File operation adapter • Mail adapter • HTTP adapter		N	N
Following asynchronous service adapters are to be used • MDB (WS-R) adapter • MDB (DB queue) adapter		Y	Y
DB adapter is to be used		N#	N
Business process is to be used	Persistence business process is to be used	Y	N
	Non-persistence business process is to be used	N	N

Function to be used	Database	Reliable Messaging
Process instance execution history is to be managed	Y	N

(Legend)

Y: Mandatory

N: Not mandatory.

Note #

When you use DB adapter, database is not required on the machine on which HCSC server operates. However, database is required on the service component operation machine connected from DB adapter.

When you use non-persistence business process, take note that the processes that you can define in the development environment have a limitation. For details, see "*5.2.1(3) Setting up status persistence*".

From above mention table, one can understand that the configuration status depending on the usage existence of database and Reliable Messaging has following 3 patterns:

- Using database and Reliable Messaging both
- Not using database and Reliable Messaging both
- Using database but not using Reliable Messaging

Settings contents of test environment differ depending on the pattern by which the concerned environment is set up. For the details on usage existence of database and Reliable Messaging to be set up in test environment and the concerned environment, see "1.3 Relationship between Test Environment and Production Environment" in "Service Platform System Setup and Operation Guide". Also, for setting contents in test environment, see "*2.4 Easy Setup of the Test Environment*".

2.2.2 SOAP mode to be used

Select the SOAP mode to be used, before developing system with the service platform.

This section describes the type of SOAP mode and support range of each mode.

(1) Types of SOAP mode

Types of SOAP mode and standard environment as well as execution environment of the corresponding Web service are as follows:

- **SOAP1.1 mode**
  Select this mode to develop a system corresponding to SOAP1.1. SOAP1.1 mode corresponds to WS-I Basic Profile1.0a.
  SOAP1.1 mode uses the SOAP communication base for transmission of SOAP messages.
- **SOAP1.1/1.2 combined mode**
  Select this mode to develop a system corresponding to SOAP1.1 or SOAP1.2. SOAP1.1/1.2 combined mode corresponds to WS-I Basic Profile1.1.
  SOAP1.1/1.2 combined mode uses JAX-WS engine for transmission of SOAP messages.

(2) Support range of the SOAP modes

Following table shows the functions of reception, service adapter and business process as well as correspondence of SOAP mode:

Table 2–2: Support range of the SOAP modes

Classification	Function name	SOAP mode	
		1.1	1.1/1.2 combined
Reception	Standard reception(Web service(SOAP1.1))	Y	Y

Classification	Function name	SOAP mode	
		1.1	1.1/1.2 combined
Reception	Standard reception(Web service(SOAP1.2))	N	Y
	Standard reception(SessionBean)	Y	Y
	Standard reception(MDB(WS-R))	Y	N
	Standard reception(MDB(DB queue))	Y	Y
	User-defined reception	Y	Y
Service adapter	SOAP adapter	Y	Y
	SessionBean adapter	Y	N
	MDB(WS-R)adapter	Y	N
	MDB(DB queue)adapter	Y	Y
	DB adapter	Y	Y
	TP1 adapter	Y	Y
	File adapter	Y	Y
	Object Access adapter	Y	Y
	Message Queue adapter	Y	Y
	FTP adapter	Y	Y
	File operation adapter	Y	Y
	Mail adapter	Y	Y
	HTTP adapter	Y	Y
	Custom adapter	Y	Y
Business process	Business process	Y	Y

(Legend)

Y: Supports.

N: Does not support.

Following table describes the correspondence between WSDL definition style and the SOAP mode:

Table 2–3: Support range of the SOAP modes (WSDL definition style)

SOAP version	WSDL definition style	SOAP mode	
		1.1	1.1/1.2 combined
SOAP1.1	`rpc/literal`	Y	N
	`document/literal`	Y	Y
SOAP1.2	`rpc/literal`	N	N
	`document/literal`	N	Y

Legend:

Y: Supports.

N: Does not support.

For the relation between service components that use SOAP mode and Web service, see "*2.6.1 Applicability of the service components that use Web service*".

2.2.3 SOAP mode settings

You can set up the SOAP mode using the HCSC Easy Setup screen or the `cscsoapmode` command.

You can set up the SOAP mode for the development environment and execution environment using the HCSC Easy Setup screen. For details about the settings in the HCSC Easy Setup screen, see "*2.4.2(1)(c) Input items of HCSC easy setup screen*".

You can set up the SOAP mode for the development environment using the `cscsoapmode` command. For details about how to set up the SOAP mode for the execution environment, see the manual *Cosminexus Service Platform System Setup and Operation Guide*. For details about the `cscsoapmode` command, see the manual *Cosminexus Service Platform Reference*.

You can set up the SOAP mode for the development environment using the `cscsoapmode` command. For details about how to set up the SOAP mode for the execution environment, see "Service platform System Setup and Operation Guide". For details about the `cscsoapmode` command, see "Service Platform Reference Guide".

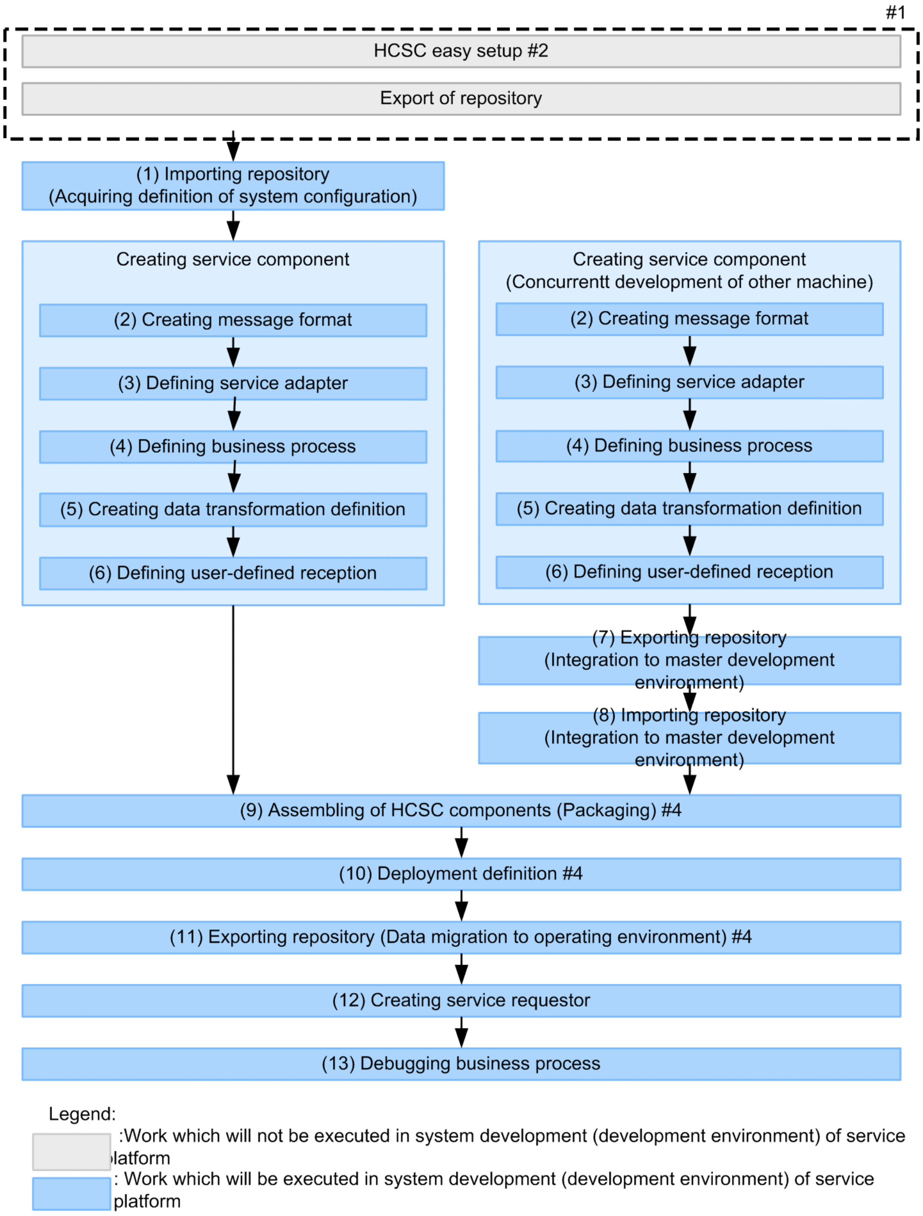
For details about how to check the specified SOAP mode, see *2.4.5 Checking the SOAP modes*.

For details about how to change the specified SOAP mode, see *3.1.5 Changing SOAP modes*.

2.3 Development Work Flow

The following figure shows the workflow for using a Cosminexus Service Platform to develop a system.

Figure 2–1: System development work flow


#1
HCSC easy setup #2
Export of repository
(1) Importing repository
(Acquiring definition of system configuration)
Creating service component
(2) Creating message format
(3) Defining service adapter
(4) Defining business process
(5) Creating data transformation definition
(6) Defining user-defined reception
Creating service component
(Concurrentt development of other machine)
(2) Creating message format
(3) Defining service adapter
(4) Defining business process
(5) Creating data transformation definition
(6) Defining user-defined reception
(7) Exporting repository
(Integration to master development environment)
(8) Importing repository
(Integration to master development environment)
(9) Assembling of HCSC components (Packaging) #4
(10) Deployment definition #4
(11) Exporting repository (Data migration to operating environment) #4
(12) Creating service requestor
(13) Debugging business process
Legend:
:Work which will not be executed in system development (development environment) of service platform
: Work which will be executed in system development (development environment) of service platform

Note #1

For HCSC easy setup, see "1.3 Relationship between Test Environment and Production Environment" in "Service Platform System Setup and Operation Guide". For exporting the repository, see "4.2 Exporting Repository Information" in "Service Platform System Setup and Operation Guide".

Note #2

Create the test environment, using HCSC easy setup. For HCSC easy setup, see " *2.4 Easy Setup of the Test Environment*". If you do not use the HCSC easy setup function, see the "Service Platform System Setup and Operation Guide" to operate and create the execution environment.

Note #3

For the definition of service adapter and user-defined reception, see "2. Defining user-defined reception" and "3. Defining Adapters" in "Service Platform Reception and Adapter Definition Guide".

Note#4

You can execute above-mentioned work in a batch. However, batch execution is possible at the time of system development or in the period from unit test through combined test. Deployment and start of HCSC components implemented in operating environment is executed along with the above-mentioned batch execution. For details, see "*7.5 Batch execution of processes for deploying HCSC components on the HCSC Server and then starting*".

Work to be performed in each process is as follows:

(1) Importing the repository (acquiring system configuration definition)

Import only the system configuration definition, to integrate the information of HCSC server created in operating environment, with the master development environment. For details on importing the repository, see "*3.2.3 Importing a Repository*".

(2) Creating message format

Define a format of messages (message format) exchanged between the service adapter and service components. Created message format differs depending on whether XML format data or binary format (other than XML format) data is to be handled in message used for executing service component. For details on creating message format, see "*4. Creating Message Formats*".

(3) Defining a service adapter

When you develop a system using a service adapter, add the service adapter by using a wizard for adding a service adapter as well as already defined service adapter. Define the added service adapter, by using the Define service adapter screen. For details on defining a service adapter, see "3. Defining Adapters" in "Service Platform Reception and Adapter Definition Guide".

(4) Defining a business process

When you develop a system by using a business process, add the business process by using the wizard for adding a business process or already defined business process. Define the added business process, by using the Define business process screen. For details on defining a business process, see "*5.Defining Business Processes*".

Also, when you receive a business process execution request, with non-standard reception, you must define the user-defined reception. For details on defining a user-defined reception, see "2. Defining User-Defined Reception" in "Service Platform Reception and Adapter Definition Guide".

(5) Creating data transformation definition

Define the data transformation, when data transformation is required in message exchange with service component. For details on the data transformation definition, see "*6. Defining Data Transformation*".

(6) Defining the user-defined reception

Define the user-defined reception, to receive service component execution requests and define any format as the interface that returns the response. For details on defining the user-defined reception, see "2. Defining User-Defined Reception" in "Service Platform Reception and Adapter Definition Guide".

(7) Exporting a repository (integrating to master development environment)

When you develop HCSC components in multiple development environments (distributed development), export the repository in which HCSC components developed in environment other than master development environment are included.

For exporting the repository, see " *3.2.2 Exporting a Repository*".

(8) Importing a repository (integrating to master development environment)

To integrate the information of HCSC component for which distributed development in performed on machines other than master development environment, to the master development environment, import a part of repository exported in " *(7) Exporting a repository (integrating to master development environment)*" to the master development environment. For importing a repository, see " *3.2.3 Importing a Repository*".

(9) Assembling (packaging) HCSC components

Consolidate the definition files related to definition of HCSC components defined from "*(3) Defining a service adapter*"*(6) Defining the user-defined reception*" as well as the files provided by execution environment, and perform packaging in EAR file. For details on packaging, see "7. Packaging HCSC Components and Defining Deployment".

(10) Deployment definition

Define the information of cluster (or unit HCSC server) that deploys HCSC components. For details on deployment definition, see "*7. Packaging HCSC Components and Defining Deployment*".

Reference note

Before deploying the procedure to check Usage existence of database and Reliable Messaging, set in the cluster (or unit HCSC server) to be deployed, is as follows:

1. From menu of Eclipse, select [HCSC-Definer]-[System configuration definitions list].

   List of clusters to be deployed (or unit HCSC servers) is displayed in the system configuration definitions list of the Tree view.
2. From the list of clusters (or unit HCSC servers) to be deployed, select the cluster (or unit HCSC server) to be checked and double click the same.

   Information about usage existence of database and Reliable Messaging set in the cluster (or unit HCSC server) to be deployed is displayed.

(11) Exporting repository (data migration to operating environment)

Export the repository that stores the definition file required for deploying HCSC components, in master development environment. Information exported from the master development environment is imported to operating environment. For exporting repository, see "*3.2.2 Exporting a Repository*".

(12) Creating a service requester

Create a service requester for sending the message that requests HCSC components to execute service components. You can create a service requester and define a service adapter or a business process at the same time. When you want to create concurrently, check the interface information to be set in the definition of service adapter or business process beforehand and set up the same information in service requester as well. For details on how to create HCSC components, see "*8. Creating Service Requesters*".

(13) Debugging business processes

Test and debug the defined business processes in development environment, before using them in operating environment. For details on debugging business processes, see "*9. Debugging Business Processes*".

2.4 Easy Setup of the Test Environment

Hitachi recommends that you test and debug the adapters and business processes developed with uCosminexus Service Architect before deploying them in the execution environment and starting actual operations. It is, therefore, necessary to build an operating and execution environment (test environment) for testing and debugging.

The uCosminexus Service Architect provides the *HCSC Easy Setup functionality* that supports the building of a test environment. You need to set a variety of information in the operating and execution environment for testing and debugging, so the building of a test environment takes a reasonable amount of time; however if you use the HCSC Easy Setup functionality, all the required information can be automatically set for the test environment. The HCSC Easy Setup functionality also has functionality to automatically unsetup the test environment set up by using the HCSC Easy Setup functionality.

! Important note

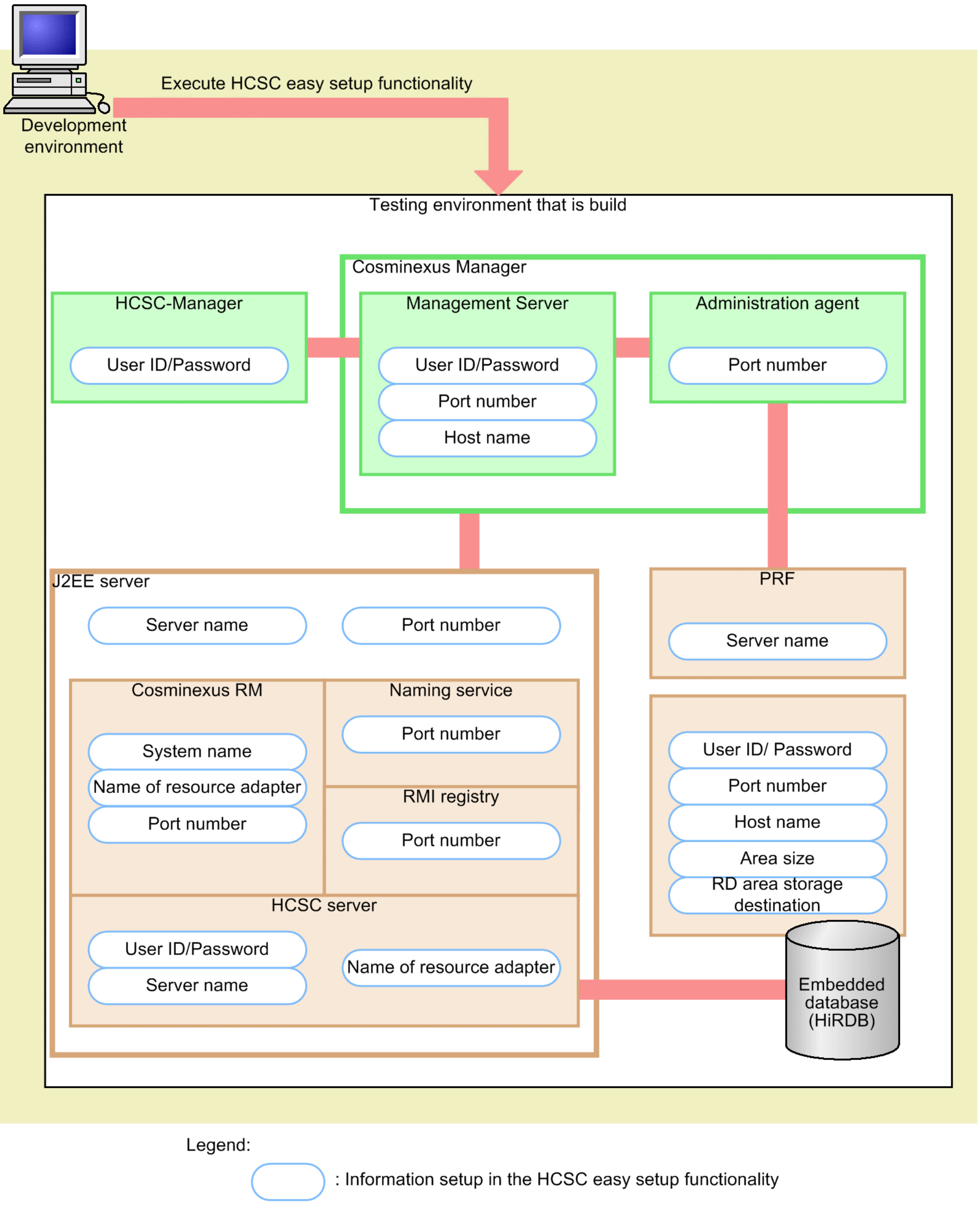
You cannot use the HCSC Easy Setup functionality (`csceasysetup` command) when setting up the environment variable `CSCMNG_HOME` for configuring multiple operating or execution environments (testing environments) on a single machine.

This section provides an overview of the HCSC Easy Setup functionality and explains how to use this functionality.

2.4.1 Environment that can be Built with the HCSC Easy Setup Functionality

A test environment built with the HCSC Easy Setup functionality has the configuration shown in the figure below. If you want to use processes other than those shown in the following figure in the test environment, separate settings will be required.

Figure 2–2: Configuration of test environment built with the HCSC Easy Setup functionality



If you use the HCSC Easy Setup functionality, the information such as the user ID, password, port number, and host name can be automatically set up for each process of the test environment that is shown in Figure 2-2. For details about the information set up in the test environment when you use the HCSC Easy Setup functionality, see "*2.4.3(2) Information required for operating the test environment*".

The number of concurrent executions of the Web containers in an environment set up using the HCSC Easy Setup functionality is 10. Therefore, if 11 or more requests are processed concurrently, the Web container threads might deplete. To change the number of concurrent executions of the Web containers, change the definition of the `webserver.connector.inprocess_http.max_execute_threads` property in the `simple_model.xml` file or `simple_model_cjl.xml` file.

For details about the `webserver.connector.inprocess_http.max_execute_threads` property, see the manual *Cosminexus Application Server Function Guide - Basic Development for Web Container*.

2.4.2 Executing HCSC easy setup functionality

This section describes the pre-conditions for setting up and unsetting up the test environment using HCSC easy setup functionality and also the methods for setting up and unsetting up the test environment.

(1) Setting up the test environment

This section describes pre-conditions and method for setting up the test environment using HCSC easy setup functionality.

(a) Pre-conditions

Following are pre-conditions for setting up the test environment.

Pre-requisite software for test environment

Following software must be installed for setting up the test environment by using HCSC easy setup functionality.

- Component Container#1
- XML Processor#1
- TPBroker#1
- Performance Tracer#1
- Reliable Messaging#1
- Service Coordinator#1
- WSDL4J 1.5.1#2

Note #1

This software is configuration software of Service Architect. You need not install the above mentioned software again, in case of an environment where Service Architect is installed.

If installing above software again, install from Installer. Install the software as per instructions of Installer. The user with Administrator privileges must perform the installation.

Note #2

Install WSDL4J 1.5.1 after installing the other pre-requisite software. WSDL4J 1.5.1 is required for using SOAP1.1 mode.

Status of system when using HCSC easy setup functionality

When you setup the test environment using HCSC easy setup functionality, status of the system must be as follows:

- Immediately after Service platform is newly installed
- Settings related to service platform should not have been performed.

However, value of variable name "`TZ`" should be valid in the system environment variables. Set time zone in the environment variable "`TZ`". Specify "`JST-9`" in the time zone.

If you customize the test environment after setup, the customized information is deleted at the time of unset up. If you do not want the customized information to be deleted, register the parameter information of definition files, which you added or changed at the time of customizing.

Pre-conditions when using database and Reliable Messaging both

Following conditions must be fulfilled when you use both database and Reliable Messaging (when you select [Model with DB/RM] in HCSC easy setup screen).

- User must have knowledge of HiRDB
- User must not implement high load testing or performance measurement

Note

After you have unsetup the test environment, you can set up the test environment again by using HCSC easy setup functionality. For unsetting up the test environment, see "*2.4.2(2) Unsetting up the test environment*".

(b) Setup method

Use HCSC easy setup screen for setting up the test environment by using HCSC easy setup functionality. Following is the method to set up the test environment.

1. From [Start] menu of Windows, select [Cosminexus]-[Test building]-[Test environment setup].
   HCSC easy setup screen is displayed.
2. Input the required information in HCSC easy setup screen.
   For items to be input in HCSC easy setup screen, see " *2.4.2(1)(c) Input items of HCSC easy setup screen*".
3. Click [Setup] function.
   Setup of the test environment is started. Setup state is displayed in the console of HCSC easy setup screen. When "`Setup of the HCSC Easy Setup functionality is complete`" is displayed on console, it implies that the test environment is successfully set up.

! Important note

> When error is displayed on the console, and test environment setup ends with an error, you must perform the setup again. When setup ends with an error, procedure of re-setup differs depending on enabled or disabled status of [Setup] button on HCSC easy setup screen.
>
> **When [Setup] button is enabled**
>
> Click [Setup] button, to perform setup again.
>
> **When [Setup] button is disabled**
>
> Unset up once by clicking [Un-setup] button and then perform the setup again.

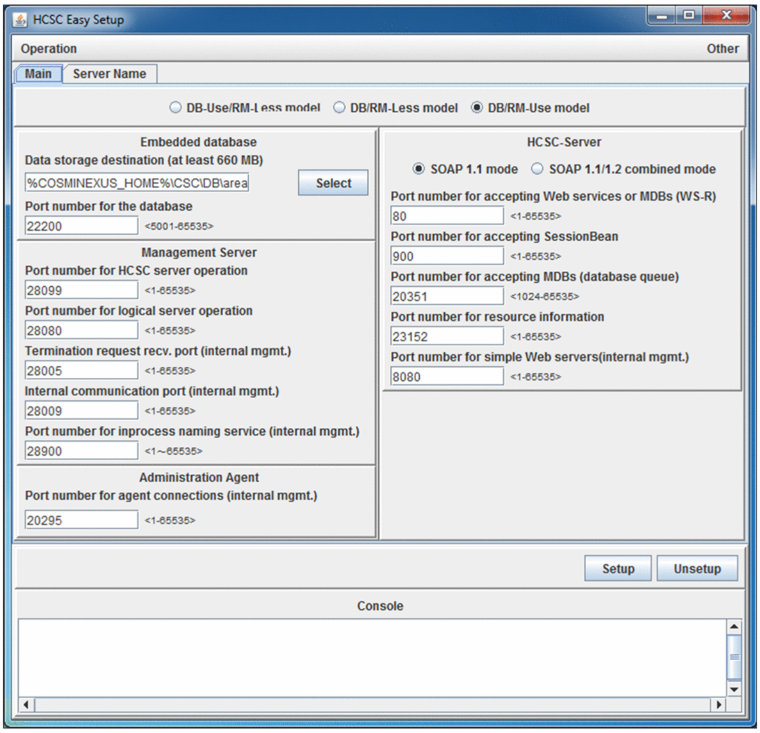
(c) Input items of HCSC easy setup screen

HCSC easy setup screen is divided in [Main] tab and [Server name] tab. With [Server name] tab, you can change the information like server name or class name at the time of setup.

This section describes the input items in HCSC easy setup screen, by using the items' relation with each process in the environment to be configured with HCSC easy setup functionality.

Following figure shows HCSC easy setup screen.

Figure 2–3: HCSC easy setup screen ([Main] tab)



○ DB-Use/RM-Less model ○ DB/RM-Less model ● DB/RM-Use model

Embedded database

Data storage destination (at least 660 MB): %COSMINEXUS_HOME%\CSC\DB\area

Port number for the database: 22200 <5001-65535>

Management Server

Port number for HCSC server operation: 28099 <1-65535>

Port number for logical server operation: 28080 <1-65535>

Termination request recv. port (internal mgmt.): 28005 <1-65535>

Internal communication port (internal mgmt.): 28009 <1-65535>

Port number for inprocess naming service (internal mgmt.): 28900 <1~65535>

Administration Agent

Port number for agent connections (internal mgmt.): 20295 <1-65535>

HCSC-Server

● SOAP 1.1 mode ○ SOAP 1.1/1.2 combined mode

Port number for accepting Web services or MDBs (WS-R): 80 <1-65535>

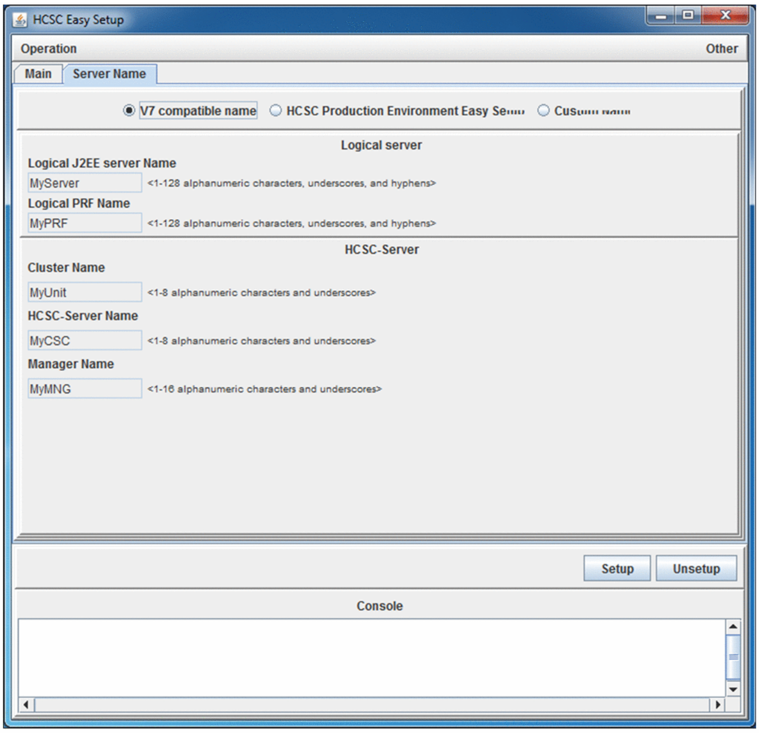
Port number for accepting SessionBean: 900 <1-65535>

Port number for accepting MDBs (database queue): 20351 <1024-65535>

Port number for resource information: 23152 <1-65535>

Port number for simple Web servers(internal mgmt.): 8080 <1-65535>

Figure 2–4: HCSC easy setup screen ([Server name] tab)



● V7 compatible name ○ HCSC Production Environment Easy Se ○ Cus

Logical server

Logical J2EE server Name: MyServer <1-128 alphanumeric characters, underscores, and hyphens>

Logical PRF Name: MyPRF <1-128 alphanumeric characters, underscores, and hyphens>

HCSC-Server

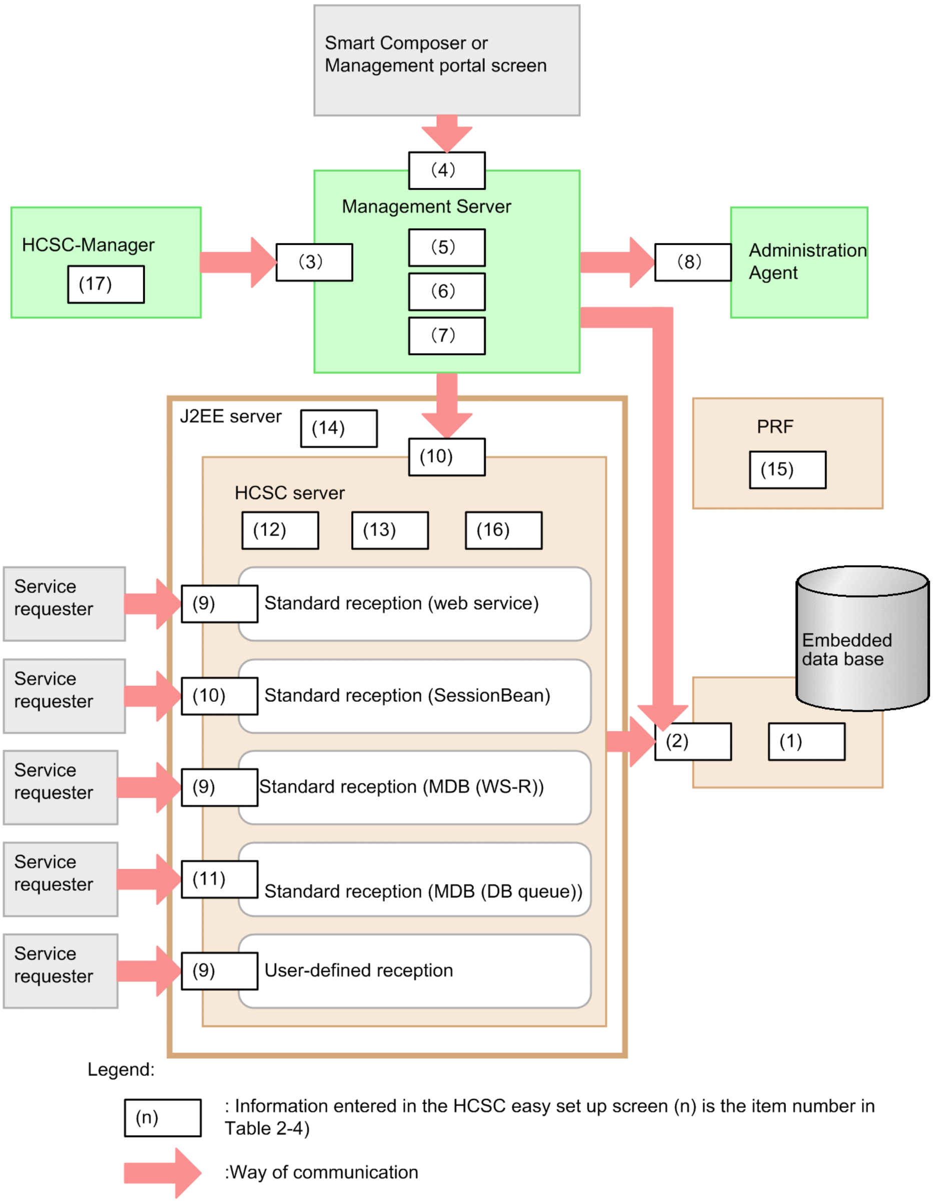
Cluster Name: MyUnit <1-8 alphanumeric characters and underscores>

HCSC-Server Name: MyCSC <1-8 alphanumeric characters and underscores>

Manager Name: MyMNG <1-16 alphanumeric characters and underscores>

As the following figure shows, the items that are input in HCSC easy setup screen are set in each process of the environment to be configured. Following figure shows the relation between input items and the values that are set.

Figure 2–5: Relation of input items in HCSC easy setup screen



Items to be set differ depending on which of [Model with DB/without RM], [Model without DB/RM] or [Model with DB/RM] is selected. Following table describes details of setting values (input items in HCSC easy setup screen) shown in (n) of Figure 2-5.

Table 2–4: Items to be input in HCSC easy setup screen

Input items				Description	Initial value#2
Tab	Classification	Item No.	Item names in HCSC easy setup screen#1		
Main	Model	--	Model with DB/without RM	Select when you want to use database and not use Reliable Messaging.	Selected
		--	Model without DB/RM	Select when you do not want to use database and Reliable Messaging both.	--
		--	Model with DB/RM	Select when you want to use database and Reliable Messaging both.	--

Input items				Description	Initial value#2
Tab	Classification	Item No.	Item names in HCSC easy setup screen#1		
Main	Embedded database	(1)	Database storing destination	Specify a directory for specifying RD area and a directory for specifying system file. Specify any available directory having size of 660MB or more. If you specify a non-existing directory, specified directory is newly created. Specify only when you select [Model with DB/without RM] or [Model with DB/RM].	Installation directory of <Service platform>\CSC\DB\area
		(2)	DB connection port number	Specify a port number to be used for accessing an embedded database from Management Server or HCSC server. Specify with any integer in the range of 5001~65535. Specify when you select [Model with DB/without RM] or [Model with DB/RM].	22200
	Management Server	(3)	HCSC server operation port number	Specify a port number to be used for accessing Management Server from HCSC-Manager. Specify with any integer in the range of 1~65535.	28099
		(4)	Logical server operation port number	Specify a port number to be used for accessing Management Server from Smart Composer or management portal screen. Specify with any integer in the range of 1~65535.	28080
		(5)	End request receipt port number (for internal management)	Specify a port number to be used by Management Server for internal management. Specify with any integer in the range of 1~65535.	28005
		(6)	Internal communication port number (for internal management)	Specify a port number (webserver.connector.ajp13.port key of mserver.properties file) to be used by Management Server for internal management. Specify with any integer in the range of 1~65535.	28009
		(7)	In-process Naming Service port number (for internal management)	Specify a port number (ejbserver.naming.port key of mserver.properties file) to be used by Management Server for internal management. Specify with any integer in the range of 1~65535.	28900
	Administration Agent	(8)	Agent connection port number (for internal management)	Specify a port number for internal management to be used for accessing Administration Agent from Management Server. Specify with any integer in the range of 1~65535.	20295
	HCSC server	--	SOAP1.1 mode	Select when you want to use SOAP1.1 mode.	Selected
		--	SOAP1.1/1.2 combined mode	Select when you want to use SOAP1.1/1.2 combined mode.	--

Input items				Description	Initial value#2
Tab	Classification	Item No.	Item names in HCSC easy setup screen#1		
Main	HCSC server	(9)	Web service /MDB (WS-R) receipt port number	Specify a port number to be used for accessing standard receipt (Web service or MDB (WS-R)) or user-defined receipt from service requester. Specify with any integer in the range of 1~65535.	80
		(10)	SessionBean receipt port number	Specify a port number to be used for accessing HSCS server from Management Server or standard reception (SessionBean) from service requester. Specify with any integer in the range of 1~65535.	900
		(11)	MDB (DB queue) reception port number	Specify a port number to be used when accessing standard reception (MDB (DB queue)) from service requester. Specify with any integer in the range of 1024~65535. Specify when you select [Model with DB/RM].	20351
		(12)	Operation check port number	Specify a port number to be used by HCSC server for internal management. Specify with any integer in the range of 1~65535.	23152
		(13)	Simple Web server port number (for internal management)	Specify a port number to be used by HCSC server for internal management. Specify with any integer in the range of 1~65535.	8080
Server name	Server name selection	--	V7 compatible name	Select to use name same as version 7, for all types of server names of test environment to be configured by HCSC easy setup.	--
		--	HCSC main environment simple setup name	Select to use name same as name to be setup with HCSC actual environment simple setup, for all types of server names of test environment to be configured with HCSC easy setup. For HCSC actual environment simple setup, see "3.5 Easy Setup of production environment" in "Service Platform System Setup and Operation Guide".	Selected
		--	Custom name	Select to specify any name, for all types of server name of test environment to be configured with HCSC easy setup.	--
	Server name	(14)	Logical J2EE server name	Specify connection destination J2EE server name. Specify single byte alphanumeric within the range of 1~128 characters, underscore and hyphen.	J2EEServer
		(15)	Logical PRF name	Specify server name of PRF to be operated by linking with HCSC server. Specify single byte alphanumeric within the range of 1~128 characters, underscore and hyphen.	PRF

Input items				Description	Initial value#2
Tab	Classification	Item No.	Item names in HCSC easy setup screen#1		
Server name	Server name	--	Cluster name	Specify name of the cluster to which HCSC server to be set up belongs. Specify a cluster name that is unique within classes. Specify single byte alphanumeric within 1~8 characters and underscore.	Cluster
		(16)	HCSC server name	Specify name of the HCSC server to be set up. Specify single byte alphanumeric within 1~8 characters and underscore.	HCSC
		(17)	Manager name	Specify HSCS-Manager independent identification name for HCSC-Manager to identify Manager. When multiple HCSC servers are managed in a single repository, specify identification name that is unique in Manager (host) unit. Specify single byte alphanumeric within 1~16 characters and underscore.	Manager

(Legend)

--: Not applicable

Note#1

When you temporarily place the mouse cursor on an item corresponding to HCSC easy setup screen, description of that item is temporarily displayed.

Note#2

Initial value is a value that is initially displayed when HCSC easy setup screen is displayed for the first time. When you execute setup by changing an initial value, previous setting value is displayed in next HCSC easy setup screen, as an initial value.

(2) Unsetting up the test environment

This section describes pre-conditions and unsetting up method for unsetting up the test environment set up by using HCSC easy setup functionality.

! Important note

When you unset up by using HCSC easy setup functionality, test environment returns to the status before setup. As service adapter or business process that use the test environment are automatically deleted, take a backup of required data without fail, before executing unsetup.

(a) Pre-conditions

Pre-conditions for unsetting up the test environment are as follows:

- Unsetup target should be the test environment set up using HCSC easy setup functionality
- Management Server and Administration Agent should be in started status
- When you set up the test environment by selecting [Model with DB/without RM] or [Model with DB/RM], in HCSC easy setup screen, service components of embedded database should be started

(b) Method for unsetting up

Use HCSC easy setup screen, for unsetting up the test environment. Method for unsetting up the test environment is as follows:

1. From [Start] menu of Windows, select [Cosminexus]-[Environment configuration]-[Test environment setup].
   HCSC easy setup screen is displayed.
2. Click [Unset up] button on HCSC easy setup screen.

Unset up of the test environment is started. Unsetup state is displayed on the console of HCSC easy setup screen. Display of "`Unset up of the HCSC Easy Setup functionality is complete`" on the console implies that the test environment is successfully unset up.

2.4.3 Operating the test environment set up with HCSC easy setup functionality

This section describes method to operate the test environment set up by using HCSC easy setup functionality and information required for operating the test environment.

(1) Starting and stopping the test environment

Following table describes the method to start and stop the test environment.

Table 2–5: Method to start and stop the test environment

Start and stop target	Method to start and stop
Embedded database#	To start: From [Start] menu of Windows, select [Cosminexus]-[Start database]. To stop: From [Start] menu of Windows, select [Cosminexus]-[Stop database].
Services of embedded database#	When you start or stop the machine of test environment, the test environment is automatically started or stopped.
• Management Server • Administration Agent	When you start the machine of test environment, the test environment is automatically started and when you stop the machine of test environment, the test environment is automatically stopped.
• Performance Tracer • J2EE server • HCSC server • Standard reception	To start: From [Start] menu of Windows, select [Cosminexus]-[Start test server]. To stop: From [Start] menu of Windows, execute [Cosminexus]-[Stop test server].

Note#

Implement only when you have selected [Model with DB/without RM] or [Model with DB/RM] in HCSC easy setup screen.

(2) Information required for operating the test environment

When you set up the test environment by using HCSC easy setup functionality, the required information is automatically set up. To perform testing and debugging by using the test environment, you must know the information set when setting up the test environment with HCSC easy setup functionality.

Information set in test environment set up by using the HCSC easy setup functionality is described as follows.

Tip

You can customize the information set in the test environment set up by using the HCSC easy setup functionality, after setting up the test environment. For details on how to customize, see "*2.4.4 Customizing a Test Environment*".

(a) Information of user ID and password

Following table shows the information of user ID and password set in the test environment set up with HCSC easy setup functionality:

Table 2–6: User ID and password set in the test environment

Setting destination	User ID and password that is set	Initial value	Description
Embedded database[#1]	User ID	`admin`[#2]	Authentication identifier of table owner (`USRID` of environment variable)
	Password	`admin`[#2]	Password of table owner (`PSWD` of environment variable)
Management Server	Management user ID	`admin`	Management user ID for logging in to Management Server(use with `cmx_build_system -change` command[#3])
	Password	`admin`	Password or logging in to Management Server(use with `cmx_build_system -change` command[#3])
HCSC-Manager	HCSC-Manager login user ID	`admin`	User ID for logging in to HCSC-Manager(use with `cscsvstart` command [#4])
	HCSC-Manager login password	`admin`	Password for logging in to HCSC-Manager(use in `cscsvstart` command[#4])
HCSC server	User ID of the database used by HCSC server	`admin`	User ID of database used by HCSC server(use with `csccompodeploy` command[#4])
	Password of database used by HCSC server	`admin`	Password of database used by HCSC sever(use with `csccompodeploy` command[#4])

Note#1

Set when you select [Model with DB/without RM] or [Model with DB/RM] in HCSC easy setup screen.

Note#2

Handled with upper case characters "ADMIN" in the embedded database.

Note#3

For this command, see "cmx_build_system (building Web system)" in "Application Server Reference Guide".

Note#4

For cscsvstart command, see "cscsvstart(starting HCSC server)" in "Service Platform Reference Guide". For csccompodeploy command, see "csccompodeploy(deploying HCSC component)" in "Service Platform Reference Guide".

(b) Information of the port number

Following port number is set in the test environment set up with HCSC easy setup functionality. For port number, you can specify any value in HCSC easy setup screen. For details on value specified in HCSC easy setup screen, see "*Table2-4 Items to be input in HCSC easy setup screen*".

Embedded database(only when [Model with DB/without RM] or [Model with DB/RM] is selected in HCSC easy setup screen)

- DB connection port number

Management Server

- HCSC server operation port number
- Logical server operation port numberr
- End request receipt port number
- Internal communication port number
- In-process Naming Service port number

Administration Agent

- Agent connection port number

J2EE server

- Web service MDB(WS-R)receipt port number
- SessionBean receipt port number
- MDB(DB queue)receipt port number
- Operation check port number
- Simple Web server port number

(c) Information of name

Following table describes information such as host name or server name set in the test environment set up with HCSC easy setup functionality. When you change the server name with [Server name] tab on HCSC easy setup screen, name after change is set.

Table 2–7: Name set in the test environment

Setting destination	Name that is set	Initial value	Description
Embedded database#1	Host name	`Localhost`	Host name of embedded database(`DB_HOST` of environment variable)
Management Server	Host name	`Localhost`	Host name of Management Server(use with `cmx_build_system -change` command#2)
HCSC-Manager	Manager name	`Manager`	HCSC-Manager unique identification name for HCSC-Manager to identify Manager(use with `cscsvstart` command#3)
PRF	Server name	`PRF`	Server name of PRF
J2EE server	Server name	`J2EEServer`	Name of connection destination J2EE server(use with `cjstartsv` command#2)
	Host name	`Localhost`	Host name of connection destination J2EE server
	System name unique in entire system to which Reliable Messaging is linked	`RM`	System name unique in entire system to be linked with Reliable Messaging(`HRM_SYSTEM_NAME` of environment variable)
	Resource adapter name	`Reliable Messaging`	Display name of target RAR file(use with `cjdeployrar` command#2)
		`DB_Connector_for_HiRDB_Type4`	
		`DB_Connector_for_HiRDB_Type4_Cosminexus_RM`	
Cluster	Cluster name	`Cluster`	Cluster name
HCSC server	HCSC server	`HCSC`	HCSC server name(use with `cscsvstart` command#3)

Note#1

Set only when you select [Model with DB/without RM] or [Model with DB/RM] in HCSC easy setup screen.

Note#2

For cmx_build_system -change command, see "cmx_build_system (setting up Web system)" in "Application Server Command Reference Guide". For cjstartsv command, see "cjstartsv (starting J2EE server)" in "Application Server Command Reference Guide". For cjdeployrar command, see "cjdeployrar (deploying resource adapter)" in "Application Server Command Reference Guide".

Note#3

For this command, see "cscsvstart (starting HCSC server)" in "Service Platform Reference Guide".

(d) Other information

When you select [Model with DB/without RM] or [Model with DB/RM] in HCSC easy setup screen, following information is set in the embedded database of the test environment set up with HCSC easy setup functionality.

Data (RD area) storage destination

For information of RD area storing destination, you can specify any value in HCSC easy setup screen. For details on the value specified in HCSC easy setup screen, see "*Table2-4 Items to be input in HCSC easy setup screen*".

Area size of the database e to be created

Area size (`DB_SIZE` of environment variable) of the database to be created is set.

Initial value is "660MB".

(3) Operating the embedded database

For the embedded database set up by using HCSC easy setup functionality, you must regularly check the unused area of database. If unused area is less, release blank pages or blank segments, or delete the unnecessary execution history. For details, see " *2.5 Operating an embedded database set up with the HCSC Easy Setup functionality*".

(4) Troubleshooting

If you continue to set up or unset up HCSC server, deploying or deleting HCSC components and sending the requests by using HCSC easy setup functionality, there is a risk of insufficiency of unused segments in the embedded database.

If you face the insufficiency of the unused segments, implement the following countermeasures:

Method 1: Release the empty segments within embedded database

1. Search the table within the embedded database with SQL Executer.

   For details on how to search the table, see pdsql command in "HiRDB SQL Executer Online Help".

2. Issue pdreclaim command for all the tables starting with "CSC" and "RM" from the result of searching with step 1., and re-use blank pages by releasing.

   For details on pdreclaim command, see the "HiRDB Command Reference Guide".

Method 2: Re-edit tables within the embedded database

1. Search tables within the embedded database with SQL Executer.

   For details on how to search tables, see pdsql command in "HiRDB SQL Executer Online Help".

2. Issue pdrorg command for all the tables starting with "CSC" and "RM" from the search results of step 1, and reorganize tables.

   For the details on how to reorganize tables, see the contents related to reorganization of table in "HiRDB System Operation Guide".

   For details on pdrorg command, see the "HiRDB Command Reference Guide".

Method 3: Resetting up the test environment built with HCSC easy setup functionality

Unset up the test environment built with HCSC easy setup functionality once and then reset up. Embedded database is recreated with this work.

However, when you reset up the test environment, created service adapter or business processes are also deleted. Therefore, export repository in advance, then reset up the test environment and import the repository after reset up.

2.4.4 Customizing a Test Environment

The information described in "*2.4.3(2) Information required for operating the test environment* is specified in the test environment that is set up with the HCSC Easy Setup functionality. You can customize this information after setting up the test environment.

To customize a test environment, edit the values specified in definition files during the setup of the test environment with the HCSC Easy Setup functionality.

The table below describes the editable definition files. For details about each definition file, see the manuals specified in the Reference Destination column of the table.

Table 2–8: Editable definition files

File name	Explanation	Storag e destina tion	Refere nce destina tion
`adminagent.properties` (Administration Agent property file)	The port number used in the communication with the Administration Agent is set in this property file.	A	a
`cdsetupconfig.bat` (Embedded database setup batch file)	The environment variables of the embedded database are set in this batch file.	B	c
`cmdconf.bat` (HCSC-Messaging command definition file)	The class path of HiRDB type4 JDBC Driver for accessing the embedded database using the `csmXXX` command of the uCosminexus Service Platform is set in this batch file.	C	d
`Cosminexus_Reliable_Messaging.xml` (Connector attribute file)	The resource adapter attributes (configuration property value and property value) are specified in this XML file.	D	b
`csccmd.properties` (HCSC-Manager command definition file)	The omitted values (login user ID and login password) of the commands used in the operating environment are specified in this property file.	E	d
`cscmng.properties` (HCSC-Manager definition file)	The information required for HCSC-Manager operations is specified in this property file.	E	d
`cscsvconfig.properties` (HCSC server runtime definition file)	The runtime information (execution history management information and database information) required for starting the HCSC server is specified in this property file.	D	d
`cscsvsetup.properties` (HCSC server setup definition file(model with DB/without RM))#1	This is property file in which information (J2EE server, Reliable Messaging, database related information) required for HCSC server setup to be used in model without DB/RM is set.	D	d
`cscsvsetup.properties.esb` (HCSC server setup definition file(model with DB/RM))# 1	This is property file in which information (J2EE server, Reliable Messaging, database related information) required for HCSC server setup to be used in model with DB/RM is set.	D	d
`cscsvsetup.xml` (HCSC server configuration definition file(model with DB/without RM))	This is XML file in which HCSC server configuration information (definition of cosminexus-manager, jms-physical-reception, ejb-reception) to be used in model with DB/without RM is set.	D	d
`cscsvsetup.xml.esb` (HCSC server configuration definition file (model without DB/RM))	This is XML file in which HCSC server configuration information (definition of cosminexus-manager, jms-physical-reception, ejb-reception) to be used in model without DB/RM is set.	D	d
cscsvsetup.xml.rm (HCSC server configuration definition file(model with DB/RM))	This is XML file in which configuration information (definition of cosminexus-manager, jms-physical-reception, ejb-reception) of HCSC server to be used in the model with DB/RM is set	D	d
`DB_Connector_for_HiRDB_Type4.xml` (Connector attribute file)	The resource adapter attributes (configuration property value and property value) are specified in this XML file.	D	b

File name	Explanation	Storag e destina tion	Refere nce destina tion
DB_Connector_for_HiRDB_Type4_LT .xml (Connector attribute file)	This is XML file in which properties (value of configuration property, property value) of resource adapter is set. This is definition for local application to be used in environment without RM.	D	b
DB_Connector_for_HiRDB_Type4_Co sminexus_RM.xml (Connector attribute file)	The resource adapter attributes (configuration property value and property value) are specified in this XML file.	D	b
grantuser (User-defined file)	This file is used for creating a HiRDB user and schema.	B	c
hrmqueue-transmit.xml (Application attribute file)	The application attributes are specified in this XML file.	D	b
input.properties (Property file wherein system information is coded)	The information entered in the HCSC Easy Setup screen and system information is specified in this property file.	D	-
inserttableshirdb.sql (Insert table SQL file)	This SQL file is used to insert records in the system management information table.	D	-
mserver.cfg (Option definition file for Management Server)	The start option of JavaVM that runs the Management Server is specified in this file.	A	a
mserver.properties (Management Server environment setup file)	The port number used by the Management Server and command operations in the case of failure detection are specified in this property file.	A	a
qconf.txt (Queue definition file)	The queue information (DisplayName and QueueName) is specified in this text file.	E	e
setupscript (Setup script file(For model with DB/without RM and SOAP1.1 mode))	This is script file in which execution order of tasks in the setup process, to be used in model with DB/ without RM is set. Use this file to setup environment of SOAP1.1 mode.	D	-
setupscript.cjw (Setup script file(for model with DB/without RM and SOAP1.1/1.2 combined mode))	This is script file in which tasks of the setup process to be used in the model with DB/without RM are described in execution order. Use this file to setup the environment of SOAP1.1/1.2 combined mode.	D	--
setupscript.esb (Setup script file(for model without DB/RM and SOAP1.1 mode))	This is script file in which execution order of tasks of setup process to be used in model without DB/RM is set. Use this file to setup the environment of SOAP1.1 mode.	D	-
setupscript.esb.cjw (Setup script file(for model without DB/RM and SOAP1.1/1.2 combined mode))	This is script file in which tasks o setup process to be used in model without DB/RM is described in execution order. Use this file to setup the environment of SOAP1.1/1.2 combined mode.	D	--
setupscript.rm (setup script file(for model with DB/RM and SOAP1.1 mode))	This is script file in which tasks of setup process to be used in model with DB/RM are described in execution order. Use this file to setup the environment of SOAP1.1 mode.	D	--
setupscript.rm.cjw	This is script file in which tasks of the setup process to be used in model with DB/RM are described in execution order.	D	--

File name	Explanation	Storage destination	Reference destination
(setup script file(for model with DB/RM and SOAP1.1/1.2 combined mode))	Use this file to setup environment of SOAP1.1/1.2 combined mode.	D	--
`simple_model.xml` (HCSC Easy Setup definition file or SOAP1.1 mode)[#2]	The Web system built with the commands of Smart Composer functionality is defined in this XML file. This file is used to set up the environment for the SOAP1.1 mode.	D	--
`simple_model_cjl.xml` (HCSC Easy Setup definition file or SOAP1.1/1.2 combined mode)[#2]	This XML file defines the Web system set up using the commands of the Smart Composer functionality. This file is used to set up the environment for the SOAP1.1/1.2 combined mode.	D	--
`tablecreate` (Table definition file)	This file is used for creating the HiRDB schema and tables.	B	c
`unsetupscript` (unsetup script file (model with DB/without RM))	This is the script file in which execution order of tasks for unsetup process to be used in model with DB/without RM is set.	D	-
`unsetupscript.esb` (unsetup script file (model without DB/RM))	This is script file in which execution order of tasks for unsetup process to be used in model without DB/RM is set.	D	-
`unsetupscript.rm` (unsetup script file (model without DB/RM))	This is script file in which execution order of tasks for unsetup process to be used in model with DB/RM is set.	D	-

Legend:

A: *Cosminexus-installation-directory*`\manager\config`

B: *Cosminexus-installation-directory*`\CSC\DB\bats`

C: *Cosminexus-installation-directory*`\CSC\config\msg`

D: *Cosminexus-installation-directory*`\CSC\system\manager\setup`

E: *Cosminexus-installation-directory*`\CSC\config\manager`

--: There is no manual for reference.

a: *Cosminexus Application Server Server Definitions*

b: *Cosminexus Application Server Application and Resource Definitions*

c: *Cosminexus Application Server Application Development Guide*

d: *Cosminexus Application Server Cosminexus Service Platform Reference*

e: *Cosminexus Application Server Cosminexus Reliable Messaging*

#1

You cannot change properties (`db-use` property, `rm-use` property, and `hcscserver-data-filepath` property) related to the setup configuration.

#2

To change the number of concurrent executions of the Web containers in an environment set up using the HCSC Easy Setup functionality, change the `webserver.connector.inprocess_http.max_execute_threads` property.

For details about the `webserver.connector.inprocess_http.max_execute_threads` property, see the manual *Cosminexus Application Server Function Guide - Basic Development for Web Container*.

Reference note

To customize the test environment, you can use the files in the following directory as a reference for customization:

```
Cosminexus-installation-directory\CSC\log\manager\setup
```

If multiple testing environments (setup of the environment variable `CSCMNG_HOME`) are built in a single machine, you can use the files saved in the following directory for customization:

```
%CSCMNG_HOME%\log\manager\setup
```

The files saved in these directories are described below:

Files with the names same as the definition files described in Table 2-8

These files (log files of the HCSC Easy Setup functionality) have the contents same as the definition files set up during the execution of the HCSC Easy Setup functionality.

`adminagent.properties.bak` **(back up file of** `adminagent.properties`**)**

The settings of `adminagent.properties`, one version prior to the current `adminagent.properties` are saved.

`cmdconf.bat.bak` **(back up file of** `cmdconf.bat`**)**

The settings of `cmdconf.bat`, one version prior to the current `cmdconf.bat` are saved.

`csccmd.properties.bak` **(back up file of** `csccmd.properties`**)**

The settings of `csccmd.properties`, one version prior to the current `csccmd.properties` are saved.

`cscmng.properties.bak` **(back up file of** `cscmng.properties`**)**

The settings of `cscmng.properties`, one version prior to the current `cscmng.properties` are saved.

`mserver.cfg.bak` **(back up file of** `mserver.cfg`**)**

The settings of `mserver.cfg`, one version prior to the current `mserver.cfg` are saved.

`mserver.properties.bak` **(back up file of** `mserver.properties`**)**

The settings of `mserver.properties`, one version prior to the current `mserver.properties` are saved.

2.4.5 Checking the SOAP modes

This subsection describes how the SOAP modes are checked currently.

(1) Checking the SOAP mode of the development environment

You can check the SOAP modes of the development environment on Eclipse.

1. From the Eclipse menu, choose **Window** and then **Setup**.
   The **Setup** dialog box appears.
2. In the tree view on the left-hand side of the dialog box, choose **HCSC-Definer**.
   On the right-hand side of the dialog box, the SOAP mode being used appears in **Current SOAP mode**.

(2) Checking the SOAP mode of the execution environment

To check the SOAP mode that is currently being used in the execution environment, use the `cscrepls` command. When you execute the command, one of the following appears in the SOAP-mode display item:

- `1.1`: Indicates the SOAP1.1 mode.
- `1.1/1.2`: Indicates the SOAP1.1/1.2 combined mode.

Note that you can execute the `cscrepls` command when the setup of the HCSC server is complete.

For details about the `cscrepls` command, see the manual *Cosminexus Service Platform Reference*.

2.5 Operating an embedded database set up with the HCSC Easy Setup functionality

This section describes operations of the embedded database set up with the HCSC Easy Setup functionality.

2.5.1 Checking unused RD area

Review the database usage status by checking the unused RD area. You must periodically check the unused RD area. This section describes the pre-conditions and methods for checking the unused RD area.

(1) Pre-conditions

To check unused RD area, database must be in active status.

Also, connection information of HiRDB server other than the embedded database must not be set in the system environment variables (PDHOST, PDNAMEPORT, PDUSER) of OS. If those are set, delete the system environment variable with set command, after starting the database console.

(2) Method for checking

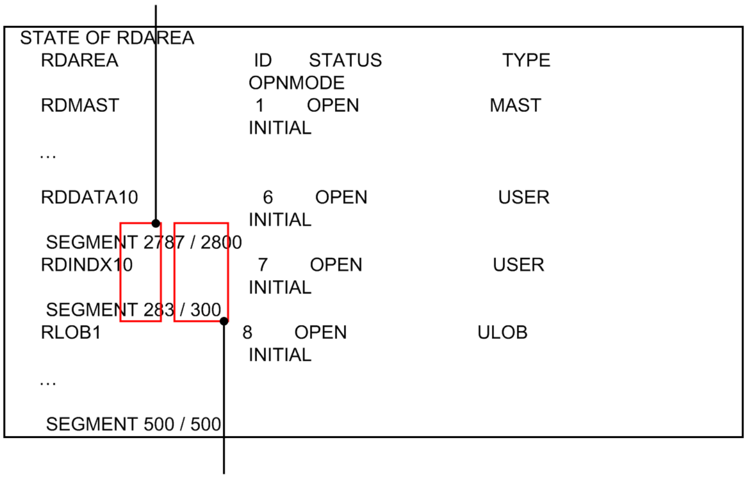
Method for checking the unused RD area is as follows:

1. From [Start] menu of Windows, select [Cosminexus]-[Database console].
   Environment variables for operating the embedded database are set.
2. Execute pddbls command to display the status of RD area.
   For details on pddbls command, see the "HiRDB Command Reference Guide".
   Execution format o `pddbls` command is as follows:

```
pddbls -r ALL -a
```

3. Check the unused RD area from the command execution result.
   Following figure shows the example of execution result.

The number of unused segments in RD area (eleven digit decimal number)



```
STATE OF RDAREA
  RDAREA                ID    STATUS              TYPE
                        OPNMODE
  RDMAST                 1    OPEN               MAST
                        INITIAL
  ...

  RDDATA10               6    OPEN                USER
                        INITIAL
   SEGMENT 2787 / 2800
  RDINDX10               7    OPEN               USER
                        INITIAL
   SEGMENT 283 / 300
  RLOB1                 8    OPEN              ULOB
                        INITIAL
  ...

   SEGMENT 500 / 500
```

The number of all segments in RD area (eleven digit decimal number)

4. When unused RD area is less, eliminate the shortage of unused area using the following methods:
   - Delete the execution history of process instances.
     See "*2.5.2 Deleting the execution history of process instances*".
   - Releasing empty pages and empty segments
     See "*2.5.3 Releasing empty pages and empty segments*".

2.5.2 Deleting the execution history of process instances

When the execution status of process instances of a business process is recorded in a database as history, the execution history will be added day after day, which reduces the capacity of the database. Therefore, you must delete the execution history of process instances periodically. The following are the prerequisites for deleting the execution history of process instances and the confirmation method:

(1) Prerequisites

To delete the execution history of process instances, the database must be in the running status.

(2) Deletion methods

1. Determine the HCSC server to be accessed.
   If the information set up with the HCSC Easy Setup functionality is not customized, `HCSC` will be set up in the HCSC server name.
2. Execute the `cscpidelete` command in the test environment (Operating environment).
   For details about how to delete the execution history of process instances, see the contents about deleting the execution history of process instances (deletion of the execution history by commands), in *Cosminexus Service Platform System Setup and Operation Guide*.
3. Release empty pages and empty segments.
   For details about how to release empty pages and empty segments, see "*2.5.3 Releasing empty pages and empty segments*".

2.5.3 Releasing empty pages and empty segments

Release the empty pages and empty segments generated due to data deletion and restore them to unused status. Pre-conditions and method for releasing empty pages and empty segments is as follows:

! Important note

Release not only empty pages but empty segments as well without fail.

(1) Pre-conditions

Following conditions must be fulfilled for releasing empty pages and empty segments.

- HCSC server must be stopped
- Database must be started

(2) Method for releasing

Method for releasing empty pages and empty segments is as follows: If password input is requested, enter " `admin` " as password.

1. From [Start] menu of Windows, select [Cosminexus]-[Database console].
   Environment variables for operating the embedded database are set.
2. Execute pdreclaim command to release the empty pages in use.
   Release the empty pages in use (pages in which data is not stored due to data deletion) and set those as unused pages.
   For details on pdreclaim command, see the "HiRDB Command Reference Guide".
   Execution format of pdreclaim command is as follows:
   For releasing empty pages, which are in use in table

```
pdreclaim -u admin -k table -t all -o
```

For releasing empty pages, which are in use in index

```
pdreclaim -u admin -k index -t all
```

3. Execute pdreclaim command to release empty segments in use.

   Release the empty segments (segments where in all pages are empty) that are in use and set those as unused segments.

   For details on pdreclaim command, see the "HiRDB Command Reference Guide".

   Execution format of pdreclaim command is as follows:

   For releasing empty segments, which are in use in table

   ```
   pdreclaim -u admin -k table -t all -o -j
   ```

   For releasing empty segments, which are in use in index

   ```
   pdreclaim -u admin -k table -t all -j
   ```

4. Check whether the insufficiency of unused area has been removed.

   For checking method, see "*2.5.1 Checking unused RD area*".

2.6 Types of Available Service Components and Their Application Scopes

This section explains the types of service components that can be used in the SOA provided by Service Platform, as well as their application scopes.

Service components that can be used in the SOA must have interface information defined.

The following table lists the types of service components for SOA that can be used in Service Platform.

Table 2–9: Available service component types

Service component type	Communication mode	Protocol
Web Services	Synchronous	SOAP (HTTP)
SessionBean		RMI-IIOP
MDB	Asynchronous	WS-R
Database queue		JMS

Note

In addition to the service components listed in this table, other service components can also be used by connecting to a system based on non-SOA architecture. For details about calling service components when connecting to a system based on non-SOA architecture, see *Chapter 2. Functionality for Connecting to Various Types of Systems* in the manual *Service Platform Overview*.

The following subsections explain the application scopes for each type of service components that can be used in SOA.

2.6.1 Applicability of the service components that use Web service

The applicability of service components that use Web services is as follows:

Tip

WSDL and XML schema in this manual are described by mapping specific prefix and namespace URI. The following table describes the mapping of prefix and namespace URI:

Table 2–10: Mapping between the prefix and name space URI

Prefix	Name space URI
`wsdl`	`http://schemas.xmlsoap.org/wsdl/`
`xsd`	`http://www.w3.org/2001/XMLSchema`
`soap`	`http://schemas.xmlsoap.org/wsdl/soap/`
`soap12`	`http://schemas.xmlsoap.org/wsdl/soap12/`

(1) Pre-requisite specifications

This section separately describes pre-requisite specifications for service components that use Web services, for SOAP1.1 mode and for SOAP1.1/1.2 combined mode.

- **When using SOAP1.1 mode**

  When you use SOAP1.1 mode, service components that use Web services must be created by conforming to the specifications of the following versions.

  - SOAP1.1
  - WSDL1.1
  - SAAJ1.2

Also, in service platform, it is recommended to create service components by conforming to WS-I Basic Profile 1.0a.

- **When using SOAP1.1/1.2 combined mode**

  When using SOAP1.1/1.2 combined mode, service components that use Web services must be created by conforming to specifications of following versions:

  - SOAP1.1 or SOAP1.2
  - WSDL1.1
  - SAAJ1.3

  Also, in service platform, it is recommended to create service components by conforming to WS-I Basic Profile 1.1.

(2) Format of SOAP message

SOAP message is configured with SOAP header and SOAP body. SOAP header includes additional information (such as identification information) of message and SOAP body includes the actual message.

In service components that use Web services, following conditions must be fulfilled as the format of SOAP message.

- Request message and reply message must be stored in the actual text of SOAP message.

Also, in service platform having version prior to 08-53, you can create SOAP adapter based on WSDL that defines `soap:header` element, but `soap:header` element is ignored.

(3) Notes when defining WSDL (SOAP1.1 mode)

This section describes WSDL description format and notes when defining WSDL, when using SOAP1.1 mode. For support range of WSDL1.1 specifications, see "*2.6.1(5) Support range of WSDL1.1 specifications*".

(a) wsdl:types elements

`wsdl:types` element defines the information related to types used in SOAP message. Describe `wsdl:types` element according to the following rules.

- Describe `wsdl:types` element as the child element of `wsdl:definitions` element.
- You can describe 0 or 1 element. You cannot describe 2 or more elements.
- Describe this element after `wsdl:documentation` element and `wsdl:import` element.
- Describe it before all other elements except `wsdl:documentation` element and `wsdl:import` element.

(b) xsd:schema element

`xsd:schema` element describes XML schema. Describe `xsd:schema` element as the child element of `wsdl:types` element. For the rules applicable when describing XML schema, see "*2.6.5 Scoping of XML schema*".

(c) wsdl:import element

`wsdl:import` element is to be defined when importing WSDL. Describe `wsdl:import` element as a child element of `wsdl:definitions` element. Describe `wsdl:import` element according to the following rules:

- Describe `wsdl:import` element after `wsdl:documentation` element.
- Describe before all the elements, excluding `wsdl:documentation` element.
- Describe `namespace` attribute without fail.
- When you want to set percent encoded value in the value, set hexadecimal value in upper case characters.
  Example is as follows:
  Correct: location="%E3%81%82.wsdl"
  Incorrect: location="%e3%81%82.wsdl"
- Notes when specifying location attribute of `wsdl:import` element are as follows:

- Use characters prescribed with RFC2396 and character that fulfills `xsd:anyURI`. However, you cannot use RFC2732 (IPv6).
- When specifying `location` attribute of `wsdl:import` element with absolute URI, use either of http, https and file protocol.

Example of specifying `location` attribute is as follows:

(Example1)Specify WSDL in local, with relative path

```
./example/sample.wsdl
```

(Example2) Specify WSDL in local with absolute path of URL format

```
file:///C:/example/sample.wsdl
```

(Example3)Specify WSDL in remote with URL

```
http://example.com/sample.wsdl
```

(d) soap:binding element

`soap:binding` element defines SOAP binding. Describe `soap:binding` element according to the following rules.

- Describe `soap:binding` element as child element of `wsdl:binding` element.
- Describe only 1 element. You cannot describe 2 or more elements.
- In `transport` attribute, specify " `http://schemas.xmlsoap.org/soap/http`" that shows HTTP binding.
- Declare SOAP binding, as the binding declaration. You cannot declare binding other than SOAP binding.

(e) wsdl:operation element

Describe `wsdl:operation` element according to the following rules:

- Specify `wsdl:operation` element, which is child element of `wsdl:portType` element and `wsdl:operation` element that is child element of `wsdl:binding` element, in 1 as to 1 relation.
- Describe `name` attribute within 255 bytes.

(f) soap:operation element

`soap:operation` element defines the operation information in SOAP binding. Describe `soap:operation` element according to the following rules:

- Describe `soap:operation` element as child element of `wsdl:operation` element, which is grandchild element of `wsdl:binding` element.
- Describe only 1 rlrmrny. You cannot describe 2 or more elements.
- When `style` attribute is "`rpc`", do not specify `element` attribute having "operation name" or "operation name" + "Response" as `name` element, in `namespace` attribute of `soap:body` element.

(g) soap:body element

`soap:body` element defines messages under `soap:body` element of SOAP message. Describe `soap:body` element according to the following rules.

- Describe `soap:body` element as child element of `wsdl:input` element or `wsdl:output` element, which is child grandchild element of `wsdl:binding` element.
- Describe only 1 element. You cannot describe 2 or more elements.
- You cannot describe child element.

When `style` attribute is "document", you can specify `parts` attribute but it is ignored.

(h) wsdl:fault element

`wsdl:fault` element defines fault. Describe `wsdl:fault` element according to the following rules:

- Describe wsdl:fault element as child element of wsdl:portType element and wsdl:binding element.
- Do not describe multiple wsdl:fault elements having value of same name attribute.
- You must define wsdl:fault element in wsdl:operation element, which is child element of wsdl:binding element, so that correspondence with wsdl:fault element defined in wsdl:operation element, which is child element of wsdl:portType element.

(i) soap:fault element

soap:fault element defines messages under detail child element included in soap:fault element of SOAP message. Describe soap:fault element according to the following rules:

- Describe soap:fault element as child element of wsdl:fault element, which is child element of wsdl:binding element.
- Specify only 1 element. You cannot describe 2 or more elements.

Conditions when using fault

In SOAP Fault returned from service components, fault name is set as faultCode and handled as fault, when following conditions are fulfilled:

- SOAP communication base(Web Services) must be used in service components
- Element name# that shows exception type complexType defined in WSDL fault name(name attribute value of wsdl:fault element)and schema(in wsdl:types), must match including namespace.

When you use the WSDL that do not fulfill these conditions and you want to handle SOAP Fault returned from service component, define SOAP Fault operation definition file and set it as user-defined exception.

Note#

Element nameimplies "element that wsdl:part element referred by wsdl:fault specifies by using element attribute". Element name indicates element that WSDL can specify by striding over the following order.

1. wsdl:fault
2. wsdl:message (striding over of this element does not happen in some cases)
3. wsdl:part
4. xsd:element

When you generate WSDL file having style attribute of operation as "rpc", by using Java2WSDL command, which is development support command of service platform, WSDL file referred by type element and not by element is generated.

Therefore, use WSDL file by revising or re-generating, rather than using the generated WSDL file as it is. For details on revising or re-generating WSDL file, see Notes in "*4.3.2 Creating a Service Component Message (for Web Services)*".

When you use fault in service platform, it is recommended to use WSDL of document style.

(j) soap:header element

soap:header element defines messages under soap:header element of SOAP message. Describe soap:header element according to the following rules:

- Describe soap:header element as child element of wsdl:input element and wsdl:output element, which is grandchild element of wsdl:binding element.
- You cannot describe child element.
- In message attribute, specify wsdl:message element different than wsdl:message element referred from wsdl:input element or wsdl:output element.
- In part element, specify wsdl:part element, which is child element of wsdl:message element specified in message attribute.

(k) wsdl:service element

`wsdl:service` element defines SOAP services. Specify only 1 `wsdl:port` element in 1 `wsdl:service` element. You cannot describe 2 or more elements.

(4) Notes when defining and using WSDL(SOAP1.1/1.2 combined mode)

When you use SOAP1.1/1.2 combined mode, describe WSDL with reference of "4.3.1 Creating WSDL file" in "Application Server Web Service Development Guide".

However, contents described in "4.3.1 Creating WSDL file" in "Application Server Web Services Development Guide" differ with operations of service platform in some aspects. This section describes notes at the time of defining WSDL, when using SOAP1.1/1.2 combined mode.

(a) Available WSDL definition styles

WSDL definition style is `document/literal`.

(b) wsdl:port element

If `name` attribute of `wsdl:port` element gets duplicated, error occurs. Specify unique `name` attribute in entire WSDL.

(c) wsdl:types element

When you define schema as child element of `wsdl:types` element, describe comments by using `xsd:annotation`.

Also, JAXB namespace "`http://java.sun.com/xml/ns/jaxb`" is added to the `namespace` declaration of message format created from WSDL in the service adapter.

(d) wsdl:operation element

Describe `name` attribute of `wsdl:operation` element with less than 255 bytes.

(e) Definition when using SOAP communication base in service components

When you use SOAP communication base in service components, match `name` attribute of `wsdl:fault` element with name element of `xsd:element` referred by `wsdl:part` element.

(f) Validating WSDL

- Errors occurring in Basic Profile based verification are considered as warning in service platform. Following table describes warning message ID, corresponding error ID and occurrence conditions due to the Basic Profile based verification.

Table 2–11: Warnings (errors) occurring due to Basic Profile-based validation

No.	Warning message ID	Corresponding error ID	Occurrence condition
1	KDJW51209-W	KDJW51208-E	This warning occurs when you describe `namespace` attribute in `soap:body` element.
2	KDJW51211-W	KDJW51210-E	This warning occurs when you describe `namespace` attribute in `soap:fault` element.
3	KDJW51213-W	KDJW51212-E	This warning occurs when you describe `wsdl:required` attribute in extended element.

- Verification based on the following conditions is not executed in service platform:
  - Verification related to number of elements defined in WSDL
  - Verification when reserved expressions of Java are described

(5) Support range of WSDL1.1 specifications

Following table shows the support range of WSDL1.1 specifications when using SOAP1.1 mode.

For support range of WSDL 1.1 specifications when using SOAP1.1/1.2 combined mode, see "20.1 Support range of WSDL 1.1 specifications" in "Application Server Web Service Development Guide".

Table 2–12: Support range of the WSDL1.1 specifications

Classification		Support	Remarks
Major classification	Minor classification		
Service definition: WSDL document structure	Document Naming and Linking	Y	Has description about scoping of namespace
	Authoring Style(acquisition of components by import element)	Y	Has description about acquisition by import element of other file
	Language Extensibility and Binding	N	Has description about deploying `wsdl:required` attribute
	Documentation	Y	Has description about comments in the element.
Service definition: `wsdl:types`		Y	Has description about data types that are handled
Service definition: `wsdl:message`		Y	Has description about logical definition of message
Service definition: Port type	One-way Operation	N	Has description about one way operations of the message.
	Request-response Operation	Y	Has description about request/response operations of the message.
	Solicit-response Operation	N	Has description about send request/response operations of the message.
	Notification Operation	N	Has description about notification operation of the message.
	Names of Elements within an Operation	Y	Shows name attribute of input and output element. You cannot perform overload. Name must be unique in WSDL.
	Parameter Order within an Operation	Y	Has description about order of parameter within operation. You can specify list of parameters according to parameterOrder attribute.
Service definition: `wsdl:binding`		Y	Has description about the definition of details on message format and protocol.
Service definition: `wsdl:port`		Y	Has description about physical definition of services.
Service definition: `wsdl:service`		Y	Has description about position of services. You cannot have correspondence of multiple SOAP services to a single WSDL.
SOAP binding	`soap:binding`	Y	Has description about binding of SOAP format
	`soap:operation`	Y	Has description about information of SOAP operation in SOAP message
	`soap:body`	Y	Has description about display method of message part of SOAP body within SOAP message. You can describe parts attributes but it is ignored.
	`soap:fault`	Y	Has description about contents of SOAP fault within SOAP message.
	`soap:header`	Y	Has description about contents in SOAP header element within SOAP message.
	`soap:headerfault`	N	
	`soap:address`	Y	Has description about address of port element

Classification		Support	Remarks
Major classification	Minor classification		
MIME binding	`mime:content`	N	Has description about MIME type
	`mime:multipartRelated`	N	Optional set of MIME parts has been consolidated.
	`mime:part`	N	Has description about each MIME part.
	`mime:mimeXml`	N	Has description about XML payload having specific schema. It does not conform to SOAP.

(Legend)

Y: Supports

N: Does not support

(6) Life cycle

Even if it is set (setting up "`Session`", "`Application`" in `DeployScope`) to maintain session between execution environment and service components of service platform, you cannot perform stateful calling. It is always calling in "`Request`" in service platform.

2.6.2 Application Scopes of Service Components That Use SessionBean

This subsection describes the application scope of service components that use SessionBean.

(1) Prerequisite specifications

Service components that use SessionBean must be created according to the EJB 2.0 specifications.

(2) Deployment destination of the available service component

Only service components deployed on Cosminexus can be used as service components that use SessionBean.

(3) SessionBean creation conditions

Service components that use SessionBean must be created according to the following conditions:

Classes that can be specified for the arguments and return values

The classes to be specified for the arguments and return values of service components that use SessionBean must satisfy the following conditions:

- A user-defined class must consist of the basic class.
- The user-defined class specified for an argument or return value must have a `private` field and a `public` access method (`set`*XX* and `get`*XX*) for allowing access to that field from outside (JavaBeans format).
- A user-defined `interface` class must not be specified for an argument or return value of a method that is public.
- A user-defined `abstract` class must not be specified for an argument or return value of a method that is public.
- A user-defined class that inherits a class other than `java.lang.Object` must not be specified for an argument or return value of a method that is public.
- A user-defined class of a different package must not be specified for an argument or return value of a method that is public.
- The following table shows the Java data types that can be used as an argument or return value of a method. Only those data types that are shown as usable in a given mode can be used.

Table 2–13: Java data type usability

Data type in Java	Usability as an argument or return value of a method		
	When used directly	When used as an array	When used as a member variable of a user-defined data class
`boolean`	Y	Y	Y
`javax.xml.rpc.holders.BooleanHolder`	Y	--	--
`byte`	Y	Y	Y
`javax.xml.rpc.holders.ByteHolder`	Y	--	--
`byte[]`	Y	Y	Y
`javax.xml.rpc.holders.ByteArrayHolder`	Y	--	--
`double`	Y	Y	Y
`javax.xml.rpc.holders.DoubleHolder`	Y	--	--
`float`	Y	Y	Y
`javax.xml.rpc.holders.FloatHolder`	Y	--	--
`int`	Y	Y	Y
`javax.xml.rpc.holders.IntHolder`	Y	--	--
`long`	Y	Y	Y
`javax.xml.rpc.holders.LongHolder`	Y	--	--
`short`	Y	Y	Y
`javax.xml.rpc.holders.ShortHolder`	Y	--	--
`java.lang.Byte`	Y	Y	Y
`javax.xml.rpc.holders.ByteWrapperHolder`	Y	--	--
`java.lang.Byte[]`	Y	Y	Y
`java.lang.Double`	Y	Y	Y
`javax.xml.rpc.holders.DoubleWrapperHolder`	Y	--	--
`java.lang.Float`	Y	Y	Y
`javax.xml.rpc.holders.FloatWrapperHolder`	Y	--	--
`java.lang.Integer`	Y	Y	Y
`javax.xml.rpc.holders.IntegerWrapperHolder`	Y	--	--
`java.lang.Long`	Y	Y	Y
`javax.xml.rpc.holders.LongWrapperHolder`	Y	--	--
`java.lang.Object`	--	--	--
`javax.xml.rpc.holders.ObjectHolder`	--	--	--
`java.lang.Object[]`	--	--	--

Data type in Java	Usability as an argument or return value of a method		
	When used directly	When used as an array	When used as a member variable of a user-defined data class
`java.lang.Short`	Y	Y	Y
`javax.xml.rpc.holders.ShortWrapperHolder`	Y	--	--
`java.lang.String`	Y	Y	Y
`javax.xml.rpc.holders.StringHolder`	Y	--	--
`java.math.BigDecimal`	Y	Y	Y
`javax.xml.rpc.holders.BigDecimalHolder`	Y	--	--
`java.math.BigInteger`	Y	Y	Y
`javax.xml.rpc.holders.BigIntegerHolder`	Y	--	--
`java.util.Date`	Y	Y	Y
`javax.xml.namespace.QName`	Y	Y	Y
`javax.xml.rpc.holders.QNameHolder`	Y	--	--
`java.lang.Boolean`	Y	Y	Y
`javax.xml.rpc.holders.BooleanWrapperHolder`	Y	--	--
`java.util.Calendar`	Y	Y	Y
`javax.xml.rpc.holders.CalendarHolder`	Y	--	--

Legend:

Y: Can be used.

--: Cannot be used.

Method overload

Methods having the same name cannot be specified for service components that use SessionBean.

Transactions

Set the transaction attribute (`trans-attribute`) to be assigned to the transaction control type (`transaction-type`) and method of SessionBean so that it does not inherit transactions from the HCSC server. Transactions cannot be inherited by service components that use SessionBean.

The following table shows whether the transaction attribute can be specified for the service component transaction modes.

Table 2–14: Specification of the transaction attribute in service components

Transaction mode of the service component	Tx attribute	Usability
BMT	N/A	Y
CMT	`Required`	--
	`RequiresNew`	Y
	`Supports`	--
	`NotSupported`	Y
	`Mandatory`	--

Transaction mode of the service component	Tx attribute	Usability
CMT	`Never`	--

Legend:

Y: Can be used (works according to the EJB specifications).

--: Cannot be used.

N/A: Not applicable

Method exceptions

Classes inherited from `java.rmi.RemoteException` and `java.lang.RuntimeException` are the only exceptions that can be described in the `throws` clause of a method.

SessionBean type

The SessionBean service component must be created as Stateless Session Bean.

In the execution environment of a Cosminexus Service Platform, connection (`create`) and disconnection (`remove`) are issued for each execution request to the EJB service. Therefore, service components that maintain an internal state by using Stateful Session Bean cannot be used.

(4) Schema format

The schema used in a service component in which SessionBean is used must satisfy the conditions explained in *2.6.5 Scoping of XML schema*. For details on schema conditions, see "*2.6.5 Scoping of XML schema*".

2.6.3 Application Scopes of Service Components That Use the Local Queue of Cosminexus RM

This subsection describes the application scopes of service components that use the local queue of Cosminexus RM.

(1) Prerequisite specifications

Service components that use the local queue of Cosminexus RM must be created according to the EJB 2.0 specifications.

(2) Message type

When a service component uses the local queue of Cosminexus RM, you can use one of the following message types for sending a message from a service requester to a service component:

- BytesMessage
- ObjectMessage
- TextMessage

Only these message types can be used.

(3) Schema format

The schema used in a service component in which Cosminexus RM local queue is used must satisfy the conditions explained in "*2.6.5 Scoping of XML schema*". For details on schema conditions, see "*2.6.5 Scoping of XML schema*".

2.6.4 Application Scopes of Service Components That Use a Database Queue

This subsection describes the application scopes of service components that use a database queue.

(1) Prerequisite specifications

Service components that use a database queue must be created according to the following version specifications:

- TP1/Server Base Enterprise Option 02-00 or later
- EJB 2.0 (when the service component is installed as MDB)

(2) Message type

When a service component uses a database queue, you can use BytesMessage as the message type for sending a message from a service requester to a service component.

BytesMessage is the only message type that can be used.

(3) Database that can be used

HiRDB is the only database that can create a database queue.

(4) Schema format

The schema used in the service component in which a database queue is used must satisfy the conditions explained in "*2.6.5 Scoping of XML schema*". For details on schema conditions, see "*2.6.5 Scoping of XML schema*".

2.6.5 Scoping of XML schema

This section describes the scoping of XML schema to be used in each service component.

(1) Format of XML schema to be used

- XML schema must fulfill the following conditions:
  - XSD namespace of schema must be " `http://www.w3.org/2001/XMLSchema`".
  - Root element must be described in the selected XML schema.
  - When you specify a file for XML schema, length of the file name must be 128 bytes or less.
  - White space character (single byte space (#`x20`), tab (#`x9`) and linefeed code (#`xA` or #`xD`) must not be used in start or end of XML schema attributes.
- If you specify a large value in `maxOccurs` attribute, large size memory is consumed. It is recommended to use `unbounded` as much as possible, in `maxOccurs` attribute.
- You can use `type` attribute, `ref` attribute or internal definition (`complexType` or `simpleType`) for defining types of `xsd:element` element.
- If you use standard reception as the reception of business process, it is recommended to specify `qualified` in `elementFormDefault` attribute of `xsd:schema` element.

! Important note

- If a SOAP specific attribute (mustUnderstand attribute, role attribute, encodingStyle attribute, actor attribute and relay attribute) appears in the child element of SOAP header or SOAP body and you convert with data transformation activity without defining this attribute in XML schema, SOAP specific attribute is not reflected in variable after conversion. Also, verification error occurs when you execute the verification process. Therefore, when SOAP specific attribute appear, mention xsd:anyAttribute in XML schema.
- When you want to specify a percent encoded value in XMLschema file, specify a hexadecimal value with upper case characters. Example is as follows:

  Correct: schemaLocation="%E3%81%82.xsd"

  Incorrect: schemaLocation="%e3%81%82.xsd"

(2) Notes when referring to external XML schema

Notes when referring and using external schema from XML schema, are as follows:

- When `schemaLocation` attribute of `xsd:import` element of XML schema defined under `wsdl:types` element of WSDL is not specified, namespace must be resolved by using the XML schema under `wsdl:types` element.
- Specify `schemaLocation` attribute of `xsd:import` element of XML schema without fail.
- Specify namespace attribute in `xsd:import` element without fail.
- In `xsd:import` element and `xsd:includ` element, call reference and set hierarchy to less than 20 hierarchy.
- In `schemaLocation` attribute, do not specify schema (`redefine` having 2 or more hierarchies) in schema having `redefine` element.
- Do not include schema for which `targetNamespace` attribute is not specified, rom the schema for which `targetNamespace` attribute of `xsd:schema` element is specified.
- When using SOAP1.1 mode, you can set value percent encoded with UTF-8 as well as value not percent encoded, in `schemaLocation` attribute of `xsd:import` element and `schemaLocation` attribute of `xsd:include` element. However, when percent encoding is done and directory is included in file path, do not encode the directory delimiter (/).
- When using SOAP1.1/1.2 combined mode, specify value percent encoded with UTF-8, in `schemaLocation` attribute of `xsd:import` element and `schemaLocation` attribute of `xsd:include` element. However, when value is percent encoded and directory is included in file path, do not encode the directory delimiter (/).
- Use a forward slash (`/`) as a directory delimiter in the `schemaLocation` attribute. (You cannot use `\`).
- Use "/" in directory delimiter of `schemaLocation` attribute (you cannot use "\").
- Do not use following characters in the file path of XML schema.
  ";", "?", ":", "@", "&", "=", "+", "$", ", ", "<", ">", "#", "%", """, "{", "}", "|", "^", "[", "]", "`".
- Notes when specifying `schemaLocation` attribute are as follows:
  - User character prescribed in RFC2396 and character that fulfills `xsd:anyURI`. However, you cannot use RFC2732 (IPv6).
  - When you want to specify `schemaLocation` attribute with absolute URI, use enter of the http, https, and file protocol.

  Example of specifying `schemaLocation` attribute is as follows:

  (Example1) Specify XML schema in local with relative path

  ```
  ./example/sample.xsd
  ```

  (Example 2) Specify XML schema in local with absolute path of URL format

  ```
  file:///C:/example/sample.xsd
  ```

  (Example 3) Specify XML schema in remote, with URL

  ```
  http://example.com/sample.xsd
  ```

2.7 Expanding code conversion

You can expand the code conversion functionality provided by service platform. For expanding, you must separately purchase the following products.

- Code conversion - Development Kit
- Code conversion - Server Runtime (for Windows)
- Code conversion - Runtime (for UNIX)

The following sub-section describes the tasks which you must perform for expanding the code conversion.

2.7.1 Creating character code conversion UOC

By creating character code conversion UOC, you can handle the character code that is not supported by service platform, as standard process. When you want to use API of code conversion in character code conversion UOC, you must install following products on the execution environment on which Service Platform is installed.

- Code conversion - Server Runtime (for Windows)
- Code conversion - Runtime (for UNIX)

The settings to be performed after installing required products in the execution environment are as follows:

(1) Embedding JAR

Procedure for embedding JAR is as follows:

1. Take backup of <Service platform installation directory>\CSC\lib\external\codeconv.jar.
2. Copy codeconv.jar of the installed unit version code conversion.
   Copy destination is as follows:
   ```
   <Service platform installation directory>\CSC\lib\external\codeconv.jar
   ```

(2) Setting up the path for code conversion table

To set the path of code conversion table, add following path in <Service platform installation directory>\CC\server \usrconf\ejb\<J2EE server name>\usrconf.properties.

```
csc.dt.codetablepath=<Code conversion installation directory>/table
```

It is recommended to execute these settings with Smart Composer functionality or management portal. For Smart Composer functionality and management portal, see "1.1 Tools of management functionality" in "Application Server System Setup and Operation Guide".

2.7.2 Embedding the user mapping table

You can create a user mapping table of user-specific external character mapping by using Code conversion - Development Kit and embed it in Service platform.

You require the following products for creating the user mapping table.

- Code conversion - Development Kit

For order for creating the user mapping, see the "Code Conversion User Guide (Java version) ".

This kit describes settings for embedding the user mapping table required for execution environment.

1. Take backup of <Service platform installation directory>#\CSC\lib\external\table directory.
2. Overwrite the created user mapping table on the <Service platform installation directory>#\CSC\lib\external\table directory.

Note#

When Code conversion - Server Runtime has been installed, replace <Service platform installation directory> with <Code conversion installation directory>.

3 Managing Project and Managing Repository

This chapter explains project management, including creating and deleting HCSCTE projects in Eclipse with HCSCTE embedded. It also explains how to manage the repository specified when the project properties are set up.

3.1 Managing a Project

Before developing a HCSC component, create an HCSCTE project and set up its properties. Additionally, export or import the created project as needed.

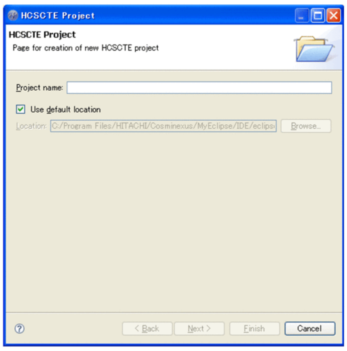
3.1.1 Creating a Project

Before creating an HCSCTE project, select **Window** - **Preferences** - **Java** - **Build Path** from the Eclipse menu, and then click **Project** in **Source and Output Folder**.

When you create an HCSCTE project, you can specify a desired prefix for the ID (service ID and reception ID) used in the service and user-defined reception. Note that you can also specify a prefix after creating the HCSCTE project.

To create an HCSCTE project:

1. Start Eclipse.
2. On the File menu, choose New, and then Project.
   The New Project dialog box opens.
3. Select HCSCTE Project and click Next.
   The HCSCTE Project dialog box (the page for creating a new HCSCTE project) opens.



HCSCTE Project

HCSCTE Project

Page for creation of new HCSCTE project

Project name:

☑ Use default location

Location: C:/Program Files/HITACHI/Cosminexus/MyEclipse/IDE/eclips

4. Specify the following items and click Next.

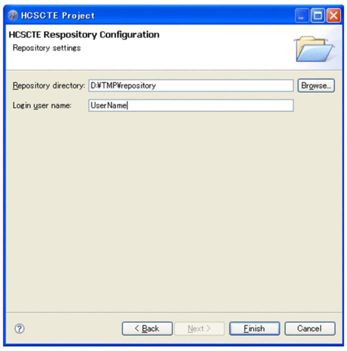
   Project name

   Specify the desired name.

   Use default location

   Select the **Use default location** check box.

   The HCSCTE Project dialog box (the page for specifying an HCSCTE repository) opens.



HCSCTE Project

HCSCTE Respository Configuration

Repository settings

Repository directory: D:¥TMP¥repository

Login user name: UserName

5. Specify the following items.

   Repository directory

   Specify the desired directory for saving the repository information. You consider the following points to specify a repository directory:

   - Do not specify the same path for a repository directory and for a project.

- Specify the path as an absolute path.

- The length of the path is to be checked as a normalized absolute path.

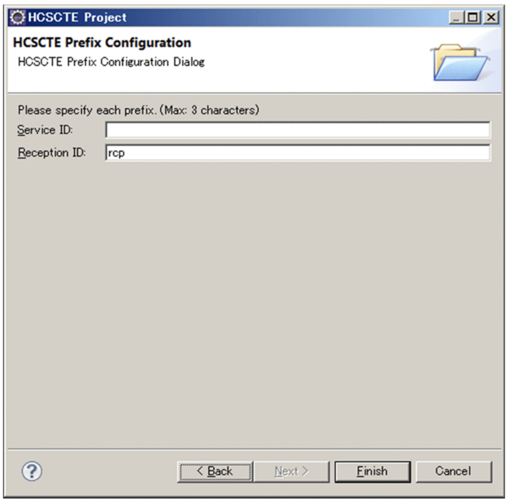
- You cannot use a repository directory that was set up in an HCSCTE project of a SOAP mode different from the SOAP mode specified before creating the HCSCTE project.

Login user name

Specify the user name to be used for logging on to the repository. Only single-byte alphanumeric characters can be used in the user name having a length of 1 to 16 characters.

6. To set up a prefix for the service ID and reception ID, click the **Next** button. If you do not want to set up a prefix, go to step 8.

The **HCSCTE Project** dialog box (page for HCSCTE prefix settings) appears.


HCSCTE Project
HCSCTE Prefix Configuration
HCSCTE Prefix Configuration Dialog
< Back
Next >
Finish
Cancel

Please specify each prefix. (Max: 3 characters)

Service ID:

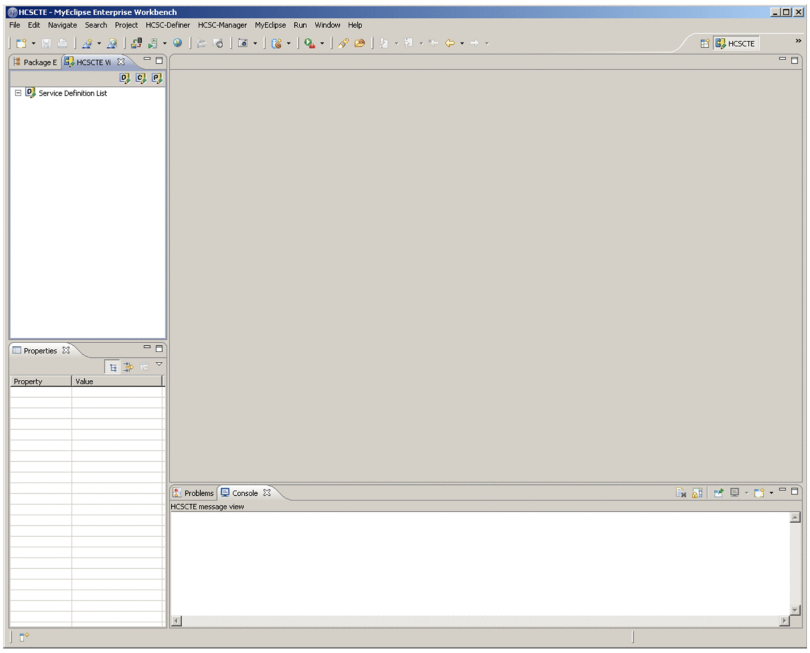
Reception ID: rcp

7. Specify the prefix for the service ID and reception ID.

Specify the prefix for the service ID and reception ID using up to 3 one-byte alphanumeric characters and underline (_).

8. Click the **Finish** button.

The HCSCTE project is created and the perspective of the created HCSCTE project starts.


HCSCTE - MyEclipse Enterprise Workbench
File Edit Navigate Search Project HCSC-Definer HCSC-Manager MyEclipse Run Window Help
HCSCTE
Package E
HCSCTE W
Service Definition List
Properties
Property
Value
Problems
Console
HCSCTE message view

! Important note

Only a single HCSCTE project can be created in each workspace.

3.1.2 Setting up Properties

This section describes the property settings to be implemented after creating the HCSCTE project.

(1) Setting up a Repository

Specify a directory to be used as repository as and when needed, when using a repository different than the repository at the time of creating HCSCTE project.

For details on setting up the repository, see "*3.2.1 Setting Up a Repository*".

(2) Specifying the Java build path

Add the JAR files used for compilation into the build path.

1. From Eclipse menu, select [Project]-[Property].
   [Property: HCSCTE#] dialog is displayed.

   Note#

   Project name is displayed in "HCSCTE" part.
2. Select [Java build and path] from Tree view on the left side of dialog and add the following JAR file.
   `<Service Platform installation directory>\CSC\lib\cscbp_ejb.jar`

(3) Validation Settings

In Validation settings, specify the character set to be used in the database.

! Important note

Note

Implement the character code change of the database to be set in HCSCTE project, before creating HCSC components#. Do not make changes after creating HCSC components#.

Note#

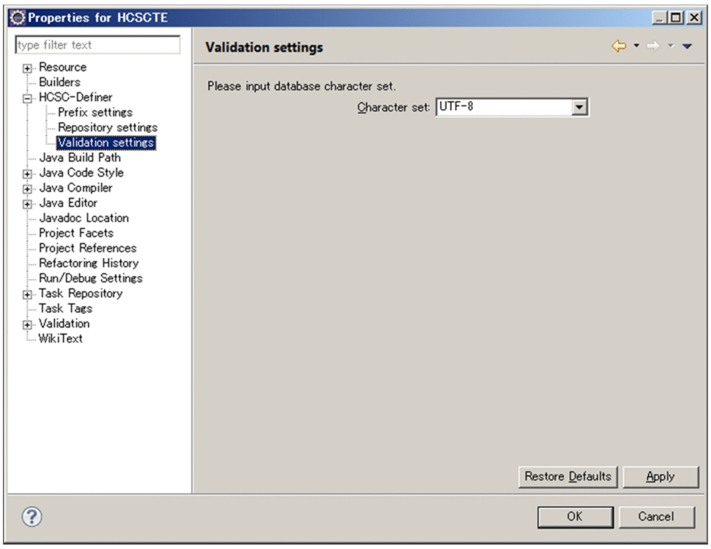
Specifies the service adapter, business process and user-defined reception.

Procedure for specifying the character code in verification settings is as follows:

1. From Eclipse menu, select [Project]-[Property].
   [Property: HCSCTE#] dialog is displayed.

   Note#

   Project name is displayed in "HCSCTE" part.
2. From Tree view on the left side of dialog, select [HCSC-Definer]-[Verification settings].
   Setting items related to verification settings are displayed on right side of dialog.



Properties for HCSCTE

Validation settings

Please input database character set.

Character set: UTF-8

3. In [Character code], specify character code to be used in database.

   The value specified here is used to verify the character string length of service name and activity name of business process.

4. Click the [Apply] button and then [OK] button.

(4) Settings for the prefix added to the service ID and reception ID

To specify a desired prefix to the service ID and reception ID:

1. From Eclipse menu, select [Project]-[Property].

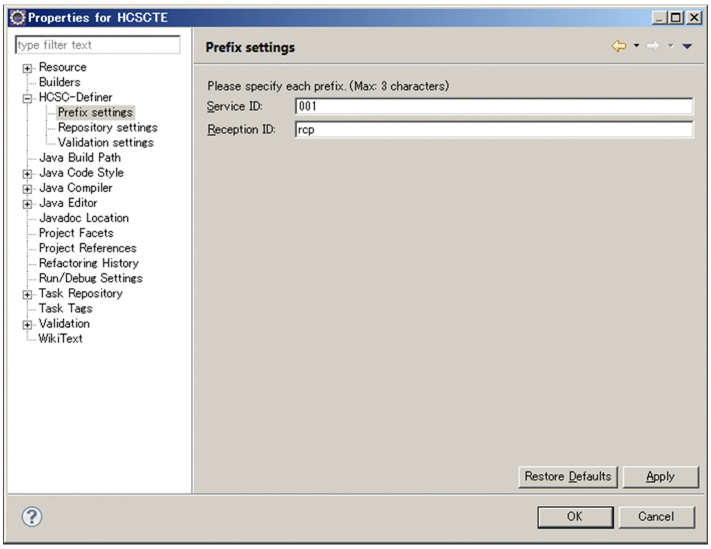
   [Property: HCSCTE#] dialog is displayed.

   Note#

   Project name is displayed in "HCSCTE" part.

2. From Tree view on the left side of dialog, select [HCSC-Definer]-[Prefix settings].

   Setting items related to prefix settings are displayed on the right side of the dialog.



Properties for HCSCTE

Prefix settings

Please specify each prefix. (Max: 3 characters)

Service ID: 001

Reception ID: rcp

3. Specify ID to be added to [Service ID] and [Reception ID].

   Specify prefix of service ID and reception ID with maximum 3 single byte alphanumeric characters and underscore (_).

4. Click the [Apply] button and then [OK] button.

3.1.3 Exporting/Importing a Project

You can use the standard functionality of Eclipse to export an HCSCTE project. You can also import an HCSCTE project that has been exported. However, you cannot create more than one HCSCTE project in each workspace at the same time.

For details about exporting and importing methods, see the documentation supplied with Eclipse.

3.1.4 Deleting a Project

To delete an HCSCTE project:

1. In the Package Explorer of Eclipse, select the project to be deleted.
2. On the Edit menu, choose Delete.
   The selected project is deleted.

3.1.5 Changing SOAP modes

To change the SOAP mode after creating the HCSCTE project, the HCSCTE project must be re-created.

To change the SOAP mode:

1. Delete the created HCSCTE project.
   For details about deleting the HCSCTE project, see "*3.1.4 Deleting a Project*".
2. Terminate Eclipse.
3. Set up the SOAP mode.
   For details about how to set up the SOAP mode, see "*2.2.3 SOAP mode settings*".
4. Start Eclipse.
5. Create an HCSCTE project.
   For details about creating an HCSCTE project, see "*3.1.1 Creating a Project*".

3.1.6 Notes at the time of development

Do not register link of the files in file system, under the lib directory or src directory of HCSCTE project. The files registered with link are not stored in EAR file at the time of service packaging.

3.2 Managing a Repository

A repository is a directory that stores the information shared by the development environment and the operating environment. Set up a repository when you create an HCSCTE project in the development environment. You can also export or import the specified repository as needed.

! Important note

> In the version 07-60 or later, you cannot share the same repository in a development environment and an operating environment. If the repository is shared in an earlier version, prepare the separate repositories in the development environment and the operating environment, and then migrate to a higher version. For details about how to migrate, see "*Appendix A.3 Migrating procedure when a repository is shared between development environment and operating environment in earlier version*".

3.2.1 Setting Up a Repository

This sub section explains how to change a repository specified while creating an HCSCTE project.

(1) Specifying a repository

Specify the directory that is to be used as the repository.

To specify a repository:

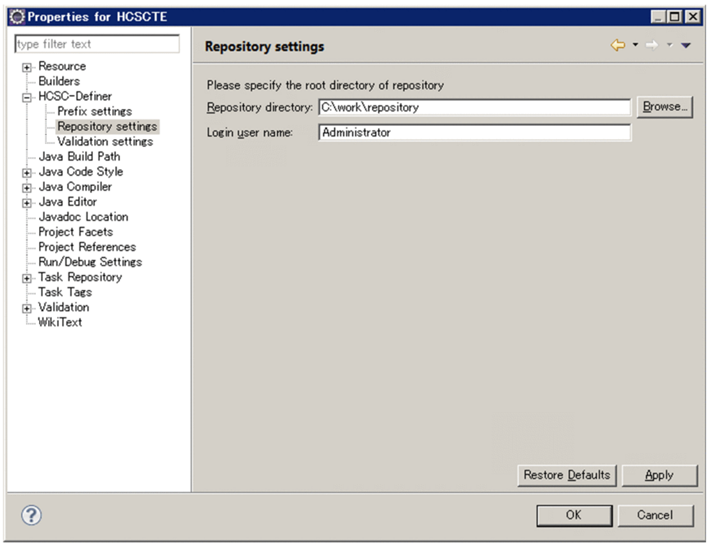
1. On the Eclipse menu, choose Project, and then Property.

   The Properties for HCSCTE# dialog box opens.

   # The actual project name is displayed in place of HCSCTE.

2. In the directory tree in the left pane of the dialog box, choose HCSC-Definer, and then Repository settings.

   Repository settings items are displayed in the right pane of the dialog box.



Properties for HCSCTE

Repository settings

Please specify the root directory of repository

Repository directory: C:\work\repository

Login user name: Administrator

3. In Repository directory, specify the desired directory to be used as the repository.

   When specifying a *repository directory*, consider the following points:

   - Specify a directory that already exists.
   - Do not specify the same path for a repository directory and for a project.
   - Specify the path as an absolute path.
   - The length of the path is checked as a normalized absolute path.
   - You cannot use a repository directory that was set up in an HCSCTE project of a SOAP mode other than the SOAP mode specified before creating the HCSCTE project.

4. In Login user name, enter the desired user name to be used as the login user name.
   Only single-byte alphanumeric characters can be used in the user name having a length of 1 to 16 characters.
5. Click Apply and then OK.

(2) Initializing a repository

Initializing a repository erases all of its contents and returns it to its initial state.

Before initializing a repository, be sure to close all editors. Also, do not access the `lib` directory and the `src` directory of the HCSCTE project. If an error occurs, confirm the following points:

- Make sure that editors (including external editors), used for opening files below the `src` directory of the HCSCTE project, are closed.
- Make sure that programs, that use libraries below the `lib` directory of the HCSCTE project, are not running.

To initialize a repository:

1. On the Eclipse menu, choose HCSC-Definer, Definition information management, and then Definition information initialization.
   A message box opens.

(3) Changing a repository

The procedure for changing a repository is the same as the procedure in *3.2.1(1) Specifying a repository*. When changing a repository, terminate all tasks such as adapter definition and business process definition, except for the repository change task.

Clicking **Restore Default** sets up the repository that was previously specified.

(4) Notes on repositories

Do not directly manipulate the repository file or directory. If the file or directory name is changed or deleted, of if the contents of the file or directory are modified, correct operation cannot be guaranteed.

3.2.2 Exporting a Repository

In exporting the repository, you can consolidate repository in 1 ZIP file and save in the specified directory.

Following are the 2 methods to export a repository.

- Saving all the information in a repository, by consolidating in a ZIP file
- Selecting information of the required services and saving in a ZIP file

(1) Exporting a repository

Procedure for exporting a repository is as follows:

1. From Eclipse menu, select [HCSC-Definer]-[Definition information management]-[Export entire definition information].
   [Export repository] dialog is displayed.
2. Specify a directory for saving and input a name for the ZIP file to be saved.
3. Click the [Save] button.
   ZIP file is saved in the specified directory.
   If the save destination directory has a file with the same name, a dialog is displayed to confirm overwriting of the file. If you want to overwrite the file, click the [Yes] button and save file.

   Reference note

   It is recommended to export the repository as and when required and take backup of the data of repository.

(2) Exporting only a part of service definition

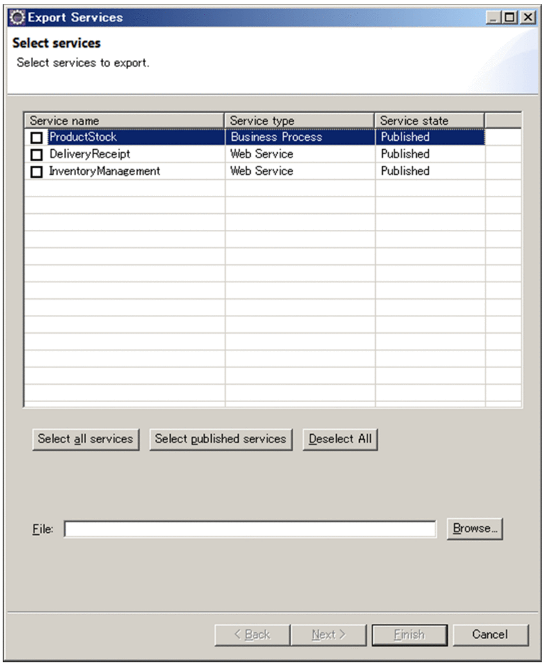
When exporting by selecting the required services, you can save the following information in specified directory.

- Definition related to HCSC components (service adapter, business process)
- Source file of user-defined Java class
- Pre-requisite library file of user-defined Java class

You can select multiple export targets. User-defined reception related to the business process to be exported, is also exported concurrently.

Procedure for exporting only a part of service definitions is as follows:

1. From Eclipse menu, select [HCSC-Definer]-[Definition information management]-[Export services].
   Select service screen of the service export wizard is displayed.



Export Services

Select services

Select services to export.

Service name	Service type	Service state
□ ProductStock	Business Process	Published
□ DeliveryReceipt	Web Service	Published
□ InventoryManagement	Web Service	Published

File:

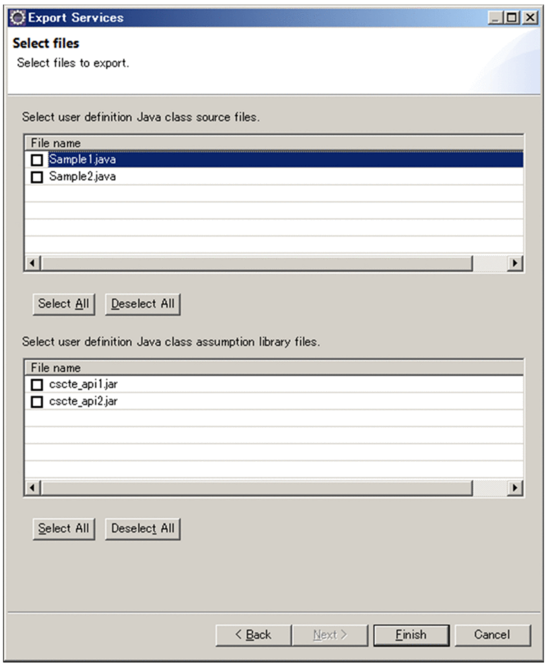
2. Check the checkbox of the service to be exported.
   When you want to check only the publicized services, click [Select only the publicized services].
3. Specify directory for saving by clicking [Browse] button and input the name of ZIP file to be saved.
   Existence of user-defined Java class is checked and either of [Next] button or [Finish] button is enabled.
4. Perform either of the following operations:

   When user-defined Java class does not exist

   Click [End] button
   Selected service definition is saved in ZIP file.

   When user-defined Java class exists

   Click [Next] button.
   Select file screen of service export wizard is displayed. Proceed with the following steps:



Export Services

Select files

Select files to export.

Select user definition Java class source files.

File name
□ Sample1.java
□ Sample2.java

Select user definition Java class assumption library files.

File name
□ cscte_api1.jar
□ cscte_api2.jar

5. Check the checkbox of user-defined Java class to be exported and pre-requisite library file for the same.
6. Click [End] button.
   Selected service definition is saved in ZIP file.

3.2.3 Importing a Repository

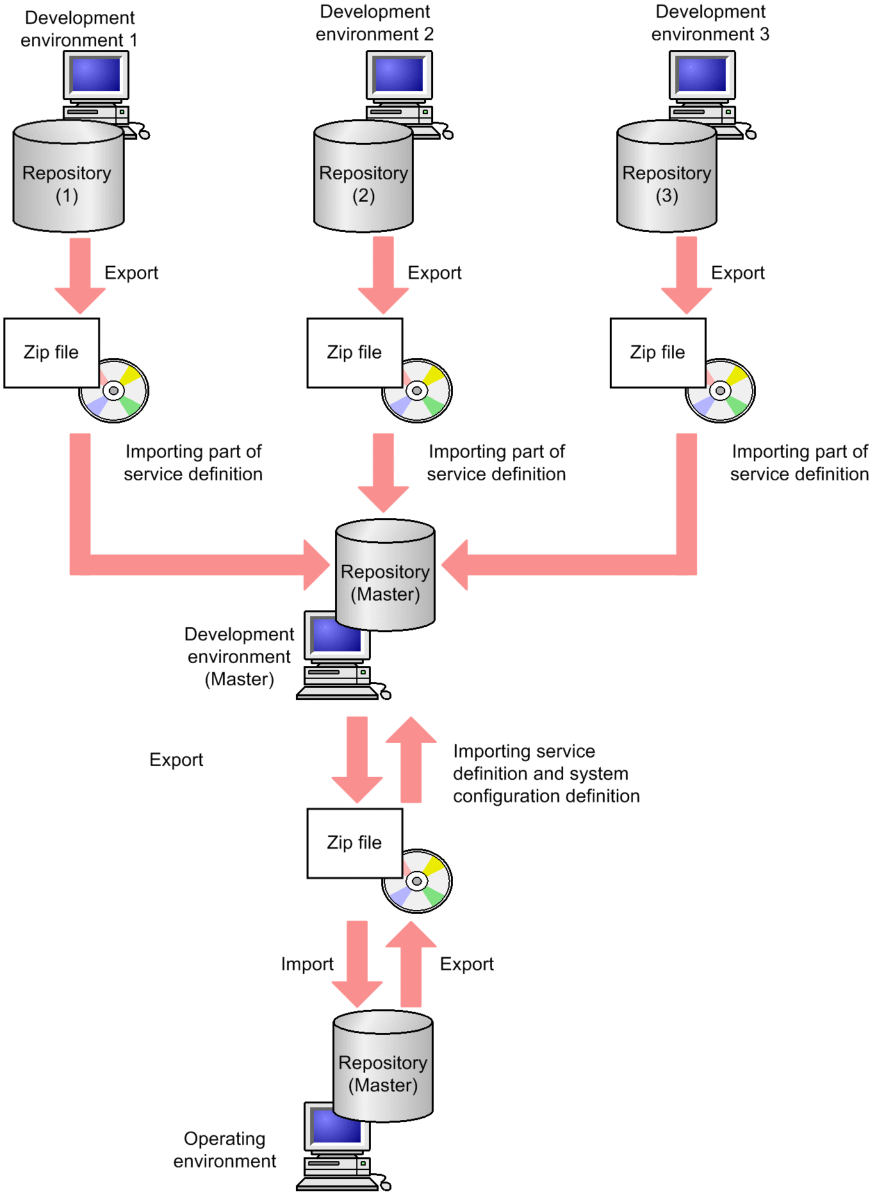
In importing the repository, you can read the repository which was consolidated to ZIP file during export. You can import the definition information such as service definition and system configuration definition saved in the repository. You can import a repository, when SOAP mode of the exported repository match with SOAP mode of the import destination repository.

Following are the 2 methods to import a repository.

- Import service definition and system configuration definition both, or either of those in ZIP file.
- Import only a part of service definitions in the ZIP file.

Following figure shows the example of different usages of the 2 methods of importing.

Figure 3–1: Different usages of the 2 methods of importing (example)


Development environment 1
Development environment 2
Development environment 3
Repository (1)
Repository (2)
Repository (3)
Export
Export
Export
Zip file
Zip file
Zip file
Importing part of service definition
Importing part of service definition
Importing part of service definition
Repository (Master)
Development environment (Master)
Export
Importing service definition and system configuration definition
Zip file
Import
Export
Repository (Master)
Operating environment

As shown in the example in the above figure, when developing HCSC components concurrently in multiple development environments, export the repositories of respective development environments and import in the master repository. At that time, import only the information required in the service definition. Also, when importing the information of a repository exported in operation environment, to the repository (master) of development environment, import service definition and system configuration definition.

You can also receive the exported information through network, without using media.

! Important note

When importing repository, close all the editors such as service adapter definition or business process definition. Also, set such that lib directory or src directory of HCSCTE project is not accessed. If error occurs, check the following points.

- Whether the editor (including external editor) in which file under src directory of HCSCTE project is opened, is closed
- Whether program in which library under lib directory of HCSCTE project is used, is executed

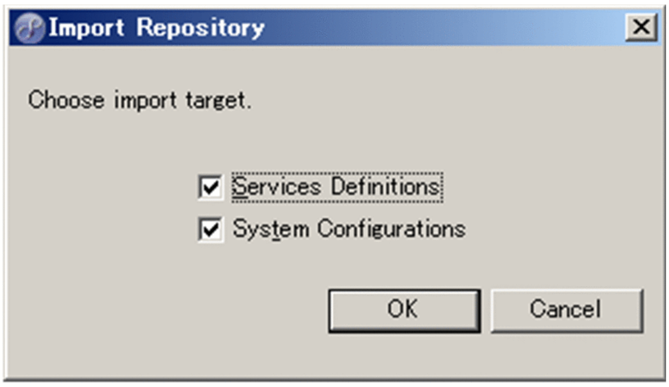
Details of the 2 methods of importing are as follows:

(1) Importing the service definition and the system configuration definition

(a) Procedure of importing

Procedure for importing service definition and system configuration definition in the repository is as follows:

1. From the Eclipse menu, select [HCSC-Definer]-[Definition information management]-[Import entire definition information].
   Dialog for confirming the importing is displayed.
2. Click [Yes] button.
   Dialog [Import repository] for selecting the ZIP file of repository information is displayed.
3. Specify ZIP file to be read.
4. Click [Open] button.
   Dialog [Import repository] for selecting the definition information to be imported is displayed.



Import Repository

Choose import target.

☑ Services Definitions

☑ System Configurations

5. Insert check mark in the check box of definition to be imported.
   When importing for the first time, insert check mark in [Service definition] and [System configuration definition] both.
6. Click [OK] button.
   Repository is read in the directory specified in "*3.2.1(1) Specifying a repository*".

(b) Notes

- When you import the repository of old version, the repository information is inherited to current version.
- When you imported user-defined Java class or pre-requisite library of user-defined Java class to be used in Java calling activity, build the HCSCTE project.
- When you want to import only either of service definition or system configuration definition, integration between definition information of import target within ZIP file and information (service definition or system configuration definition) within current repository must be consistent. In case of non-consistency, you cannot import the service definition or system configuration definition in the repository.

  Contents for checking the consistency during the import of repository are as follows:

  When importing only the service definition

  Following table describes error conditions when checking consistency between publicized service definition in repository and service definition in corresponding ZIP file.

  Table 3–1: Error conditions in consistency check performed while importing a repository (in case of importing only service definition 1)

No.	Error conditions
1	Service IDs do not match
2	Service names do not match[#1]
3	Types (service adapter or business process) do not match[#1]

No.	Error conditions
4	Versions of business process do not match#1#2
5	Number of user-defined receptions defined in business process do not match#1
6	IDs of user-defined reception defined in the business process do not match#1
7	Names of the user-defined reception defined in business process do not match#1
8	Packaging of service definition in ZIP file is not performed#1#2

Note#1

Check the interrelation between service definitions for which service IDs match.

Note#2

Do not check in case of repository created in old version.

Following table describes error conditions when checking consistency between publicized service definition in repository and service definition in corresponding ZIP file for which service IDs match. Error conditions differ depending on server configuration defined in system configuration definition in the repository.

Table 3–2: Error conditions in consistency check performed while importing a repository (in case of importing only system configuration definition 2)

No.	Information in repository			Error conditions (service definition in ZIP file for which service IDs match)
	Server configuration defined in system configuration definition		Publicized service definition (types of service)	
	Usage of database	Usage of Reliable Messaging		
1	Used	Not used	All	--
2	Not used	Not used	Business process	yes is set in persistence of status
3			• SOAP adapter • SessionBean adapter • DB adapter • TP1 adapter • File adapter • Object Access adapter • Message Queue adapter • FTP adapter • File operation adapter • Mail adapter • HTTP adapter • General custom adapter	MDB_WSR or MDB_DBQ has been set in types of service
4	Used	Not used	• SOAP adapter • SessionBean adapter • DB adapter • TP1 adapter • File adapter • Object Access adapter • Message Queue adapter • FTP adapter • File operation adapter • Mail adapter	MDB_WSR or MDB_DBQ has been set in types of service

No.	Information in repository			Error conditions (service definition in ZIP file for which service IDs match)
	Server configuration defined in system configuration definition		Publicized service definition (types of service)	
	Usage of database	Usage of Reliable Messaging		
4	Used	Not used	• HTTP adapter • General custom adapter	`MDB_WSR` or `MDB_DBQ` has been set in types of service

(Legend)

--: Error condition does not exist.

Troubleshooting in case of error occurrence

Non-publicize all services in the current repository and re-import only the service definition.

When importing only the system configuration definition

Following table describes error conditions when checking consistency between publicized service definition in repository and service definition in corresponding ZIP file.

Table 3–3: Error conditions in consistency check performed while importing a repository (in case of importing only the system configuration definition 1)

No.	Error conditions
1	Service IDs do not match
2	Service names do not match#1
3	Types (service adapter or business process) do not match#1
4	Versions of business process do not match#1#2
5	Number of user-defined receptions defined in business process do not match#1
6	ID of user-defined reception defined in business process do not match#1
7	Names of user-defined receptions defined in business process do not match#1
8	Packaging is not performed of service definition in ZIP file#1#2

Note#1

Check the interrelation between service definitions for which service ID match.

Note#2

Do not check in case of repository created in old version.

Following table describes error conditions when checking consistency between publicized service definition in ZIP file and service definition in repository for which service IDs match. Error conditions differ depending on server configuration defined in system configuration definition in ZIP file.

Table 3–4: Error conditions in consistency check performed while importing a repository (in case of importing only system configuration definition 2)

No.	Information in ZIP file			Error condition (service definition in ZIP file for which service IDs match)
	Server configuration defined in system configuration definition		Publicized service definition (type of service)	
	Usage of database	Usage of Reliable Messaging		
1	Used	Used	All	--
2	Not used	Not used	Business process	yes is set in persistency of status

No.	Information in ZIP file			Error condition (service definition in ZIP file for which service IDs match)
	Server configuration defined in system configuration definition		Publicized service definition (type of service)	
	Usage of database	Usage of Reliable Messaging		
3	Not used	Not used	• SOAP adapter • SessionBean adapter • DB adapter • TP1 adapter • File adapter • Object Access adapter • Message Queue adapter • FTP adapter • File operation adapter • Mail adapter • HTTP adapter • General custom adapter	MDB_WSR or MDB_DBQ has been set in types of service
4	Used	Not used	• SOAP adapter • SessionBean adapter • DB adapter • TP1 adapter • File adapter • Object Access adapter • Message Queue adapter • FTP adapter • File operation adapter • Mail adapter • HTTP adapter • General custom adapter	MDB_WSR or MDB_DBQ has been set in types of service

(Legend)

--: Error condition does not exist.

Troubleshooting in case of error occurrence

Take measures by executing the following procedure:

1. Export (rep.zip) and save the current repository.
2. Import service definition and system configuration definition both from the archive file tried to be imported.
3. Non-publicize all the service definitions imported in procedure 2.
4. Import only the service definition from the rep.zip saved in procedure 1.

(2) Importing only a part of the service definitions

When importing only a part of service definitions, you can select following information as import target:

- Definitions related to HCSC components (service adapter and business process)
- Source file of user-defined Java class
- Pre-requisite library files of the user-defined Java class

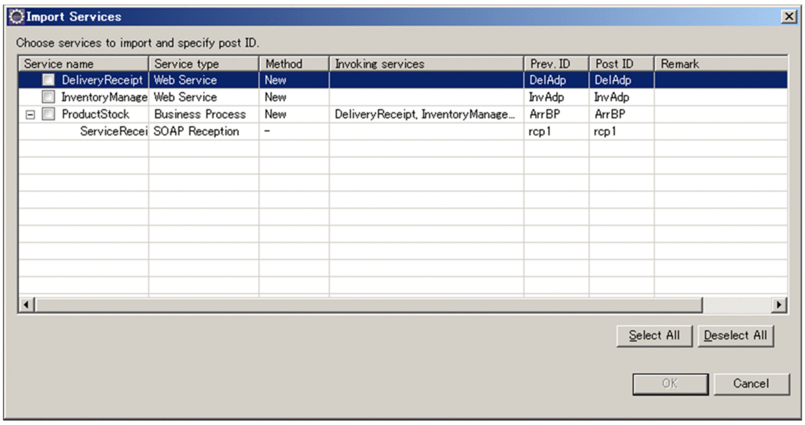
You can select multiple sets of information for importing. Service adapters and business processes invoked from business processes that are selected for importing are imported simultaneously as needed. User-defined reception related to the business process to be imported is also imported simultaneously.

While importing, you can specify any service ID or reception ID to import the service and user-defined reception.

(a) Procedure for importing

Procedure for importing only a part of service definition is as follows:

1. From the Eclipse menu, select [HCSC-Definer]-[Definition information management]-[Import service].
   Dialog for selecting archive file (ZIP file) is displayed.
2. Select the archive file that includes the information to be imported and click [Open] button.
   [Import service] dialog (service selection) is displayed.

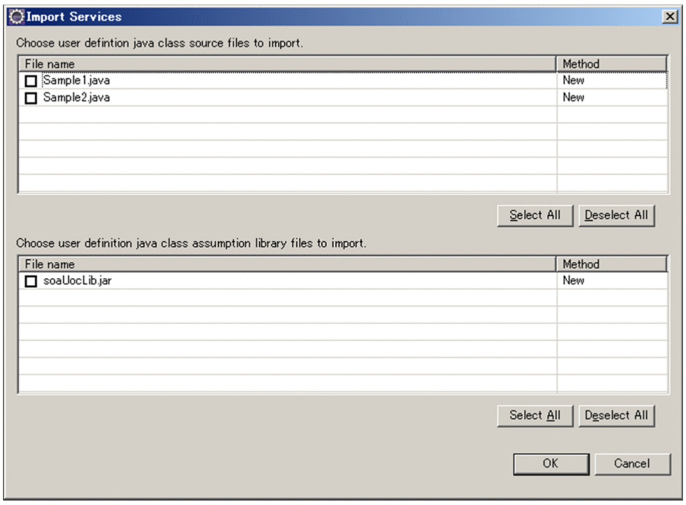


Import Services

Choose services to import and specify post ID.

Service name	Service type	Method	Invoking services	Prev. ID	Post ID	Remark
□ DeliveryReceipt	Web Service	New		DelAdp	DelAdp	
□ InventoryManage	Web Service	New		InvAdp	InvAdp	
⊟ □ ProductStock	Business Process	New	DeliveryReceipt, InventoryManage...	ArrBP	ArrBP	
ServiceRecei	SOAP Reception	-		rcp1	rcp1	

3. Check the checkbox of the service to be imported.
4. Change the [Post-incorporation ID] of the service to be imported, as and when required.
   Specify ID by taking note of the following points:
   - Specify with 8 or less alphanumeric characters and underscore (_).
   - You cannot specify an ID that is already used in import destination.
     However, even if it is ID used in import destination, if that ID is no longer used depending on the concurrently imported services, you can specify that ID.
   - You cannot specify ID same as other services or user-defined reception.
   - When integrating services distributed and developed across multiple repositories, interface inconsistency occurs till all the services are imported and warming message might be displayed in Remarks column. If inconsistency occurring till service integration is anticipated, there is no problem even if warning message is displayed in Remarks column.
     However, when inconsistency occurs in interfaces, correct design information is not output and hence do not output the design information.
5. Click the [OK] button.
   Existence of user-defined Java class is checked.
   When user-defined Java class exists, [Import service] dialog (selection of user-defined Java class) is displayed. Proceed with the following steps:
   When user-defined Java class does not exist, contents of the selected repository are read in the directory specified in "3.2.1(1) Specifying a repository".
6. Check the checkbox of user-defined Java class to be imported and its pre-requisite library files.



Choose user defintion java class source files to import.

File name	Method
□ Sample1.java	New
□ Sample2.java	New

Choose user definition java class assumption library files to import.

File name	Method
□ soaUocLib.jar	New

7. Click the [OK] button.
   Existence of user-defined class is checked.
   When user-defined Java class exists, [Import service] dialog (selection of user-defined Java class) is displayed.
   Contents of selected repository are read in the directory specified in "3.2.1(1) Specifying a repository".

Tip

- If you cannot select an import target such as in case where HCSC component of same name is publicized in the repository, the display of service in [Import service] dialog (service selection) becomes inactive. Eliminate the cause with reference to contents in [Remarks] column and import again.
- Always perform packaging of the imported HCSC component (service adapter, business process). For details on the method of packaging, see "7.2 Packaging".
- When you import user-defined Java class or pre-requisite library of user-defined java class to be used in Java calling activity, build HCSCTE project.

(b) Notes

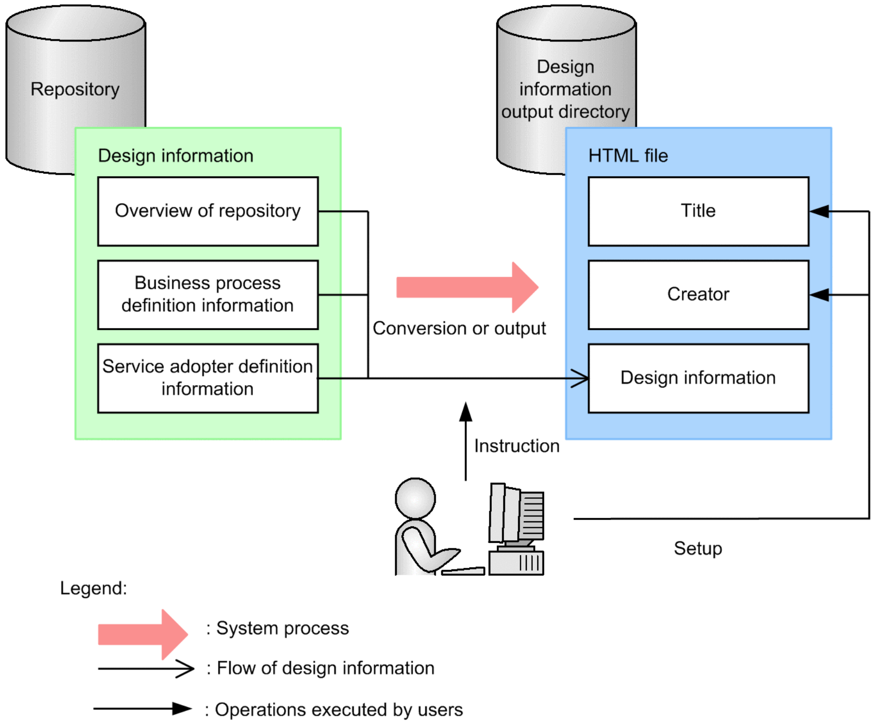
- You cannot import, when HCSC component with same name has been publicized in the repository. Non-publicize that component before importing.
- When you import only a part of service definitions of repository, system configuration definition is not imported. When you import a publicized HCSC component, the imported HCSC component is non-publicized.
- When import destination directory has HCSC component with same name as the HCSC component to be imported, all the contents are overwritten. Also, all the format IDs are updated.
- You cannot import only a part of service definitions from repository created in old version. Move the repository as per the following procedure and then import a part of service definition.
  1. Export the current directory to archive file.
  2. Import the entire definition information from archive file of old version.
  3. Export the definition information imported in Step.2 to a different archive file.
  4. Import the entire definition information from the archive file imported in Step.1.
  5. Import a part of service definitions from the archive file exported in Step.3
- When you want to import an exported repository to the development environment, immediately after setting up HCSC server in operation environment, do not import only a part of service definitions, but import the entire definition information by selecting [HCSC-Definer]-[Definition information management]-[Import entire definition information] from the Eclipse menu.

3.3 Output of design information

In the development environment, you can output an overview of repository, the definition information of business processes, and the definition information of service adapters as the design information in an HTML file. When you output the design information, the efficiency of operations can be improved because the defined information can be checked without using windows of the development environment. Also, the design information is output into the HTML file in a uniform format.

The following figure shows an overview of the design information output:

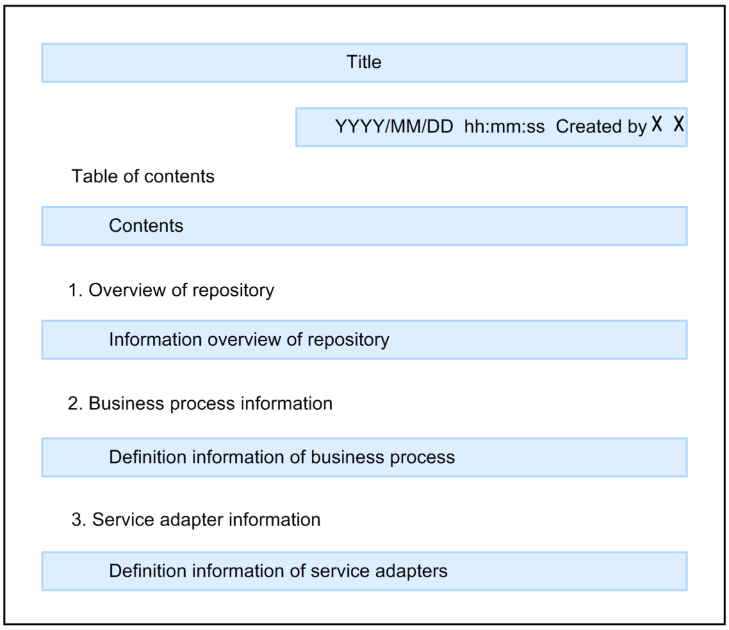
Figure 3–2: Overview of design information output


Repository
Design information output directory
Design information
Overview of repository
Business process definition information
Service adapter definition information
Conversion or output
HTML file
Title
Creator
Design information
Instruction
Setup
Legend:
: System process
: Flow of design information
: Operations executed by users

3.3.1 Information that can be output as design information

You can select the information that can be output as the design information using the Design Information Output Wizard. The following figure shows the information that can be output as the design information:

Figure 3–3: FigureInformation that can be output as design information


Title
YYYY/MM/DD hh:mm:ss Created by X X
Table of contents
Contents
1. Overview of repository
Information overview of repository
2. Business process information
Definition information of business process
3. Service adapter information
Definition information of service adapters

Title

The character string specified in *Title* in the output wizard of the design information is output.

Date and time of creation

This is the date and time when the output of design information is started. This is displayed in the `YYYY/MM/DD hh:mm:ss` format.

Created by

When a creator who created the document is specified in *Created by* in the output wizard of the design information, the specified creator's name is output. This is displayed in the `Created by` × × format. × × indicates the name of the creator.

If the name of the creator is not specified, the name will not output and only `Created by` will be displayed.

Table of contents

The table of contents of the output information is output.

Overview of the repository

The product version and overview about all the service components that exist in the repository and the dependence of each service is output.

Business process information

The information defined by the user in the Business Process Definition screen is output in each service component. The definition information of a business process also contains a figure showing the placement of activities defined in the Business Process Definition screen.

Service adapter information

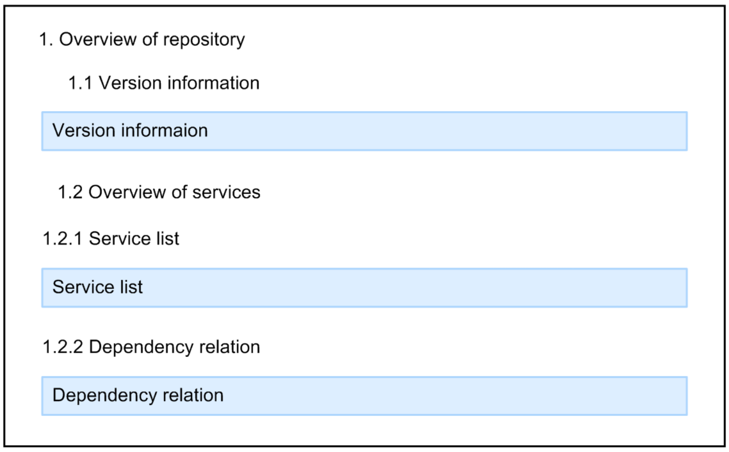
The information defined by the user in the Service Adapter Definition screen is output in each service component.

The information that can be output as the repository overview, business process information, and service adapter information is described here. The definition information about the data transformation that is included in the business process information is also described.

(1) Overview of the repository

The following figure provides an overview of repository.

Figure 3–4: FigureOverview of the repository



1. Overview of repository

1.1 Version information

Version informaion

1.2 Overview of services

1.2.1 Service list

Service list

1.2.2 Dependency relation

Dependency relation

Version information

The product version of uCosminexus Service Architect is output.

Service list

The overview information of the service components selected in the Design Information Output Wizard is output.

Dependency relation

The dependence of each service component selected in the Design Information Output Wizard is output.

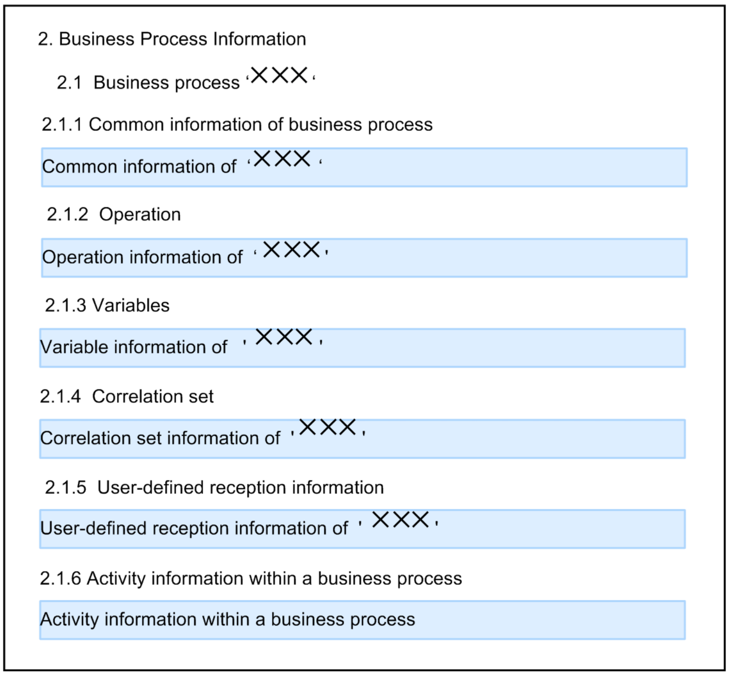
(2) Business process information

The output business process information differs depending upon whether the `scope` activity and the `iteration` activity is defined.

The following figure shows the business process information:

Figure 3–5: FigureBusiness process information

- When scope activity and while activity are not defined



2. Business Process Information

2.1 Business process 'XXX'

2.1.1 Common information of business process

Common information of 'XXX'

2.1.2 Operation

Operation information of 'XXX'

2.1.3 Variables

Variable information of 'XXX'

2.1.4 Correlation set

Correlation set information of 'XXX'

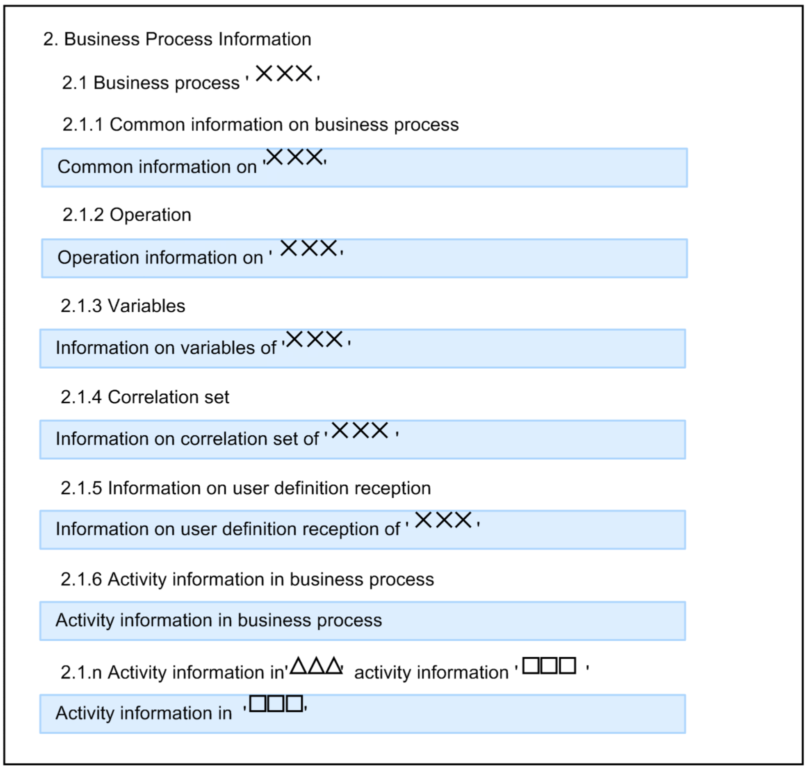
2.1.5 User-defined reception information

User-defined reception information of 'XXX'

2.1.6 Activity information within a business process

Activity information within a business process

- When scope activity and while activity are not defined



2. Business Process Information

2.1 Business process '×××'

2.1.1 Common information on business process

Common information on '×××'

2.1.2 Operation

Operation information on '×××'

2.1.3 Variables

Information on variables of '×××'

2.1.4 Correlation set

Information on correlation set of '×××'

2.1.5 Information on user definition reception

Information on user definition reception of '×××'

2.1.6 Activity information in business process

Activity information in business process

2.1.n Activity information in'△△△' activity information '□□□'

Activity information in '□□□'

Common information of × × ×

The information set up when creating a business process, dependency of service components, and a list of activities will be output. The business process name will be displayed in × × ×.

Operation information of × × ×

A list of operations of a business process and the definition information of the communication model and messages of each operation will be output as the detailed operation information.

Variable information of × × ×

The information of all the global variables defined in a business process will be output.

Correlation set information of × × ×

The information of all the correlation sets defined in a business process will be output.

User-defined reception information of × × ×

A list of user-defined receptions defined in a business process and the detailed information of each user-defined reception will be output.

Activity information within a business process

The information of each activity placed on the campus will be output.

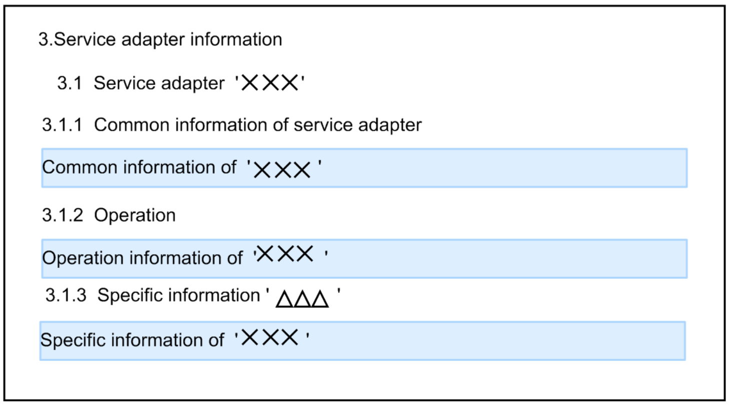
Activity information within □ □ □

The information of the `scope` activity and the `while` activity will be output in loops. `Scope` or `While` is displayed in △ △ △, and the `scope` activity name or the `while` activity name is displayed in □ □ □.

(3) Service adapter information

The following figure shows the service adapter information:

Figure 3–6: FigureService adapter information



3.Service adapter information

3.1 Service adapter '×××'

3.1.1 Common information of service adapter

Common information of '××× '

3.1.2 Operation

Operation information of '××× '

3.1.3 Specific information ' △△△ '

Specific information of '××× '

Common information of × × ×

The service component control information and invoking service are output. The service name is displayed in × × ×.

Operation information of × × ×

A list of operations of an adapter and the definition information of the communication model and messages of each operation will be output as the detailed operation information.

Specific information of × × ×

The information set up when an adapter is defined in the Development Environment screen will be output in each service type. The service type is displayed in △ △ △ . The following are the points to be considered when referencing the specific information for each service type:

Web Services

- When using basic authentication, **** is displayed irrespective of the set password.
- If the client definition file contains the characters that cannot be used in XML, the client definition file will not be output properly.

SessionBean

If the client definition file contains the characters that cannot be used in XML, the client definition file will not be output properly.

MDB (WS-R)

When using basic authentication, **** is displayed irrespective of the set password.

MDB (DBQueue)

The specific information is not output.

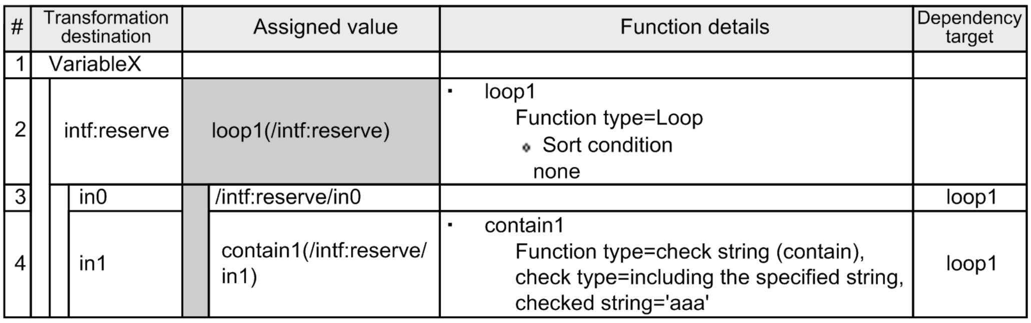
For a general purpose custom adapter

When the self-defined file is not in the XML format, the information will not be output properly.

(4) Definition information of data transformation

When the data transformation is defined in a business process or a service adapter, the corresponding table of the data transformation will be displayed. The following figure shows the corresponding table of the data transformation:

Figure 3–7: FigureCorresponding table of data transformation



#	Transformation destination	Assigned value	Function details	Dependency target
1	VariableX			
2	intf:reserve	loop1(/intf:reserve)	・ loop1 Function type=Loop ⋄ Sort condition none	
3	in0	/intf:reserve/in0		loop1
4	in1	contain1(/intf:reserve/in1)	・ contain1 Function type=check string (contain), check type=including the specified string, checked string='aaa'	loop1

Transformation destination

Each node included in the transformation destination schema is output.

Assigned value

A value assigned to the transformation-destination node is output.

One or more functions and the transformation source node will be output.

Function details

If the function output in the `assigned value` contains detailed information, the detailed information will be output. If the function output in `assigned value` contains multiple instances of the detailed information, the multiple instances of detailed information will be output.

Dependency target

The loop node function on which the `assigned value` depends will be output.

The following table describes the information output in the `assigned value` and `function details` for the function:

Table 3–5: TableInformation output in the assigned value and function details

No.	Function name	Output format of the substitution value	Output format of function details
1	Concatenate	`concat*` (*target-to-be-concatenated*, ...)	--
2	Acquire substring	`substr*` (*target-to-be-split*)	Any of the following is output after `.substr*`: • *acquisition-method* = *range-specified-from-the-beginning*, *start-position* = *start-position*, *character-count* = *character-count* | *acquire-from-the-start-position-until-the-end* • *acquisition-method* = *range-specified-from-the-end*, *start-position* = *start-position*, *character-count* = *character-count* | *acquire-from-the-start position-until-the-beginning* • *acquisition-method* = *split-character-string-specified*, *split-character-string* = *'character-string-used-for-splitting'*, *acquisition-position* = *forward* | *backward*
3	Acquire string length	`length*` (*target-to-acquire-character-count*)	--
4	Check string	`contain*` (*target-to-check-whether-or-not-a-specific-character-string-is-included*)	`.contain*` *check-type* = *specified-character-string-included* | *start-from-specified-character-string*, *check-character-string* = '*check-character-string*'
5	Trim node	`trim*` (*target-from-which-blanks-are-to-be-removed*)	--

No.	Function name	Output format of the substitution value	Output format of function details
6	Convert number format	`format*`(*number-to-be-formatted*)	`.format*` *pattern* = '*format-pattern*', *symbol-change* = *none* | (*decimal-point-character* = '*decimal-point-character*', *digit-delimiter* = '*digit-delimiter*')
7	Perform node operation	`calc*`(*target-for-first-operation*, '+ | - | * | / | %', *target-for-second-operation*)	--
8	Round node	`round*`(*target-for-which-fraction-processing-is-to-be-performed*)	`.round*` *fraction-processing-type* = *round-off* | *round-down* | *round-up*
9	Sum up nodes	`sum*`(*target-for-which-the-sum-is-to-be-acquired*, ...)	--
10	Acquire node count	`count*`(*node-for-which-the-node-count-is-to-be-acquired*)	--
11	Acquire node name	`name*`(*node-for-which-the-node-name-is-to-be-acquired*)	--
12	Check node	`exist*`(*node-whose-existence-is-to-be-confirmed*)	--
13	Loop node	`loop*`(*path-forming-the-reference-for-the-loop*)	`.loop*` *sort-conditions none* | *key* = *key-node*, *order* = *ascending-order* | *descending-order*, *language* = *auto* | *Japanese* | *English*, *type* = *text* | *number*, *priority-order* = *upper-case* | *lower-case*
14	Choose node	`choose*`([*condition-value*]=>[*output-value*], ..., [When no condition is matching]=>[*output-value*]) Any one of the following is output in [*output-value*]: • Node is not output • Empty node • *output-value* • Value is output	--
15	Set constant	`const*`(*character-string*: '*value*' | *number*: '*value*' | *logical-value*: *real* | *fake* | *special-node*: *no-node-output* | *empty-node*)	--
16	Replace value	`replace*`(*node-to-be-replaced*)	`.replace*` *transformation-table-ID* = '*Transformation-table-ID*', *path-property* = 'csc.dt.valueTable.*transformation-table-ID*', *code-property* = 'csc.dt.encodeType.*transformation-table-ID*', *search-key-column-specification* = *left-column* | *right-column*, *operation-during-search-failure* = *replace-default-value*, *value* = '*value*' | *transformation-error*
17	Custom	`custom*`(*input-value-of-argument*, ...)	`. custom*`

No.	Function name	Output format of the substitution value	Output format of function details
17	Custom	Note If no arguments exist, nothing is displayed within the round brackets.	`jar file ='`*jar-file-name*`', Class='`*class-name*`', Method='`*method-name*`', Method description ='`*method-description*`'` Note If arguments exist, the argument name and the argument description are displayed in a list immediately under the information displayed in the above format.

Legend:

--: No information is to be output.

3.3.2 Items to be checked before output

The following are the items to be checked before the output of design information:

(1) Usable Web browsers

Use the following Web browsers to reference the output design information (HTML):

- Internet Explorer 6
- Internet Explorer 7
- Internet Explorer 8

(2) Repository status

When you output the design information, Hitachi recommends that you use a repository in which no error occurs during the packaging of HCSC components. If an error occurs during the packaging, due to a validation error in the data transformation definition, the corresponding table of the data transformation definition, in which the validation error occurred, will not be output.

Note that when a validation error of a data transformation definition occurs, the message reporting the occurrence of an error in the console view is displayed. When the design information output terminates, a dialog box reporting the occurrence of an error is displayed.

(3) Output destination of the design information

Specify a location containing no directories or files as the output destination of the design information. If directories or files exist, a file might be overwritten or unnecessary files might remain. Note that if a directory or file exists at the output destination of the specified design information, a dialog box enquiring whether to proceed with the processing will be displayed.

(4) Output time of the design information

A significant amount of time might be consumed in the output of the design information if the data to be output is large, or depending upon the performance of the machine used. Note that during the output of the design information, you will not be able to perform any other task on Eclipse.

(5) Printing the design information

When you print from a browser, the right end of the output contents might be truncated during the printing if the size of the print paper is small. Therefore, Hitachi recommends that you check the printing contents with the print preview functionality before printing, and set up the correct paper size and margins.

(6) When using with Windows Vista and Windows 7

When you want to specify a directory that requires administrator privileges, as the output destination of the design information, execute Eclipse for Cosminexus with administrator privileges. If administrator privileges are not granted, an error will occur during the output of the design information.

3.3.3 How to output the design information

The method of outputting design information is described below.

1. From the menu, select **HCSC-Definer - Design Information Output**.

   When a service or user-defined reception is being edited

   A dialog box that confirms whether to save (privatize) the service or user-defined reception being edited appears. If multiple services or user-defined receptions are being edited, the dialog box appears multiple times. To save, click the **Yes** button. To save all without displaying the confirmation dialog box thereafter, click the **Yes to All** button. If you do not want to save the service or user-defined reception, click the **Cancel** button and end the processing.

   Proceed to step 2.

   When a service or user-defined reception is not being edited

   The output wizard of the design information will be displayed. Proceed to step 3.

2. Click Yes or Yes to All.

   The output wizard of the design information (Input Basic Information page) will be displayed. Note that the processing will differ as follows depending upon whether the deployment definition is performed (public) or not performed (private) for the service or user-defined reception being edited:

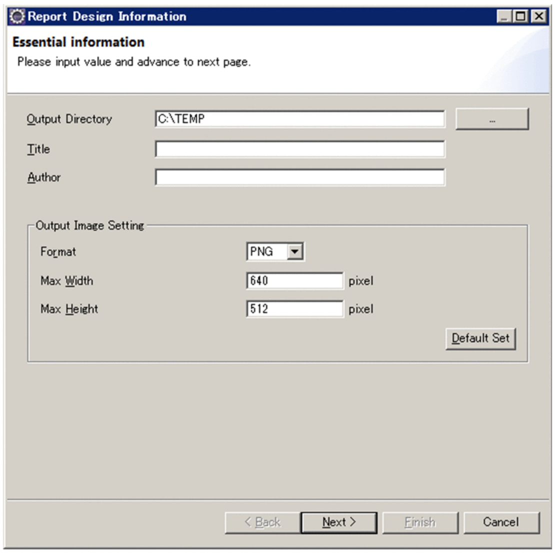
   When deployment definition is performed for a service or user-defined reception

   If deployment definition is performed for a service or user-defined reception, set to a state in which deployment definition is not performed. Subsequently, save the service or user-defined reception being edited and proceed with the processing.

   When deployment definition is not performed for the service or user-defined reception

   Save the service or user-defined reception being edited and proceed with the processing.

3. Set up the required information, such as the output destination directory, in the output wizard of the design information (Input Basic Information page):



Report Design Information

Essential information

Please input value and advance to next page.

Output Directory: C:\TEMP

Title:

Author:

Output Image Setting

Format: PNG

Max Width: 640 pixel

Max Height: 512 pixel

Output destination directory

Specify the directory to output the design information. When you specify the output destination of the design information, consider the following points:

- Specify a location containing no directories or files as the output destination. If the specified directory contains a directory or a file with the same name, that directory or file will be overwritten. If the number of business processes is different, unnecessary image files will remain.

- Specify the path as an absolute path.

- Specify a path of 239 bytes or lesser with the character code `MS932`.

- When an error occurs after the output of the design information is started and the output processing is interrupted, directories or files will remain until the time of interruption.

Title

Specify the character string to be output as the title of the design information.

Note that you can use the following character codes as the character strings to be specified in the title:

`U+0009`, `U+0020-U+D7FF`, `U+E000-U+FFFD`, `U+10000-U+10FFFF`.

You can also use a tab (`U+0009`), single-byte blank (`U+0020`), and double-byte blank (`U+3000`) only between the characters.

Created by

Specify a character string to be output as the name of the creator who created the design information.

Note that you can use the following character codes as the character strings to be specified as the name of the creator:

`U+0009`, `U+0020-U+D7FF`, `U+E000-U+FFFD`, `U+10000-U+10FFFF`.

You can also use a tab (`U+0009`), single-byte blank (`U+0020`), and double-byte blank (`U+3000`) only between the characters

Format

Specify the format of the output image of the activity of the activity placement figure. Choose from the PNG format, BMP format, and JPEG format.

Maximum width

Specify the maximum width of the output image. You can specify a value within the range from 250 to 2500.

When you exceed the maximum width specified for the width size of the output image, reduce the output image while maintaining the ratio between the width and height so as to fit within the maximum width.

Maximum height

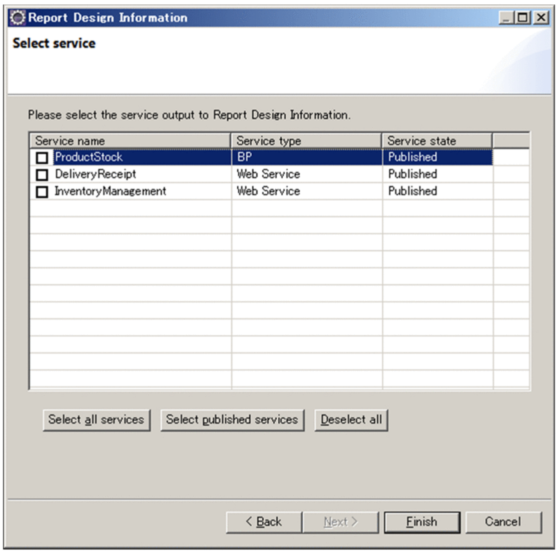
Specify the maximum height of the output image. You can specify a value within the range from 200 to 2000.

When you exceed the maximum height specified for the height size of the output image, reduce the output image while maintaining the ratio between the width and height so as to fit within the maximum height.

4. Click the **Next** button.

   The Design Information Output Wizard (Select Service page) appears.

5. In the Design Information Output Wizard (Select Service page), select the check box of the service that will be output as the design information.



Report Design Information

Select service

Please select the service output to Report Design Information.

Service name	Service type	Service state
□ ProductStock	BP	Published
□ DeliveryReceipt	Web Service	Published
□ InventoryManagement	Web Service	Published

6. Click Finish.

   The output processing status of the design information is displayed in a dialog box.

   When the output processing finishes, the processing results are displayed in the dialog box, and the design information is output in the output destination directory.

3.3.4 Notes on outputting design information

The following are notes on outputting design information.

- If the following methods are used to specify an XPath expression, the reference destination link to variables in the XPath expression is not output. Use the search function of your browser (or other method) to refer to information about the variables.
    - Method for obtaining information by specifying variable names directly
    - Method for obtaining information within variables by using the following extension functions:
      - `csc:getMessageInitialize`
      - `csc:getHexVariableData`
      - `csc:getHexString`
- If the self-defined file is an XML file where one of the following character encodings is specified, an attempt to output design information might fail to output the contents of the file.
    - ISO-10646-UCS-2
    - UTF-32
    - UTF-32BE
    - UTF-32LE
    - ISO-10646-UCS-4

3.4 Notes regarding Eclipse

If Eclipse is started even if HCSCTE project and HCSCTE repository are properly set, Tree view is not displayed in HCSCTE view and error dialog is also not displayed in some cases. Troubleshooting is as follows:

1. Close HCSCTE view once and then re-open the HCSCTE view by the following method:
    - Open a dialog with [Windows]-[View]-[Others].
    - From the opened dialog, select [HCSCTE-Definer]-[HCSCTE view] and click [OK] button.
2. If the problem is not resolved with process in step 1, end Eclipse and perform the following procedure.
    - Add following item in <Eclipse installation directory>\eclipse.ini:
      -Dequinox.statechange.timeout=<timeout>
      Set timeout value of 20000 or above (unit: milliseconds) in <timeout> part.
    - Restart Eclipse.

      ! Important note

      When using Eclipse on Windows Vista or Windows 7, set the desktop theme as Windows Classic.

4 Creating Message Formats

This chapter explains the process of creating message formats.

4.1 Message Formats and Data Transformations

This section provides an overview of the messages to be exchanged among service requesters, service adapters, and service components, as well as their message formats and data transformations.

(1) Message types

A message that requests the execution of a service component from a service requester via a service adapter is called a *request message*.

Conversely, a message that returns the service component execution result to the service requester via a service adapter is called a *response message*.

(2) Message format

The format of a message that is exchanged among service requesters, service adapters, and service components is called a *message format*.

A service adapter has a standard message format for receiving requests. This is called a *standard message*. A message format unique to a service component is called a *service component message*.

A file that defines a message format is called a *message format definition file*.

Depending on the type of data handled, a message format definition file can be in either XML format or binary format. For details about the types of message format definition files, see *4.2 Message Format Types*.

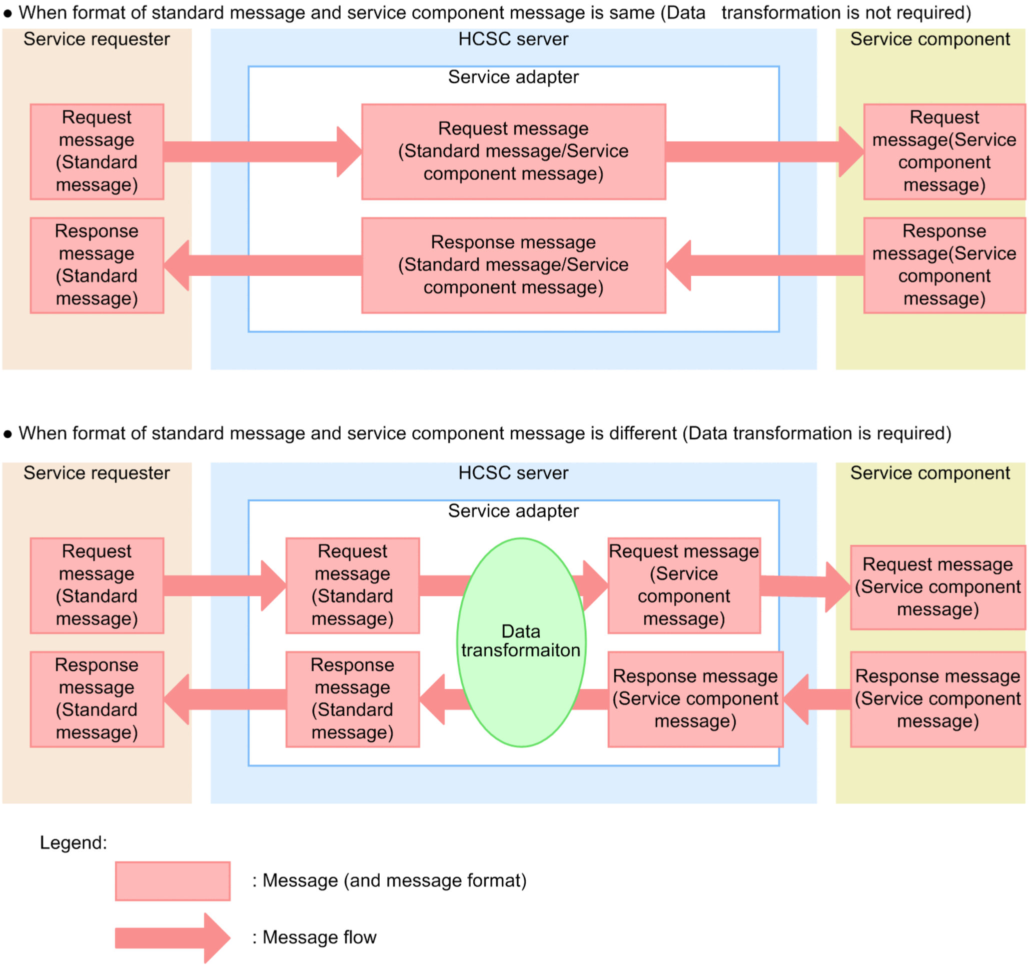
(3) Data transformation

Normally, when a service component invocation request is received from a service requester, a standard message can be used to invoke the service component. However, if the service component cannot be invoked with the request message (in standard message format) received from the service requester, the message format of the request message must be converted to the message format of the service component (service component message). This conversion is called a *data transformation*. Likewise, when a response message from the service component cannot be returned to the service requester, the message format must be converted to the standard message format.

For details about how to define data transformation, see *Chapter 6. Defining Data Transformations*.

The following figure shows the flow of messages between a service requester and a service component when a service adapter is used.

Figure 4–1: Flow of messages between a service requester and a service component



● When format of standard message and service component message is same (Data transformation is not required)

● When format of standard message and service component message is different (Data transformation is required)

Service requester
HCSC server
Service component
Service adapter
Request message (Standard message)
Request message (Standard message/Service component message)
Request message(Service component message)
Response message (Standard message)
Response message (Standard message/Service component message)
Response message(Service component message)
Request message (Standard message)
Request message (Service component message)
Request message (Service component message)
Data transformaiton
Response message (Standard message)
Response message (Service component message)
Response message (Service component message)
Legend:
: Message (and message format)
: Message flow

4.2 Message Format Types

The message format to be created differs depending on whether the message used for executing a service component handles XML data or binary (non-XML) data.

(1) For handling XML data

To handle XML data, create an *XML format definition file*.

An XML format definition file is created as an XML schema file (extension: .xsd). The contents of the XML format definition file to be created and how to create it differ depending on the type of service component to be executed. For details about how to create an XML format definition file for each service component, see *4.3 Creating Message Formats (XML Format Definition File)*.

(2) For handling binary data

To handle binary (non-XML) data, create a *binary format definition file*.

In a binary format definition file, you can define the text, binary, and CSV format data structures (such as fixed length and separators) and data character codes (such as shift JIS, JIS Kanji, Unicode, KEIS, IBM Kanji and JEF).

For details about how to create a binary format definition file, see *4.4 Creating Message Formats (Binary Format Definition File)*.

4.3 Creating Message Formats (XML Format Definition File)

This section explains how to create a message format when the message to be used for executing a service component handles XML data.

4.3.1 Creating a Message

To create an XML format definition file for a message, you can use WST (Web Standard Tool, which is provided by Eclipse) regardless of the type of service component to be executed.

The XML format definition file to be created must satisfy the conditions described in *2.6.5 Scoping of XML schema*. For details about the conditions, see *2.6.5 Scoping of XML schema*.

While selecting the message format file to be used, if an error (error messages beginning with message ID KECX) occurs in the service adapter and business process definition, there is an error in the XML schema file. In such a case, reference the messages in the manual *uCosminexus Application Server Messages* and take appropriate measures.

4.3.2 Creating a Service Component Message (for Web Services)

If the service component you want to use is a Web Service, you need not create an XML format definition file for the service component message. An XML format definition file is automatically created when the service adapter is created.

However, before creating the service adapter, you must prepare a WSDL file that fulfills the conditions described in *2.6.1 Applicability of the service components that use Web service*.

Notes: When Web Service throws a user-defined exception with the SOAP Communication Infrastructure

When the communication style of the WSDL generated by the WSDL generating function of the SOAP application development support function is *rpc*, you must modify the WSDL and create a corresponding adapter.

When the communication style of the generated WSDL is document, there is no need to modify the WSDL. When a Cosminexus Service Platform uses `fault`, Hitachi recommends that you use WSDL in the document format.

To modify the WSDL:

1. Define `xsd:element`.

   Define the `xsd:element` element in the `xsd:schema` element inside the `wsdl:types` element that has the same `targetNamespace` attribute as the name space indicated by the `targetNamespace` attribute of the `wsdl:definitions` element.

2. Obtain the corresponding `wsdl:message` element from the value of the `message` attribute of the `wsdl:fault` element in the `wsdl:operation` element inside the `wsdl:portType` element. Check the `wsdl:part` element contained inside.

3. Add the name attribute to the `xsd:element` element defined in step 1.

   Use the same value as the local name of the `type` attribute of the `wsdl:part` element obtained in step 2.

4. Following the same procedure as in step 3, add the `type` attribute.

   Use the same value as the `type` attribute of the `wsdl:part` element obtained in step 2.

5. From the `wsdl:part` element checked in step 2, delete the `type` attribute and add a new `element` attribute.

   For the name space prefix, use the prefix that indicates the `xsd:schema` element inside the `wsdl:types` element referenced in step 1. Additionally, set the local name to the same value as the `name` attribute added in step 3.

After you have modified the WSDL, you need to re-create Web Services itself.

Execute the command shown below and re-create a skeleton for Web Services from the modified WSDL.

```
WSDL2Java -s WSDL-file
```

For details about the WSDL2Java command, see the section related to skeletons in the manual *Cosminexus Application Server SOAP Application Development Guide*

4.3.3 Creating a Service Component Message (for SessionBean)

If the service component to be used is SessionBean, you need not create an XML format definition file for the service component message. The XML format definition file is automatically created when you create a service adapter.

4.3.4 Creating a Service Component Message (for MDB (WS-R or Database Queue))

To create an XML format definition file for a service component message, you can use WST (Web Standard Tool, which is provided by Eclipse).

The XML format definition file to be created must satisfy the conditions described in *2.6.5 Scoping of XML schema*. For details about the conditions, see *2.6.5 Scoping of XML schema*.

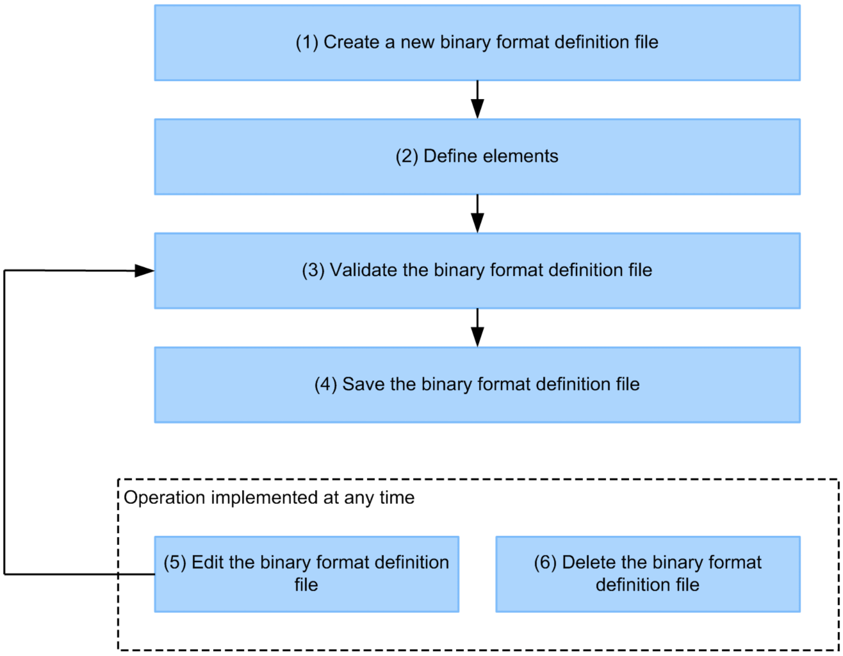
4.4 Creating Message Formats (Binary Format Definition File)

To use binary (non-XML) format data in a message to be used for service component execution, create a binary format definition file.

For details about the data types and character code types used in the binary format definition file, see *4.4.1 Data types and character code types in the binary format definition file*.

The following figure shows the workflow for creating a binary format definition file.

Figure 4–2: Workflow for creating a binary format definition file


(1) Create a new binary format definition file
(2) Define elements
(3) Validate the binary format definition file
(4) Save the binary format definition file
Operation implemented at any time
(5) Edit the binary format definition file
(6) Delete the binary format definition file

(1) Creating a new binary format definition file

Use a wizard to create a new binary format definition file.

For details about how to create a new binary format definition file, see *4.4.2 Creating a New Binary Format Definition File*.

(2) Defining elements

Define elements in the newly created binary format definition file. The method of defining elements differs when the binary data to be used is in the non-CSV format and in the CSV format.

For the details about how to define elements, see *4.4.3 Defining Elements (for Non-CSV Format)* or *4.4.4 Defining Elements (for CSV Format)*.

(3) Validating the binary format definition file

Validate the conformity of the created binary format definition file. You can perform validation at any time, such as when the elements of the binary format definition file are being defined, or after they have been defined.

For details about how to validate a binary format definition file, see *4.4.6 Validating a Binary Format Definition File*.

(4) Saving the binary format definition file

Save the binary format definition file in which elements have been defined. Unlike adapters and business processes, binary format definition files are not saved in a repository. To save the defined binary format definition file, use the Package Explorer of Eclipse.

When the binary format definition file is being saved, it is automatically validated (in this case, the validation result is not displayed in the console view).

(5) Editing the binary format definition file

You can modify the contents of a binary format definition file that has already been created. For details about how to edit a binary format definition file, see *4.4.5 Editing a Binary Format Definition File*.

(6) Deleting a binary format definition file

As necessary, you can delete a binary format definition file that is no longer needed. To delete a binary format definition file, use the Package Explorer of Eclipse.

4.4.1 Data types and character code types in the binary format definition file

In the binary format definition editor, you can use the following data types to define the formats:

- Numeric value
- Character string
- Byte string
- Bit string
- Date and time

For details on types of character codes to be handled in each data, see "1.3.2 Simple content element dialog" in "Service Platform Reference Guide".

The following points describe the details for each data type:

(1) Numeric value

Numeric value type is divided in character string type numeric value and byte string type numeric value depending on the numeric value attribute. Numeric value attributes depending on the respective numeric value types are as follows:

Character string type numeric value

- Integer
- Real number
- Numeric value with fixed decimal part

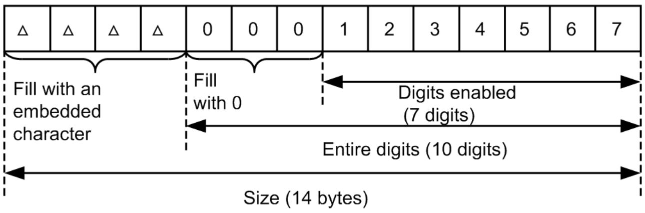
Description about procedure to transform from binary type to XML type is as follows.

- Read the numeric value string of specified size and starting embedded character (in case of right justification), end embedded character (in case of left justification), valid digits and code except heading 0 and transform those to XML schema decimal type.

Description about transformation specifications from XML type to binary type is as follows.

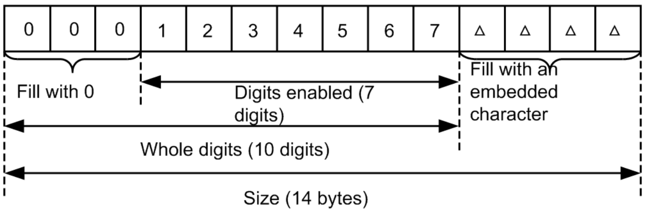
- Always delete 0 and single byte space at the beginning of numeric value.
- Handle embedded character and non-required character as different characters.
- Non-required character is 0 at the beginning of numeric value.
- Specification of all digits count becomes valid only in case of settings of not deleting the non-required character.
- When size is fixed length and all digits count is less than size, fill the trailing or leading part with embedded character, by specifying right justification or left justification.

Figure 4–3: Relation between all digits, size and embedded character (example of right justification)


△ △ △ △ 0 0 0 1 2 3 4 5 6 7
Fill with an embedded character
Fill with 0
Digits enabled (7 digits)
Entire digits (10 digits)
Size (14 bytes)

Size: Fixed length (14 bytes)
Entire digit count: 10 digits
Left/right alignment: Right
Unnecessary characters: Do not delete
Embedded character: Single byte space (△ shows Single byte space)

Figure 4–4: Relation between all digits, size and embedded character (example of left justification)



Size: Fixed length (14 bytes)
Whole digit count: 10 digits
Left/right alignment: Left
Unnecessary characters: Do not delete
Embedded character: Single byte space (△ shows Single byte space))

Byte column type numeric value

- Zone format numeric value
- Pack format numeric value
- Coded binary integer
- Unsigned binary integer

You can omit the coding of character string type numeric value. Coding expression position is before the numeric character string. If not even one number is expressed in input data, 0 is considered. Also, you can use 0 or space in embedded character.

All numeric value attributes are treated as decimal, in the data transformation. Maximum number of digits (total of digits in integer part and fraction part) is 34.

When you perform the XML transformation for character string numeric value type, transformation to `decimal` type of the XML schema is performed. When you perform XML transformation for the bytes string type numeric value, transformation to `decimal` type of XML schema is performed in the case of zone format numeric value and pack format numeric value and transformation to integer type is performed in the case of singed binary integer and unsigned binary integer.

Respective numeric value attributes are explained as follows:

(a) Integer

This numeric value attribute is configured by "Numeric value" and "Symbol". This numeric value attribute is internally processed in `BigInteger`r class.

Integer has maximum 34 digits and does not include the fraction part.

Examples of integer are as follows:

(Example 1)

```
1234567890
```

(Example 2)

`-1234567890`

(b) Real number

This is general format numeric value attribute configured by "Number", "Symbol" and "Decimal point". This numeric value attribute is internally processed in `BigDecimal` class.

Real number has maximum 34 digits and includes fraction part. However, when input data does not have a number and has only fraction part, error occurs.

Examples of real number are as follows:

(Example 1)

1234567890

(Example 2)

-123.456

(c) Fixed fraction numeric value

Fixed fraction numeric value is a numeric value attribute, which is configured from "Symbols" and "Numbers". For this attribute, if you specify digits in fraction part, decimal point is implicitly set. This numeric value attribute is internally processed in `BigDecimal` class.

Fixed fraction part numeric value has maximum 34 digits and fraction part is of maximum 33 digits.

Examples of fixed fraction part numeric value when you specify 4 digits for fraction part are as follows:

(Example 1) For "`1234567890`"

`123456.789`

(`1234567890` --> `123456.7890` --> `123456.789`)

(Example 2) For "`+123`"

`0.0123`

(`+123` --> `+0.0123` --> `0.0123`)

(Example 3) For "`-1230000`"

`-123`

(`-1230000` --> `-123.0000` --> `-123`)

(d) Zone format numeric value

Zone format numeric value is numeric value attribute of zone format. When you specify digits of fraction part, decimal point is implicitly set. This numeric value attribute is internally processed in `BigDecimal` class.

Zone format numeric value has maximum 34 digits and symbol is expressed in zone part of last byte in the item.

When character code is JIS8, plus symbol is expressed by 0x3 and minus symbols is expressed by 0x7.

When character code is EBCDIK, plus symbols is expressed by 0xC and 0xF, minus symbols is expressed by 0xD.

You can change type plus or minus of the sign.

Zone part other than final byte is expressed with plus sign of each code and numeric value is expressed by last 4 bits of each byte (0x0~0x9).

Examples of zone format numeric value when character code is JIS8, are as follows:.

(Example 1) For "`0x31323334`" (no specification of fraction part)

`1234`

(Example 2) For "`0x31323334`" (when you specify fraction part to 2 digits)

`12.34`

(Example 3) For "`0x31323374`" (no specification of fraction part)

`-1234`

(e) Pack format numeric value

This is pack format numeric value attribute. If you specify digits in fraction part, the decimal point is implicitly set. This numeric value attribute is internally processed in `BigDecimal` class.

Pack format numeric value has maximum 34 digits and when digits count is odd number, the first 4 bits of the topmost byte have to be set to 0.

Symbols are expressed with last 4 bits of the final byte of item data. Plus symbol is expressed with 0xC or 0xF, minus symbol is expressed with 0xD and numeric value is expressed in 4 bits (0x0~0x9). You can change plus or minus of the symbol. For data transformation, 0xC is output as plus symbol.

Examples of pack format numeric value when character code is JIS8 are as follows:

(Example 1) For "`0x01234F`**"** (no specification of fraction part)

```
1234
```

(Example 2) For "`0x01234F`**"** (you specified the fraction part to 2 digits)

```
12.34
```

(Example 3) For "`0x01234D`**"** (no specification of fraction part)

```
-1234
```

(f) Signed binary integer

This is numeric value attribute configured with signed binary integer having bytes in the range of 1~8 bytes. This numeric value attribute is internally processed in `BigInteger` class.

Endian depends on the format information.

Examples of singed binary integer in case of Big Endian are as follows:

(Example 1) For "`256`"

```
0x0100
```

(Example 2) For "`-257`"

```
0xFEFF
```

(g) Unsigned binary integer

This is numeric value attribute configured with unsigned binary integer having bytes in the range of 1~8 bytes.

Endian depends on the format information.

Examples of unsinged binary integer in case of Big Endian are as follows:

(Example 1) For "`-256`**"**

```
0x0100
```

(Example 2) For "`65279`**"**

```
0xFEFF
```

(2) Character string

This section describes specifications for transforming from binary type to XML type.

- Transforms the character string of specified size to character string as XML schema string type.
- When space is specified in the embedded character and settings are done not to delete the unnecessary characters, input character string with space is extracted.
- When space is specified in the embedded characters and settings are done to delete the unnecessary characters, deletes the opposite side spaces on left and right justification.
- When blank character (0x00) is specified is embedded character, transforms it according to the following specifications.
  - In case of settings to consider characters till blank character (0x00) as considers the characters from the first character of a string till character just before the blank character as user data and transforms to the XML character string.

    - In case of settings not to consider characters till blank character (0x00) as data and to delete the unnecessary character, deletes the blank characters on opposite side of left and right justification.
    - In case of settings not to consider characters till blank character (0x00) as data and not to delete the unnecessary character, character string with blank character is extracted.
- In case of settings to replace the characters, which cannot be used in XML (non-available characters), with a substitute character, handles the non-available characters on XML as normal characters.

Specifications to transform from XML type to binary type are explained as follows:

- Transform the XML data with the specified character code.
- In case of fixed length, embed the embedded characters when the size after transformation is less than the specified size.
- When XML data comprises of all single byte spaces, data is considered to have only 1 single byte space.
- In case blank character (0x00) is specified in embedded character and settings are to consider characters till blank character (0x00) as data, embeds a blank character at the end of character string.
- In case of settings to delete the unnecessary character, delete the single byte space at the end of input character string.
- In case of settings not to delete the unnecessary character, does not delete single byte space or blank character from the input string.

(3) Bytes string

Byte string is the byte string saved between platforms in bit pattern. When transforming byte string to XML, it is transformed to the `hexBinary` type or base64Binary type of XML schema.

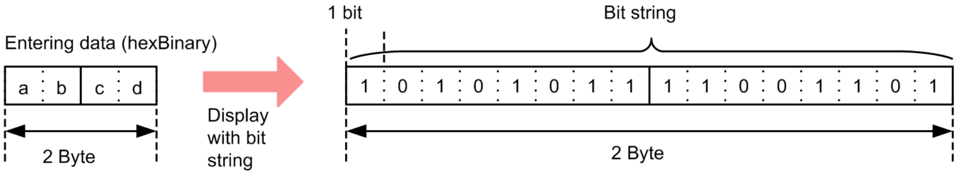
In `hexBinary` type, the 8 bit (1 byte) binary data is transformed to the hexadecimal binary characters. For example, when expressing the source data having 300 bytes, in `hexBinary` type, 300 bytes x 2 characters (600 characters) are required.

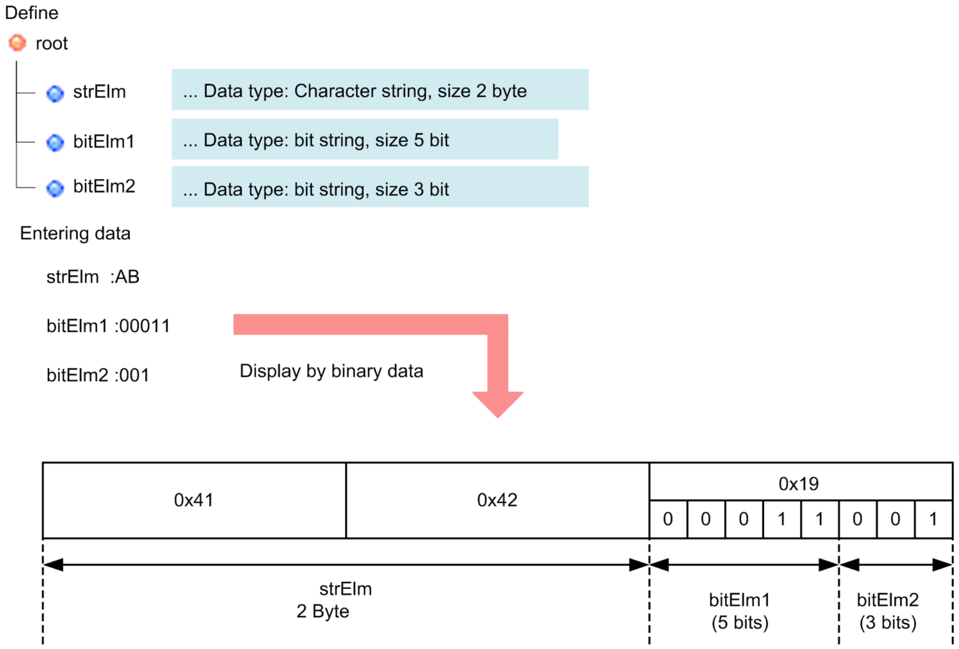
In base64Binary type, 6 bit (6/8 byte) binary data is transformed to 1 digit character encoded as Base64. For example, to express the source data of 300 bytes, in base64Binary type, 300 x 8/6 characters (400 characters) are required.

(4) Bit string

Bit string is a string in which bits expressed in binary are aligned. Bit string type is the data type that allows user to handle the binary data in 1 digit unit. Bit string type is processed in Big Endian.

Figure 4–5: Images of bit string and bit string type





Transformation from binary type to XML type is performed as follows:

- Reads elements from the beginning of input binary, equal to the specified bit size and transforms to hexBinary format of XML schema, in 8 bit unit.
- When data of size less than 1 byte remains after completing the reading of all elements, considers the remaining data as digits alignment bit and rounds down the same. If data having size of 1 byte or more is remaining, error occurs.

Transformation from XML type to binary type is performed as follows:

- Transforms the data in hexBinary format to binary format, for the size equal to the specified bit size from child bit.
- If the XML data is bigger than the transformation destination bits count, rounds down the upper bits from transformation destination bit and error does not occur.
- If the XML data is smaller than transformation destination bits count, supplements the bits till transformation destination bits count, with 0.
- If the XML data equal to the specified bits size is not the hexadecimal string, error occurs.

(5) Date and time

Date and time is the type configured by combination of parts like Year, Month, Date, Hours, Minutes and Seconds. You can specify numeric value for individual part and part that expresses seconds can include a fraction part.

When transforming the date and time type from XML type to fixed length element of binary type, if the size of input XML after transformation is smaller than the size of fixed length element, you can specify whether to express by aligning the element data to either left or right side (left or right justification).

Limitations for specifying the date and time type are as follows:

- You cannot use the 2 byte character code.
- "." (period) is the only character identified as decimal point.

Following table describes parts of date and time type.

Table 4–1: Parts of the date and time type

Parts	Meaning	Number of specification digits
CCYY	Christian year (0~9999)	1~4 digits
YY	Year (last 2 digits of Christian year)	Maximum 2 digits
MM	Month (1~12)	Maximum 2 digits
DD	Date (1~31)	Maximum 2 digits
hh	Hour (0~23)	Maximum 2 digits
mm	Minutes (0~59)	Maximum 2 digits
ss	Seconds (0~59, fraction part can be included)	Maximum 2 digits in integer part and maximum 3 digits in fraction part

You can specify 3 data types (date type, time type and date and time type), depending on the combination of parts.

Following table shows the combination of data type and part.

Table 4–2: Combination of data type and part

Data type	Combination	Meaning
Date type	CCYYMMDD	Christian year/month/date
	YYMMDD#	Last 2 digits in Christian year/month/date
Time type	Hhmmss	Hour/minutes/second
Date and time type	CCYYMMDDhhmmss	Christian year/month/date/hour/minutes/second
	YYMMDDhhmmss	Last 2 digits of Christian year/month/date/hour/minutes/second

Note#

In case of "YYMMDD" format of date type, the range you can specify is Year 1951~Year 2050, as per the Christian year calculation.

When transforming date type, time type and date and time type to XML, the respective types are transformed as follows:

- Date type is transformed to the `date` type of XML schema.
- Time type is transformed to `time` type of XML schema.
- Date and time type is transformed to `dateTime` type of XML schema.

Examples of date and time type are as follows:

(Example 1) For January 24, 2001

```
20010124
```

(Example 2) For December 31, 1999, 1:30:59

```
991231013059
```

4.4.2 Creating a New Binary Format Definition File

This subsection explains how to create a new binary format definition file.

Before you create a binary format definition file, you must first create a project.

1. On the Eclipse File menu, choose New, and then Other.
   A dialog box for selecting a wizard opens.
2. In HCSCTE format definition, select Binary format definition file and click Next.
   The New binary format definition file wizard opens.
3. In Enter or choose parent folder, either enter or choose the folder in which to save the binary format definition file.
4. In Format name, specify the format name of the binary format definition file to be saved.
   When you specify a format name, the `.fdx` extension is automatically added to the name set in **File name**.
5. If the binary data to be used is in the CSV format, select the CSV format checkbox.
   Note that the data using the following separation characters can be handled in the CSV format data:
   - Intervening separation character: Comma (,)
   - End character: Linefeed

   If the element value contains characters same as the intervening separation character, enclose the element in double quotations ("). Note that if you use double quotation as a value in the element, use two continuous double quotations for escape.
   Also, note that if a start character is being used, it cannot be handled as a CSV format data.
6. Click Finish.
   A new binary format definition file has been created and the Binary Format Definition screen opens. In this window, you can proceed to define elements.

! Important note

Unlike adapters and business processes, binary format definition files are not saved in a repository. For file management operations, such as saving and deleting the binary format definition file that has been defined, use the Package Explorer of Eclipse.

4.4.3 Defining Elements (for Non-CSV Format)

You define the contents of a binary format definition file in the Binary Format Definition screen. The Binary Format Definition screen opens when you double-click a binary format definition file displayed in the Package Explorer of Eclipse.

For details about the Binary Format Definition screen, see the manual *Cosminexus Service Platform Reference*.

Reference note

In the Binary Format Definition screen, you can open multiple binary format definition files and edit them. However, you cannot copy and paste definition information between multiple binary format definition files.

In the Binary Format Definition screen, perform the following tasks to define elements:

- Specifying the format information
- Specifying globally defined simple content elements
- Specifying complex content elements
- Specifying components
- Specifying a component selection condition
- Specifying the root element
- Specifying a separator

The method for defining elements is described below.

(1) Specifying the format information

For the format information of the binary format definition file, specify the character code used by the binary data and the endian included in the binary data.

To specify the format information:

1. In the Binary Format Definition Editor, select and right-click a displayed format name, and choose Setting.
   The Format dialog box opens.
2. Enter the necessary information in the Format dialog box.
   For details about the display and input contents of the Format dialog box, see the manual *Cosminexus Service Platform Reference*.
3. Click OK.
   The information entered in the Format dialog box is set.

(2) Specifying globally defined simple content elements

A simple content element is a simple-type element (`xsd:simpleType` element) when the format of binary data it uses is expressed as an XML schema, and it can also be expressed as a globally defined element. When a binary format definition file is displayed in the Data Transformation Definition screen, a simple content element is displayed as a node having simple content.

When specifying a simple content element, specify an element data type and size.

To specify a simple content element:

1. In the Binary Format Definition Editor, select and right-click a displayed format name, and choose Add Simple Content Element.
   The Simple Content Element dialog box opens.
2. Enter the necessary information in the Simple Content Element dialog box.
   For details about the display and input contents of the Simple Content Element dialog box, see the manual *Cosminexus Service Platform Reference*.
3. Click OK.
   A simple content element is set up using the information entered in the Simple Content Element dialog box. The specified simple content element is also displayed in the Binary Format Definition Editor.

(3) Specifying complex content elements

A complex content element is a complex-type element (`xsd:complexType` element) when the format of binary data it uses is expressed as an XML schema, and it can also be expressed as a globally defined element. When a binary format definition file is displayed in the Data Transformation Definition screen, a complex content element is displayed as a node having complex content.

When specifying a complex content element, specify a component type for the constituent elements of the complex content element. For the component type, specify whether the constituent element is a sequential element (`xsd:sequence` element) or selection element (`xsd:choice` element).

To specify a complex content element:

1. In the Binary Format Definition Editor, select and right-click a displayed format name, and choose Add Complex Content Element.
   The Complex Content Element dialog box opens.
2. Enter the necessary information in the Complex Content Element dialog box.
   For details about the display and input contents of the Complex Content Element dialog box, see the manual *Cosminexus Service Platform Reference*.

   When choosing Selection as the component type

   > When you choose **Selection** as the component type, you must set up the selection conditions in components with either of the following methods:
   >
   > \- Specify **Selection condition node**, and select the node containing the judgment value of the selection conditions

- Specify **Starting separator of components**

For details about how to specify a selection condition, see *4.4.3(5) Specifying a component selection condition*.

Before specifying a selection condition for a component, you must specify the constituent elements of the complex content element. Carry out step 3 below, and then specify a component by following the procedure described in *4.4.3(4) Specifying components*. After that, you can specify a selection condition.

When a simple content element that is a part of a complex content element is a separator data

If a simple content element that is a part of a complex content element is the data (separator data) in which each element is separated with separation characters (separator), specify the separators in the complex content element.

For details about how to specify a separator, see *4.4.3(7) Specifying a separator*.

3. Click OK.

A complex content element is set up using the information entered in the Complex Content Element dialog box. The specified complex content element is also displayed in the Binary Format Definition Editor.

Reference note

The first complex content element that is specified when elements are being defined for a binary format definition file becomes the root element. To change the root element, see *4.4.3(6) Specifying the root element*.

(4) Specifying components

A *component* is an element that constitutes a complex content element. The following two methods are available for specifying a component.

- Specifying a globally defined element as a component
- Specifying a locally defined simple content element as a component

These component specification methods are described below.

(a) Specifying a globally defined element as a component

You can specify, as a component of a complex content element, a globally defined element (simple content element or complex content element) that has already been set up:

To specify a component:

1. In the Binary Format Definition Editor, select and right-click a displayed complex content element, and choose Add Component.

   The Element Selection dialog box opens. The Element Selection dialog box displays all globally defined elements that can be selected as components of a complex content element.

   For details about the display and input contents of the Element Selection dialog box, see the manual *Cosminexus Service Platform Reference*.
2. In the Select Element dialog box, select the element to be specified as a component from the element list.
3. Click OK.

   Component is added in the complex content element and closes the Select Element dialog box.
4. Right-click the added component, and choose Setting.

   The Component dialog box opens.
5. Enter the necessary information in the Component dialog box.

   For details about the display and input contents of the Component dialog box, see the manual *Cosminexus Service Platform Reference*.
6. Click OK.

   The globally defined element is set up as a component of the complex content element, using the information entered in the Component dialog box. The specified component is displayed as a lower-order element of a complex content element in the Binary Format Definition Editor.

(b) Specifying a locally defined simple content element as a component

You can specify a locally defined simple content element as a component of a complex content element:

To specify a component:

1. In the Binary Format Definition Editor, select and right-click a displayed complex content element, and choose Add Local Simple Content Element.

   The Simple Content Element dialog box opens.

2. Click the Simple Content Element tab and enter the necessary information.

   For details about the display and input contents of the **Simple Content Element** tab in the Simple Content Element dialog box, see the manual *Cosminexus Service Platform Reference*.

3. Click the Component tab and enter the necessary information.

   In the Component tab, specify an occurrence count and size for the component.

   For details about the display and input contents of the **Simple Content Element** tab in the Simple Content Element dialog box, see the manual *Cosminexus Service Platform Reference*.

   You can also make the occurrence count and size dependent on another element. In this case, you can select the base node from the Node Selection dialog box, which opens when you click Select Node on the Component tab.

   For details about the display and input contents of the Node Selection dialog box, see the manual *Cosminexus Service Platform Reference*.

   Clicking **OK** in the Node Selection dialog box commits the specified value to the **Component** tab of the Simple Content Element dialog box.

4. Click OK.

   A locally defined simple content element is set up as a component of the complex content element, using the information entered in the Simple Content Element dialog box. The specified component is displayed as a lower-order element of a complex content element in the Binary Format Definition Editor.

(5) Specifying a component selection condition

If you choose **Selection** as the component type when specifying a complex content element, you need to specify a selection condition for the component. When a selection condition is set up, the determination of which multiple elements (components) contained in the complex content element are to be used can be made based on the value of another element.

The following two methods are available for specifying a selection condition:

- Select **Selection condition node**, and then set up the selection condition

  From the condition value set up in the node that is selected in the selection condition node, determine the component to be used.

- Select **Starting separator of components**, and then set up the selection condition

  From the starting separator set up in the components, determine the component to be used.

These methods for specifying a selection condition are described below.

(a) Selecting a selection condition node and setting up the selection condition

To select a selection condition node, and then set up the selection condition:

1. In the Binary Format Definition Editor, select and right-click the complex content element containing the component for which a selection condition is to be set up, and click Setting.

   The Complex Content Element dialog box opens.

2. In **Component Type**, choose **Selection**.

3. In Selection type, click Selection condition node, and then Select Node.

   The Node Selection dialog box opens. In this dialog box, select a selection condition node.

   For details about the display and input contents of the Node Selection dialog box, see the manual *Cosminexus Service Platform Reference*.

   Clicking **OK** in the Node Selection dialog box commits the specified value to **Selection condition node** in the Complex Content Element dialog box.

4. In Components, select the element for which a condition is to be specified, and click Set Condition Value.

   The Set Condition Value dialog box opens. In this dialog box, enter a condition value.

   For details about the display and input contents of the Set Condition Value dialog box, see the manual *Cosminexus Service Platform Reference*.

   Clicking **OK** in the Set Condition Value dialog box commits the specified value to **Selection condition value** of the Complex Content Element dialog box. To set up selection conditions for other components, repeat Step 3.

   Any component for which no selection condition is specified becomes a component that occurs when none of the selection conditions specified for other components is satisfied. Only a single such component can be set up.
5. Click OK.

   The selection condition is set up using the information entered in the Complex Content Element dialog box.

(b) Selecting the starting separator of components and setting up the selection condition

To select the starting separator of components and set up the selection condition:

1. In the Binary Format Definition Editor, select and right-click the complex content element containing the component for which a selection condition is to be set up, and click Setting.

   The Complex Content Element dialog box opens.
2. In **Component Type**, choose **Selection**.
3. In **Selection Type**, select **Starting separator of components**.

   The component and the starting separator set up in the component will be displayed in **Components List**.
4. Click OK.

   The selection condition is set up using the information entered in the Complex Content Element dialog box. Note that when the starting separator is not set up in the component (complex content element), an error will occur during the validation of the binary format definition file.

(6) Specifying the root element

Specify the complex content element that is to be treated as the root element when the binary data used is expressed as an XML schema.

The first complex content element that is specified when elements are being defined for a binary format definition file becomes the root element. To make another complex content element the root element, in the Binary Format Definition Editor, select and right-click a displayed complex content element, and click **Set as Root Element**.

When **Set as Root Element** is chosen, the complex content element selected in the Binary Format Definition Editor becomes the root element. In the Binary Format Definition Editor, a root element icon is displayed next to the complex content element that is treated as the root element.

If a complex content element is already specified as the root element, specifying another complex content element as the root element changes the root element. The display in outline view also switches to start with the new root element.

(7) Specifying a separator

If a simple content element, specified as part of a complex content element, is the data (separator data) in which each element is separated using a separation character (separator), specify the separators in the complex content element.

The details about how to specify a separator is explained below:

1. Select the complex content element, right click, and then click Setting.

   The Complex Content Element dialog box opens.
2. Choose the Separator tab.

   The contents of the **Separator** tab in the complex content element dialog box are displayed.
3. Select the separator format to be used from the drop down list.

   Note that when you select CSV as a separator format, the starting separator and the intermediate separator will be set up automatically.
4. Select the Separator to be used check box and click Select.

The Select Separator dialog box opens. For details about the display and input contents of the Select Separator dialog box, see the manual *Cosminexus Service Platform Reference*.

5. Choose a separator that you want to use from the list and click OK.

   The information about the separator selected in the **Separator** tab of the complex content element dialog box is displayed.

   Note that if the separator you want to use is not present in the Select Separator dialog box; click **Add** in the Select Separator dialog box. The Add/Change Separator dialog box opens. You can add or change separators in this dialog box.

   For details about the display and input contents of the Add/Change Separator dialog box, see the manual *Cosminexus Service Platform Reference*.

6. Repeat step 3. and 4. as needed.

7. Click OK.

   The separator is specified in the information entered in the **Separator** tab of the Complex Content Element dialog box.

4.4.4 Defining Elements (for CSV Format)

In case of the CSV format, define the contents of the binary format definition file in the Binary Format Definition screen. The Binary Format Definition screen opens when you double-click a binary format definition file displayed in the Package Explorer of Eclipse.

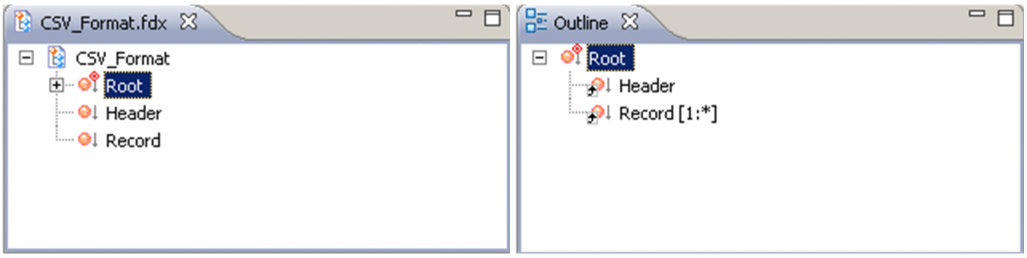
For details on the Binary Format Definition screen, see the manual *Cosminexus Service Platform Reference*.

Reference note

In the Binary Format Definition screen, you can open multiple binary format definition files and edit them. However, you cannot copy and paste definition information between multiple binary format definition files.

In the case of CSV format, if you create a new binary format definition file and view the Binary Format Definition screen, the root, header, and record elements are displayed as deployed.

Figure 4–6: Initial display of the Binary Format Definition Screen when using CSV format data



The root, header, and record elements are all complex content elements. The header and record elements are specified as components of the root element. Specify the simple content element of the local definition as a component in the header and record elements.

You perform the following operations in the Binary Format Definition screen:

- Specifying the format information
- Specifying occurrence count of header and record elements
- Specifying components of header and record elements
- Specifying the root element

Tip

When using a CSV format data, you cannot change the configuration of the root element, header element, and record element displayed initially in the Binary Format Definition screen.

The method for defining elements is described below.

(1) Specifying format information

Specify the character code and linefeed used in binary data as the format information of the binary format definition file.

The specification method is same as when non-CSV format binary data is used. For details about how to specify the format information, see *4.4.3(1) Specifying the format information*.

(2) Specifying occurrence count of header and record elements

How to specify the occurrence count of header and record elements is explained below:

1. Select the header element or record element displayed as a root element component, right click, and then choose Setting.
   The Component dialog box opens.
2. Enter the occurrence count in the Component dialog box.
   Specify `0` as the occurrence count of header element if there is no header in the CSV data and `1` if a header exists.
   For details about the display and input contents of the Component dialog box, see the manual *Cosminexus Service Platform Reference*.
3. Click OK.
   The occurrence count entered in the Component dialog box is set.

(3) Specifying components of header and record elements

The details about how to specify the components of the header and record elements that are displayed as the root element components in the binary format definition editor is explained below:

1. Select the header element or record element displayed as a root element component, right click, and choose Add local simple content element.
   The Simple Content Element dialog box opens.
2. Enter the necessary information in the Simple Content Element dialog box.
   For details about the display and input contents of the Simple Content Element dialog box, see the manual *Cosminexus Service Platform Reference*.
3. Click OK.
   The information entered in the Simple Content Element dialog box is specified.

(4) Specifying the root element

If you use CSV data, you cannot specify elements other than the root elements initially displayed in the Binary Format Definition screen, in a root element. However, you can change only the root element name.

How to change a root element name is explained below:

1. Select a root element, right click, and choose Setting.
   The Complex Content Element dialog box opens.
2. Enter Name.
3. Click OK.
   The entered name is specified.

4.4.5 Editing a Binary Format Definition File

You can modify the contents of a binary format definition file that has already been created and saved.

To modify the contents of a binary format definition file, in the Package Explorer of Eclipse, double-click the binary format definition file to be edited. In the Binary Format Definition screen that opens, you can modify the contents.

For contents that can be specified in the Binary Format Definition screen, see *4.4.3 Defining Elements (for Non-CSV Format)* or *4.4.4 Defining Elements (for CSV Format)*.

You can also copy a binary format definition file that has already been created and modify its contents to create another binary format definition file. However, you cannot copy and use for other purposes part of the definition content of a binary format definition file that has already been created.

4.4.6 Validating a Binary Format Definition File

If the required information has not been specified or settings contents or relation are not proper, you cannot successfully perform data transformation definition creation or data transformation, using the binary format definition file.

Therefore, validate whether the created binary format definition file has correct setting contents. You can validate the file at any time. Also, in case of input through dialog for setting each element, the entered value is automatically validated.

(1) Validation contents

You can validate the settings contents of each element in binary format definition file.

Following table describes validation contents, the actions in case of error occurrence and the details on which validation contents are to be implemented in case of using which validation method.

Table 4–3: Validation contents of binary format definition file and actions in case of error or warning occurrence

Validation target	Validation contents	Action in case of error or warning occurrence	Validation method	
			Auto	Optional
Overall	Whether name is `NCName` type?	Specify format name and element name in `NCName` type.	Y	--
Format	Whether the complex contents elements of sequential configuration are set?	Set the complex contents element of sequential configuration.	--	Y
	Whether the root element is set?	Set the complex contents element within binary format definition as the root element.	--	Y
	Whether code conversion library is available? (whether you can load the class?)	Set the jar file for code conversion properly and the restart Eclipse.	--	Y
	Whether complex contents element that defines a separator is used, when selecting KEIS, IBM_CODE, JEF as character code?	Change character code of format to other than KEIS, IBM_CODE. Or JEF, or cancel the specifications of separator of complex contents.	Y	--
	Whether simple contents element in which other than character type has been set in data type, when KEIS, IBM_CODE and JEF is selected as character code?	Set character code of format to other than KEIS, IBM_CODE and JEF or specify character string in the data type of simple contents element.	Y	--
	Whether bit value (hexadecimal) of pack format numeric value and zone format numeric value has been set with hexadecimal or the value has not been set?	Set the corresponding signed bit or character with hexadecimal.	Y	--
	Whether value of singed character (hexadecimal) of zone format numeric value has been set with hexadecimal or value has not been set?	Set the corresponding signed bit or character with hexadecimal.	Y	--
	Whether escape character is Latin character or 1 single byte space?	Set escape character in the range (0x21~0x7E) of Latin character.	Y	--
	Whether same character has been specified in escape character and separator value?#	Specify a character that is not duplicated in the escape character and separator value.	Y	--

Validation target	Validation contents	Action in case of error or warning occurrence	Validation method	
			Auto	Optional
Simple contents element	Whether the elements having same name exist in the same hierarchy?	Specify unique name in the same hierarchy.	Y	--
	Whether simple contents element of global definition have been set in the components of complex contents element?	Set the corresponding simple contents element in the component of the complex contents element.	Y	--
	Whether the value of specified size fulfills the following expression, when the size is fixed? `0<Size=2,147,483,647`	Specify the size in the range of 1~2,147,483,647.	Y	--
	Whether the value of specified size is odd number, when all the following conditions are fulfilled? • Data type is character string, integer, real number, fixed fraction part numeric value or date and time • Size is fixed • Character code is UTF-16 (BE/LE)	Specify other than UTF-16 (BE/LE) in the character code or specify odd number in the size of corresponding simple contents element.	Y	Y
	Whether the byte size is 2 or more when all the following conditions are fulfilled? • Data type is character string • Size is fixed • Character code is KEIS, IBM_CODE and JEF	Set the character code of format or individual character code to other than KEIS, IBM_CODE and JEF or specify value of 2 or greater in the size of corresponding simple contents	Y	Y
	Whether the separator has not been set in ancestor when all the following conditions are fulfilled? • Size is fixed • Individual character code has been set	Cancel the specifications of separator for the simple contents or cancel the specifications of individual character code.	Y	Y
	Whether the value of 0020~D7DE, E000~FFDC (case insensitive) has been set when all the following conditions are fulfilled? • Data type is character string • "Replace to substitute character" has been specified	Specify hexadecimal of 0020~D7DE, E000~FFDC.	Y	--
	Whether embedded character is other than "0" when all the following conditions are fulfilled • Data type is integer, real number or fixed fraction part numeric value • "Left" has been specified in left and right justification	Set left or right justification to other than "Left" or specify other than "0" in embedded character.	Y	--
	Whether the specified integer fulfills following expression when the size is fixed? `All digits count>Signed digits count`	In all digits count, specify value greater than signed digits count.	Y	--
	Whether specified integer fulfills the following expression when the size is fixed and all digits count is not specified? `Size>Signed size`	In size, specify a value greater than sign byte size.	Y	Y

Validation target	Validation contents	Action in case of error or warning occurrence	Validation method	
			Auto	Optional
Simple contents element	Whether the specified integer fulfills following expression, when the size is fixed and all digits count is specified? Size=Signed size + Integer part size	In size, specify value greater than bytes size of all digits count.	Y	Y
	Whether character code is other than KEIS, IBM_CODE and JEF, when all the following conditions are fulfilled? • Data type is integer, real number, fixed fraction part numeric value, zone format numeric value, pack format numeric value, signed binary integer, unsigned binary integer, bytes string, bit string or date and time	Set character code of format to other than KEIS, IBM_CODE and JEF or specify character string of data type.	Y	--
	Whether the specified integer, real number or fixed fraction part numeric value fulfills the following expression? 1=All digits count=34	Set the corresponding digits count in the numeric value within configurable range.	Y	--
	Whether the specified real number fulfills the following expression, when the size if fixed? Overall digits count=Signed digits count _ digits count of decimal point‾+ digits count of fraction part And Overall digits count>Signed digits count	In overall digits count, specify value greater than the number of digits acquired by combining signed digits count and decimal point as well as fraction part digits count.	Y	--
	Whether the specified real number or fixed fraction part numeric value fulfills the following conditions when the size is variable? Overall digits count=Fraction part digits count	In overall digits count, specify value greater than the fraction part digits count.	Y	--
	Whether the specified real number fulfills the following expression, when size is fixed part and overall digits count and fraction part digits count are not specified? Size>Signed size	In size, specify value greater than the signed byte size.	Y	Y
	Whether the specified real number fulfills the following expression, when size is fixed, overall digits count is not specified and fraction part digits count is specified? Size=Signed size + decimal point size + fraction part size And Size>Signed size	In size, specify a value greater than byte size acquired by combining the signs and decimal point and fraction part.	Y	Y
	Whether the specified real number fulfills the following expression when size is fixed and overall digits count is specified? Size=Signed size+integer size=decimal point size+fraction part size	In size, specify value greater than the bytes size of overall digits count.	Y	Y
	Whether the specified fraction part numeric value fulfills the following expression, when the size is fixed?	In overall digits count, specify value greater than the digits count having combination of sign and fraction part.	Y	--

Validation target	Validation contents	Action in case of error or warning occurrence	Validation method	
			Auto	Optional
Simple contents element	Overall digits count=Signed digits count +fraction part digits count and Overall digits count>Signed digits count	In overall digits count, specify value greater than the digits count having combination of sign and fraction part.	Y	--
	Whether the specified fixed fraction part numeric value fulfills the following expression, when size is fixed and overall digits count is not specified? Size=Signed size+fraction part size and size>signed size	In size, specify a value greater than bytes size acquired by combining the sign and fraction part digits count.	Y	Y
	Whether the specified fixed fraction part numeric value fulfills the following expression, when the size is fixed and overall digits count is specified? Size=Signed size+integer part size+fraction part size	In size, specify a value greater than bytes size of overall digits count.	Y	Y
	Whether the specified fraction part digits count fulfill the following expression, when the data type is real number, fixed decimal point numeric value, zone format numeric value, or lack format numeric value? 0=fraction part digits count=33	Specify fraction part digits count in the range of 0~33.	Y	--
	Whether "Separator format" is "user specified", when separator is set in the ancestor on the following simple contents element? • Data type is zone format numeric value, pack format numeric value, signed binary integer, unsigned binary integer or bytes string	Specify data type other than zone format numeric value, pack format numeric value, signed binary integer, unsigned binary integer or bytes string. Or, change the format of separator for the simple contents, to user specified format.	Y	Y
	Whether the "fraction part digits count=maximum fraction part digits count acquired from the specified size" is set when the all the following conditions are fulfilled? • Data type is pack format numeric value • Size is fixed	In size, specify value greater than the bytes count of fraction part digits count.	Y	Y
	Whether the specified size fulfills the following expression, when the data type is signed binary integer or unsigned binary integer? 1≡size≡8	Set the corresponding digits count with the numeric value in the configurable range.	Y	--
	Whether the specified size fulfills the following expression, when the data type is bits string? 1≡size≡64	Specify size in the range of 1~64.	Y	--
	Whether the complex contents element for which separator is defined exist in the binary format definition, when data type is bits string?	Specify other than bits string in the data type or cancel the specification of separator.	Y	--
	Whether the specified fraction part digits count fulfill the following expression, when the data type is date and time type and second fraction part digits count is specified?	Set the seconds fraction part digits count in the range of 0~3.	Y	--

Validation target	Validation contents	Action in case of error or warning occurrence	Validation method	
			Auto	Optional
Simple contents element	`0≦Seconds fraction part digits count≦3`	Set the seconds fraction part digits count in the range of 0~3.	Y	--
	Whether the specified size fulfills the following expression when the data type is date and time type and size is fixed? `Size=format size+decimal point size+seconds fraction part digits count`	In size, specify value greater than the bytes count of date and time format.	Y	Y
	Whether "fraction part digits count=maximum fraction part digits count acquired from specified size" is set when all the following conditions are fulfilled? • Data type is zone format numeric value • Size is fixed • Settings of sign is custom • Type of sign is sign bits Or, whether "Fraction part digits count=maximum fraction part digits count acquired from the specified size" is set when all the following conditions are fulfilled? • Data type is zone format numeric value • Size is fixed • Settings of sign is auto	In size, specify value greater than bytes count of fraction part digits count.	Y	Y
	Whether "maximum fraction part digits count of specified size=Fraction part digits count_sign character" is set when all the following conditions are fulfilled? And whether the size is 2 bytes or more? • Data type is zone format numeric value • Size is fixed • Settings of sign is "Custom" • Type of sign is "Sign character"	In size, specify value greater than bytes count of fraction part digits count+bytes count of sign character and value having 2 bytes or more.	Y	Y
Complex contents element	Whether the elements having same name exist in the same hierarchy?	Specify unique name in the same hierarchy.	Y	--
	Whether complex contents element set in the root element has been set as the component of other complex contents element?	Delete the complex contents element set in the root element, from the component of the complex contents element.	--	Y
	Whether the complex contents element has been set in the components of other complex contents element (excluding the cases of root element)?	Set the corresponding complex contents element to the component of other complex contents element.	--	Y
	Whether the component has been set?	Set the component in corresponding complex contents element.	--	Y
	Whether following simple contents elements exist, when "CSV" is specified in the separator format? • Data type is zone format numeric value, pack format numeric value, signed binary	Change the separator format to user-specified format or specify the data type other than zone format numeric value, pack format numeric value, singed binary integer, unsigned binary integer and bytes string in the simple contents element	Y	Y

Validation target	Validation contents	Action in case of error or warning occurrence	Validation method	
			Auto	Optional
Complex contents element	integer, unsigned binary integer or bytes string	Change the separator format to user-specified format or specify the data type other than zone format numeric value, pack format numeric value, singed binary integer, unsigned binary integer and bytes string in the simple contents element	Y	Y
	Whether the specification of start separator is correct	Specify the correct separator in the start separator	--	Y
	Whether the specification of intermediate delimitation separator is correct	Specify correct separator in the intermediate delimitation separator	--	Y
	Whether the specification of end separator is correct	Specify the correct separator in end separator.	--	Y
	Whether the character code of format is KEIS, IBM_CODE and JEF, when the separator is specified	Cancel the specification of complex contents separator or specify character code of format to other than KEIS, IBM_CODE and, JEF	Y	--
	Whether the simple contents element in which individual character code is specified exists when the separator is specified	Cancel the specification of individual character code of simple contents or cancel the specification of the separator of complex contents	Y	Y
	Whether the separator name is duplicated	Specify separator that is not specified in other separators, within single complex contents	Y	--
	Whether separator value is duplicated	Specify separator value that does not duplicate with other separator values, in single complex contents	Y	--
	Whether the code conversion library is valid when the separator is specified	Set the correct jar file of code conversion and re-start Eclipse.	--	Y
	Whether code conversion table for code conversion and validation target character code exists, when the separator is specified	Specify path of code conversion table, in the code conversion table.	--	Y
	Whether error occurs during the character code conversion, when the separator is specified	See the exception information that is cause of error and take actions accordingly.	--	Y
	Whether character code conversion fails when the separator is specified	See the error code and take actions accordingly	--	Y
	Whether the character code is supported by code conversion library, when the separator is specified	Select character code that is supported by the code conversion library. Or, use the supported code conversion library.	--	Y
	Whether the simple contents element of bit string type exists in binary format definition, when the separator is specified	Cancel the specification of separator or specify other than bit string in the data type of simple contents element	Y	--
	Whether the complex contents element in which size node is specified exists in the binary format definition, when separator is specified	Cancel the specification of separator or delete the size node	Y	--
	In sequential configuration, when last child element is simple contents element of bit string type, does it correspond to the following conditions (excluding the case of root element)	Check the size of child element of bits string type. Calculate the size of element of bits string type as follows:	--	Y

Validation target	Validation contents	Action in case of error or warning occurrence	Validation method	
			Auto	Optional
Complex contents element	- Total the size of consecutive bits string type and size is not byte unit	When occurrence count is fixed, "size x maximum occurrence count". When occurrence count is not fixed, calculate as follows: - When actual occurrence count is unknown and size is not in byte unit, consider it as warning. - When size is of bytes unit, even if the occurrence count is unknown, value of "size x occurrence count" is always in the byte unit and hence calculate occurrence count as 1 and consider as "size x 1".	--	Y
	In selected configuration, when child element is the simple contents element of bits string type, does it correspond to following condition (excluding the case of root element) - Size is not in byte unit	Check the size of child element of bits string type. Calculate the size of element of bits string type as follows: When occurrence count is fixed, "size x maximum occurrence count". When occurrence count is not fixed, calculate as follows: - When actual occurrence count is unknown and size is not in byte unit, consider it as warning. - When size is of bytes unit, even if the occurrence count is unknown, value of "size x occurrence count" is always in the byte unit and hence calculate occurrence count as 1 and consider as "size x 1".	--	Y
	Whether the selected condition node has been set in case of selected configuration	Set the selected condition node in corresponding complex contents element	--	Y
	Whether the element set in the selection condition node is simple contents element, in case of selected configuration	Set node of the simple contents element of selected condition node	Y	Y
	Whether selection condition value is set in case of selected configuration (however, selection condition value might not be set in case of only 1 selected configuration)	Set the selection condition value in each component and set the count of components for which selection condition value is not set, to 1 or less	--	Y
	Whether value specified in the data type of the simple contents element set in the selection condition node matches with value specified in selection condition value of each component, in case of selected configuration	Validation contents and actions differ depending on the data type of simple contents element set in the selection condition node. For details, see "*Table4-4 Validation contents and actions related to the value specified in selection condition value*".	Y	Y
	Whether selection condition value of each component is duplicated in case of selected configuration	In component for which selection condition value is duplicated, specify a selection condition value that is not specified in other components	--	Y
	Whether occurrence count of selection condition node and all ancestor nodes is fixed as 1 time, in case of selected configuration	Set the occurrence count of selection condition node and its ancestor node to the fixed count that is 1 time	--	Y

Validation target	Validation contents	Action in case of error or warning occurrence	Validation method	
			Auto	Optional
Complex contents element	Whether the size of selection condition node is fixed, in case of selected configuration	Set the size of selection condition node as fixed	--	Y
	Whether the path of selection condition node is correct, in case of selected configuration	Path that shows selection condition node, might be invalid. For details on action, see "*Table4-5 Cause and actions when the path is invalid*"	Y	Y
	Whether the complex contents element of selected configuration are included in the path of selection condition node, in case of selected configuration (however, when the path is absolute path, exclude the ancestor node of the setting source node)	Selection condition node may not occur in the actual binary data. Check whether the corresponding data occur in the selection condition node, in actual binary data.	--	Y
	Whether the node specified in selection condition node includes the complex contents element specified in size node, in case of selected configuration (when the path is absolute path, exclude the node that is ancestor of the setting source node. When the path is standard path, exclude the standard node)	Sometimes, selection condition node might not occur in the actual binary data. Check whether the corresponding data occurs in the selection condition node, in actual binary data.	--	Y
	Whether code conversion library is used in case of selected component (whether the class can be loaded)	Set the correct jar file of code conversion and restart Eclipse.	Y	--
	Whether the correct path of the code conversion table is set, in case of selected configuration	Set the occurrence count of corresponding selected configuration and its ancestor as 1, which is a fixed value	Y	Y
	Whether error occurs during character code conversion, in case of selected configuration	See the exception information that is cause of error occurrence and take actions accordingly.	Y	Y
	Whether character code conversion fails, in case of selected configuration	See the error code and take actions accordingly.	Y	Y
	Whether character code is reported in the code conversion library, in case of selected configuration	Select the character code supported in the code conversion library. Or, use the supported code conversion library.	Y	Y
	Whether simple contents element count set in the component is 1 or less, when the start separator of component is set in case of selected configuration	Set the simple contents element to be set in component to l or less	--	Y
	Whether the start separator is set in each component (complex contents element) in case of selected configuration (however, when simple contents element has not been set in the component, you may not set start separator if the component is only 1)	When simple contents element is set in the component, set start separator in each component. If the simple contents element has not been set in the component, set the count of components for which start separator has not been set, to 1 or less.	--	Y
	Whether the value of start separator set in each component is duplicated, in case of selected configuration	Set the value of start separator that has not been set in other components, in the component for which start separator is duplicated.	--	Y
Component	Whether the specified occurrence count fulfills following expression, when the occurrence count is fixed `0<Occurrence count=2,147,483,647`	Specify the occurrence count in the range of 1~2,147,483,647.	Y	--

Validation target	Validation contents	Action in case of error or warning occurrence	Validation method	
			Auto	Optional
Component	Whether the occurrence count node is set, when the occurrence count is "Occurrence count node"	Set the occurrence count node in the corresponding component	--	Y
	Whether data type of occurrence count node is numeric value type, when the occurrence count is "Occurrence count node"(Numeric value type implies integer, real number, fixed fraction part numeric value, zone format numeric value, pack format numeric value, signed integer, or unsigned integer)	In occurrence count node, specify simple contents element having data type as numeric value type.	Y	Y
	Whether occurrence count of occurrence count node and its all ancestors is fixed and that is 1, when the occurrence count is "Occurrence count node"	Set the occurrence count of occurrence count node and its ancestor nodes to a fixed value, that is 1	--	Y
	Whether path of the occurrence count node is correct, when the occurrence count is "Occurrence count node"	Path that shows occurrence count node might not be correct. For details on actions, see "*Table4-5 Cause and actions when the path is invalid*"	Y	Y
	Whether complex contents element of selected configuration is included in the path of occurrence count node, when the occurrence count is "occurrence count node" (however, exclude the ancestor node of setting source node, when the path is absolute path)	Occurrence count data might not occur in actual binary data. Check whether the data corresponding to occurrence count node occurs in the actual binary data	--	Y
	Whether the node specified in occurrence count node includes the complex contents element specified in size node, when the component is simple contents element (exclude the node that is ancestor node of the setting source node, when the path is absolute path. Exclude standard node when the path is standard path)	Selection condition node may not occur in the actual binary data. Check whether corresponding data occurs in the selection condition node, in the actual binary data.	--	Y
	Whether the recursive structure is set	Delete the recursively set components.	Y	--
	Whether all the following conditions are fulfilled when parent element is sequential configuration • Simple contents element or complex contents element having other than bit string type • Sibling element existing at 1 previous position is the simple contents element of the bit string type • Size of consecutive bit string type is totaled and size is not in byte unit	Check the size of sibling element of bit string type. Calculate the size of element of bits string type as follows: When occurrence count is fixed, "size x maximum occurrence count". When occurrence count is not fixed, calculate as follows: • When actual occurrence count is unknown and size is not in byte unit, consider it as warning. • When size is of bytes unit, even if the occurrence count is unknown, value of "size x occurrence count" is always in the byte unit and hence calculate occurrence count as 1 and consider as "size x 1".	--	?
	Whether all the following conditions are fulfilled when the component is simple contents element and data type is "bit string" • Specified size is less than 8 bit	Change the occurrence count of simple contents element to "fixed" or "occurrence count node" or specify value of 8 bit or more in the size.	--	Y

Validation target	Validation contents	Action in case of error or warning occurrence	Validation method	
			Auto	Optional
Component	• Occurrence count is "Till end of data" or "Range specified" • Element at the end of binary format definition • Size node is not specified in ancestor node	Change the occurrence count of simple contents element to "fixed" or "occurrence count node" or specify value of 8 bit or more in the size.	--	Y
	Whether size node is set when the component is simple contents element and size is "size node"	Set the size node in the corresponding component	--	Y
	Whether the data type of size node is numeric value type, when the component is simple contents element and size is "size node" (numeric value type implies integer, real number, fixed fraction part numeric value, zone format numeric value, pack format numeric value, singed binary integer or unsigned binary integer)	In size node, set the simple contents component having data type as numeric value type.	Y	Y
	Whether occurrence count of size node and all the ancestor nodes is fixed, that is 1, when the component is simple contents element and size is "size node"	Set the occurrence count of size node and its ancestor node to fixed value, which is 1.	Y	Y
	Whether the path of size node is correct when the component is simple contents element and size is "size node"	Path that shows the size node might be invalid. For details on actions, see "*Table4-5 Cause and actions when the path is invalid*".	Y	Y
	Whether path of size node includes complex contents element of selected configuration, when the component is simple contents element and size is "size node" (however, when the path is absolute path, exclude the ancestor node of setting source node)	Size node may not occur in the actual binary data. Check whether the data corresponding to size node exists in the actual binary data.	--	Y
	Whether the node specified in size node includes the complex contents element specified in size node when component is simple contents element (exclude the node that is ancestor node of the setting source node, when the path is absolute path. Exclude standard node when the path is standard path)	Size node may not occur in the actual binary data. Check whether the data corresponding to size node exists in the actual binary data.	--	Y
	Whether the separator or size node that shows the end of occurrence is set, when the component is simple contents element and occurrence count is "till end of the data"	Specify separator that shows end of occurrence or size node of the complex contents element, in the element having occurrence count as variable.	Y	Y
	Whether the separator or size node that shows the end of data is set, when the component is simple contents element and size is "till end of the data"	Specify separator that shows end of data or size node of the complex contents element, in the element having size as variable.	Y	Y
	Whether the separator or size node that shows the end of occurrence is set, when the component is complex contents element and occurrence count is "till end of the data"	Specify separator that shows end of occurrence or size node of the complex contents element, in the element having occurrence count as variable.	Y	Y
	Whether all the following conditions are fulfilled when the component is complex contents element and occurrence count is "Till end of data" or "range-specified".	Change the occurrence count of complex contents element to "fixed" or "occurrence count node" or define such that size of complex contents element becomes 8 bits or more.	--	Y

Validation target	Validation contents	Action in case of error or warning occurrence	Validation method	
			Auto	Optional
Component	1. Data type of last element in the binary format definition is "bit string" 2. Occurrence count of element in 1. Is "fixed" or "occurrence count node" 3. Size node is not specified from element in 1. To ancestor node 4. Total size till element in 1. Is less than 8 bits	Calculate the size of each element as follows: • When the data type is bits string and occurrence count is fixed, the size is "value specified in size x maximum occurrence count". • When data type is bits string and occurrence count is not fixed, size is the value specified in size. However, when not the last element, but the smallest occurrence count is 0, consider size as 0. • When data type is other than bits string, consider size as 8 (as validation ends when total size exceeds 8 bits). However, when smallest occurrence count is element of 0 or the complex contents element for which size node is specified, consider size as 0.	--	Y
	Whether total size of bit string is in byte unit when the component is complex contents element, size node is set and bit string exists in grandchild node	Check the size of simple contents element of bit string	--	Y
	Whether complex contents element in which separator is defined exists in the binary format definition, when component is complex contents element and size node is set	Delete the specification of size node or cancel the specification of separator.	Y	--
	Whether data type of the size node is numeric value type when component is complex contents element (numeric value type implies integer, real number, fixed fraction part numeric value, zone format numeric value, pack format numeric value, singed binary integer and unsigned binary integer)	In size node, specify simple contents of numeric value type	Y	Y
	Whether size node and all its occurrence count is fixed, that is 1 time, when component is complex contents element	Define such that the node has occurrence count fixed as 1 time, for size node and its ancestor node.	Y	Y
	Whether the path of size node is correct when the component is complex contents element	Path specified in size node may get discarded. Specify the side node again, with reference to detail information.	Y	Y
	Whether the path of size node has complex contents element of selected configuration, when component is complex contents element	Size node may not occur in the actual binary data. Check that the corresponding data occur in the size node, in actual binary data.	--	Y
	Whether the node specified in size node include the complex contents element included in size node, when component is complex contents element (when the path is absolute path, exclude the node that is ancestor of the setting source node. When path is standard path, exclude the standard node).	Size node may not occur in the actual binary data. Check that the corresponding data occur in the size node, in actual binary data.	--	Y

Legend:

Y: Validated.

--: Not validated

Note#

Separator having the type of separator value as byte string, is not validated.

Table 4–4: Validation contents and actions related to the value specified in selection condition value

Data type	Validation contents	Action
Character string	Whether following expression is fulfilled when "Replace with substitute character" is not specified#1 `Size after character code conversion=Specified size`	Specify such that byte size of selection condition value becomes less than the size of selection condition node.
Integer	Whether the format of specified character string is n#2#3	In selection condition value, specify character string matching to the data type of selection condition node.
	Whether the following expression is fulfilled `Size after character code conversion=Specified size`	Specify such that byte size of selection condition value is less than the size of selection condition node.
	Whether it is 0=value, when the existence of sign is "No"	Specify a positive value in selection condition value.
	Whether digits count of condition value is less than the specified digits count of "Overall digits count", when the "overall digits count" is specified	Specify the digits count of selection condition value as less than overall digits count.
	Whether the digits count of condition value is 34 digits or less, when "Overall digits count" is not specified	In selection condition value, specify numeric value having 34 or less digits, in the valid digits count.
Real number	Whether the format of specified character string is n or n.m#2#3	In selection condition value, specify the character string matching with data type of selection condition node.
	Whether the following expression is fulfilled#1#4 `Size after character code conversion=Specified size`	Specify the bytes size of selection condition value such that it is smaller than the size of selection condition node.
	Whether it is 0=value, when the existence of sign is "No"	Specify positive value in the selection condition value.
	Whether the fraction part digits count of condition value is less than specified digits count of "Fraction part digit count", when the "Fraction part digits count" has been specified	Specify the digits count in fraction part of selection condition value to the count less than fraction part digits count.
	Whether digits count of condition value is less than specified digits count of "overall digits count", when "overall digits count" has been specified#4	Specify the digits count of input value of the selection condition value to count less than overall digits count.
	Whether the digits count of condition value is 34 digits or less, when "overall digits count" has not been specified	Specify numeric value having less than 34 digits as valid digits count, in selection condition value
Fixed fraction part numeric value	Whether the format of specified character string is n or n.m#2#3	In selection condition value, specify character string matching with data type of selection condition node.
	Whether the following expression is fulfilled#1 `Size after converting to encrypted numeric value with decimal point=Specified size`	Specify such that byte size of selection condition value is less than the size of selection condition node
	Whether it is 0=value, when the existence of sign is "No"	Specify positive value in the selection condition value.

Data type	Validation contents	Action
Fixed fraction part numeric value	Whether the fraction part digits are less than specification of "Fraction part digits count	Specify the fraction part digits count of selection condition value to less than fraction part digits count
	Whether the digits count of condition value after converting to encrypted numeric value with decimal point is less than specified digits count of "overall digits count", when the "overall digits count" has been specified	Specify digits count of input value of selection condition value to count less than overall digits count.
	Whether digits count of condition value is 34 digits or less, when "overall digits count" has not been specified	In selection condition value, specify a numeric value having 34 digits or less as valid digit count
Zone format numeric value	Whether the format of specified character string is n or n.m [#2]	In selection condition value, specify character string matching with data type of selection condition node.
	Whether the following expression is fulfilled `Size after converting to zone format=Specified size`	Specify such that byte size of selection condition value is less than the size of selection condition node
	Whether digits in integer part+digits in fraction part are real number having 34 digits or less	Specify numeric value having less than 34 digits as valid digits count, in selection condition value
	Whether the fraction part digits are within the range specified in "fraction part digits count"	Specify digits count of fraction part of selection condition value, as the digits less than fraction digits count
	Whether negative value is set in the condition value, when the settings of sign of simple contents element is "custom" and existence of sign is "does not exist"	Specify positive value in the selection condition value
Pack format numeric value	Whether the format of specified character string is n or n.m [#2]	In selection condition value, specify character string matching with data type of selection condition node.
	Whether the following expression is fulfilled Size after converting to pack format=Specified size	Specify such that byte size of selection condition value is less than the size of selection condition node
	Whether digits in integer part+digits in fraction part are real number having 34 digits or less	Specify numeric value having less than 34 digits as valid digits count, in selection condition value
	Whether the fraction part digits are in the range specified in "fraction part digits count"	Specify digits count of fraction part of selection condition value to count less than fraction part digits count.
	Whether negative value is set in the condition value, when the settings of sign of simple contents element is "custom" and existence of sign is "does not exist"	Specify positive value in selection condition value
Signed binary integer	Whether the format of specified character string is n or -n[#2]	In selection condition value, specify character string matching with data type of selection condition node.
	`Whether the integer fulfills following expression when the size is n bytes (n=1~8)` -1x2$^{(8 \text{ x } n)-1}$=Value=2$^{(8 \text{ x } n)-1}$-1	Specify valid value in the selection condition value.
Unsigned binary integer	Whether the format of specified character string is n[#2]	In selection condition value, specify character string matching with data type of selection condition node.
	`Whether the integer fulfills following expression when the size is n bytes (n=1~8)`	Specify valid value in the selection condition value.

Data type	Validation contents	Action
Unsigned binary integer	0<=Value<=$2^{8 \times n}$-1	Specify valid value in the selection condition value.
Bytes string	Whether the format of specified character string is 0~9, a~f and A~F, when the code format is hexBinary Whether the format of specified character string is 0~9, a~z, A~Z, -, , =, or blank character, when encode format is base64Binary	In selection condition value, specify character string matching with data type of selection condition node.
	Whether following expression is fulfilled `Size after converting to bytes=specified size`	Specify such that byte size of condition value is same as the byte size of selection condition node.
Bit string	Whether the format of specified character string is 0~9, a~f and A~F	In selection condition value, specify character string matching with data type of selection condition node.
	Whether the following expression is fulfilled A=(<specified size>+7)/8(round up of digits after decimal point) A x 2=input character count	Specify required digits count.
Date and time	Whether the specified character string format is CCYYMMDD, when the format is CCYYMMDD	In selection condition value, specify character string matching with data type of selection condition node.
	Whether the specified character string format is YYMMDD, when the format is YYMMDD	In selection condition value, specify character string matching with data type of selection condition node.
	Whether the specified character string format is hhmmss, when the format is hhmmss	In selection condition value, specify character string matching with data type of selection condition node.
	Whether the specified character string format is CCYYMMDDhhmmss, when the format is CCYYMMDDhhmmss	In selection condition value, specify character string matching with data type of selection condition node.
	Whether the specified character string format is YYMMDDhhmmss, when the format is YYMMDDhhmmss	In selection condition value, specify character string matching with data type of selection condition node.
	Whether the specification of seconds decimal point match with the digits count of second decimal point	In selection condition value, specify character string matching with data type of selection condition node.
	Whether the specified date and time is correct date and time	Specify correct date and time in the selection condition value

Note#1

When other than CUSTOM has been set in character code

Note#2

n, m indicate numeric value in the range of 0~9. However, start of n and end of m has numeric value in the range of 1~9.

Note#3

Sign to be added to condition value differs depending on the settings of "existence of sign" on the selection condition node. Sign is necessarily added in case of "Always exist". Sign is not added in case of "does not exist". In case of "only minus exists", you can specify only "-".

Note#4

When condition value has fraction part and integer part is only 0, count after deleting 0.

Table 4–5: Cause and actions when the path is invalid

Cause for invalid path	Action
Node of the path specified in selection condition, occurrence count or size node has been deleted from the definition	Specify the selection condition, occurrence count or size node again.
Hierarchy structure was changed after specifying the selection condition, occurrence count or size node	Specify the selection condition, occurrence count or size node again.
Path specified in selection condition, occurrence count or size node occurs after element of setting condition.	Specify element that occurs before setting destination, in the selection condition, occurrence count or size node.
Multiple components that reference the node of node path settings destination are defined	Set the component that references the node path setting destination to 1.
Topmost element of absolute path is not the root element	Set the root element to topmost element of absolute path, or reset the path.
Standard node does not exist in ancestor on the setting source node	Reset the node path.
Standard node of standard path is not the element of sequential configuration	Set the path such that standard node is sequential configuration.
Common parent of path and setting source node is complex contents of selection configuration	Set the common parent of path and setting source node to complex contents of sequential configuration.

(2) Validation methods

Following 2 validation methods exist. Contents to be validated differ depending on the validation method. For details, see "*(1) Validation contents*".

Validation in case of dialog input

In the dialog used when setting each element of the binary format definition file, entered value is automatically validated in sequential manner.

Validation in optional timing

You can validate at any timing such as after creating or during creation of the binary format definition file.

To implement validation, select and right click element in the binary format definition editor and select [Validate]. The binary format definition file displayed in the binary format definition editor is validated.

Elements to be selected in binary format definition editor, when validating are optional. Even if you select any element, entire binary format definition file is validated.

For details on display of validation result, see " *(3) Displaying the validation results*".

(3) Displaying the validation results

The validation results are displayed in a window and in a dialog box.

(a) Displaying result of auto validation at the time of dialog input

Auto validation result at the time of dialog input is displayed in the [Information] of the dialog related to the binary format definition. If there is no problem, nothing is displayed.

For details on the dialog related to a binary format definition, see "1.2.1 Binary Format Definition Window" in "Service Platform Reference Guide".

(b) Displaying result of validating at optional timing

Validation result is displayed on the screen.

For screen display contents, see the following locations.

- "1.1.1 Configuration of the screen for creating the binary format definition file" in "Service Platform Reference Guide"
- "1.2.1 Binary format definition screen" in "Service Platform Reference Guide"

When validation is successful

Message indicating the success of validation is displayed in the Console view.

When validation fails

Message indicating error, warning and information notification contents is displayed in the Console view. Also, contents of error, warning and information notification are displayed in each part on the screen.

Binary format definition editor and outline view

Icon indicating error or warning is displayed in the erroneous node. When error and warning both exist, icon of error is given priority for display.

Package explorer

Icon indicating error or warning is displayed in erroneous file and its ancestor folder. When error and warning both exist, icon of error is given priority for display.

Problem view

Message, resource name (file name) and folder name showing problems are displayed in list format.

4.4.7 Notes regarding binary format definition

You cannot specify simple contents element of variable length in size node and occurrence count node, in case of transformation from XML type to binary type.

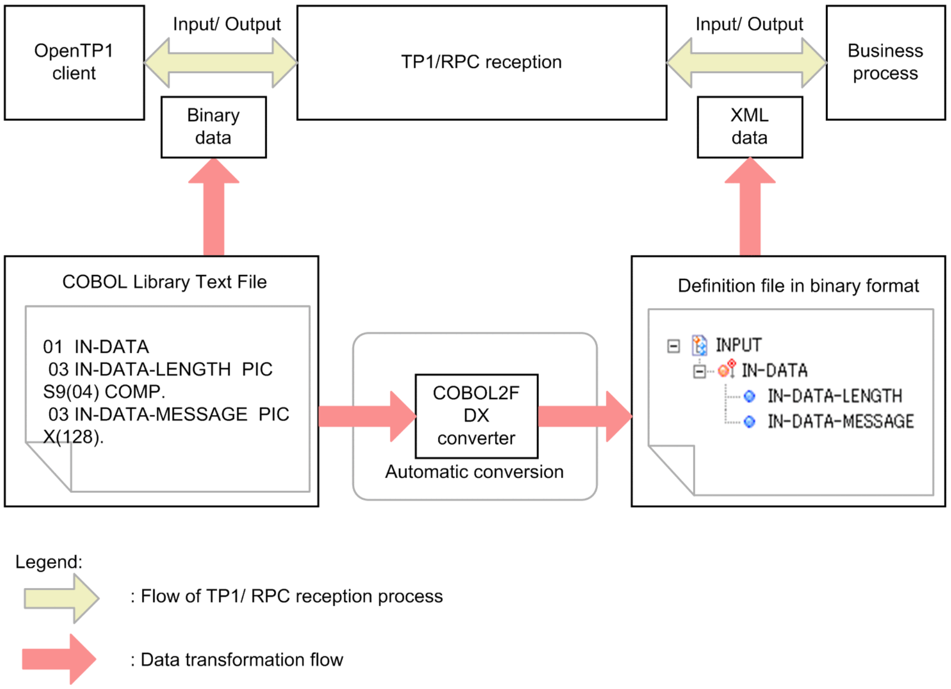
4.5 Generating the binary format definition file from COBOL Library Text File

Generating the binary format definition file from COBOL Library Text file is a function that converts the specified COBOL Library Text File to the binary format definition file.

When you use this function, you can automate the generation of the binary format definition file from COBOL Library Text File. Therefore, effectivity of data transformation work can be plotted.

Following figure shows the image of data transformation by using the function for generating the binary format definition file from COBOL Library Text File:

Figure 4–7: Data transformation by using function for generating binary format definition file from COBOL Library Text File (example of using TP1/RPC reception)



This function uses the COBOL2FDX converter. The COBOL2FDX converter parses the record definition of the specified COBOL Library Text File and outputs as the binary format definition file.

4.5.1 Description format of COBOL Library Text File that can be transformed

This section describes description format of COBOL Library Text File that can be used in function for generating binary format definition file from COBOL Library Text File.

This function considers the usage of COBOL85 language as prerequisite. However, you can use COLBOL Library Text File described according to rules and format explained here. For details on functions and methods of COBOL85 language, see "COBOL85 language".

(1) File format that can be specified

You can specify only fixed format (extension: other than cbf) as the COBOL Library Text File that you can use in this function. File having free format (extension: .cbf) is not supported.

(2) Description rules for COBOL Library Text File

COBOL Library Text File that you can use in this function conforms to the following description rules.

- Character code of COBOL Library Text File must be MS932.
- You must use the correct syntax. Pre-requisite is the condition like messages output by compiler do not include the "Fatal error" or "Warning error".
- First~sixth column are not used as row number and are handled as 6 digits single byte spaces (0x20). It is not used even as the line number output in the error information.
- Comment line (seventh column is "*", "/") is ignored.
- Comments from 73rd column onwards are ignored.
- Debug line (line having seventh column as "D" and "d") is ignored.
- Similar to normal COBOL, line having seventh column as "-" is handled as consecutive lines.
- Type character is handled as 1 blank character.
- Double byte blank character is handled as seperative sign, but ";" and "," is not handled as seperative sign.
- Extended code character and standard code character is handled as non-equivalent.
- Following words are handled as reserved words. Other words are handled as user-words.
    - COMP
    - COMP-3
    - COMPUTATIONAL
    - COMPUTATIONAL-3
    - DISPLAY
    - FILLER
    - IS
    - NATIONAL
    - OCCURS
    - PIC
    - PICTURE
    - TIMES
    - USAGE

(3) Description format of data item

Description of data items such as alphabets or numbers must conform to the rules described in this point. If rules are not followed, syntax error occurs.

(a) Writing style of data item

Writing style of data item of COBOL is as follows:

```
Record number {Data name|FILLER}
[{PICTURE|PIC}[IS] character string]
[[USAGE [IS]]{COMPUTATIONAL-3|COMP-3|COMPUTATIONAL|COMP|DISPLAY|NATIONAL}]
[OCCURS integer [TIMES]].
```

(b) Syntax rules

1. Level number must be unsigned integer of 1~2 digits and must be in the range of 1~49.
2. Character count of data name must be in the range of 1~30.
3. Count of data item and FILLER item must not exceed 32,760.

4. Integer value that you specify in OCCURS clause must be in the range of 1~2,147,483,647.
5. In PICTURE clause, you can specify only the contents given in "Table4-6 Characters that you can specify in PICTURE clause".
6. Regarding the syntax rules of level number, data name, PICTURE clause, USAGE clause and OCCURS clause, rules other than the rules described in syntax rules in 1.~5. Conform to the syntax rules of COBOL85 language.
7. Syntax rules of USAGE clause are as follows:
8. You cannot specify USAGE in aggregation item.
    - If you specify DISPLAY, NATIONAL in the numeric value item, it is ignored.
    - If you specify USAGE in other than numeric value item, it is ignored.
    - Data item must comprise of only 1 record (group item of level number 01) and dependent items.
9. You cannot specify COPY statement.

Table 4–6: Characters that you can specify in PICTURE clause

No	Data type		Character that you can specify in PICTURE clause		Fraction part	Range of available digits (including fraction part)
			Character	Digits count		
1	Alphabet		"A"	1	--	1~2,147,483,647
2	Alphanumeric		"A", "X", "9"	1		
3	Number	External decimal format	"9"	1	Alphabets from "V" onwards	1~18
			"S", "V"	0		
4		Internal decimal format	Similar to external decimal format			
5		Binary format	"9"	1	N	
			"S"	0		
6	Alphanumeric edition items		"A", "X", "9", "B", "0", "/"	1	--	1~2,147,483,647
7	Number edition items		"B", "/", "Z", "0", "9", ",", ".", "*", "+", "-" currency edition character	1	N	1~249
			"CR", "DB"	2		
			"V"	0		
8	Japanese item		"N"	2	--	2~32,766
9	Japanese edition item		"N", "B"	2		

Legend:

--: Corresponding item does not exist.

N: Not supported in this function.

Note

Similar to normal COBOL, when "nn" (nn is number in the range of 1~2,147,483,647) is specified immediately after each character, character repetition count is set.

(c) Notes

The currency edition character or edition character such as "Z", or "0" that you specify in PICTURE clause is used only with the purpose of calculating the digits count. Edition function for embedding currency edition character in data item is not supported.

4.5.2 Support to data type of COBOL Library Text File and binary format definition file

This section describes the support to data type of COBOL Library Text File before transformation and binary format definition file after transformation.

(1) Support to element name

When optional name is set in data item of COBOL, the already set optional name serves as the element name of the binary format definition file. When optional name has not been set, data name of COBOL before transformation as it becomes the element name of the binary format definition file.

For details on setting the optional name in element name, see " *4.5.3(3) Setting up data items*".

(2) Support to data type

Following table describes the mapping of data type of COBOL and data type o binary format definition file after transformation.

Table 4–7: Mapping of data type of COBOL and data type of binary format definition file

No.	Data type of COBOL			Data type of binary format definition file			Data type in TP1/ COBOL adapter
				Data type	Embedded character	XML schema type	
1	Alphabet			Character string	Space	string	Character string data (string)
2	Alphanumeric			Character string	Space	string	Character string data (string)
3	Number	External decimal format		Zone format numeric value	--	decimal	Decimal data (BigDecimal)
4		Internal decimal format		Pack format numeric value	--	decimal	N
5		Binary format	Without decimal point	Signed (unsigned) binary number	--	integer	Short data (Short) ~Long data (Long)
6			With decimal point	N			Decimal data (BigDecimal)
7		External floating decimal point		N			N
8		Internal floating decimal point	4 bytes				Single precision data (Float)
9			8 bytes				Double precision data (Double)
10	Alphanumeric edition item			Character string	Space	string	Character string data (string)
11	Number edition item			Character string	Space	string	Character string data (string)
12	Index data item			N			N
13	Japanese item			Character string	Space	string	Character string data (string)

No.	Data type of COBOL	Data type of binary format definition file			Data type in TP1/ COBOL adapter
		Data type	Embedded character	XML schema type	
14	Japanese edition item	Character string	Space	string	Character string data (string)
15	External bool item	N			N
16	Internal bool item	N			N

Legend:

--: Corresponding item does not exist

X: Not supported in this function

(3) Support to sign of zone format/pack format numeric value

When generating zone format numeric value and pack format numeric value when transforming to data type of binary format definition file, generate when the settings of sign is in "Custom" status and perform settings in the item of simple contents element dialog. Following table describes the contents set in the item of simple contents element dialog.

Table 4–8: Setting contents of attribute

No.	Attribute	Setting contents
1	Settings of sign	Custom
2	Existence of sign	• When "S" is specified in PICTURE character string: exists • In other cases: Do not exist
3	Type of sign	Sign bit
4	Position of sign	Later

(4) Method of calculating the size of data type of COBOL

Following table describes the method for calculating size (byte) of data type.

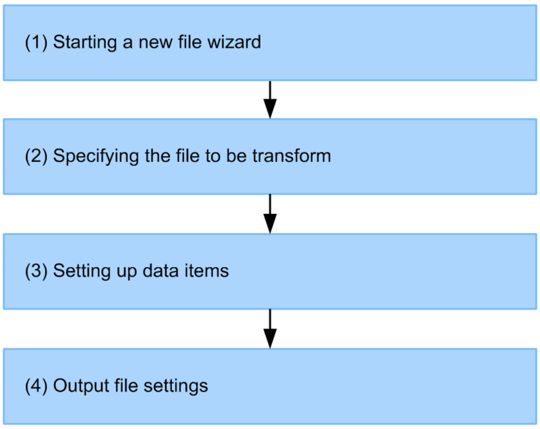
Table 4–9: Method for calculating size of data type

Data type of COBOL	Size (byte)
Alphabet	Digits count is same.
Alphanumeric	
External decimal format	
Internal decimal format	((Digits count) / 2) + 1
Binary format	Calculate from digits count. • In case of 1 digit~4 digits: 2 • In case of 5 digits~9 digits: 4 • In case of 10 digits~18 digits: 8
Alphanumeric edition item	Digits count is same.
Number edition item	
Japanese item	
Japanese edition item	

4.5.3 Method of generating the binary format definition file

Following figure shows the flow of data transformation operation using the binary format definition file from COBOL Library Text File:

Figure 4–8: Flow of data transformation operation


(1) Starting a new file wizard
(2) Specifying the file to be transform
(3) Setting up data items
(4) Output file settings

This section describes about the data transformation related operation.

(1) Starting a new file wizard

Method to start the new file wizard is as follows:

1. Open New file wizard, by either of the following methods:
    - From menu of Eclipse, select [File]-[New]-[Others].
    - Right click any location in the Package explorer and select [New]-[Others].
2. In New file wizard, select [HCSCTE format definition]-[Binary format definition file(transformation from COBOL Library Text File)].
3. Click [Next] button.
   Proceed to [COBOL Library Settings page].

(2) Specifying the file to be transformed

Following section describes the method of specifying the COBOL Library Text File to be transformed.

1. Click [Browse] button on [COBOL Library Settings page] and specify the COBOL Library Text File to be transformed.
2. In [Currency edition character of PICTURE clause], specify the currency edition character of PICTURE clause, to be used.
   For details on the characters that you can specify in currency edition character of PICTURE clause, see "1.3.10 Conversion dialog from COBOL Library Text" in "Service Platform Reference Guide".
3. In [Character code of basic items], select character code used in alphanumeric items of COBOL Library Text File.
4. In [Binary format Endian], select either of Big Endian or Little Endian.
5. Click [Next] button.
   COBOL Library Text File is transformed and process proceeds to [Data item Settings page]. If an error occurs during transformation, an error message is displayed.

(3) Setting up data items

This section describes a method to set up data items.

1. To change the element name of the binary format definition file after transformation, select the cell of [Optional name] in the [Data item settings page] and enter optional name.

   If the name you have specified is incorrect, a message is displayed in the upper part of page. For details on the character that you can specify in [Optional name], see "1.3.10 Conversion dialog from COBOL Library Text" in "Service Platform Reference Guide".

2. Click [Next] button.

   Process proceeds to [Output file settings page].

   When you click [Back] button, a message for confirming whether to discard the already set contents is displayed. If you want to return to [COBOL Library text settings page] and specify COBOL Library Text File once again, click [OK] button.

(4) Output file settings

This section describes a method to set up the output file.

1. Specify output destination folder, from Tree view of [Output file settings page].

   For details on [Output file settings page], see "1.3.10 Conversion dialog from COBOL Library Text" in "Service Platform Reference Guide".

2. In [Format name], enter format name of the binary format definition file.

3. Click [Finish] button.

   Binary format definition file after conversion is output and the Binary format definition editor is started. During the file output, a progress bar is displayed on the lower part of the screen.

Reference

You can edit the binary format definition file after transformation, by using the Binary format definition editor, in the same way as the file directly created using the Binary format definition editor. For details on the Binary format definition editor, see "4.4.1 Types of data type and character code of binary format definition file".

4.6 Generating an XML schema file from the binary format definition file

Use the `cscfdx2xsd` command to generate an XML schema file from the binary format definition file. If you use this command to define transformation of the binary data into an XML data with the same structure, you can easily create an XML schema file for the transformation destination.

For details about the `cscfdx2xsd` command, see the manual *Cosminexus Service Platform Reference*.

The following is the procedure for defining the file adapter for reading the binary data described as an example of the proper use of the `cscfdx2xsd` command:

1. Creating the binary format definition file

   Use the Wizard to create the binary format definition file.

   For details about how to create the binary format definition file, see *4.4 Creating Message Formats (Binary Format Definition File)*.

2. Creating the XML format definition file

   Based on the created binary format definition file, execute the `cscfdx2xsd` command and generate the XML schema file. If you use the command to transform the binary data into an XML data with the same structure, you need not create the XML schema definition using an editor.

3. Defining the business process and service adapter

   Use the Wizard to set up the created binary format definition, to set up the XML schema definition generated by the command, and to create the data transformation definition.

   For details about defining the adapters, see *5. Defining Business Processes* and for details about the data transformation definition, see *6. Defining Data Transformation*.

4.7 Changing the message formats

If you change the message format, you must re-define the data transformation. After changing the message format, if you do not define the data transformation again, an error occurs in validation (adapter or business process) or packaging.

When you re-define the data transformation, you can use the previously created data transformation definition. You need not create a new definition. For details about the definition procedure and the precautions when the message format is changed, see *6.3.2 Procedure for defining changed message formats*.

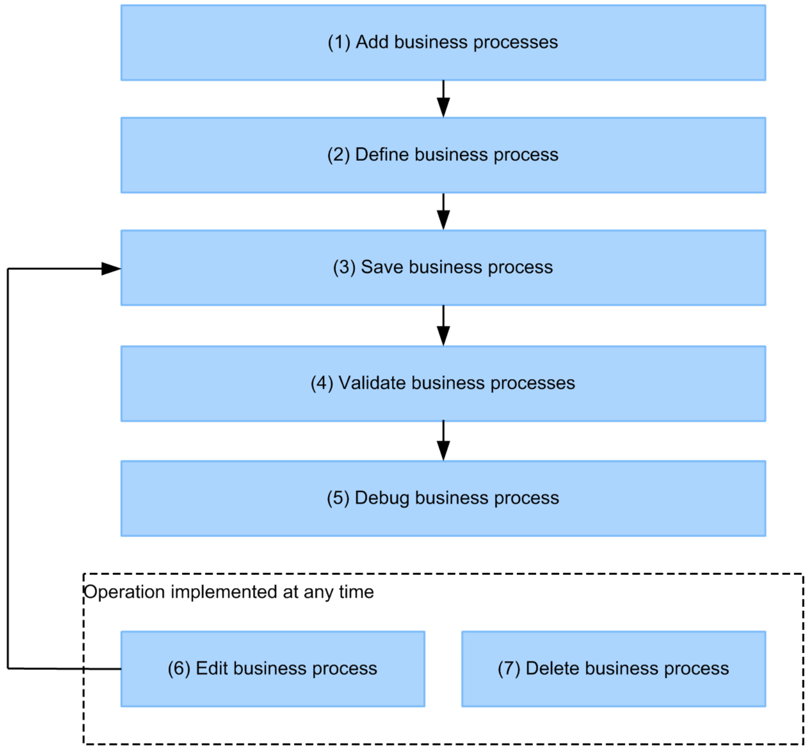
5 Defining Business Processes

This chapter explains the workflow of defining business processes.

5.1 Definition Work Flow

The following figure shows the workflow for defining business processes.

Figure 5–1: Workflow for defining business processes


(1) Add business processes
(2) Define business process
(3) Save business process
(4) Validate business processes
(5) Debug business process
Operation implemented at any time
(6) Edit business process
(7) Delete business process

The tasks involved in defining business processes are explained below.

(1) Adding business processes

To add business processes, use either one of the methods described below.

To add a new business process

Use a wizard to add new business processes.

There are two methods for adding a new business process with a wizard, one method is to add an undefined business process, and the other method is to first import the BPEL file created by the high level design tool in which BPMN is used, convert the BPEL file into a business process definition, and then add a business process.

For details about how to add new business processes, see *5.2.1 Adding New Business Processes*.

Using already defined business processes

You can add a business process by copying an already defined business process. When you copy an already defined business process, a business process with the same definition contents is added. You can also edit the definition contents of the copied business process.

For details about how to add a business process by copying an already defined business process, see *5.2.2 Using an Already Defined Business Process to Add Business Processes*.

(2) Defining business process contents

Define business process contents in the Business Process Definition screen.

For details about definition methods and definition contents, see *5.3 Defining Business Process Contents*. For details about the Business Process Definition screen, see the manual *Cosminexus Service Platform Overview*.

(3) Saving business processes

You need to save the definition information of edited business processes in a repository as needed.

For details about how to save business process definition information and the timing for saving it, see *5.8 Saving Business Processes*.

(4) Validating business processes

Validate the conformity of the business processes you have defined. You can perform validation at any time, such as when the business processes are being defined, or after they have been defined.

For details about the validation method, see *5.10 Validating Business Processes*.

(5) Debugging business processes

Debug the defined business process in the development environment. After defining the business process you can debug them at any time.

For the method to debug a business process, see *9. Debugging Business Processes*.

(6) Editing business processes

You can edit and change the definition contents of already defined business processes as needed.

For details about the business process editing method, see *5.9 Editing Business Processes*.

(7) Deleting business processes

You can delete business processes that are no longer needed as necessary.

For details about the business process deletion method, see *5.11 Deleting Business Processes*.

5.2 Adding Business Processes

Adds business processes.

To add new business processes, use the Business Process Addition Wizard. You can also add new business processes by copying already created business processes.

5.2.1 Adding New Business Processes

The two methods to add a new business process with the Business Process Addition Wizard are as follows:

- Adding a new undefined business process
  This method is used to add a new undefined business process. The new business process added with this method has no defined information.
- Importing and adding a new BPEL file
  In this method, you first create a BPEL file using the high level design tool in which BPMN is used. After that, you import the BPEL file, convert it into a business process definition, and add a new business process. The new business process added with this method inherits the definition information of the original BPEL file.

The methods are explained below:

(1) Adding a new undefined business process

The method to add a new business process in which nothing is defined is as follows:

1. In the service definition list in the tree view, choose and right-click Add Business Process.
   The dialog box for adding a business process opens.
2. Enter a business process name and choose whether to make its status persistent (whether to save the execution status of the business process in a database).
   Make sure that the business process name (service name of the business process) does not exceed 64 bytes.
3. Click Finish.
   The required files are created and saved in a repository.
   When the business process is saved normally, the Business Process Definition screen opens.

(2) Adding a new business process by importing a BPEL file

The following points explain how to create a BPEL file in advance with the high level design tool, import a BPEL file, and then add a new business process:

(a) Creating a BPEL file

Create a BPEL file conforming to BPEL1.1 with the high level design tool. For details about how to create a BPEL file, see the documents for the high level design tool to be used.

For the relationship between the contents of elements and attributes defined in the BPEL file, and the contents of business process definition, see *Appendix E. Support Range of BPEL Used by Linking with an High Level Design Tool*.

(b) Adding a new business process

To import a BPEL file and add a new business process:

1. In the service definition list in the tree view, choose and right-click Add Business Process.
   The dialog box for adding a business process opens.
2. Enter a business process name and choose whether to make its status persistent (whether to save the execution status of the business process in a database).
   Make sure that the business process name (service name of the business process) does not exceed 64 bytes.
3. Choose the Import checkbox of the BPEL file.

4. Specify the BPEL file to be imported in File name.

   You can click **...** to choose the BPEL file to be imported.

5. Choose the activity deployment method.

6. Click Finish.

   The definition of the BPEL file is converted into a business process definition. After the end of the conversion, if the required file is created, the file is stored in the repository.

   When the business process is saved normally, the Business Process Definition screen opens. In the canvas of the displayed Business Process Definition screen, the activity is deployed with the method chosen in step 5.

! Important note

Note the following points when importing the BPEL file:

Unconverted BPEL definition

The BPEL definition that is not supported in the development environment of the Cosminexus Service Platform is either ignored or converted into an empty activity.

For the relationship between the contents of elements and attributes defined in the BPEL file, and the contents of business process definition, see *Appendix E. Support Range of BPEL Used by Linking with an High Level Design Tool*.

When a message type variable is defined in the BPEL file

If the message name is defined with the high level design tool and a message type variable is defined in the BPEL file, the variable type is converted into string type (string) in the business process that is created by importing the BPEL file.

In this case, import the file, and then change the variable type to message type (messageType). For details on defining the variables, see *5.5.1(6) Variable definition methods*.

When the invoke service activity is defined

When importing a BPEL file to create a business process, the `invoke` element in the BPEL file is converted into an invoke service activity. The local name of the `portType` attribute in the `invoke` element is converted into an invoke service name and the `operation` attribute is converted into an operation name.

However, if the service or operation is not relevant to the repository, the service name and operation name are not converted.

If the service name and operation name are not converted, after importing the BPEL file, you need to create the service adapter and business process to be invoked, and then allocate the service name and operation name.

When more than one activity with a long name is defined in the BPEL file

When a horizontal-direction arrangement is specified and more than one activity with a long name is defined, activities might overlap at the tip of the canvas after import. In such cases, you re-arrange activities.

(3) Setting up status persistence

Business processes have two types such as a business process for which execution status and execution history has persistence to the database and a business process for which execution status and execution history does not have persistence to the database. Persistence implies recording the execution status and execution history of a business process to the database.

As a business process that has persistence has record of execution status and execution history of a process and hence you can understand the progress of process execution and re-execute the process in case of failure Whereas in case of business process that does not have persistence , you cannot acquire record of execution status and execution history of a process and hence set this business process when there is a necessity of realizing a high performance.

There are limitations such as some operations that you can perform in business process having persistence, cannot be performed in the business process that does not have persistence. You can define a business process that does not have persistence, when the following conditions are fulfilled:

1. Invoke service activity for invoking a service is not included for an asynchronous service adapter.
2. Standby activity is not included.
3. Multiple Receive activities are not included.
4. Re-execution of a process instance is not used (in case of failure, re-execute Invoke service request (invoking a business process) from the service requester.

! Important note

You cannot use list specification method of while activity, in case of a business process that has persistence.

5.2.2 Using an Already Defined Business Process to Add Business Processes

You can copy already defined business processes to add business processes.

1. In the service definition list in the tree view, choose and right-click the business process to be copied.

   The Service List pop-up menu opens.

2. On the pop-up menu, click Copy.

   A copy of the chooseed business process is created. A different name and ID# are automatically assigned to the copied business process (service name of the business process) and service ID to prevent any conflict within the system.

#

A number that is unique within the system and that is automatically assigned when an adapter or business process is added. The service ID can be changed, but a service ID that is already in use cannot be assigned.

! Important note

If a business process is duplicated, the user-defined reception included in the duplicated business process is also duplicated. However, you cannot duplicate user-defined reception only.

5.3 Defining Business Process Contents

You define the business process contents in the Business Process Definition screen.

The Business Process Definition screen opens when you double-click a business process inside a service displayed in the service definition list in the tree view.

For details about the Business Process Definition screen, see the manual *Cosminexus Service Platform Overview*.

In the Business Process Definition screen, you define activities, variables, and correlation sets.

(1) Activities

A business process is defined by linking multiple activities.

An activity is a component that becomes the configuration element of a business process and also expresses a processing structure. You link multiple activities to define a business process processing flow, and define a business process by deploying and linking activities on-screen.

For details about how to deploy and link activities, see *5.4 Deploying and Linking Activities*.

! Important note

The maximum number of activity instances that can be created in the business process when executing a business process is 2,147,483,648. In the following cases, the maximum number that can be created is 32,768.

- When the `activitynumber-maximum-compatible` property is set as `ON`
- When overwrite installation is executed for environment using database during migration from old version

The HCSC server must be re-setup to change the maximum number. Note that the execution log of business processes is deleted during re-setup.

If an attempt is made to create activity instances exceeding the maximum number, an error occurs in the business process.

Note that the activity instance includes start activities, end activities and sequence activities.

If a process is repeated in a while activity, activity instances are created only for the number of activities to be executed.

(2) Variables and correlation sets

You can define variables and correlation sets in a business process.

(a) Variables

When a variable is used as a value of a term in a conditional expression inside a business process, you need to declare that variable in activity definition.

For details about the types of variables that can be used inside business processes, their relationship to activities, and how to define these variables, see *5.5.1 Defining Variables*.

(b) Correlation sets

A correlation set is a character string that is used to uniquely identify the request message to be sent from a service requester to a service component via an HCSC server.

For details about correlation sets, their relationship to activities, and how to define correlation sets, see *5.5.2 Defining Correlation Sets*.

! Important note

During business process definition, values being entered are not checked for validity. The correctness of the entered values is checked during business process validation. If an entered value is incorrect, the validation returns an error.

5.4 Deploying and Linking Activities

In the Business Process Definition screen, you deploy activities on a canvas and link them to define a business process processing flow. To specify fault handling, define fault handling for the link between the activities.

This section explains how to deploy activities, link them, and define fault handling.

5.4.1 Deploying Activities

This subsection explains how to deploy activities in the Business Process Definition screen.

1. In the basic activities or structure activities on the palette, click the activity to be deployed on the canvas.
   The clicked activity becomes chooseed.
2. Click an approp
3. riate location on the canvas.
   The chooseed activity is deployed on the canvas. You can move the deployed activity by dragging it to another location.

For details about the content that can be defined for each activity, see *5.6 Defining Activities*.

Tip

If you import a BPEL file created by the high level design tool and add a business process, the deployment of the activity initially displayed on the canvas differs depending on the deployment method chooseed while adding a business process.

If you choose Vertical
The activity is initially displayed with the process flowing from the top to bottom.

If you choose Horizontal
The activity is initially displayed with the process flowing from the left to the right.

Drag & drop or align the activity as needed. For details about operations in the Business Process Definition screen (canvas), see the manual *Cosminexus Service Platform Reference*.

5.4.2 Linking Activities

Linking two activities deployed on the canvas defines a business process execution sequence.

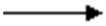
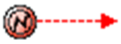
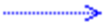
(1) Link types

The following types of links can be used for activities:

- Link based on connection
  This type is used to link normal activities.
- Link based on fault connection
  This type is used to define fault handling to be executed when a fault occurs in activities. For details about defining fault handling, see *5.4.3 Defining Fault Handling*.
- Link based on link connection
  This type is used to define a link in parallel processing that uses flow activities. For details about parallel processing and links, see *5.6.15 Flow Activities*.

How the linking line is displayed on the canvas differs depending on the linking method. The following table describes the relationship between linking methods and linking lines.

Table 5–1: Linking lines displayed on the screen

Linking method	Linking line displayed	Line color	Line type	Termination
Link based on connection		Black	Solid line	Triangular arrow
Link based on fault connection		Red	Broken line	Triangular arrow
Link based on link connection		Blue	Dotted line	Line arrow

(2) Link setting method

To link activities:

1. On the palette, click **Connection**, **Link**, or **Fault**.
   Connection, fault connection, or link connection is selected.
   In this state, move the cursor to an activity. If that activity can be specified as a link source, its background color changes.
2. From the activities deployed on the canvas, click the activity that is to become the link source.
   The activity that is to become the link source is set.
   In this state, move the cursor to another activity. If that activity can be specified as a link destination, its background color changes.
3. From the activities deployed on the canvas, click the activity that is to become the link destination.
   The link-source and link-destination activities become linked.

(3) Changing links

To change an already specified link, drag and drop the starting point or termination point of a linking line into another activity.

(4) Bending linking lines

You can bend and display linking lines. You can use this operation to rearrange the linking lines on the canvas when the links among activities become complex. The method for bending linking lines is described as follows:

1. On the canvas, select a linking line.
   Points at which the linking line can be bent (bend points) are displayed between the starting and termination points of the linking line.
2. Align the pointer of the selection tool to a bend point.
   A cross-shaped arrow (✥) appears.
3. Drag and drop the arrow into any position.
   The linking line bends at the bend point.
   On a linking line that has been bent at a bend point, the following positions can also be used as bend points:
   - Between the starting point of the linking line and the bend point
   - Between the bend point and the termination point of the linking line
   - Between the bend point and another bend point

(5) Conditions for linking activities

Whether an activity can be used as a link source or link destination differs depending on the type of activity and the link used. The following table describes activities that can or cannot be used as link sources or link destinations.

Table 5–2: Activities that can or cannot be used as link sources or link destinations

Activity	Connection		Fault connection		Link connection	
	Link source	Link destination	Link source	Link destination	Link source	Link destination
Start	Y	--	--	--	--	--
Receive	Y	Y	--	Y	Y	Y
Reply	Y	Y	--	Y	Y	Y
Invoke service	Y	Y	Y	Y	Y	Y
Invoke Java	Y	Y	Y	Y	Y	Y
Data transformation	Y	Y	--	Y	Y	Y
Assign	Y	Y	--	Y	Y	Y
Empty	Y	Y	--	Y	Y	Y
Throw	--	Y	--	Y	Y	Y
Standby	Y	Y	--	Y	Y	Y
Validate	Y	Y	--	Y	Y	Y
Scope	Y	Y	Y	Y	Y	Y
While	Y	Y	--	Y	Y	Y
Switch (Start)	Y	Y	--	Y	Y	Y
Switch (End)	Y	Y	--	--	--	--
Flow (Start)	Y	Y	--	Y	Y	Y
Flow (End)	Y	Y	--	--	--	--
End	--	Y	--	--	--	--

Legend:

Y: Activity can be used as a link source or link destination.

--: Activity cannot be used as a link source or link destination.

When activities are linked through a connection, an activity that becomes the transition source to another activity is called a *transition-source activity*, and an activity that becomes the transition destination from another activity is called a *transition-destination activity*.

The number of transition-source activities and transition-destination activities that can be linked using a connection (excluding fault connections and link connections) varies according to each activity.

The following table describes the number of transition-source and transition-destination activities that can be linked for each activity.

Table 5–3: Number of transition-source and transition-destination activities that can be linked

Activity	Number of transition-source activities that can be linked	Number of transition-destination activities that can be linked
Start	0	1
Receive	1	1
Reply	1	1
Invoke service	1	1
Invoke Java	1	1

Activity	Number of transition-source activities that can be linked	Number of transition-destination activities that can be linked
Data transformation	1	1
Assign	1	1
Empty	1	1
Throw	1	0
Standby	1	1
Validate	1	1
Scope	1	1
While	1	1
Switch (Start)	1	1 or more
Switch (End)	1 or more	1
Flow (Start)	1	1 or more
Flow (End)	1 or more	1
End	1	0

! Important note

A receive activity that generates instances is the only basic activity that can be executed first in a business process.

(6) Notes on using connections

- If **yes** is selected for **Transition condition** for the link destination in the Link dialog box, specify transition conditions.
- Do not specify a link connection as a loop.
- Link names must be unique within each business process.
- Do not extend a link connection from the outside of fault handling into the inside of the fault handling.
- If two activities linked with a connection are brought close to each other, the directions of the arrows on the linking line might appear reversed from their intended directions. To avoid this problem, deploy linked activities at a distance from each other.

5.4.3 Defining Fault Handling

To enable process execution when a fault occurs in activities, deploy an activity for fault handling and execute fault handling.

You define fault handling by linking the activity in which a fault occurs with the activity that executes fault handling. Use fault connection for the link.

(1) Procedure for defining fault handling

To define fault handling:

1. Deploy the activity in which a fault occurs and the activity that executes fault handling on the canvas, and define their details.

   You can define multiple activities that execute fault handling.

   For details about how to deploy activities, see *5.4.1 Deploying Activities*. For details about how to define individual activities, see *5.6 Defining Activities*.
2. Use fault connection to link the activity in which a fault occurs with the activity that executes fault handling.

For details about how to link activities, see *5.4.2(2) Link setting method*. For details about activities that can use fault connection, see *5.4.2(5) Conditions for linking activities*.

3. Double-click a link line.

   The Fault Handler dialog box opens.

   For details about the input and display contents of the **Fault Handler** dialog box, see the manual *Cosminexus Service Platform Reference*. Note that only the message type variables are allocated for fault handling.

4. Define fault handling conditions.

   In the Allocated variable drop-down list, specify the variable that is subject to a fault. In the Transition destination drop-down list, specify the link-destination activity that executes processing when a fault occurs.

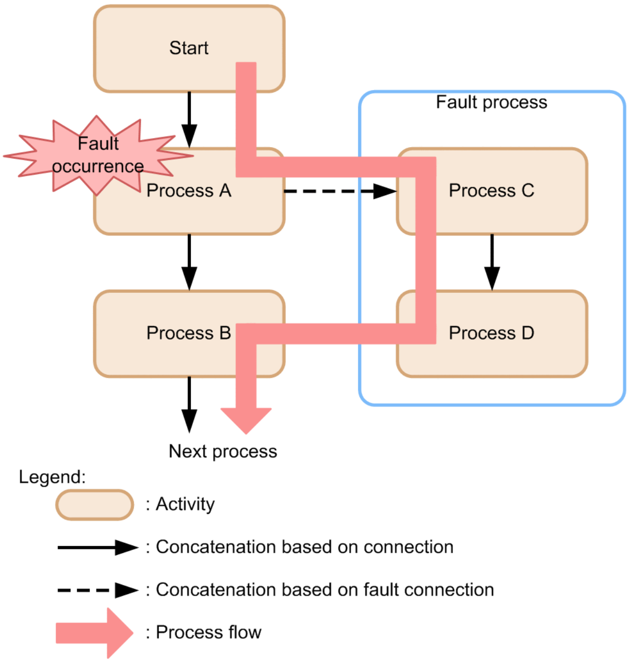
   To specify fault handling to be executed when an undefined fault occurs, click **catch-all** in the **Allocated variable** drop-down list. If the activity in which a fault occurs is a Java-invoking activity, only catch-all fault handling can be specified.

   For converting an exception for error occurred when invoking service component (Web Services) from service adapter to a fault message in service adapter, choose the definition file to be set in the variable definition from the **Allocated variable** dropdown list. For details about how to set the definition file, see *5.5.1(6)(a) Defining new variables*.

5. Click OK.

As shown in the following figure, when a fault occurs, the process moves to the next activity from the one in which the fault occurred after the fault handling is executed:

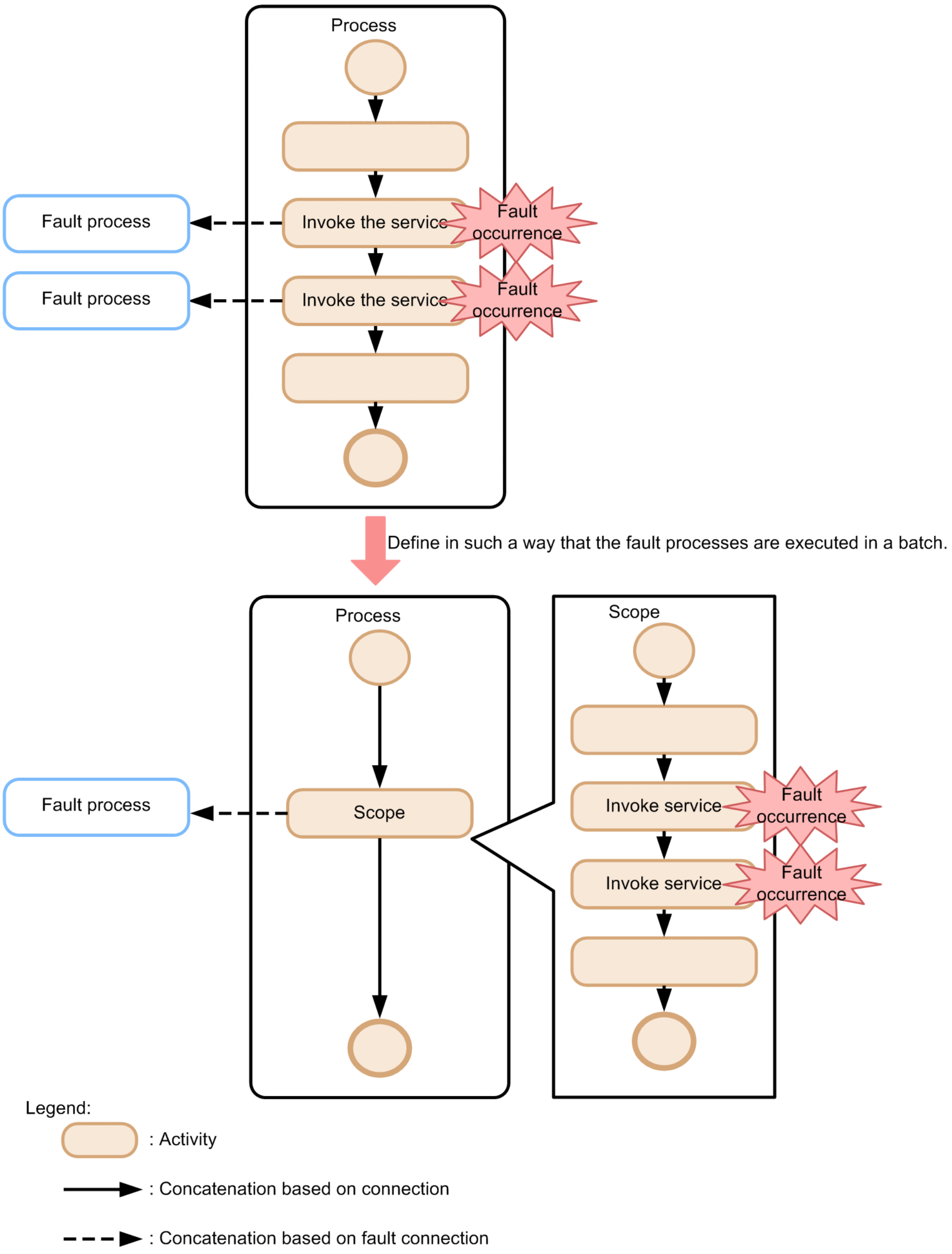
Figure 5–2: Processing after fault handling is executed



(2) Defining fault handling when executed in a batch

When a process contains multiple activities that might return a fault, you can execute fault handling for processes, in a batch instead of defining fault handling for each activity. In such a case, compile processes within the processing into one, and then define fault handling.

Figure 5–3: Fault handling defined by using the scope activity



(3) Message format used for defining fault handling in a business process

This point describes the message format used for defining the fault processing in a business process.

(a) Message format corresponding to multiple fault names

From a Java program that throws multiple user-defined exceptions, use the SOAP application development support functionality (`Java2WSDL` command) and create a WSDL of the `Document` style. If you define an adapter using this WSDL as the input, the message format corresponding to the multiple fault names displayed in *Fault messages* of the Adapter Definition screen will become the same. In such a case, the message format includes multiple root elements as shown in the following example:

Example of a message format (root schema) corresponding to multiple fault names

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema targetNamespace="http://service"
```

```
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:tns="http://service"
  xmlns:tns2="http://data.service"
  xmlns:tns3="http://fault.service">
  <xsd:import namespace="http://data.service"
              schemaLocation="cscformat2.xsd"/>
  <xsd:import namespace="http://fault.service"
              schemaLocation="cscformat3.xsd"/>
  <xsd:element name="order">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="in0" type="tns2:OrderData"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="orderResponse">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="orderReturn" type="tns2:OrderResult"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xsd:element name="StockShortageFault"
               type="tns3:StockShortageFault"/>
  <xsd:element name="InvalidCustomerFault"
               type="tns:InvalidCustomerFault"/>
  <xsd:complexType name="InvalidCustomerFault">
    <xsd:sequence>
      <xsd:element name="CustomerName" type="xsd:string"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```

In the above example, fault messages corresponding to the two faults `StockShortageFault` and `InvalidCustomerFault` are handled by a single message format `cscformat1.xsd`.

The same format is applicable even when an adapter is defined based on the WSDL corrected according to the procedure described in *Notes* of *4.3.2 Creating a Service Component Message (for Web Services)*, so as to handle the faults after creating an `rpc` style WSDL using the SOAP application development support functionality (`Java2WSDL` command).

When this message format is defined as a separate variable for each fault, and `catch` is defined twice as the allocated variable in the **Fault Handler** dialog box to separate the processing according to the generated faults, the format of the two variables allocated as `catch` will become the same. Therefore, no matter which fault occurs, the `catch` allocated variable defined first will be applicable, and the corresponding fault handling will not be invoked for the cause.

When the fault message has such a format, make sure to take action by defining the business process according to the following method:

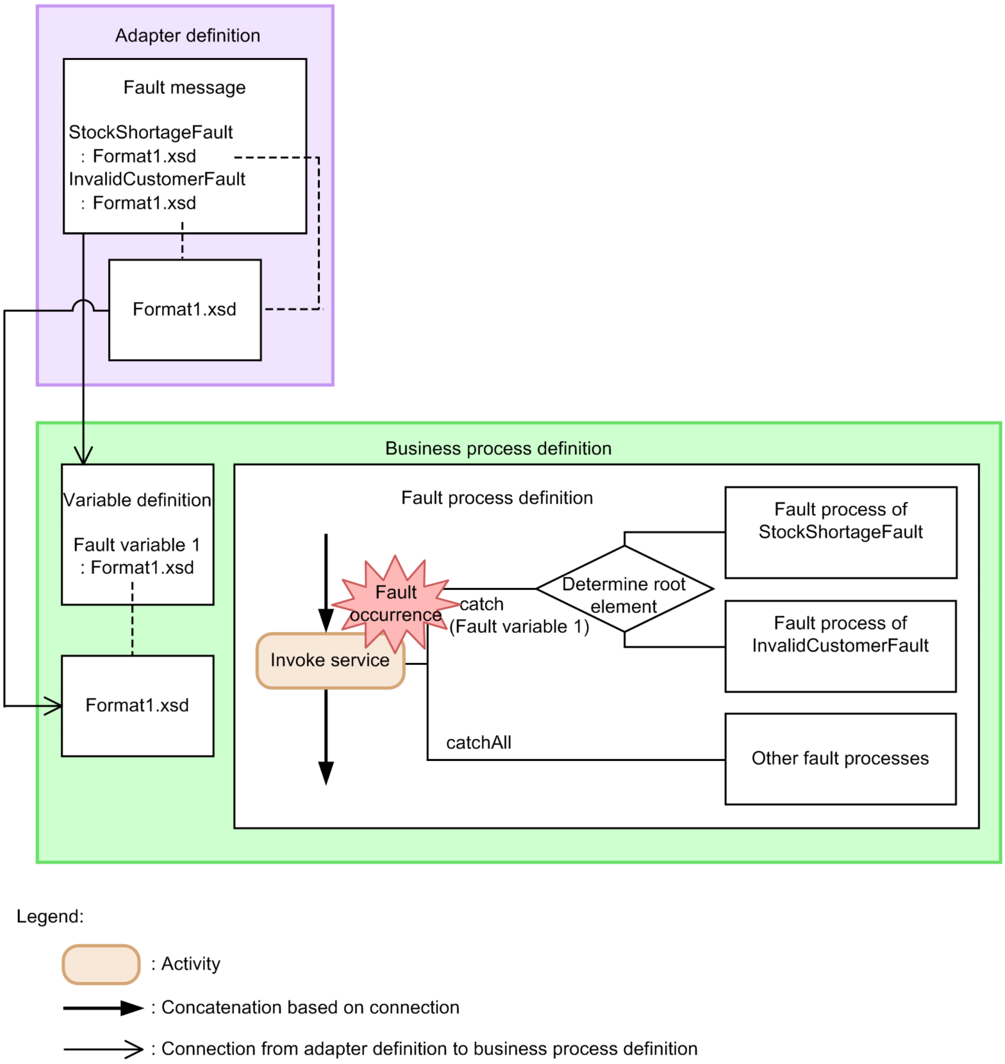
(b) Definition method

Instead of defining variables for each fault, define a single variable that will be common for faults having the same message format. Also specify a single transition destination for fault handling, and allocate a common variable in the **Fault Handler** dialog box. However, for unexpected fault handling, you can additionally set up the transition destination for `catchAll`.

Arrange a switch activity in the transition destination of fault processing, and describe the switch conditions for separating the faults that occur based on the element names of the root.

The following figure shows the schematic diagram when a common variable is allocated:

Figure 5–4: Schematic diagram when a common variable is allocated



An example of describing the switch condition for acquiring the root element name and then judging the match (**Root element judgment** in the figure) is as follows:

```
csc:getVariableData('Fault variable 1','local-name(/*)')="StockShortageFault"
```

In `Fault variable 1`, specify the variable name of the fault common variable, and in `StockShortageFault`, specify the root element name.

5.5 Defining Variables and Correlation Sets

5.5.1 Defining Variables

When defining a business process, you can include characters, numeric values to be included in messages, numeric values that become terms in conditional expressions, or messages as variables. Because variable values can be recorded in a database, they can be managed as execution history.

(1) Variable types

There are two types of variables: global variables and local variables.

- *Global variables*
  A global variable can be referenced from anywhere within a single process instance. Additionally, because the values of variables that have been processed are recorded in a database, they can be used for analyzing the progress of the entire business process. Variable names must be unique within each process.
- *Local variables*
  A local variable can be referenced with the declared scope. When the declared scope is terminated, the values of the variables are also deleted from a database. Therefore, local variables can be used for temporarily referencing values such as condition determination during process execution. Variable names must be unique within each scope, but an identical variable name can be declared in a different scope. However, when a variable is to be referenced, the variable that is declared in the innermost scope at the referencing location is referenced.

(2) Variable formats

In variables, you can define the values of following types. You can save the variable only for the value of defined type.

- `boolean` type: positive or negative (`true` or `false`)
- `numeric` type: Numeric value (floating decimal of 64 bit)
- `string` type: Character string
- `message` type: Values that substitute variable conform to the format definition related to variables. Define the following formats, depending on the types of variable:
  XML: Define the format in XML schema
  non-XML: Define the format in binary format definition file
  any: Do not define the format

(3) Activities to which variables can be assigned

You can assign variables to the activities and items shown in the following table.

Table 5–4: Activities and items to which variables can be assigned

Activity	Item	Allocated contents	Remarks
Receive activity	Request message	Variable of message type for referencing the request message received from the service requester	It is used in also in specification of correlation set.
Reply activity	Response message	Variable of message type for referencing response message or fault message to the service requester	
Invoke service activity (synchronous)	Request message	Variable of message type for referencing the request message when invoking the synchronous service	
	Response message	Variable of message type for referencing the response message received from the invoked synchronous service	

Activity	Item	Allocated contents	Remarks
Invoke service activity (synchronous)	Fault message	Variable of message type for referencing response message to be received when fault occurs in the invoked synchronous service	--
Invoke service activity (asynchronous)	Request message	Variable for message type for referencing the request message when invoking the asynchronous service	It is used also in the specification of correlation set
Invoke java activity	Argument	Variable for referencing the value, which serves as argument when invoking the user-created Java class	--
	Return value	Variable for referencing the return value from the invoked Java class	--
Data transformation activity	Transformation source data	This is variable of message type for referencing the transformation source data when transforming the data in data transformation activity and you can specify multiple variables	--
	Transformation destination data	Variable of the message type for referencing the transformation destination data when transforming the data in the data transformation activity	--
Assign activity	Copy source	Variable for referencing the data that serves as copy source when copying the data in assign activity	--
	Copy destination	Variable for referencing the data that serves as copy destination when copying the data in assign activity	--
Throw activity	Fault message	Variable of message type for referencing the fault message, when throwing fault in the throw activity	--
Standby activity	Standby time	Variable for referencing the value of standby time	--
Validate activity	Variables to be validated	Variable to be validated	--
While activity	Repetition condition	Variable for referencing the value of while loop variable or end condition expression	--
	Repetition list	Variable for referencing in the expression for acquiring the repetition list	--
	Repetition element variable	Variable for storing the repetition element	--
Switch activity	Distribution condition	Variable for referencing the value that serves as item of distribution condition expression	--

(Legend)

--: Not applicable

(4) Showing variables and acquiring paths

Clicking the **Show Tree** button in the Variable-Correlation Set dialog box opens XML schema viewer Parameters. You can show XML schemas that match variables in a tree view. When you choose a variable displayed in the tree view, the path to the chooseed variable is displayed in Chooseed path as an absolute path from the root element.

When you choose and right-click a variable and choose Get Path, you can obtain the absolute path from the root element. You can paste the acquired path in the desired location.

! Important note

Take note of the following points when operating the **Show Variables** dialog box:

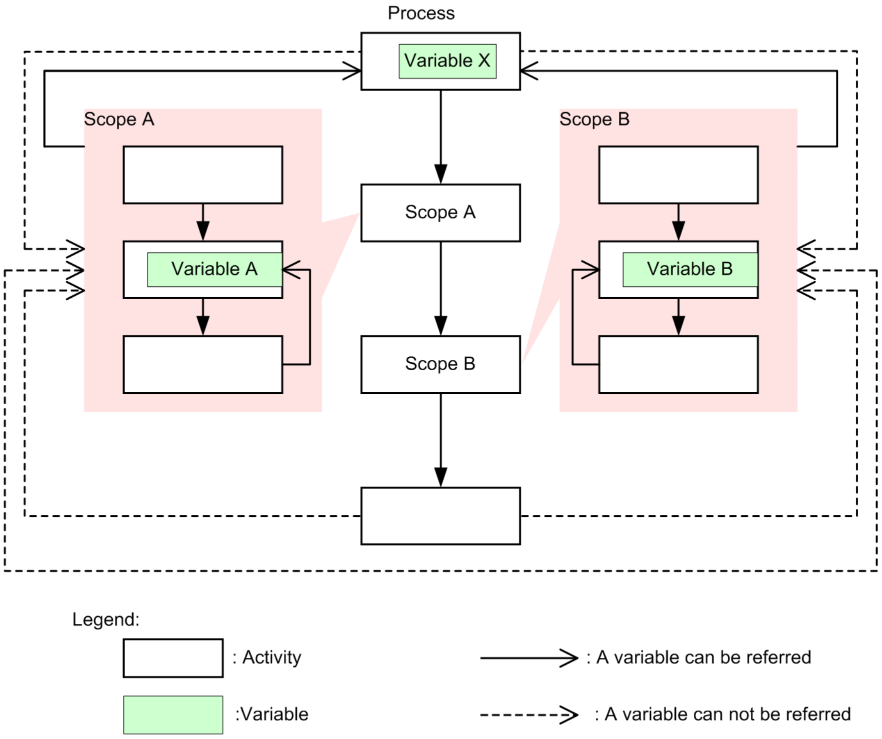
- You cannot edit the XML schema displayed in the **Show Variables** dialog box.
- You can choose only one variable in the Tree view.
- If elements re-enter the XML schema of the variables that you want to display in the **Show Variables** dialog box, elements from the second hierarchy onwards will not be displayed.
- The `substitutionGroup` attribute of the XML schema will be ignored.

(5) Notes on using variables

(a) Use of variables when scope activities are used

You can add or edit variables that are used by activities inside business processes or scopes. In this case, variables declared outside a scope can be referenced from inside the scope, but variables declared inside a scope cannot be referenced from outside the scope. Note that the same variable names can be declared within different scopes. In the following figure, while Scope A and Scope B can reference variables outside their scope in other parts of the process, the variables of Scope A and Scope B cannot be referenced from the process (from outside their scope).

Figure 5–5: Relationship between a scope and variables



(b) Use of variables when defining fault handling

When defining fault handling for a scope activity, choose the variable to be used as an assign variable in the Fault Handler dialog box. The chooseable variables are those that are defined in the fault-target scope (including a scope activity) or a scope on the outside. However, if a variable having the same name as the variable chooseed in the Fault Handler dialog box is also defined in the fault-target scope, the variable defined in the fault-target scope, and not the variable chooseed in the Fault Handler dialog box, is used during execution, and may result in unintended operations. Therefore, when defining fault handling for a scope activity, ensure that no variable having the same name as the variable chooseed in the Fault Handler dialog box is defined in the fault-target scope.

(c) Message formats used when defining variables

File name of a message format

When defining variables, you can specify definition files in the message format having the same file name but different XML schemas for multiple variables. However, because only one of these XML schemas is used during execution, an unintended operation may occur. Therefore, ensure that the file names in the message format specified when defining variables correspond to XML schemas on a one-to-one basis.

Setting a message format that contains external XML schema references

For specifying a message format that references an external XML schema, you must specify a file for the root schema. The external XML schema file referenced from the root schema is automatically imported.

Format of a message format

The message format used in the variable must satisfy the conditions described in *2.6.5 Scoping of XML schema*. For details about the conditions, see *2.6.5 Scoping of XML schema*.

(d) Modifying the definition information of variables

You can change or delete the definition information (variable name, type, message format and part) of variable and correlation set as and when required.

! Important note

Take note of the following points, when changing the definition information of variables,

- When you change the message format of variables used in data transformation activity, re-define the data transformation. For details on definition when changing the message format, see "*6.3.2 Procedure for defining changed message formats*".
- When you change the name of a variable used in data transformation activity for which data transformation definition file has been defined, warning message is displayed. In that case, mapping related to the changed variable is discarded.
- When you change the variable definition information, you must re-assign to the location where that variable is assigned.

  For example, when you assign variable X to some activity and then change the name of that variable X to variable Y, perform either of the following:

  Re-assign a variable again to the activity

  Re-define variable X.
- You must take care when changing variable name after upgrading the version of a business process. For details, see "*5.9.4(3) Points to be noted when upgrading the version of business process*".

(e) Initializing definition information of variables

When executing a business process, you must set a numeric value or message, which can serve as item of numeric value or condition expression, included in the character or message, in the variable.

When receiving a request from the service requester or receiving a response from the service by Invoke service activity, a value has been entered in the variable. However, in other cases, you must set a value to variable ,(initialize) by using data transformation activity or Switch activity.

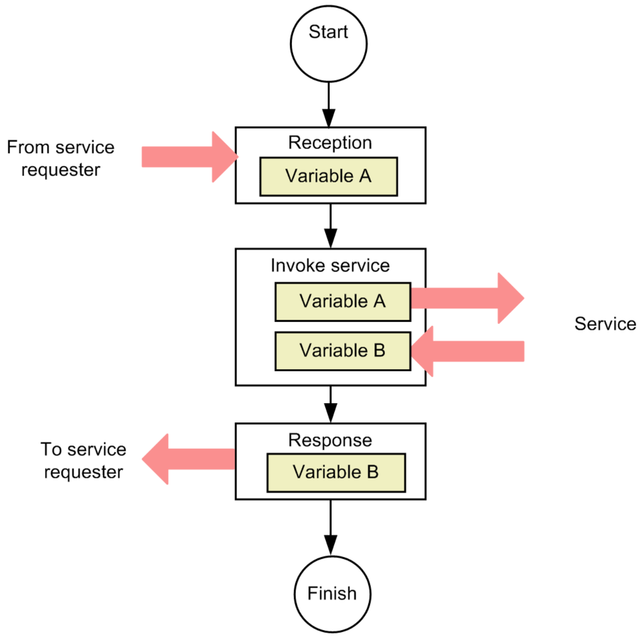
Following section describes the cases when variable initialization is not required:

For the variables specified in the following locations, value has been set before referencing the variable (message) and hence you need not initialize these variables:

- Variable that received a request from the service requester
- Body assigning variable/header assigning variable of the reply message of Invoke service activity
- Assigned variable for return value of Invoke java activity
- Transformation destination variable of data transformation activity
- Copy destination variable of Assign activity
- Assigned variable specified in a fault connection

Following figure shows the example when initializing a variable is not required:

Figure 5–6: Examples when initialing a variable is not required



Variable A:

Request message from the service requester is set as the value of variable. Also, as variable A for which service invoking value has already been set, initializing is not required.

Variable B:

As variable has been set in the reply message from a service, initializing is not required.

Description about the cases when initializing a variable is required, is as follows:

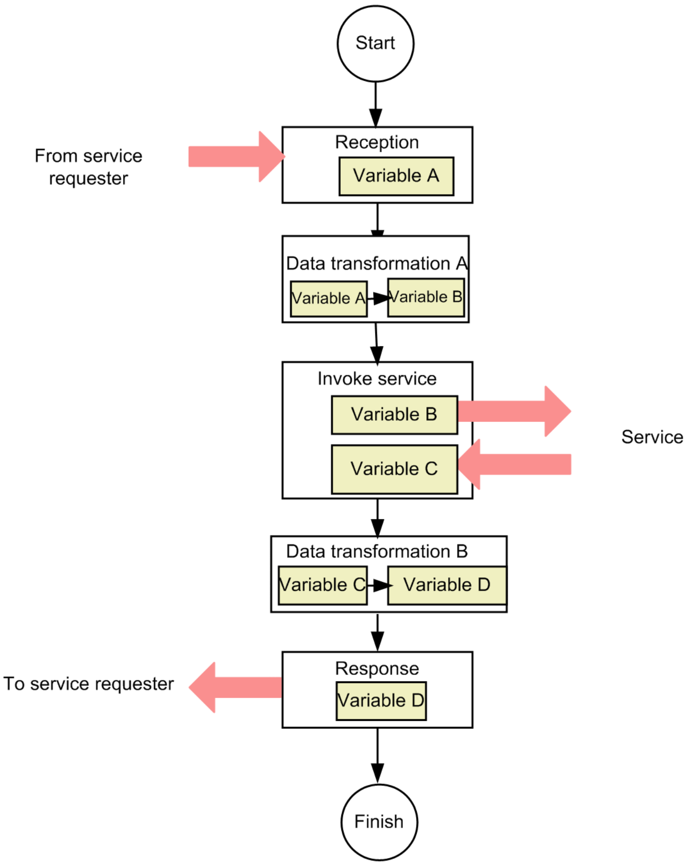
As value is not set for the variable not specified in the location shown when initializing the variable is not required, if you reference such variables in a business process, an error occurs. You must set the value beforehand and initialize the variable.

Following are the locations where variables for which value has not been set, can be specified:

- Body assigning variable/header assigning variable of the request message of Invoke service activity
- Assigned variable for arguments of Invoke java activity
- Transformation source variable o the data transformation activity
- Copy source variable of the Assign activity
- Location where variable name is described in Standby activity, Validate activity, While activity, Switch activity, XPath expression of link connection
- Body assigning variable/header assigning variable of the reply activity

Following figure shows the examples of cases when initializing a variable is required:

Figure 5–7: Example when initializing a variable is required



Variable A:

As a request message from the service requester has been set in the variable, initializing is not required.

Variable B:

Initializing a variable is required. In data transformation A, you can set a value in variable B, based on the information of variable A.

Variable C:

Initializing is not required as the reply message from the service is set in the variable.

Variable D:

Initializing a variable is required. In data transformation B, you can set value to variableD, based on the information of variableC.

(6) Variable definition methods

(a) Defining new variables

The procedure for defining new variables is described below. Note that variables cannot be defined on a canvas inside a while activity.

1. Use one of the following methods to open the Variable-Correlation Set dialog box:
   - Double-click the **Variable-Correlation Set** icon on the canvas in the Business Process Definition screen.
   - Click **Edit** in the dialog boxes for the following activities:
     - Receive activity
     - Reply activity
     - Service invocation activity
     - Java invocation activity

- Data transformation activity
- Assign activity
- Throw activity

2. In the list that is displayed, choose Variable List.
3. Enter a variable name.
4. Choose a variable type from the drop-down list.
5. Perform one of the following operations as needed:

   If the variable type is message type and you want to set the message format specified in the request message, response message, or fault message of a service component to be invoked or user-defined reception to variable:

      Click **Get Format** to view the Get Message Format dialog box. Specify the message format you want to use in the Get Message Format dialog box.

      For details about the input and display items of the **Get Message Format** dialog box, see the manual *Cosminexus Service Platform Reference*.

      Proceed to step 6. after the operations in the Get Message Format dialog box are complete.

   If the variable type is message type, and for the cases other than mentioned above

      Click **...** and set the definition file of the message format that uses the variable.

      For converting an exception for error occurred when invoking service component (Web Services) from service adapter to a fault message in service adapter, click the **...** button and choose `Cosminexus installation directory\CSC\system\msg\cscfault.xsd`.
6. When the variable type is Message, choose the Part Specifications check box.
7. Click Add Line and specify a part name, an expression, and a type.
8. Click Add.

   The added variable is displayed in the variable list.
9. Click OK.

(b) When a message format definition file is modified

If the message format definition file specified during the definition of the variable is modified, the contents of the modified definition file can be committed using the following procedure:

1. Open the Variable-Correlation Set dialog box.

   For details about how to open the Variable-Correlation Set dialog box, see step 1 in *5.5.1(6)(a) Defining new variables*.
2. Perform one of the following operations:

   If the variable type is message type and you want to set the message format specified in the request message, response message, or fault message of a service component to be invoked or user-defined reception to variable:

      Click **Get Format** to view the Get Message Format dialog box. Specify the message format you want to use in the Get Message Format dialog box.

      For details about the input and display items of the **Get Message Format** dialog box, see the manual *Cosminexus Service Platform Reference*.

      Proceed to step 3. after the operations in the Get Message Format dialog box are complete.

   If the variable type is message type, and for the cases other than mentioned above

      Click **...** and set the definition file of the message format that uses the variable.
3. Click **Update** button.

(7) Output of message format definition file

Description about a method to output a message format definition file is as follows:

When you want to set the message format of variable set in the activity at copy source, to the variable of activity at copy destination, you can output a message format definition file in the Variable/correlation set list dialog.

Format of the message format definition file that you can output, is either of the following formats:

- Extension of the message format definition file: xsd (variable type: XML)

- Extension of the message format definition file: fdx (variable type: non-XML)

Procedure to output the message format definition file is as follows:

1. Click [Output] button of the Variable/correlation set list dialog.
   Dialog for specifying the output destination is displayed.
2. Specify an output destination and click [OK] button.
   The message format definition file is output to the specified folder. In case of message type (XML) variable formed from definition file having multiple output targets, output file name is automatically assigned.

5.5.2 Defining Correlation Sets

A *correlation set* is a character string that is used to uniquely identify a request message sent from a service requester.

(1) Identifying process instancesbased oncorrelation sets

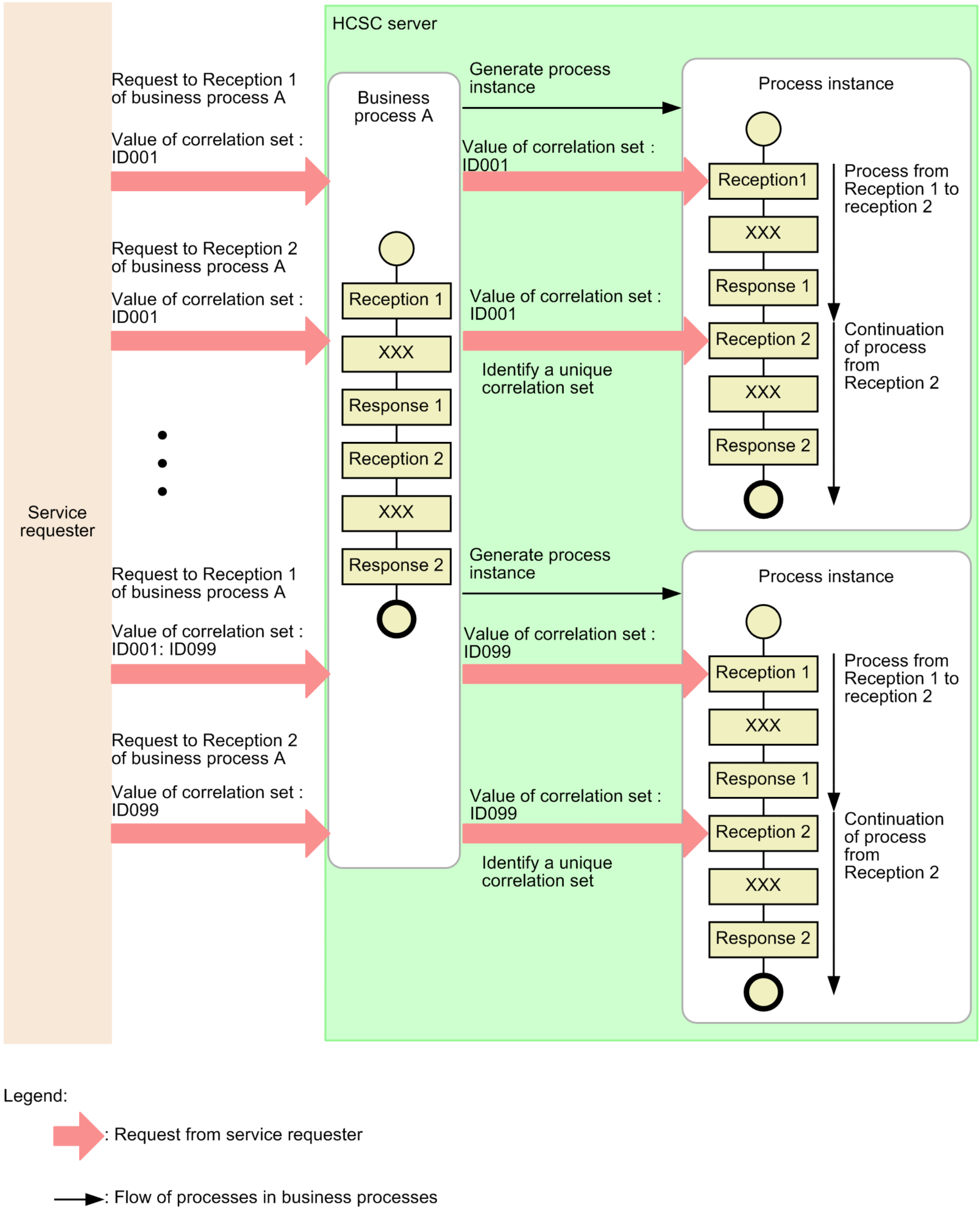
The business process generates a process instance for each request to a receive activity for which **yes** is specified for **Instance generation** in the Receive Activity dialog box.

If the HCSC server contains multiple process instances that can receive request messages from service requesters, the HCSC server identifies each process instance by using the correlation set value contained in each request message.

For example, assume that two receive activities that receive requests from service requesters are defined in a business process. A request is sent to the first receive activity, and the second receive activity places the business process in a wait request status. In this case, the value of the correlation set included in a request message to the first receive activity must be included in a request message to the second receive activity. Specifying the same value for these correlation sets enables the system to identify that these two requests go to the same process instance. This enables subsequent processing to continue after the second receive activity.

The following figure shows an example of identifying process instances based on correlation sets.

Figure 5–8: Example of identifying process instances based on correlation sets


HCSC server
Service requester
Request to Reception 1 of business process A
Value of correlation set : ID001
Request to Reception 2 of business process A
Value of correlation set : ID001
Request to Reception 1 of business process A
Value of correlation set : ID001: ID099
Request to Reception 2 of business process A
Value of correlation set : ID099
Business process A
Reception 1
XXX
Response 1
Reception 2
XXX
Response 2
Generate process instance
Value of correlation set : ID001
Value of correlation set : ID001
Identify a unique correlation set
Generate process instance
Value of correlation set : ID099
Value of correlation set : ID099
Identify a unique correlation set
Process instance
Reception1
XXX
Response 1
Reception 2
XXX
Response 2
Process from Reception 1 to reception 2
Continuation of process from Reception 2
Process instance
Reception 1
XXX
Response 1
Reception 2
XXX
Response 2
Process from Reception 1 to reception 2
Continuation of process from Reception 2
Legend:
: Request from service requester
: Flow of processes in business processes

To implement a business process that has multiple receive activities, which of the values included in a request message is to be used for the correlation set must be defined in the business process. When designing a business process and service requester, a unique key value identifying each process instance must be set in a request message from the service requester.

A correlation set can be configured from a single part contained in a request message or by linking multiple parts contained in a request message.

When using correlation sets in a business process, you can search the execution history of process instances by using the value of the correlation set as a key. You can search the execution history to confirm the execution status of a particular request. For details about the execution history of process instances, see *6.1 Management of execution log of process instances* in the *uCosminexus Service Platform Setup and Operation Guide*.

(2) Activities to which correlation sets can be assigned

In a business process, you can specify correlation sets for receive activities, reply activities, and invoke service activities. You can also specify multiple correlation sets for a single message.

Whether a correlation set must be specified and what role it plays differ depending on the activity type.

(a) Receive activities and reply activities

For receive activities and reply activities, the specification requirement and contents of correlation sets differ depending on the instance generation settings, and on whether the assigned correlation set is to be initialized.

You can specify instance generation in the Receive Activity dialog box. You can use the Allocating Correlation Set Group dialog box to specify whether to initialize correlation sets.

For details about the Receive Activity dialog box, see *1.4.7 Receive Activity Dialog* in the manual *uCosminexus Service Platform Reference Guide*. For details about the Allocating Correlation Set Group dialog box, see *1.4.3 Allocating Correlation Set Group Dialog* in the manual *uCosminexus Service Platform Reference Guide*.

Table 5–5: Specification requirement and contents of correlation sets (receive and reply)

Activity	Instance generation	Initialization specification	Specification requirement	Maximum number of specifications	Contents of specified correlation set
Receive	yes	yes	Optional	1	When a process instance is generated, a correlation set for identifying process instances is generated from a receive message defined as a variable.
		no	Cannot be specified	--	--
	no	yes	Optional	1	After a generated process instance is retrieved, a new correlation set corresponding to the retrieved process instance is generated from a receive message defined as a variable.
		no	Required	1	The specified correlation set is used as a correlation set within the receive message defined as a variable, to retrieve generated process instances.
Reply	--	yes	Optional	1	When returning a reply to a service requester, a new correlation set is generated from a reply message defined as a variable.
	--	no	Optional	1	When returning a reply to a service requester, the system checks whether the correlation set value within the reply message defined as a variable matches the correlation set value of the process instance.

Legend:

--: Not applicable

(b) Invoke service activities

For invoke service activities, the specification requirement and contents of a correlation set differ depending on the pattern settings for the assigned correlation set and whether the correlation set is to be initialized.

You can use the Allocating Correlation Set Group dialog box to specify pattern settings for the assigned correlation set and whether to initialize the correlation set.

For details about the Allocating Correlation Set Group dialog box, see *1.4.3 Allocating Correlation Set Group Dialog* in the manual *uCosminexus Service Platform Reference Guide*.

Table 5–6: Invoke service activities to which correlation sets can be assigned

Activity	Pattern specification	Initialization specification	Specification requirement	Maximum number of specifications	Contents of specified correlation set
Invoke service	out	yes	Optional	1	When sending a message to a service component, a new correlation set is generated from a send message defined as a variable.
		no	Optional	1	When sending a message to a service component, the system checks whether the correlation set value within the send message defined as a variable matches the correlation set value of the process instance.
	in	yes	Optional	1	When a reply is received from a service component, a new correlation set is generated from a reply message defined as a variable.
		no	Optional	1	When a reply is received from a service component, the system checks whether the correlation set value within the reply message defined as a variable matches the correlation set value of the process instance.
	out-in	yes	Optional	1	During transmission: When sending a message to a service component, a new correlation set is generated from a send message defined as a variable. During reception: When a reply is received from a service component, the system checks whether the correlation set value within the reply message defined as a variable matches the correlation set value of the process instance.
		no	Optional	1	During transmission: When sending a message to a service component, the system checks whether the correlation set value within the send message defined as a variable matches the correlation set value of the process instance. During reception: When a reply is received from a service component, the system checks whether the correlation set value within the reply message defined as a variable matches the correlation set value of the process instance.

(3) Notes on using correlation sets

(a) Use of correlation sets when scope activities are used

You can add or edit the correlation sets to be used by activities within a business process or scope, in the same way you can add or edit variables. For details, see *5.5.1(5)(a) Use of variables when scope activities are used*.

(b) Changing the definition information for correlation sets

If necessary, you can change or delete the definition information for correlation sets.

For notes on changing the variable definition information used in correlation set definitions, see *5.5.1(5)(d) Modifying the definition information of variables*.

Care is required when changing the correlation set after upgrading the version of a business process. For details, see *5.9.4(3) Points to be noted when upgrading the version of business process*.

! Important note

If you change the correlation set definition information, you might need to reallocate the correlation set to the sections where it was allocated.

For example, if correlation set A is allocated to a certain activity, and then the name of correlation set A is changed to correlation set B, perform one of the following operations:

- Reallocate the correlation set to the activity.
- Redefine correlation set A.

(c) Scope of correlation set

If a correlation set is declared by a global variable of a business process, the correlation set is valid until the processing of the applicable process instance finishes.

If a correlation set used by an activity in a scope is also declared, the correlation set is valid until the processing within the scope finishes.

The correlation set becomes invalid when the processing finishes.

(4) Correlation set definition methods

(a) Defining new correlation sets

Before defining a correlation set, you must define variables and specify some of these variables to be used as part of the correlation set. The procedure for defining new correlation sets is described below. Note that correlation sets cannot be defined on the canvas within a while activity.

1. Use one of the following methods to open the List Of Variables And Correlation Sets dialog box:
   - In the Define Business Process window, double-click the **Variable-Correlation** icon on the canvas.
   - Click **Edit** in the dialog boxes for the following activities:
     - Receive activity
     - Reply activity
     - Invoke service activity
2. In the list, select **CorrelationSet List**.
3. Enter a correlation set name.
4. Click **Add Line** and add a variable name and a part name in **Acquisition Part** to specify the part to be used as the correlation set.

   For correlation sets, you can specify a variable of the message type (XML) and a part name specifying a type other than the message type.

   If multiple message type variables are used, select the variables and part names to be used from the drop-down list.

   To link multiple parts and set the linked character string as the correlation set value, click **Add Line** again and add the variables and part names to be used.
5. Click **Add**.

   The added correlation set is displayed in the **Correlation Set List**.
6. Click **OK**.

(b) Specifying correlation sets from activities

The procedure for specifying correlation sets from the activities shown in *Table 5-6 Invoke service activities to which correlation sets can be assigned* is described as follows:

1. Click **Setting** in the dialog box for each activity.

The Allocating Correlation Set Group dialog box appears.

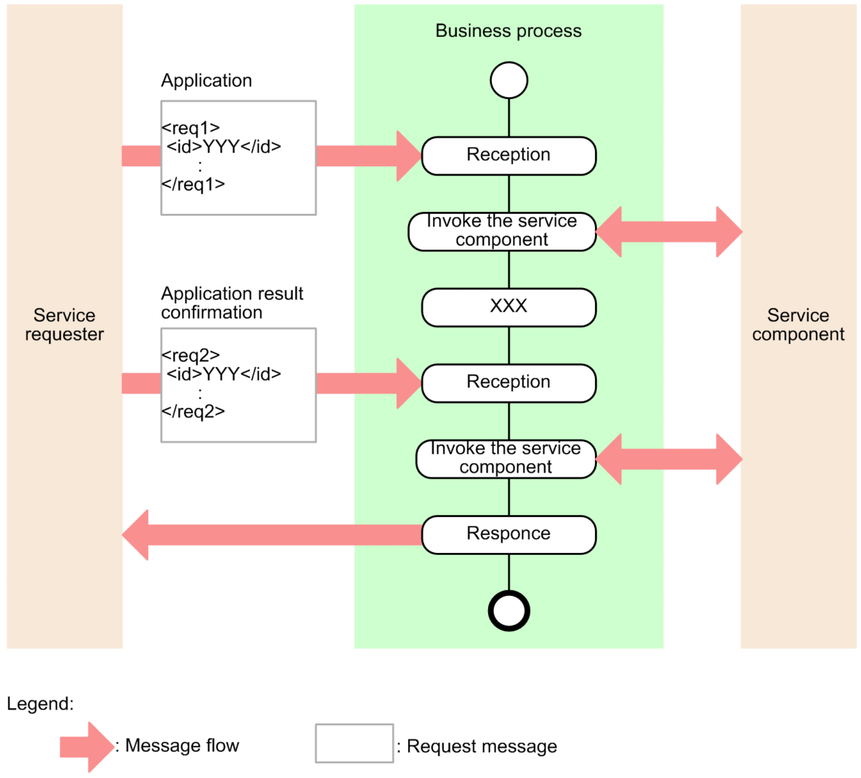
2. Select the correlation set to be assigned to the activity from the **Correlation set** drop-down list.
3. If you cannot find the desired correlation set in the **Correlation set** drop-down list, click **Edit**.
4. The List Of Variables And Correlation Sets dialog box appears. Add a correlation set.
5. In the Allocating Correlation Set Group dialog box, specify whether to initialize the correlation set.

(5) Definition example that uses correlation sets

The process of defining a business process that uses correlation sets is explained by using an example.

The example in the following figure uses a business process that asynchronously receives application requests and application result confirmation requests.

Figure 5–9: Example of the business process to be defined



The contents of the variables, correlation sets, and receive activities for defining the business processes shown in the above figure are described below.

Defining the variables and correlation sets

Define the variables and correlation sets to be used, as shown in the tables below. You can define variables and correlation sets in the List Of Variables And Correlation Sets dialog box. For details about the List Of Variables And Correlation Sets dialog box, see *1.4.1 List Of Variables And Correlation Sets Dialog* in the manual *uCosminexus Service Platform Reference Guide*.

Table 5–7: Definition contents of variables for application request messages (example)

Setting items in the List Of Variables And Correlation Sets dialog box (for displaying variable information)	Values specified
Variable name	ApplicationMessage
Type	XML
Message format	*application-message*.xsd

Setting items in the List Of Variables And Correlation Sets dialog box (for displaying variable information)		Values specified
Part specifications	**Part name**	ApplicationID
	Expression	`/req1/id`
	Type	`string`

Table 5–8: Definition contents of variables for application result confirmation request messages (example)

Setting items in the List Of Variables And Correlation Sets dialog box (for displaying variable information)		Values specified
Variable name		ApplicationConfirmationMessage
Type		`XML`
Message format		*application-confirmation-message*.xsd
Part specifications	**Part name**	ApplicationID
	Expression	`/req2/id`
	Type	`string`

Table 5–9: Definition contents of correlation sets (example)

Setting items in the List Of Variables And Correlation Sets dialog box (for displaying correlation set information)		Values specified
Correlation set name		ApplicationCorrelationID
Acquisition Part	**Variable name**	ApplicationMessage#
	Part name	ApplicationID

#

For **Variable name**, you can specify either the application message or application confirmation message.

Defining the receive activities

Define the receive activities as shown in the tables below. You can define receive activities in the Receive Activity dialog box. For details about the Receive Activity dialog box, see *1.4.7 Receive Activity Dialog* in the manual *uCosminexus Service Platform Reference Guide*.

Table 5–10: Definition contents of the receive activities for receiving application requests (example)

Setting items in the Receive Activity dialog box		Values specified
Activity name		ApplicationReception
Operation name		`request`
Allocated variable		ApplicationMessage
Correlation set group	**Correlation set name**	ApplicationCorrelation ID
	Initialization	`yes`
Communication model		`Async`
Instance generation		`yes`

Table 5–11: Definition contents of the receive activities for receiving application result confirmation requests (example)

Setting items in the Receive Activity dialog box	Values specified
Activity name	ConfirmationReception

Setting items in the Receive Activity dialog box		Values specified
Operation name		confirm
Allocated variable		ApplicationConfirmationMessage
Correlation set group	**Correlation set name**	ApplicationCorrelation ID
	Initialization	no
Communication model		Synch
Instance generation		no

5.6 Defining Activities

You need to deploy and link activities on the canvas and define their details.

The following table describes the activities that can be defined and their definition contents.

Table 5–12: Activities that can be defined

Activity	Definition
Start	Indicates the start of a business process.
Receive#	Defines an interface for receiving a request message from a service requester.
Reply#	Defines an interface for returning a response or fault to request messages received synchronously from a service requester.
Invoke service#	Defines the transmission of a request message to a defined HCSC component.
Invoke Java	Defines the invocation of a Java class that implements a dedicated interface.
Data transformation	Defines data transformation processing.
Assign	Creates definitions for executing the following processes: • Assigning a variable (basic type or message type) to another variable • Assigning part of a variable to another variable • Creating a value (numeric value, character string, or true/false value) and assigning it to a variable
Empty	Defines that the activity does nothing even if executed.
Throw	Defines a fault notification to a higher-order scope activity.
Standby#	Defines the process for putting a process flow in standby mode at regular intervals or until a certain time limit.
Validate	Validates messages transferred within a business process.
Scope	Defines a process flow consisting of one or more activities as a single unit (scope).
While	Defines repetitive processing of one or more activities under specified conditions.
Switch (Start)	Defines switching of the processing of a business process according to the result of a conditional expression. The switching start and end points must be defined.
Switch (End)	
Flow (Start)	Defines the division of a processing flow into multiple sequences and the concurrent execution of these sequences (parallel processing of a flow). The flow start and end points must be defined.
Flow (End)	
End	Indicates the end of a business process.

#

If status persistence is specified for a business process, the processing of receive, reply, invoke service, and standby activities determines the timing of finalizing the statuses of these activities (transaction start and commitment timing).

For details about transaction start and commitment timing, see *3.4 Transaction of a business process* in the manual *Service Platform Overview*.

The following subsections provide details of the definition of each activity.

Reference note

Any activities other than start and end activities can be copied to any locations on the business process screen while their definition contents are retained.

However, variable inconsistencies might occur between activities during activity copying (for example, the variables to be referenced might not exist at the copy destination). In such cases, output the message format of the variables from the copy source and set the format to the variables at the copy destination. For details about how to output the message format set for variables, see *5.5.1(7) Output of message format definition file*.

5.6.1 Start Activity

This activity indicates the start of a business process. Only a single start activity is deployed on the canvas. This activity cannot be deleted.

There is no content to be defined for the start activity.

5.6.2 Receive Activity

This activity defines an interface required for a business process to receive a request message from a service requester. If a single business process contains multiple operations to be open to service requesters, a receive activity must be defined for each operation.

You can define the details of receive activities in the Receive Activity dialog box.

(1) Definition procedure

To define a receive activity:

1. Deploy a receive activity on the canvas.
   For details about how to deploy activities, see *5.4.1 Deploying Activities*.
2. Use one of the following methods to open the Receive Activity dialog box:
   - Double-click a receive activity on the canvas.
   - Select and right-click a receive activity on the canvas, and then select **Setting**.

   The Receive Activity dialog box appears.
3. Enter the necessary information in the Receive Activity dialog box.
   For details about the display and input contents of the Receive Activity dialog box, see *1.4.7 Receive Activity Dialog* in the manual *uCosminexus Service Platform Reference Guide*.
   - When editing the contents of the variable to be specified in **Body allocated variable**

     Click **Edit**. The List Of Variables And Correlation Sets dialog box appears. In this dialog box, you can edit the contents of the variable. In this item, you can specify a variable of the message type (`XML`, `non-XML`, or `any`). For details about the List Of Variables And Correlation Sets dialog box, see *1.4.1 List Of Variables And Correlation Sets Dialog* in the manual *uCosminexus Service Platform Reference Guide*.

     Note that if you have already defined a user-defined reception that has relevant operations, you can set the message format for the user-defined reception to a variable. Specify this setting in the Take In Message Format dialog box. The Take In Message Format dialog box appears when you click **Take In** in the List Of Variables And Correlation Sets dialog box.

     If you specify an operation name in the Receive Activity dialog box beforehand, the message format for the relevant operation within the user-defined reception is already selected when the Take In Message Format dialog box appears.

     For details about the Take In Message Format dialog box, see *1.4.5 Take In Message Format Dialog* in the manual *uCosminexus Service Platform Reference Guide*.
   - When specifying settings in **Header allocated variable**

     Click **Setting**. The Header allocated variable dialog box appears. In this dialog box, you can specify a variable to be allocated to the header. In this item, you can specify a variable of the message type (`XML`). Note that you cannot specify a variable of the message type (`non-XML` or `any`). For details about the Header allocated variable dialog box, see *1.4.2 Header allocated variable dialog* in the manual *uCosminexus Service Platform Reference Guide*.
   - When specifying settings in **Allocating Correlation Set Group**

     Click **Setting**. The Allocating Correlation Set Group dialog box appears. In this dialog box, you can specify the correlation sets to be allocated. For details about the Allocating Correlation Set Group dialog box, see *1.4.3 Allocating Correlation Set Group Dialog* in the manual *uCosminexus Service Platform Reference Guide*.

4. Click **OK**.

(2) Notes on definition

- Specify activity names that are unique within a business process (and also within a scope).
- An activity name must be no more than 64 bytes.
- Do not specify any control characters in the input fields of the dialog box.
- Specify operation names that are unique within a business process (and also within a scope).
- When a synchronous receive activity is specified, specify a reply activity that corresponds to the operation of the receive activity.
- When a synchronous receive activity is specified, do not specify another receive activity between the receive activity and the reply activity corresponding to the operation of that receive activity.
- Specify all receive activities generating instances so that they use the same correlation set.
- Create at least one receive activity that generates instances.
- Care is required when changing the definition information for a receive activity after upgrading the version of a business process. For details, see *5.9.4(3) Points to be noted when upgrading the version of business process*.

5.6.3 Reply Activity

This activity defines an interface for returning a response or fault to request messages received synchronously from service requesters by the business process. A reply activity must be defined if synchronous reception is defined for the communication model of the corresponding receive activity.

You can define the details of reply activities in the Reply Activity dialog box.

(1) Definition procedure

To define a reply activity:

1. Deploy a reply activity on the canvas.
   For details about how to deploy activities, see *5.4.1 Deploying Activities*.
2. Use one of the following methods to open the Reply Activity dialog box:
   - Double-click a reply activity on the canvas.
   - Select and right-click a reply activity on the canvas, and then select **Setting**.

   The Reply Activity dialog box appears.
3. Enter the necessary information in the Reply Activity dialog box.
   For details about the display and input contents of the Reply Activity dialog box, see *1.4.8 Reply Activity Dialog* in the manual *uCosminexus Service Platform Reference Guide*.
   - When editing the contents of the variable to be specified in **Body allocated variable**
     Click **Edit**. The List Of Variables And Correlation Sets dialog box appears. In this dialog box, you can edit the contents of the variable. In this item, you can specify a variable of the message type (`XML`, `non-XML`, or `any`). For details about the List Of Variables And Correlation Sets dialog box, see *1.4.1 List Of Variables And Correlation Sets Dialog* in the manual *uCosminexus Service Platform Reference Guide*.

     Note that if you have already defined a user-defined reception that has relevant operations, you can set the message format for the user-defined reception to a variable. Specify this setting in the Take In Message Format dialog box. The Take In Message Format dialog box appears when you click **Take In** in the List Of Variables And Correlation Sets dialog box.

     If you specify an operation name in the Reply Activity dialog box beforehand, the message format for the relevant operation within the user-defined reception is already selected when the Take In Message Format dialog box appears.

     For details about the Take In Message Format dialog box, see *1.4.5 Take In Message Format Dialog* in the manual *uCosminexus Service Platform Reference Guide*.
   - When specifying settings in **Header allocated variable**

Click **Setting**. The Header allocated variable dialog box appears. In this dialog box, you can specify a variable to be allocated to the header. In this item, you can specify a variable of the message type (`XML`). Note that you cannot specify a variable of the message type (`non-XML` or `any`). For details about the Header allocated variable dialog box, see *1.4.2 Header allocated variable dialog* in the manual *uCosminexus Service Platform Reference Guide*.

- When specifying settings in **Allocating Correlation Set Group**

  Click **Setting**. The Allocating Correlation Set Group dialog box appears. In this dialog box, you can specify the correlation sets to be allocated. For details about the Allocating Correlation Set Group dialog box, see *1.4.3 Allocating Correlation Set Group Dialog* in the manual *uCosminexus Service Platform Reference Guide*.

4. Click **OK**.

(2) Notes on definition

- Specify activity names that are unique within a business process (and also within a scope).
- An activity name must be no more than 64 bytes.
- Do not specify any control characters in the input fields of the dialog box.
- An operation name must be specified.
- If a fault name is specified, the fault returns.
- Allocated variables must be specified.
- For reply activities having the same operation name, specify the same allocated variable name.
- Specify a single allocated variable that corresponds to the fault names of the reply activities of the same operation.
- When a synchronous receive activity is specified, specify a reply activity that corresponds to the operation of the receive activity.
- Do not specify another reply activity between a receive activity and the reply activity that corresponds to the operation of the receive activity.
- Specify a variable of the message type (`XML`) for the allocated variable that is set to allocate fault handling. Note that you cannot specify a variable of the message type (`non-XML` or `any`).
- Care is required when changing the definition information for a reply activity after upgrading the version of a business process. For details, see *5.9.4(3) Points to be noted when upgrading the version of business process*.

5.6.4 Service Invocation Activity

This activity defines transmission of a request message to an already defined HCSC component.

You define the details of service invocation activities in the Invoke Service Activity dialog box.

(1) Definition procedure

To define a receive activity:

1. Deploy a invoke service activity on the canvas.

   For details about how to deploy activities, see *5.4.1 Deploying Activities*.

2. Use one of the following methods to open the Invoke Service Activity dialog box:

   - Double-click a invoke service activity on the canvas.
   - Choose and right-click a invoke service activity on the canvas, and choose Setting.

   The Invoke Service Activity dialog box opens.

3. Enter the necessary information in the Invoke Service Activity dialog box.

   For details on display/input contents of [Invoke service activity] dialog, see "1.4.9 Invoke service activity dialog" in the "Service Platform Reference Guide".

   - When editing the contents of variable to be set in [Body assigned variable for request message] and [Body assigned variable for reply message]

Click [Edit] button. You can edit the contents of variable, in the displayed [Variable/correlation set list] dialog. In this item, you can set the variable of the message type (XML, non-XML or any). For details on [Variable/correlation set list] dialog, see "1.4.1 Variable/correlation set list dialog" in the "Service Platform Reference Guide".

When setting a message format defined in invoked service components, you can set by using [Incorporate a message format] dialog. For the procedure to set the message format defined in service components invoked by using [Incorporate a message format] dialog, see "Points" described after this procedure.

For details on [Incorporate a message format], see "1.4.5 Incorporate a message format dialog" in the "Service Platform Reference Guide".

- When setting a variable in [Header assigned variable for request message] and [Header assigned variable in reply message]

  Click [Settings] button. Set the assigned variable in the displayed [Header assigned variable] dialog. In this item, you can set the variable of message type (XML). You cannot set the variable of message type (non-XML or any). For details on [Header assigned variable] dialog, see "1.4.2 Header assigned variable dialog" in the "Service Platform Reference Guide".

- When setting the [Assigned correlation sets]

- When setting a [Assigned correlation sets]

  Click [Settings] button. Set up the assigned correlation set in the displayed [Assigned correlation sets] dialog. For details on the [Assigned correlation sets], see "1.4.3 Assigned correlation sets dialog" in the "Service Platform Reference Guide".

4. Click OK.

Tip

How to set the message format defined in the service component to be invoked to variable is explained below. Note that the following steps are explained for the request messages, but the steps for the response messages are also same:

1. Specify the service name and operation name to be invoked in the Invoke Service Activity dialog box.
2. Click **Edit** in Variable allocated for Request Message.
   The Variable-Correlation Set dialog box opens.
3. Specify the variable name.
4. Click **Get Format**.
   The Get Message Format dialog box opens.
   The values specified in step 2. are set in the service name and operation name. Also, Request Message is set in Message Type.
5. Specify the message format name in the Message Format.
6. Click **OK**.
   The Get Message Format dialog box closes and the user returns to Variable-Correlation Set dialog box.
7. Click **Add**.
   The defined variable is added to the Variable List.
8. Choose the added variable from the variable list.
9. Click **OK**.
   The Variable-Correlation Set dialog box closes and the user returns to Invoke Service Activity dialog box.

(2) Notes on definition

- Specify activity name such that it is unique in the business process (and also within scope).
- Set defined HCSC components and their operation in the service name and operation name. Also, match the communication model of the set operation with the communication model of the defined operation.
- Specify activity name within 64 bytes.
- Do not specify a control character in the input field of dialog.
- Set the invoked service name.
- In case of synchronous invoking, set the assigned variables for reply message without fail.
- In case of non-persistence business process, do not set the asynchronous invoke service activity.

- Match the message type variables)XML, non-XML, any) specified in body assigned variable with the message type of invoked HCSC component. In case of mismatch, error occurs in invoked HCSC components, except for the case of invoking a business process from the business process definition.

5.6.5 Invoke Java Activity

This activity defines the invocation of a Java class that implements a dedicated interface (`jp.co.Hitachi.soft.csc.bp.receiver.ejb.CustomClassInterface`).

You can define the details of invoke Java activities in the Invoke Java Activity dialog box.

(1) Preparations before defining invoke Java activities

Before defining invoke Java activities, you must make the following preparations:

Creating an HCSCTE project

The Java class to be specified in the Invoke Java Activity dialog box is saved in the `src` directory of the HCSCTE project. Therefore, you must create an HCSCTE project beforehand.

For details about how to create an HCSCTE project, see *3.1 Managing a Project*.

Creating a Java class

Create a Java class to be used by invoke Java activities. For details about the interface of the Java class to be used, see *5.6.5(2) Interface of the Java class to be used*.

! Important note

- Java classes are shared by all HCSC components within a repository. Therefore, if you modify an existing Java class, HCSC components of the pre-modification class might be mixed with HCSC components of the post-modification class. If a Java class is changed, repackage the business process that defines the invoke Java activity for invoking the changed Java class.

  Note that even if an invoke Java activity is deleted, the Java classes used by the invoke Java activity are not deleted. Therefore, separately delete any Java classes that are no longer required.
- Resources that are allocated when a Java class is called remain allocated even after processing finishes. For this reason, if a high load is applied to the entire system, `OutOfMemoryError` might occur due to insufficient Java heap or Perm heap memory. To prevent this problem, you need to implement the processing for releasing resources when `OutOfMemoryError` occurs, and error processing such as rollback.
- A Java class that implements a dedicated interface (`jp.co.Hitachi.soft.csc.bp.receiver.ejb.CustomClassInterface`) must not be included in the container extension library. Java classes are automatically added to the EAR file when business processes are packaged.

Adding a library

To add a library to an EAR file created by packaging business processes, copy the library to the `lib` directory of the HCSCTE project beforehand.

The library copied to the `lib` directory is automatically added to the EAR file when business processes are packaged.

Do not register a directory or a library with any of the following names in the `lib` directory:

- `csbdef.jar`
- `cscbp_ejb.jar`
- `csbjava.jar`

! Important note

Do not delete links to the `src` directory or `lib` directory in the HCSCTE project.

(2) Interface of the Java class to be used

The Java class to be invoked by an invoke Java activity must implement the following interface:

```
package jp.co.Hitachi.soft.csc.bp.receiver.ejb;

public interface CustomClassInterface {
    public Object invoke(
        String processName,
        int version,
        String activityName,
        Object inputData
    ) throws CSBUserException,CSBSystemException;
}
```

The content of the Java class is explained below.

Arguments

Dummy argument name	Explanation
processName	Business process name
version	Business process version
activityName	Activity name
inputData	Variable specified in **Variable for argument** in the Invoke Java Activity dialog box

Return value

The return value is assigned to the variable specified in **Variable for return value** in the Invoke Java Activity dialog box.

Exception

Processing differs depending on whether the exception to be thrown is `CSBUserException` or `CSBSystemException`.

When `CSBUserException` is thrown

If fault handling is specified for the invoke Java activity, the activity defined for fault handling in the fault connection is executed when `CSBUserException` is thrown.

If fault handling is not specified for the invoke Java activity, processing continues assuming that `invokeJavaFault` occurred.

The interface for `CSBUserException` is shown below.

```
package jp.co.Hitachi.soft.csc.bp;

public class CSBUserException extends Exception {
    public CSBUserException() { super(); }
    public CSBUserException(String message) { super(message); }
}
```

When `CSBSystemException` is thrown

When `CSBSystemException` is thrown, the processing is halted assuming that a system exception occurred. In this case, if the process execution status has been set to persistence, rollback is executed.

The interface for `CSBSystemException` is shown as follows:

```
package jp.co.Hitachi.soft.csc.bp;

 public class CSBSystemException extends Exception {
     public CSBSystemException() { super(); }
     public CSBSystemException(String message) { super(message); }
     public CSBSystemException(String message, Throwable cause) {
         super(message, cause);
     }
     public CSBSystemException(Throwable cause) { super(cause); }
 }
```

If an exception other than `CSBUserException` or `CSBSystemException` occurs, the processing is halted. In this case, if the process execution status has been set to persistence, rollback is executed.

Note:

- The following table lists the variable types to be specified in **Variable for argument** and **Variable for return value** in the Invoke Java Activity dialog box, as well as their correspondence to the argument and return value types for this interface:

Variable types specified in "Variable for argument" and "Variable for return value"	Argument and return value types
boolean	java.lang.Boolean
numeric	java.lang.Double
string	java.lang.String
XML	byte[]
non-XML	byte[]
any	byte[]

  If a type other than those listed above is specified, an error (`KDEC20030-E`) occurs.

- The interface of the Java class used is included in `cscbp_ejb.jar`. Therefore, for compilation, you must add `cscbp_ejb.jar` to the Classpath.
- A Java program uses the default constructor to generate an instance every time it is invoked. Therefore, you cannot hold data in instances.

(3) Definition procedure

To define an invoke Java activity:

1. Deploy an invoke Java activity on the canvas.
   For details about how to deploy activities, see *5.4.1 Deploying Activities*.
2. Use one of the following methods to open the Invoke Java Activity dialog box:
   - Double-click an invoke Java activity on the canvas (only if the **Java Edit** menu is inactive).
   - Select and right-click an invoke Java activity on the canvas, and then select **Setting**.

   The Invoke Java Activity dialog box appears.
3. Enter the necessary information in the Invoke Java Activity dialog box.
   For details about the display and input contents of the Invoke Java Activity dialog box, see *1.4.10 Invoke Java Activity Dialog* in the manual *uCosminexus Service Platform Reference Guide*.
   To edit the contents of the variables to be specified in **Variable for argument** and **Variable for return value**, click **Edit**. The List Of Variables And Correlation Sets dialog box appears. In this dialog box, you can edit the contents of the variable. In **Variable for argument** and **Variable for return value**, you can specify a variable of the message type (`XML`, `non-XML`, or `any`). For details about the List Of Variables And Correlation Sets dialog box, see *1.4.1 List Of Variables And Correlation Sets Dialog* in the manual *uCosminexus Service Platform Reference Guide*.
4. Click **OK** in the Invoke Java Activity dialog box.
   The Invoke Java Activity dialog box closes.
5. Double-click an invoke Java activity on the canvas, or right-click it and then select **Java Edit**.
   The Java editor of Eclipse starts.
   The Java editor displays the source code of the class specified in the Invoke Java Activity dialog box. If you are editing the class specified in the Invoke Java Activity dialog box for the first time, a source code template is displayed when the Java editor starts.
6. Use the Java editor to edit the source code of the Java class invoked by the invoke Java activity.
7. Compile the edited source code of the Java class.
8. Save the edited source code and compiled class file, and then exit the Java editor.

(4) Notes on definition

- Specify activity names that are unique within a business process (and also within a scope).
- An activity name must be no more than 64 bytes.
- Do not specify any control characters in the input fields of the dialog box.
- A Java class name must be specified.

5.6.6 Data Transformation Activity

This activity defines data transformation processing.

You can define the details of data transformation activities in the Data Transformation Activity dialog box.

You can also use the Data Transformation Definition screen to create definition information for data transformation methods (as a data transformation definition file).

(1) Definition procedure

To define a data transformation activity:

1. Deploy a data transformation activity on the canvas.
   For details about how to deploy activities, see *5.4.1 Deploying Activities*.
2. Use one of the following methods to open the Data Transformation Activity dialog box:
   - Double-click a data transformation activity on the canvas (only if the **Launch Mapping Definition Editor** menu is inactive).
   - Select and right-click a data transformation activity on the canvas, and then select **Setting**.

   The Data Transformation Activity dialog box appears.
3. Enter the necessary information in the Data Transformation Activity dialog box.
   For details about the display and input contents of the Data Transformation Activity dialog box, see *1.4.11 Data Transformation Activity Dialog* in the manual *uCosminexus Service Platform Reference Guide*.
   To edit the contents of the variables specified in **Source Variables** and **Destination Variable**, click **Edit**. The List Of Variables And Correlation Sets dialog box appears. In this dialog box, you can edit the contents of the variable. In **Source Variables** and **Destination Variable**, you cannot specify a variable of the message type (`any`). Instead, specify a variable of the message type (`XML` or `non-XML`). For details about the List Of Variables And Correlation Sets dialog box, see *1.4.1 List Of Variables And Correlation Sets Dialog* in the manual *uCosminexus Service Platform Reference Guide*.
4. Click **OK** in the Data Transformation Activity dialog box.
   The Data Transformation Activity dialog box closes.
5. Double-click a data transformation activity on the canvas, or right-click it and then select **Launch Mapping Definition Editor**.
   The data transformation definition screen opens.
   If the message format was changed, a dialog box appears to confirm whether to apply the change in the message format. For details, see *6.3.2 Procedure for defining changed message formats*.
6. In the Data Transformation Definition screen, create a data transformation definition file.
   For details about how to create a data transformation definition file, see *6.3 Defining Data Transformation*.
   For details about the Data Transformation Definition screen, see *1.2.5 Data Transformation Definition Window* in the manual *uCosminexus Service Platform Reference Guide*.
   Note that the created data transformation definition file is saved in a temporary directory until the business process is saved. When the business process is saved, the data transformation definition file is stored in the repository.

! Important note

The data transformation definition file specified in **DataTransDefnFile** in the Data Transformation Activity dialog box is saved in the repository. Then, you can change only the part specifications for variables and message formats in the information for the variables specified in **Source Variables** and **Destination Variable**. If you want to change any information other than part specifications for variables and message formats, click **Delete File**.

In this case, even if you click **Delete File**, the data transformation definition file is not deleted from the repository. The data transformation definition file is deleted from the repository when the business process is saved.

(2) Processing when a system exception occurs in an activity

A system exception that occurs in an activity can be transformed into a general fault before being output. For details, see *4.7 General fault for converting system exceptions to faults* in the manual *Service Platform Overview*.

(3) Notes on definition

- Specify activity names that are unique within a business process (and also within a scope).
- An activity name must be no more than 64 bytes.
- Do not specify any control characters in the input fields of the dialog box.
- A destination variable must be specified.
- A source variable must be specified.
- Data transformation definitions must be specified.

5.6.7 Assign Activity

This activity creates definitions for executing the following processes:

- Assigning a variable (basic type or message type) to another variable
- Assigning part of a variable to another variable
- Creating a general expression and assigning it to a variable
  The evaluation result of a general expression is acquired (mapped to `java.lang.String`) as a character string type.

You define the details of assign activities in the Assign Activity dialog box.

When assigning a value to a variable, the format can be automatically converted and the value can be assigned even if the variable has a different format. The following table shows data convertibility and the conversion rules.

Table 5–13: Conversion rules for an assigned value

Data format of the value to be assigned		Data format after conversion					
		Boolean	Numeric	string	Message		
					XML	non-XML	any
Boolean		--	Y#1	Y#2	N	N	N
Numeric		Y#3	--	Y#4	N	N	N
string		Y#5	P#6	--	N	N	N
Message	XML	N	N	N	--	N	Y
	non-XML	N	N	N	N	--	Y
	any	N	N	N	Y	Y	--

Legend:

--: No conversion is needed.
N: Cannot be converted and assigned.
Y: Can be converted and assigned.
P: Can be converted and assigned in some cases.

#1

If `true`, `1` is assigned; if `false`, `0` is assigned.

#2

true or false is assigned.

#3

If 0, false is assigned. In all other cases, true is assigned.

#4

The value obtained by the toString(double) method of the java.lang.Double class is assigned.

#5

The value obtained by the parseBoolean(String) method of the java.lang.Boolean class is assigned.

#6

The value obtained by the parseDouble(String) method of the java.lang.Double class is assigned.

In assign activities, there are some combinations that cannot be specified for the copy source and copy destination. The following figure shows copy source and copy destination combinations.

Table 5–14: Copy source and copy destination combinations

Copy source		Copy destination							
		Variable (message type)			Variable (basic type)	Part of a variable (message type)#		Part of a variable (basic type)#	
		XML	non-XML	any		XML	non-XML	XML	non-XML
Variable (message type)	XML	Y	N	Y	N	Y	N	N	N
	non-XML	N	Y	Y	N	N	N	N	N
	any	Y	Y	Y	N	N	N	N	N
Variable (basic type)		N	N	N	Y	N	N	Y	N
Part of a variable (message type)	XML	Y	N	N	N	Y	N	N	N
	non-XML	N	N	N	N	N	N	N	N
Part of a variable (basic type)	XML	N	N	N	Y	N	N	Y	N
	non-XML	N	N	N	Y	N	N	Y	N
Expression		N	N	N	Y	N	N	Y	N

Legend:

Y: Can be specified.

N: Cannot be specified.

#

Cannot be copied if the copy-destination variable has not been initialized.

(1) Definition procedure

To define a receive activity:

1. Deploy an assign activity on the canvas.
   For details about how to deploy activities, see *5.4.1 Deploying Activities*.
2. Use one of the following methods to open the Assign Activity dialog box:
   - Double-click an assign activity on the canvas.
   - Choose and right-click an assign activity on the canvas, and choose Setting.

   The Assign Activity dialog box opens.

3. Enter the necessary information in the Assign Activity dialog box.

   For details about the display and input contents of the **Assign Activity** dialog box, see the manual *Cosminexus Service Platform Reference*.

   To add or edit data in Copy source or Copy destination, click **Add** or **Edit**. In the Assign Activity sub dialog box that opens, you can add or edit data in **Copy source** or **Copy destination**. If the value to be specified is a variable and you wish to edit its contents, click **Edit** in the Assign Activity sub dialog box. The Variable-Correlation Set dialog box opens. In this dialog box, you can edit the contents of the variable.

   For details about the **Assign Activity** sub dialog box and **Variable and Correlation Set List** dialog box, see the manual *Cosminexus Service Platform Reference*.

4. Click OK.

(2) Process when system exception occurs in the activity

You can send the system exception occurring in the activity, by transforming to general fault. For details, see "*4.7 General fault for transforming the system exception to fault*" in the manual "*Service Platform Function Guide*".

(3) Notes on definition

- Specify activity names that are unique within a business process (and also within a scope).
- An activity name must not exceed 64 bytes.
- Do not specify the control character in the input field of the dialog box.
- Specify at least one assignment operation.
- A copy source must be specified.
- A copy destination must be specified.
- Specify either variable or expression as the copy source type.
- If a copy source variable is chooseed, a variable name must be specified.
- If a copy source expression is chooseed, an expression must be specified.
- If the copy destination variable and the copy source variable have different formats, the format might be implicitly converted, or an error may occur during execution.
- When defining assignment from a variable (message type) to a variable (message type), specify definition files having the same XML schema and same file name for the message formats to be specified in the definition of the individual variables.

5.6.8 Empty Activity

This activity does nothing even if executed. An empty activity is used to indicate that no processing is to take place during switching that uses a switch activity or fault handling that uses fault connection.

You define the details of empty activities in the Empty Activity dialog box.

(1) Definition procedure

To define a receive activity:

1. Deploy an empty activity on the canvas.

   For details about how to deploy activities, see *5.4.1 Deploying Activities*.

2. Use one of the following methods to open the Empty Activity dialog box:
   - Double-click an empty activity on the canvas.
   - Choose and right-click an empty activity on the canvas, and choose Setting.

   The Empty Activity dialog box opens.

3. Enter the necessary information in the Empty Activity dialog box.

   For details about the display and input contents of the **Empty Activity** dialog box, see the manual *Cosminexus Service Platform Reference*.

4. Click OK.

(2) Notes on definition

- Specify activity names that are unique within a business process (and also within a scope).
- An activity name must not exceed 64 bytes.
- Do not specify the control character in the input field of the dialog box.

5.6.9 Throw Activity

This activity is used for defining a fault inside a business process and sending this fault to communicate it to a higher-order scope activity.

You define the details of throw activities in the Throw Activity dialog box.

(1) Definition procedure

To define a receive activity:

1. Deploy a throw activity on the canvas.
   For details about how to deploy activities, see *5.4.1 Deploying Activities*.
2. Use one of the following methods to open the Throw Activity dialog box:
   - Double-click a throw activity on the canvas.
   - Choose and right-click a throw activity on the canvas, and choose Setting.

   The Throw Activity dialog box opens.
3. Enter the necessary information in the Throw Activity dialog box.
   For details about the display and input contents of the **Throw Activity** dialog box, see the manual *Cosminexus Service Platform Reference*.
   For **Allocated variable**, specify the variable to be assigned when throwing a fault. To edit the contents of the variable specified for **Allocated variable**, click **Edit**. The Variable-Correlation Set dialog box opens. In this dialog box, you can edit the contents of the variable. For details about the **Variable and Correlation Set List** dialog box, see the manual *Cosminexus Service Platform Reference*.
4. Click OK.

(2) Notes on definition

- Specify activity names that are unique within a business process (and also within a scope).
- An activity name must not exceed 64 bytes.
- Do not specify the control character in the input field of the dialog box.
- For the higher-order scope that includes the throw activity, define fault handling that utilizes a corresponding allocated variable or catch-all.
- Set variable of message type (XML) in the assigned variable to be set in fault process assigning. You cannot set a variable of message type (non-XML or any).
- When the value of assigned variable of throw activity does not match to the assigned variable specified in fault process, process does not move to the activity specified in assigned variable of fault process. In such a case, process moves to the activity specified in catchAll.

5.6.10 Standby Activity

A standby activity defines a process that puts the processing flow of a business process in standby mode at regular intervals or until a certain time limit. You can use this activity to put a business process in standby mode for a specified period or until a specified time.

Define the details of the standby activity in the Wait Activity dialog box. Specify the standby time interval or time limit in an XPath expression.

(1) Definition procedure

1. Deploy a standby activity on the canvas.
   For details about how to deploy activities, see *5.4.1 Deploying Activities*.
2. Use one of the following methods to open the Wait Activity dialog box:
   - Double-click a standby activity on the canvas.
   - Select and right-click a standby activity on the canvas, and then select **Setting**.

   The Wait Activity dialog box appears.
3. Enter the necessary information in the Wait Activity dialog box.
   Select **For** or **Until** in **Standby time** and then specify the standby time interval or time limit in an XPath expression.
   For details about the Wait Activity dialog box, see *1.4.16 Standby Activity dialog* in the manual *uCosminexus Service Platform Reference Guide*.
4. Click **OK**.

(2) Character strings that can be specified as standby time

An XPath expression is used to specify standby time. An XPath expression is evaluated in the processing of a business process, and the character string obtained as a result of the evaluation is used as standby time. The BPEL standard imposes some limitations on the character strings that can be obtained by evaluating XPath expressions.

The following subsections describe the character strings that can be obtained by evaluating XPath expressions. For details about specifying XPath, see *5.6.18 Specifying an XPath*.

(a) If "For" is specified in "Standby time"

If **For** is specified in **Standby time**, the character strings that can be obtained by evaluating XPath expressions must conform to the XML Schema type duration (`xsd:duration`). The *XML Schema type duration* is the variable type indicating character strings that express elapsed time in PnYnMnDTnHnMnS format.

The following example shows the character strings that can be obtained by evaluating XPath expressions when **For** is specified for **Standby time**:

(Example 1) If an interval of 1 year, 2 months, 3 days, 4 hours, 5 minutes, and 6 seconds is specified as the standby time:

The character string that can be obtained by evaluating the XPath expression is `P1Y2M3DT4H5M6S`.

(Example 2) If an interval of 10 seconds is specified as the standby time:

The character string that can be obtained by evaluating the XPath expression is `PT10S`.

Instead of specifying an XPath expression in **Expression** in the Wait Activity dialog box, you can also specify a standby time interval directly by enclosing it in single quotation marks (') or double quotation marks ("). The following examples show direct specifications of standby time intervals by using single quotation marks (').

(Example 3) If an interval of 1 year, 2 months, 3 days, 4 hours, 5 minutes, and 6 seconds is specified as the standby time:

Specify `'P1Y2M3DT4H5M6S'` directly in **Expression** in the Wait Activity dialog box.

(Example 4) If an interval of 10 seconds is specified as the standby time:

Specify `'PT10S'` directly in **Expression** in the Wait Activity dialog box.

(b) If "Until" is specified for "Standby time"

If **Until** is specified for **Standby time**, the character strings that can be obtained by evaluating XPath expressions must conform to the XML Schema type dateTime (`xsd:dateTime`). The *XML Schema type dateTime* is the variable type indicating character strings that express a date and time in *CCYY*-*MM*-*DD*T*hh*:*mm*:*ss*.*sss*TZ format (where .*sss* and TZ can be omitted).

The following example shows the character strings that can be obtained by evaluating XPath expressions when **Until** is specified for **Standby time**.

(Example 1) If January 1, 2010 at 00:00:00 is specified as the standby time:

The character string that can be obtained by evaluating the XPath expression is `2010-01-01T00:00:00` or `2010-01-01T00:00:00.000TZ`.

Instead of specifying an XPath expression in **Expression** in the Wait Activity dialog box, you can also specify a standby time limit directly by enclosing it in single quotation marks (') or double quotation marks ("). The following examples show direct specifications of standby time limits by using single quotation marks (').

(Example 2) If January 1, 2010 at 00:00:00 Japan Standard Time (JST) is specified as the standby time:

Specify `'2010-01-01T00:00:00+09:00'` directly in **Expression** in the Wait Activity dialog box.

(Example 3) If January 1, 2010 at 00:00:00 Greenwich Mean Time (GMT) is specified as the standby time:

Specify `'2010-01-01T00:00:00+00:00'` or `'2010-01-01T00:00:00Z'` directly in **Expression** in the Wait Activity dialog box.

(3) Notes on definition

- Specify activity names that are unique within a business process (and also within a scope).
- An activity name must be no more than 64 bytes.
- Do not specify any control characters in the input fields of the dialog box.
- A standby time must be specified.
- Do not set a standby activity for non-persistent business processes.
- Set a standby activity after a reply activity because operations must be performed after a reply is returned from a reply activity to a service requester. If a standby activity is set before a reply activity, an error does not occur during packaging, but does occur when the business process is executed. Note that if a standby activity is set between a reply activity and a receive activity following it, an error does not occur.[#1, #2]
- After a standby activity, do not set a reply activity that is not paired with a receive activity. If a reply activity not paired with a receive activity is set after a standby activity, an error does not occur during packaging, but does occur when the business process is executed. Even if an activity other than reply activities is set after a standby activity, an error does not occur.[#1, #3]

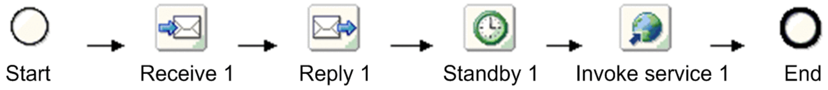
#1

The following examples show cases where no error occurs:

(Example 1)

A standby activity is set after a reply activity. No error occurs because processes following the standby process are executed as asynchronous business processes.

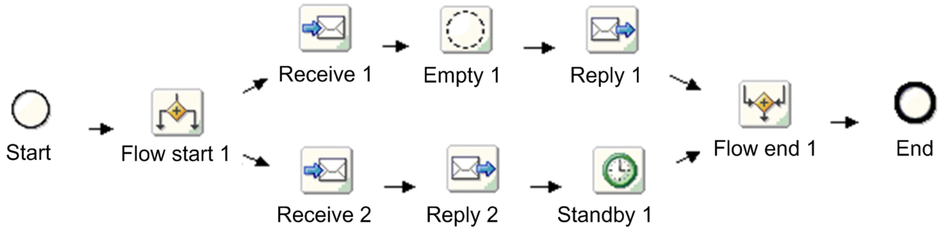
The standby activity waits for processing, and the invoke service activity is executed when the standby time limit is reached.



(Example 2)

A standby activity is set after a reply activity. No error occurs because processes following the standby process are executed as asynchronous business processes.

The empty activity executes processing as part of parallel processing in response to a request from the service requester. When another request is received from the service requester, a response is returned to the service requester once, and then the standby process is executed.



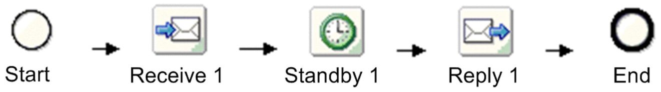
#2

The following example shows a case where an error occurs:

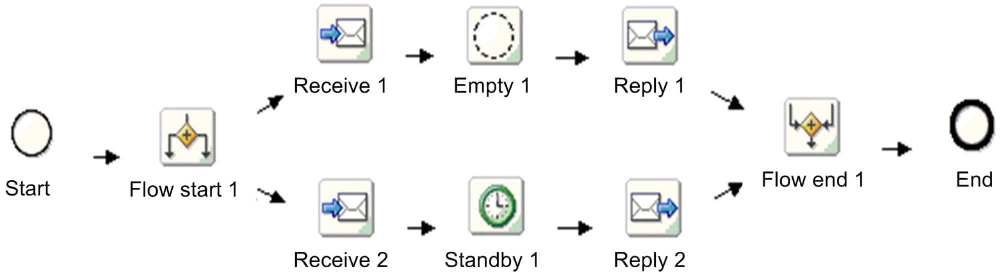
(Example 1)

A standby activity is set before a reply activity. Because a standby activity is executed within a synchronous business process, an error occurs when the business process is executed.

- When a flow activity is not used



- When a flow activity is used

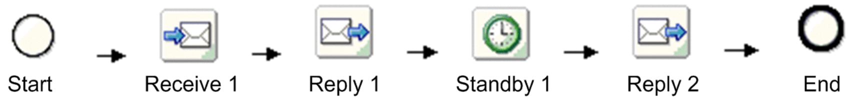


#3

The following example shows a case where an error occurs:

(Example 1)

A reply activity that is not paired with a receive activity is set after a standby activity. As a result, an error occurs when the business process is executed.



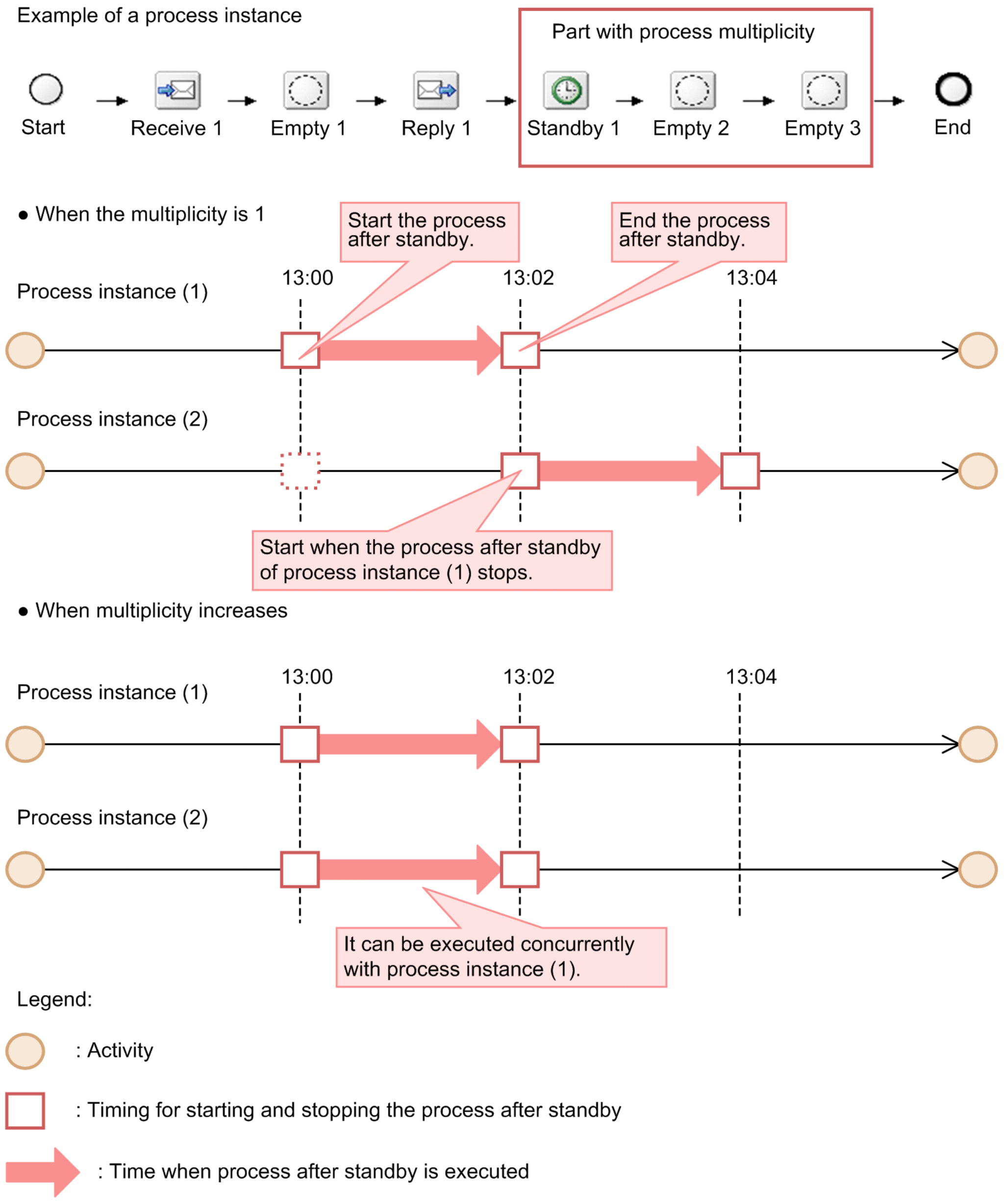
(4) Notes on multiple executions in an environment set up by the HCSC easy setup functionality

In an environment set up using the HCSC easy setup functionality, processes following a standby activity for multiple process instances are not executed concurrently because the multiplicity of processes following a standby activity is set as 1 by default. If processes following a standby activity start simultaneously from multiple process instances, the processing of a process instance that starts later is not executed until the previous processing finishes.

To simultaneously execute processes following a standby activity for multiple process instances, adjust the process multiplicity by changing the `ejbserver.ejb.timerservice.maxCallbackThreads` property in `usrconf.properties`.

The figure below shows an example of processing where multiple process instances are executed when the multiplicity of processes following a standby activity is 1 and the multiplicity is increased. This example assumes that the standby activity (Standby 1) causes multiple process instances to wait for processing until 13:00 and then it takes two minutes for post-standby processes (Empty 2 and Empty 3) to finish.

Figure 5–10: Examples of processing where the multiplicity of processes following a standby activity is 1 and where the multiplicity is increased


Example of a process instance
Part with process multiplicity
Start
Receive 1
Empty 1
Reply 1
Standby 1
Empty 2
Empty 3
End
● When the multiplicity is 1
Start the process after standby.
End the process after standby.
13:00
13:02
13:04
Process instance (1)
Process instance (2)
Start when the process after standby of process instance (1) stops.
● When multiplicity increases
13:00
13:02
13:04
Process instance (1)
Process instance (2)
It can be executed concurrently with process instance (1).
Legend:
: Activity
: Timing for starting and stopping the process after standby
: Time when process after standby is executed

For details about how to set the `ejbserver.ejb.timerservice.maxCallbackThreads` property in `usrconf.properties`, see *2.4 usrconf.properties(User property file for J2EE servers)* in the *Application Server Definition Reference Guide*.

5.6.11 Validate activity

This is the activity for validating the correctness of message sent and received in business processes. It is used to validate whether the message sent and received with external by receive activity or invoke service activity match with the schema of corresponding allocation variable.

Describe the details on validate activity, in the validation activity dialog.

(1) Definition procedure

Definition procedure is as follows:

1. Schedule the validate activity to a canvas.
   For details on how to schedule the activity, see "*5.4.1 Deploying Activities*".
2. Display Validate activity dialog by either of the following methods:
   - Double click the validate activity of canvas.
   - Select and right click the validate activity of canvas and select **Settings**.

   Validate activity dialog is displayed.
3. Enter the required information in Validate activity dialog.
   For details on the Validate activity dialog, see "*1.4.17 Validate activity dialog*" in the manual "*service Platform Reference Guide*".
   Click **Edit** button to edit the contents of variables to be set in **Variables**. You can edit the contents of variable, in the displayed Variable/correlation set list dialog. In **Variables**, you can set the variable of message type (XML), but you cannot set the variable of message type (non-XML or any). For details on Variable/correlation set list dialog, see "*1.4.1 Variable/correlation set list dialog*" in the manual "*Service Platform Reference Guide*".
4. Click [OK] button.

(2) Process in case of error occurrence in validation

If validation result is invalid, you can send following types of faults.

- Validate activity specific fault
- General fault

For details on how to select the fault to be sent, see "*4.7.1 Overview of general fault to transform the system exception to fault*" in the manual "*Service Platform Function Guide*".

Reference note

> If multiple variables have been set in the validate activity, a fault message is created with information of first variable for which error occurred during validation.

Settings required for fault process are as follows:

(a) Definition in case of executing fault process

When fault occurs in the validate activity, process is switched depending on the allocation of fault process, which has been set in the scope activity to which the validate activity belongs. If fault process allocation has not been set, fault exception occurs.

Fault process allocation

You cannot allocate fault process directly from the validate activity. Allocate the process by using the scope activity to which validate activity belongs.

Fault process switching

Use the switch activity to switch the process corresponding to the validation result.

Define such that the fault information is acquired by the variable part specification, from the variable allocated to fault message and the process is switched depending on the acquired information.

Optional fault occurrence

Use throw activity to generate the optional fault.

Define such that when fault occurs, a fault of optional variable value is generated by the throw activity.

(b) Schema file for defining the fault message to be used in fault process definition

Storage destination of the schema file for defining the validate activity specific fault message is as follows:

```
<Service Platform installation directory>\CSC\system\msg\cscvalidatefault.xsd
```

Schema file for defining the validate activity specific fault message to be used in allocation of fault process of the scope activity to which validate activity belongs, is shown in the following snippet. For details on schema file of general fault, see "*7.12.1 Schema file that defines the general fault message*" in the manual "*Service Platform System Setup and Operation Guide*".

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
 DO NOT EDIT THIS FILE.
 -->
<xs:schema
    elementFormDefault="qualified"
    targetNamespace="http://www.msg.csc.soft.Hitachi.co.jp/cscBpValidate"
    xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="fault">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="exception-name"    type="xs:string"/>
        <xs:element name="exception-message" type="xs:string"/>
        <xs:element name="scope-name"        type="xs:string"/>
        <xs:element name="activity-name"     type="xs:string"/>
        <xs:element name="variable-name"     type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Configurable elements are as follows:

exception-name

Set the exception name. In exception name, set the value of Class.getName().

exception-message

Set the exception information. In exception information, set the value of Exception.toString().

scope-name

Set the scope name to which the validate activity belongs.

activity-name

Set the name of validate activity.

variable-name

Set the name of variable to be validated.

(3) Notes at the time of definition

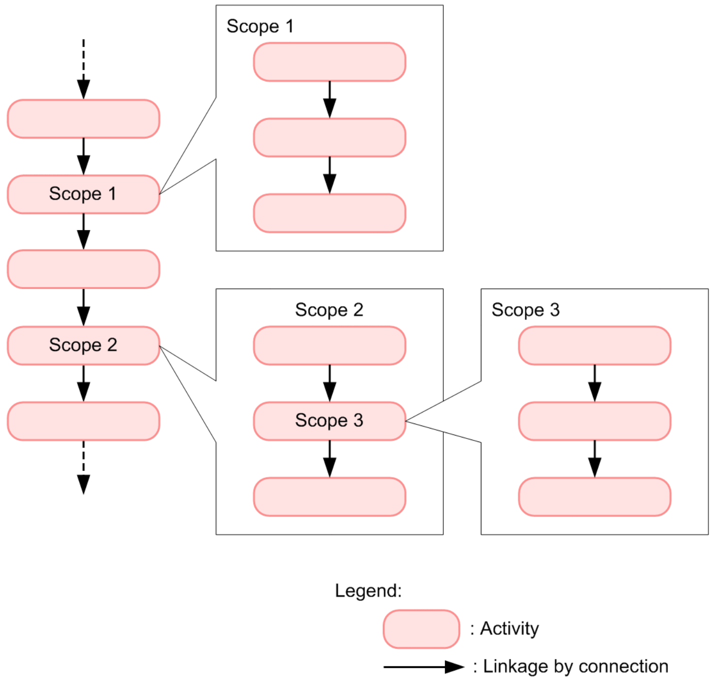
- Specify activity name such that it is unique in the business process (and also within scope).
- Specify activity name within 64 bytes.
- Do not specify a control character in input field of a dialog.

5.6.12 Scope Activity

This activity defines a process flow consisting of one or more activities as a single coherent processing unit.

The following figure shows processing units that use scope activities.

Figure 5–11: Setting up processing units using scope activities



When you define a scope activity, you can handle the processing flow inside the scope as a single coherent unit. You can also define a scope inside the processing flow of another scope, as exemplified by Scope 3 in the figure above.

Note that variables defined inside a scope are valid only within that scope. For details about the scopes and the valid range for variables, see *5.5.1(5)(a) Use of variables when scope activities are used*.

(1) Definition procedure

To define a receive activity:

1. Deploy a scope activity on the canvas.
   For details about how to deploy activities, see *5.4.1 Deploying Activities*.
2. Use one of the following methods to open the Scope Activity dialog box:
   - Double-click a scope activity on the canvas.
   - Choose and right-click a scope activity on the canvas, and choose Setting.

   The Scope Activity dialog box opens.
3. Enter an activity name.
4. Click OK.
5. Choose and right-click a scope activity on the canvas, and choose Open.
   A tab with the scope activity name is displayed in the bottom portion of the canvas.
6. Click the tab with the scope activity name.
   The canvas for specifying the processing inside the scope is displayed.
7. Deploy, link, and define the desired activities to specify the processing inside the scope.

(2) High level settings

You can set the scope activity such that transaction is not committed in the internally defined activity.

When you select **Commit at the time of start and end of this scope** radio button in **High level settings** of Scope activity dialog, transaction is committed at the time of start and end of the scope activity. The activity defined in the scope activity is not committed.

When you select **Commit in the activity unit** radio button, activity is not committed at the time of start and end of the end activity. Activity to execute commit defined in the scope activity is committed.

By default, **Commit in activity unit** radio button is in selected status.

For details on Scope activity dialog, see "*1.4.18 Scope activity dialog*" in the "*Service Platform Reference Guide*".

For details on the transaction when you select **Commit only at the time of start and end of this scope** radio button, see "*3.4.5 Transaction when selecting the settings of committing at the time of start and end of scope*" in the "*Service Platform Function Guide*".

(3) Notes on definition

- An activity name must not exceed 64 bytes.
- Do not specify the control character in the input field of the dialog box.

You must note the following points to be noted when defining the scope activity for which **Commit only at the time of start and end of this cope** radio button:

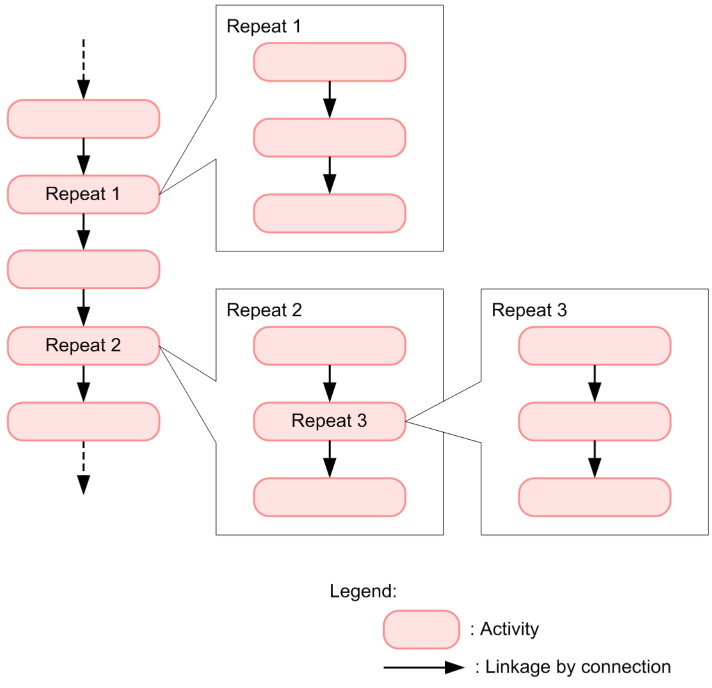
- You cannot define following activity in the scope activity:
  - Scope activity for which **Commit only at the time of start and end of this scope** radio button is selected
- If you define following activities in the scope activity, validation error occurs.
  - Receive activity
  - Reply activity
  - Standby activity
- You can specify only the DB adapter in the invoke service activity defined in the scope activity for which **Commit only at the time of start and end of this scope** radio button is selected. If you specify the service adapter other than DB adapter, and the business process, operation is not guaranteed.
- If transactions of the business process and service invoke destination are different, transaction control target is the transaction of the business process. Transaction of the service invoking destination operates without getting impacted by the transaction control (such as in case of defining invoke service activity that invokes a business process and invoke service activity that invokes the function of RequiresNew)
- When you define invoke service activity that invokes the DB adapter, in the scope activity, operating with value of " dba-separate-transaction" property of HCSC server runtime definition file as "false" or value of " dba_separate_transaction" attribute of SQL operation definition file as "N" is pre-requisite.
- When system exception occurs and rolled back business process is re-executed, you must assume that activity defined in the scope activity and service of service invoking destination are re-executed.

5.6.13 While Activity

This activity defines repeated processing of one or more activities according to a given condition.

The following figure shows repeated processing that uses while activities.

Figure 5–12: Repeated processing using while activities



When a while activity is defined, one or more processes can be repeated according to a given condition. You can also define a repeating process inside another repeating process, as exemplified by Repetition 3 in the above figure.

(1) Repetition method

In the while activity, you can select either of the following repeat method.

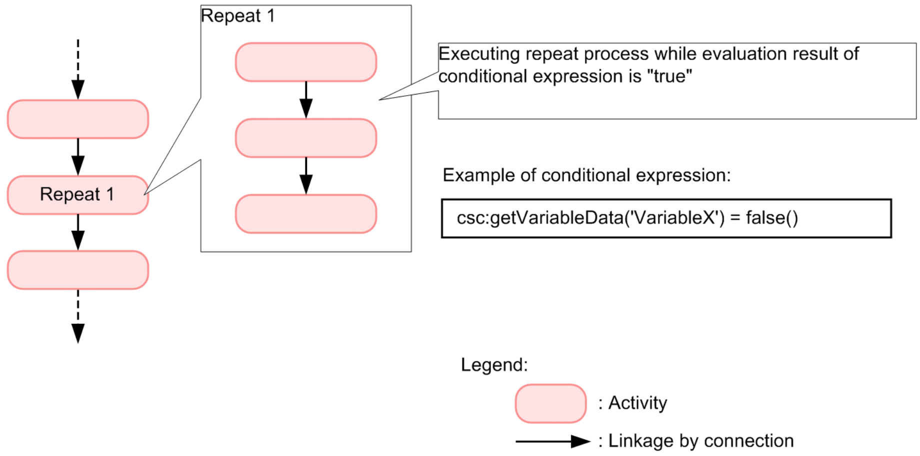
- Condition specification format of repeating according to the fixed conditions
- Method of list specification to be repeated for the number of times equal to the count of elements included in repetition list

Repeat process using the condition specification method and list specification method is described as follows:

(a) Condition specification method

Following figure shows the repeat process using the condition specification method.

Figure 5–13: Repeat method using the condition specification format

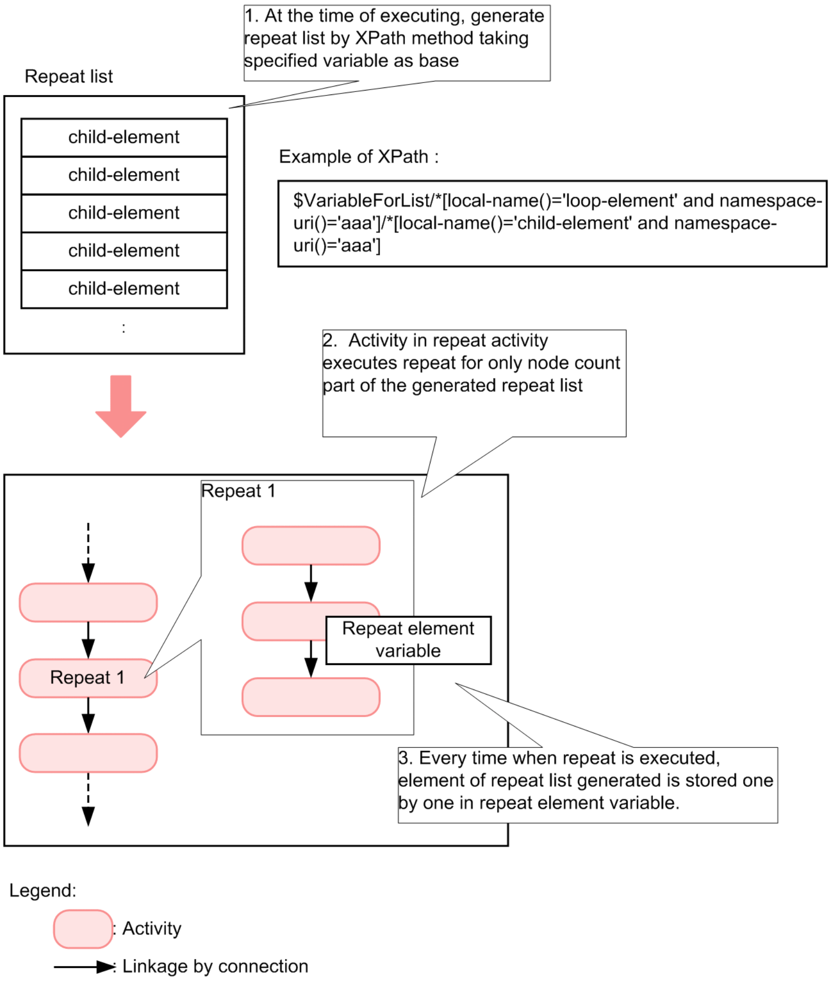

Repeat 1
Repeat 1
Executing repeat process while evaluation result of conditional expression is "true"
Example of conditional expression:
csc:getVariableData('VariableX') = false()
Legend:
: Activity
: Linkage by connection

In condition specification method, specify the repetition condition expression in XPath format. Activity within the while activity is repeated till the evaluation result of condition expression is true.

(b) List specification method

Following figure shows the repetition process using the list specification method.

Figure 5–14: Repetition process using the list specification method


1. At the time of executing, generate repeat list by XPath method taking specified variable as base
Repeat list
child-element
child-element
child-element
child-element
child-element
:
Example of XPath :
$VariableForList/*[local-name()='loop-element' and namespace-uri()='aaa']/*[local-name()='child-element' and namespace-uri()='aaa']
2. Activity in repeat activity executes repeat for only node count part of the generated repeat list
Repeat 1
Repeat 1
Repeat element variable
3. Every time when repeat is executed, element of repeat list generated is stored one by one in repeat element variable.
Legend:
: Activity
: Linkage by connection

In list specification method, repetition process is executed in the following flow:

1. Repetition list is generated based on variables specified in XPath.
2. Activity within the while activity is repeated only for the number of times equal to the number of nodes included in the generated repetition list.
3. Every time when the repeat is executed, elements of repetition list are stored one by one in the repetition element variable.

Create variables and repeat element variable for generating repetition list, in the development environment and specify those in Repetition list settings dialog.

(2) Definition procedure

To define a receive activity:

(a) Condition specification method

1. Schedule the while activity to the canvas.

For details on how to schedule the activity, see "*5.4.1 Deploying Activities*".

2. Select and right click the while activity on the canvas and select **Settings**.

   While activity dialog is displayed.

3. Enter any activity name in While activity dialog.

   For details on the While activity dialog, see "*1.4.19 While activity dialog*" in the "*Service Platform Reference Guide*".

4. Select **Condition specification method** radio button and click the **Repetition condition settings** button.

   Repetition condition settings dialog for setting up the repetition condition is displayed. For display/input contents of Repetition condition setting dialog, see "*1.4.21 Repetition condition settings dialog*" in the "*Service Platform Reference Guide*".

5. Specify the required information and click **OK** button.

6. Display the canvas for setting up the repetition process, by either of the following methods:

   - Double click the while activity of the canvas.
   - Select and right click the while activity of the canvas and select **Open**.

   Tab of the while activity name is displayed in the lower part of the canvas and the canvas for setting up the while process is displayed.

7. Deploy, link and define any activity and set the repetition process.

(b) List specification method

1. Define the message type (XML) variable for generating the repetition list.

   Create variables by schema having repetition element as child element. All the child elements must have the same format.

   For details on defining variables, see "*5.5.1(6)(a) Defining new variables*".

   Creation example (unify the name attribute of child attribute and specify elements count in maxOccurs attribute)

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
    elementFormDefault="qualified"
    targetNamespace="http://example.com/sample"
    xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="loop-element">
    <xs:complexType>
      <xs:sequence>
        <xs:element maxOccurs="5" minOccurs="0" name="child-element"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

2. Define the message type (XML) variable of repetition element.

   Create variables for storing the element of repetition list generated by variables created in step 1.

   Creation example

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
    elementFormDefault="qualified"
    targetNamespace="http://example.com/sample"
    xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="child-element"/>
</xs:schema>
```

3. Schedule the while activity to the canvas.

   For details on scheduling the activity, see "*5.4.1 Deploying Activities*".

4. Select and right click the while activity of the canvas and select **Settings**.

   While activity dialog is displayed.

5. Enter any activity name in the while activity dialog.

   For details on while activity dialog, see "*1.4.19 While activity dialog*" in the "*Service Platform Reference Guide*".

6. Select **List specification method** radio button and click **repetition list settings** button.

7. Repetition list settings dialog for setting the repetition list is displayed.

   For details on display/input contents of Repetition list settings dialog, see "*1.4.20 Repetition list settings dialog*" in the "*Service Platform Reference Guide*".

8. Specify the path of element of the variable defined in step.1, in the **Expression** of **Repetition list** of Repetition list settings dialog, in XPath format.

   For details on specifying XPath format, see "*5.6.18(2)(d) Specifying in the repetition list settings dialog*".

9. In **Variable name** of **Repetition variable**, specify the variable defined in step.2.

10. Set **Maximum repetition count** as and when required.

11. Click **OK** button.

12. Display canvas for setting the repetition process, by either of the following methods:

    - Double click while activity of the canvas.
    - Select and right click the while activity of the canvas and select **Open**.

    Tab of the while activity name is displayed in the lower part of canvas and the canvas for setting the repetition process is displayed.

13. Deploy, link and define any activity and set the repetition process.

(3) Notes on definition

- Specify activity names that are unique within a business process (and also within a scope).
- An activity name must not exceed 64 bytes.
- Do not specify the control character in the input field of the dialog box.
- A repetition condition must be specified.

(4) Notes when defining (only in case of list specification method)

You must consider following points at the time of definition, when specifying the list specification method.

- You can use repetition process of using the list specification method, only for non-persistence business process. If you define the repetition activity of list specification method in the persistence business process, error message is displayed when validating the business process.
- Variable and repetition element variable for generating the repetition list must be of message type (XML). You cannot specify variable having type other than message type (XML).
- You cannot change the variable for generating the repetition list, within the repetition process. If you change the variable, the operation is not guaranteed. You can reference and change the repetition element variable in the repetition process.
- If you specify a path of non-existing element in the XPath expression specified in **Expression** of the Repetition list settings dialog, repetition process is not executed.
- In the following cases, format of handled information might not match with XML schema of repetition element variable.
    - If you set information of format not matching to XML schema of repetition element, in the repetition element
    - If you specify node of the type other than element node in the XPath expression specified in **Expression** of Repetition list settings dialog

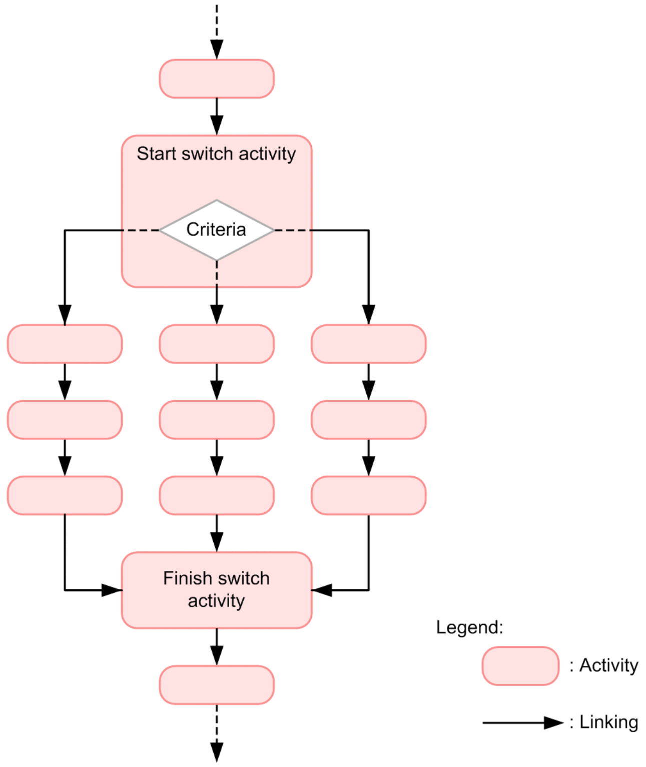
  In repetition process, if you try to reference the repetition element variable not matching to the format of above information, and referencing is not possible, system exception occurs and process stops.
- When evaluation result of the XPath expression specified in **Expression** of the Repetition list settings dialog is other than NodeList, system exception occurs and process stops.

5.6.14 Switch Activities

These activities define switching of the processing of a business process according to the condition determination result. Switch activities include switch start and switch end activities. You deploy a switch start activity at the switch start location and a switch end activity at the switch end location.

The following figure shows process switching using switch activities.

Figure 5–15: Process switching using switch activities



Several processes are provided under the switch start activity, and switching occurs according to the condition set in the switch start activity and the condition determination result. A switch end activity is deployed at the termination of the switch.

A process that branches from the switch start activity and ends in a throw activity need not be connected to the switch end activity. If all processes that branch from the switch start activity end in throw activities, there is no need to deploy a switch end activity.

(1) Defining the switch start activity

To define the switch start activity:

1. Deploy a switch activity on the canvas.
   For details about how to deploy activities, see *5.4.1 Deploying Activities*.
2. Deploy an activity that becomes the link destination of the switch activity on the canvas.
   For details about how to deploy activities, see *5.4.1 Deploying Activities*.
3. Link the switch activity to the activity at the link destination.
   For details about how to link activities, see *5.4.2(2) Link setting method*.
4. Use one of the following methods to open the Switch Activity dialog box:
   - Double-click a switch activity on the canvas.
   - Choose and right-click a switch activity on the canvas, and choose Setting.

   The Switch Activity dialog box opens.

5. Enter the necessary information in the Switch Activity dialog box.

   For details about the display and input contents of the **Switch Activity** dialog box, see the manual *Cosminexus Service Platform Reference*.

   To set a switch condition, click **Condition Setting**. The Condition Setting dialog box opens. In this dialog box, you can set a switch condition. Specify a condition for each link destination. Enter the switch conditions as XPath expressions. For details about the Condition Settings dialog box, see the manual *Cosminexus Service Platform Overview*.
6. Click OK.

(2) Defining the switch end activity

To define the switch end activity:

1. Deploy a switch activity on the canvas.

   For details about how to deploy activities, see *5.4.1 Deploying Activities*.
2. Link the switch activity to the link source activity.

   For details about how to link activities, see *5.4.2(2) Link setting method*.

(3) Process when system exception occurs in the activity

You can send the system exception occurring in the activity, by transforming to general fault. For details, see "*4.7 General fault for transforming the system exception to fault*" in the "*Service Platform Function Guide*".

(4) Notes on definition

- Specify activity names that are unique within a business process (and also within a scope).
- An activity name must not exceed 64 bytes.
- Do not specify the control character in the input field of the dialog box.
- Specify at least one switch condition.
- Assign the transition destination to a conditional switch or the default.
- A switch condition name must be specified.
- A switch condition must be specified.
- Do not connect a switch activity to a flow end activity.
- Do not make the following connections:
  - Connection that uses the start activity as the source#1
  - Connection that uses the start point of fault handling of an activity as the source#1
- Connect the switch start activity to a single corresponding end activity.#2
- Connect the corresponding switch processes to the switch end activity.#2
- A switch process must be connected to the switch end activity.#2
- Specify at least one activity between a switch activity and its corresponding switch end activity.

#1

For details about connection that uses the start activity as the source and connection that uses the start point of fault handling of an activity as the source, see *Figure 5-26 Connection example*.

#2

Excludes a case in which the termination points of all switch processes are throw activities.

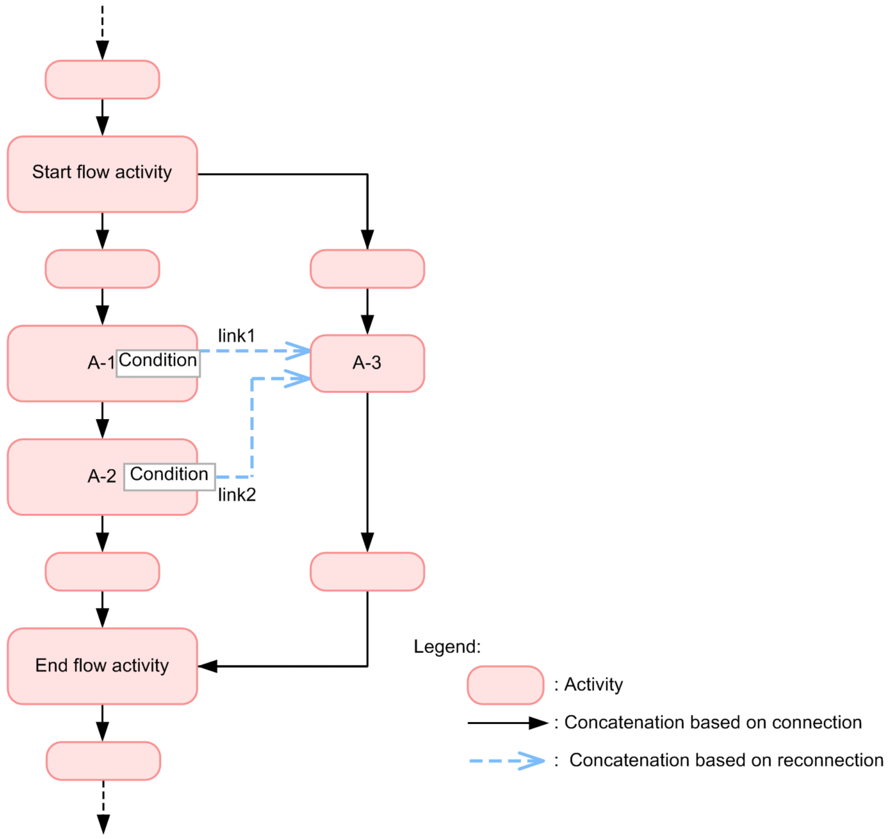
5.6.15 Flow Activities

Flow activities define the division of flow of a business process into multiple sequences and the concurrent execution of these sequences. Flow activities include flow start and flow end activities. Deploy a flow start activity at the starting point of a flow process and a flow end activity at the merging point of multiple sequences.

Of the multiple sequences that branch from the flow start activity and that are executed concurrently, a sequence that ends in a throw activity need not be linked to the flow end activity. In addition, if all sequences that branch from the flow start activity and that are executed concurrently end in throw activities, no need exists to deploy a flow end activity.

The following figure shows the concurrent execution of processes using flow activities.

Figure 5–16: Concurrent execution of processes using flow activities



You can provide multiple sequences under a flow start activity, and execute processes concurrently. You can also specify the processing order among the multiple sequences.

For example, to execute A-3 after processing of A-1 and A-2 finishes, use link connections (link1 and link2) to link the link sources A-1 and A-2 to the link destination A-3. In addition to specifying processing completion of A-1 and A-2, you can also specify linking conditions to control the linking of A-1 and A-2 to A-3.

In this example, processing of the link destination A-3 starts in the following sequence:

1. Processing of A-1 and A-2 finishes.
2. The linking conditions specified for A-1 and A-2 are evaluated.

   When the linking conditions for A-1 and A-2 are evaluated, their respective link connections link1 and link2 are activated. The link connections are activated as `true` or `false` depending on the condition evaluation result. If no condition is specified, the link connections are activated as `true` when processing of the link source activities finishes.
3. When all link connections linked to A-3 are activated, processing of A-3 starts.

   However, if all link connections are activated as `false`, processing of A-3 is omitted. Processing of A-3 starts only when at least one of the link connections is activated as `true`.

Processing of activities following the flow end activity is executed after all multiple sequences being concurrently processed finish.

(1) Defining flow activities

To define flow activities:

1. Deploy a flow start activity on the canvas.
   For details about how to deploy activities, see *5.4.1 Deploying Activities*.
2. Deploy and define the individual activities that branch from the flow start activity.
   For details about how to deploy activities, see *5.4.1 Deploying Activities*, and for details about how to define activities, see the descriptions of each activity in *5.6 Defining Activities*.
3. Deploy a flow end activity on the canvas.
   For details about how to deploy activities, see *5.4.1 Deploying Activities*.
4. Link the individual activities from the flow start activity to the flow end activity.
   For details about how to link activities, see *5.4.2(2) Link setting method*.
5. Use one of the following methods to open the Flow Activity dialog box:
   - Double-click a flow start activity on the canvas.
   - Select and right-click a flow start activity on the canvas, and then select **Setting**.

   The Flow Activity dialog box appears.
6. Enter the necessary information in the Flow Activity dialog box.
   For details about the display and input contents of the Flow Activity dialog box, see *1.4.24 Flow Activity Dialog* in the manual *uCosminexus Service Platform Reference Guide*. For details about how to set up links among multiple sequences, see *5.6.15(2) Defining links among sequences*.
7. Click **OK**.

(2) Defining links among sequences

To define links among multiple sequences that are concurrently processed:

1. Use one of the following methods to open the Flow Activity dialog box:
   - Double-click a flow start activity on the canvas.
   - Select and right-click a flow start activity on the canvas, and then select **Setting**.

   The Flow Activity dialog box appears.
2. Enter a link name for each link to be set up within the flow process.
3. Click **OK** to close the Flow Activity dialog box.
4. Connect the activities to be linked.
   For details about how to deploy activities, see *5.4.1 Deploying Activities*.
5. Use one of the following methods to open the Link dialog box:
   - Double-click the linking line that was connected in step 4.
   - Select and right-click the linking line that was connected in step 4, and then select **Setting**.

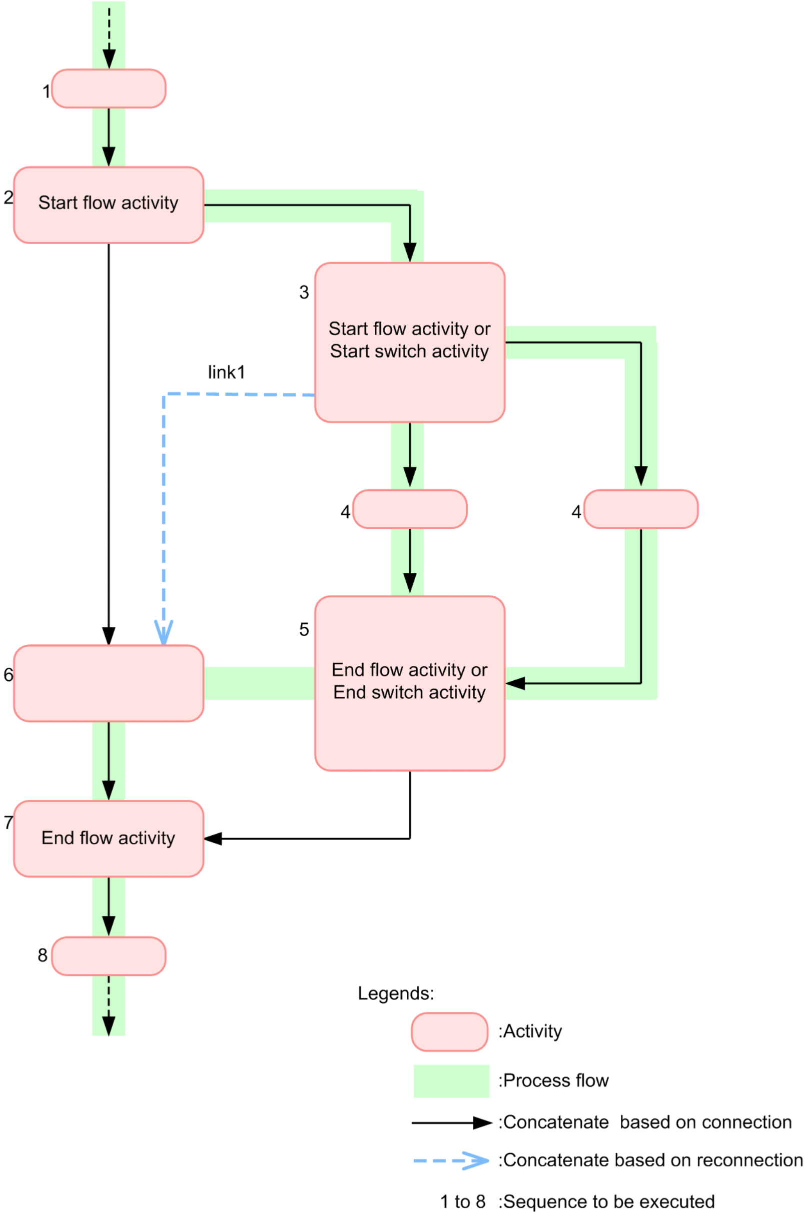
   The Link dialog box appears.
6. Enter the necessary information in the Link dialog box.
   For details about the display and input contents of the Link dialog box, see *1.4.25 Link Dialog* in the manual *uCosminexus Service Platform Reference Guide*.

   To set up link conditions, click **Set Transition Condition**. The Condition Setting dialog box appears. In this dialog box, you can specify link conditions. Enter the link conditions in an XPath expression. For details about the Condition Setting dialog box, see *1.4.26 Condition Setting Dialog (Flow Activity)* in the manual *uCosminexus Service Platform Reference Guide*.
7. Click **OK**.

Important note

The following figure shows the flow of processing performed when a link is defined from a flow start activity or switch start activity.

Figure 5–17: Processing flow when a link is defined from a flow start activity or switch start activity



(3) Notes on definition

- Specify activity names that are unique within a business process (and also within a scope).
- An activity name must be no more than 64 bytes.
- Do not specify any control characters in the input fields of the dialog box.
- A link name must be no more than 64 bytes.
- Do not connect a flow activity to a switch end activity.
- Do not make the following connections:
    - Connection that uses a start activity as the source[#1]

  - Connection that uses the start point of fault handling of an activity as the source#1
- Connect a flow start activity to a single corresponding flow end activity.#2
- Connect corresponding flow processes to a flow end activity.#2
- A flow process must be connected to a flow end activity.#2
- Specify at least one activity between a flow start activity and its corresponding flow end activity.
- Specify a valid link name for each link connection.

#1

For details about connections that use an activity as the source and connections that use the start point of fault handling of an activity as the source, see *Figure 5-26 Connection example*.

#2

This does not apply to cases where the termination points of sequences that are concurrently processed are throw activities.

5.6.16 End Activity

Indicates the end of a business process. Only a single start activity is deployed on the canvas. This activity cannot be deleted.

There is no content to be defined for an end activity.

5.6.17 Sequence Activity

This activity holds the information about a series of activities that are sequentially processed. A sequentially processed activity is generated automatically as needed. A sequentially processed activity stores the following activity information:

- A series of activities defined between a start activity and an end activity (additionally, activities defined on the canvas of a scope activity or while activity)
- A series of activities on each branch defined between a switch start activity and a switch end activity
- A series of activities on each branch defined between a flow start activity and a flow end activity
- A series of activities that are connected to a fault connection and that perform fault handling

5.6.18 Specifying an XPath

In the input column of the dialog boxes used for defining activities, you must specify an XPath. The following table describes the input column of dialog boxes used for defining activities and specifying the Xpath:

Table 5–15: Input column of dialog boxes used for defining activities and specifying the XPath

Item No.	Activity name	Input column of dialog boxes
1	Assign activity	Value of Copy source in the Assign Activity sub dialog box
2	Switch activity	Condition Expression of the **Condition Setting** dialog box (Switch activity)
3	Flow Activity	Condition Expression of the **Condition Setting** dialog box (Flow activity)
4	While activity	Condition Expression of the **Condition Setting** dialog box (While activity)
5	Standby activity	**Expression** of **Standby time** of **Standby activity** dialog box

In the XPath, the information within a variable can be acquired either by using the extension function or by directly specifying the variable name. For details about the extension function, see *5.6.18(3) Extension function*.

(1) Acquiring the information within a variable using the extension function

The methods of acquiring the information within a variable using the extension function are as follows:

(a) Specifications in the Assign Activity sub dialog box

To acquire a value of the copy-source XML, specify the XPath in *Value* with the extension function.

Acquiring the value of the propertyName attribute

Acquire the value of the `propertyName` attribute using the extension function `bpws:getVariableProperty`. You can use this method when the variable type is `message`.

Specification method

```
bpws:getVariableProperty('variableName','propertyName')
```

- *variableName*: Specify a variable name.
- *propertyName*: Specify the part name other than the `messageType` defined within the variable specified in *variableName*.

Specification example

```
bpws:getVariableProperty('VariableX','PropertyY')
```

Directly specifying the value of the variable

Acquire the value of the `propertyName` attribute using the extension function `csc:getVariableData`. You can use this method when the variable type is `message`, `string`, `numeric`, or `boolean`.

Specification method (When the variable type is message)

```
csc:getVariableData('variableName','locationPath')
```

- *variableName*: Specify a variable name.
- *locationPath*: Specify a path (acquired in Chooseed path of the **Show Variables** dialog box) that indicates the node of an XML schema.

Specification example

```
csc:getVariableData('VariableX','input/forInvoke/depositData/id')
```

Specification method (When the variable type is string, numeric, or boolean)

```
csc:getVariableData('variableName')
```

- *variableName*: Specify a variable name.

Specification example

```
csc:getVariableData('VariableX')
```

When acquiring the initialization status of header assigned variables

Use the extension function csc:getMessageInitialize to acquire the initialization status of header assigned variable. You can use this function, when the variable type is message type (XML, non-XML or any).

Specification method

```
csc:getMessageInitialize('variableName')
```

- *variableName*: Specify the header assigned variables.

Specification example

```
csc:getMessageInitialize('VariableX')
```

When acquiring the hexadecimal format character string of message type variable (non-XML)

Use the extension function csc:getHexVariableData to acquire the hexadecimal format character string (single byte upper chase characters) of message type variable (non-XML or any).

You can use the function when the variable type is message type (non-XML or any).

Specification method

```
csc:getHexVariableData('variableName','beginIndex','compNumber')
```

- *variableName***:** Specify the acquisition source variable name of message type **(**non-XML or any).
- *beginIndex***:** Specify bytes count of acquisition start position.
- *compNumber***:** Specify bytes count to be obtained.

Specification example

```
csc:getHexVariableData('VariableX','2','1')
```

When acquiring the character string in hexadecimal format

Use the extension function csc:getHexString to acquire the character string (single byte upper case characters) in hexadecimal format.

Specification method

```
csc:getHexString('convertString','characterCode')
```

- *convertString***:** Specify a character string to be transformed to hexadecimal format.
- *characterCode***:** Specify the character code to be used for encoding from character string to byte data. Character code that you can specify depends on the JavaVM of the environment in which this function is used.

Specification example

```
csc:getHexString('PNG','UTF-8')
```

(b) Specifications in the Condition Setting dialog box

To set up the condition expression for judging a value of the variable, specify the expression that acts as a condition in Condition Expression with XPath.

Setting the value of the propertyName attribute as the condition expression

Use the extension function `bpws:getVariableProperty` and specify the condition statement from the value of the `propertyName` attribute. You can use this method when the variable type is `message`.

Specification method

Use one of the following specification methods:

Method 1

```
bpws:getVariableProperty('variableName','propertyName') = "value-to-be-compared"
```

Method 2

```
bpws:getVariableProperty('variableName','propertyName') =
bpws:getVariableProperty('variableName','propertyName')
```

- *variableName*: Specify a variable name.
- *propertyName*: Specify the part name other than `messageType` defined within the variable specified in *variableName*.

Specification example

```
bpws:getVariableProperty('VariableX','PropertyY') = "HITACHI"
```

Setting the value of the variable as the condition expression

Use the extension function `csc:getVariableData` and specify the condition statement from the value of the `propertyName` attribute. You can use this method when the variable type is `message`, `string`, `numeric`, or `boolean`.

Specification method (When the variable type is message)

```
csc:getVariableData('variableName','locationPath') = "Value-to-be-compared"
```

- *variableName*: Specify a variable name.

- *locationPath*: Specify a path (acquired in Chooseed path of the Show Variables dialog box) that indicates the node of an XML schema.

Specification example

```
csc:getVariableData('VariableX','input/forInvoke/depositData/userName/
firstName') = "Taro"
```

Specification method (When the variable type is string, numeric, or boolean)

```
csc:getVariableData('variableName')= "Value-to-be-compared"
```

- *variableName*: Specify a variable name.

Specification example

```
csc:getVariableData('VariableX') = false()
```

# When `VariableX` is a boolean type variable.

When specifying the initialization status of header assigned variable in the condition expression

Use the extension function csc:getMessageInitialize to specify the initialization status of header assigned variable, to the condition expression. You can use this function when the variable type is message type (XML, non-XML or any).

Specification method

```
csc:getMessageInitialize('variableName') = "Comparison target value"
```

- *variableName***:** Specify the header assigned variable name.

Specification example

```
csc:getMessageInitialize('VariableX') = false()
```

When specifying hexadecimal format character string of message type variable (non-XML), in the condition expression

Use the extension function csc:getHexVariableData to acquire the hexadecimal format character string (single byte upper case characters) of message variable type (non-XML or any).

You can use this function when variable type is message type (non-XML or any).

Specification method

```
csc:getHexVariableData('variableName','beginIndex','compNumber') =
"hexadecimal expression of the comparison target value"
```

- *variableName***:** Specify acquisition source variable name of message type **(**non-XML or any).
- *beginIndex***:** Specify the bytes count of acquisition start position.
- *compNumber***:** Specify bytes count to be acquired.

Specification example 1 (when determining by acquiring value of third byte from the beginning)

```
csc: getHexVariableData ('VariableX','2','1') = "4e"
```

Specification example 2 (when acquiring the value of third byte from the beginning and determining that character string by using the extension function csc:getHexString)

```
csc: getHexVariableData ('VariableX','0','3') =
csc:getHexString('PNG','UTF-8')
```

When specifying the character acquired in hexadecimal format, in the condition expression

Use the extension function csc:getHexString to specify the hexadecimal format character string (single byte upper case characters), in the condition expression.

Specification method

```
csc:getHexString('convertString','characterCode') = "Hexadecimal expression of the
comparison target value"
```

- *convertString***:** Specify character string to be transformed to hexadecimal format.

- *characterCode***:** Specify the character code to be used when encoding from character string to byte data. Character code that you can specify depends on JavaVM of the environment in which this function is used.

Specification example

```
csc:getHexString('PNG','UTF-8') = "504E47"
```

(c) Specifying in the standby activity dialog box

In **Expression** of **Standby time**, specify **XPath** in the extension function.

To acquire value of the propertyName attribute

Acquire the propertyName attribute using the function name`bpws:getVariableProperty`. You can use this when the variable type is the message type.

Specification method

```
bpws:getVariableProperty('variableName','propertyName')
```

- **variableName:** Specify the variable name.
- **propertyName:** Specify the part name other than the messageType type defined in variable specified by variableName.

Example of specification

```
bpws:getVariableProperty('VariableX','PropertyY')
```

To directly specify the value of the variable

Acquire the value of the propertyName attribute using the extension function `csc:getVariableData`. You can use this when the variable type is message type or string type.

Specification method (for message type)

```
csc:getVariableData('variableName','locationPath')
```

- **variableName:** Specify the variable name.
- **locationPath:** Specify the path showing the XML schema node (path acquired by **Choose path** of the **Variable display** dialog box).

Example of specification

```
csc:getVariableData('VariableX','input/forInvoke/depositData/id')
```

Specification method (for string type)

```
csc:getVariableData('variableName')
```

- **variableName:**Specify the variable name.

Example of specification

```
csc:getVariableData('VariableX')
```

When acquiring the initialization status of header assigned variable

Use the extension function csc:getMessageInitialize to acquire the initialization status of the header assigned variable. You can use this function when the variable type is message type (XML, non-XML or any),

Specification method

```
csc:getMessageInitialize('variableName')
```

- **variableName:** Specify the header assigned variable name.

Specification example

```
csc:getMessageInitialize('VariableX')
```

When acquiring the hexadecimal format character string of message type variable (non-XML)

Use the extension function csc:getHexVariableData to acquire the hexadecimal format character string (single byte upper case characters) of message type variable (non-XML or any).

You can use this function when the variable type is message type (non-XML or any).

Specification method

```
csc:getHexVariableData('variableName','beginIndex','compNumber')
```

- **variableName:** Specify the acquisition source variable name of message type **(**non-XML or any).
- **beginIndex:** Specify bytes count of acquisition start position.
- **compNumber:** Specify the bytes count to be acquired.

Specification example:

```
csc:getHexVariableData('VariableX','2','1')
```

When acquiring the character string in hexadecimal format

Use the extension function csc:getHexString to acquire the character string (single byte upper case character) of hexadecimal format.

Specification method

```
csc:getHexString('convertString','characterCode')
```

- **convertString:** Specify the character string to be converted to hexadecimal format.
- **characterCode:** Specify the character code to be used when encoding from the character string to byte data. The character code that you can specify depends on JavaVM of the environment in which this function is used.

Specification example

```
csc:getHexString('PNG','UTF-8')
```

! Important note

Note the following when specifying standby time:

- To specify **Intervals** in standby time
  The character strings that can evaluate and acquire the XPath expression must conform to XML Schema type duration (xsd:duration).
- To specify **Term** in standby time
  The character strings that can evaluate and acquire the XPath expression must conform to XML Schema type dateTime (xsd:dateTime).

These specifications are fixed according to BPEL standard rules. Therefore, variables of numeric type and boolean type of the business process cannot be specified.

Note that you can also specify intervals and terms directly by enclosing them in " or ' instead of specifying the XPath expression.

For details, see *5.6.10(2) Character strings that can be specified as standby time*.

(2) Acquiring the information by directly specifying variable names

The methods of acquiring information by directly specifying variable names are as follows:

(a) Specifications in the Assign Activity sub dialog box

Directly specify the XPath for acquiring *Value* of the copy source.

Acquiring the value of the propertyName attribute

Execute the method for acquiring the information within the variable using the extension function.

Directly specifying the value of the variable

Directly specify the variable name in `$variableName`. This method can be used when the variable type is `message`, `string`, `numeric`, or `boolean`.

Specification method (When the variable type is message)

```
$variableName/locationPath
```

- *variableName*: Specify a variable name.
- *locationPath*: Specify a path (acquired in Chooseed path of the **Show Variables** dialog box) that indicates the node of an XML schema.

Specification example

```
$VariableX/input/forInvoke/depositData/id
```

Specification method (When the variable type is string, numeric, or boolean)

```
$variableName
```

- *variableName*: Specify a variable name.

Specification example

```
$VariableX
```

(b) Specifications in the Condition Setting dialog box

Directly specify the XPath of the Condition Expression.

Setting the value of the propertyName attribute as the condition expression

Execute the method for specifying the condition expression using the extension function.

Setting the value of the variable as the condition expression

Directly specify the variable name in `$variableName`. This method can be used when the variable type is `message`, `string`, `numeric`, or `boolean`.

Specification method (When the variable type is message)

```
$variableName/locationPath
```

- *variableName*: Specify a variable name.
- *locationPath*: Specify a path (acquired in Chooseed path of the **Show Variables** dialog box) that indicates the node of an XML schema.

Specification example

```
$VariableX/input/forInvoke/depositData/userName/firstName = "Taro"
```

Specification method (When the variable type is string, numeric, or boolean)

```
$variableName
```

- *variableName*: Specify a variable name.

Specification example

```
$VariableX = false()
```

# When `VariableX` is a boolean type variable.

(c) Specifying in the standby activity dialog box

In **Expression** of **Standby time**, specify **XPath** in the extension function.

To acquire value of the propertyName attribute

Execute the method to acquire the information in the variable using the extension function.

To directly specify the value of the variable

To directly specify the variable name as **$variableName**. You can use this when the variable type is message type or string type.

Specification method (for message type)

```
$variableName/locationPath
```

- **variableName:** Specify the variable name.
- **locationPath:** Specify the path showing the XML schema node (path acquired by **Choose path** of the **Variable display** dialog box).

Example of specification

```
$VariableX/input/forInvoke/depositData/id
```

Specification method (for string type)

```
$variableName
```

- **variableName:** Specify the variable name.

Example of specification

```
$VariableX
```

! Important note

Note the following when specifying standby time:

- To specify **Intervals** in standby time
  The character strings that can evaluate and acquire the XPath expression must conform to XML Schema type duration (xsd:duration).
- To specify **Term** in standby time
  The character strings that can evaluate and acquire the XPath expression must conform to XML Schema type dateTime (xsd:dateTime).

These specifications are fixed according to BPEL standard rules. Variables of numeric type and boolean type of the business process cannot be specified.

Note that you can also specify intervals and terms directly by enclosing them in " or ' instead of specifying the XPath expression.

For details, see *5.6.10(2) Character strings that can be specified as standby time*.

(d) Specifying in the repetition list settings dialog

Directly specify in **Expression** of **repetition list**, in XPath. You cannot use the extension function, when you use the list specification method.

When specifying the node of repetition element

Specification method

```
$variableName/locationPath
```

- **variableName:** Specify the variable name.
- **locationPath:** Specify path of repetition element (path acquired in **selection path** of display variable dialog) that shows the node of XML schema.

Specification example

```
$VariableX/*[local-name()='loop-element' and namespace-uri()='http://example.com/sample']/
*[local-name()='child-element' and namespace-uri()='http://example.com/sample']
```

When specifying multiple nodes of repetition element

Specification method

```
($variableName/locationPath | $variableName/locationPath)
```

- **variableName:** Specify the variable name.
- **locationPath:** Specify path of repetition element (path acquired in **selection path** of display variable dialog) that shows the node of XML schema.

Specification example

```
($VariableX/*[local-name()='loop-element' and namespace-uri()='http://example.com/sample']/
*[local-name()='child-element' and namespace-uri()='http://example.com/sample'][1] |
$VariableX/*[local-name()='loop-element' and namespace-uri()='http://example.com/sample']/
*[local-name()='child-element' and namespace-uri()='http://example.com/sample'][3])
```

When specifying the node of repetition element in the specific range

Specification method (range specification)

```
$variableName/locationPath[position()<10]
```

- **variableName:** Specify the variable name.
- **locationPath:** Specify path of repetition element (path acquired in **selection path** of display variable dialog) that shows the node of XML schema.

Specification example

```
$VariableX/*[local-name()='loop-element' and namespace-uri()='http://example.com/sample']/
*[local-name()='child-element' and namespace-uri()='http://example.com/sample']
[position()<10]
```

(3) Extension function

There are two types of extension functions as shown in (i) and (ii). The format and arguments of the extension function and the notes when using an extension function are as follows:

(i) bpws:getVariableProperty

Format

`bpws:getVariableProperty('`*variableName*`','`*propertyName*`')`

Argument

variableName

Specifies a variable name.

propertyName

propertyName: Specifies the part name other than the `messageType` defined within the variable specified in *variableName*.

(ii) csc:getVariableData

Format

`csc:getVariableData('`*variableName*`','`*locationPath*`')`

Argument

variableName

Specifies a variable name.

locationPath

Specifies a path that indicates the node of an XML schema.

Note:

- If the variable type is other than `message`, specify only *variableName* as the argument.
- If the variable type is other than `message`, the function returns a data type corresponding to the variable type.
- The result of evaluating the path specified in *locationPath* for the variable data specified in *variableName* is returned as a string type (mapped to `java.lang.String`).
- If *locationPath* contains `'`, enclose *locationPath* inside double quotation marks (`"`).

(iii) getMessageInitialize

Format

`csc:getMessageInitialize`('*variableName*')

Argument

variableName

Specify the variable name.

Notes

- Define the variable element of the variable to be specified in variableName, with xml type.
- Initialization status of the variable specified in variableName is returned in boolean type.
  In case of initialized status: true
  In case of non-initialized: false

(iv) getHexVariableData

Format

`csc:getHexVariableData`('*variableName*','*beginIndex*','*compNumber*')

Argument

variableName

Specify the variable name.

beginIndex

Specify the bytes count of acquisition start position.

compNumber

Specify the bytes count to be obtained.

Notes

- In beginIndex you can specify value 0 or greater than 0 and in compNumber you can specify value 1 or greater than 1. However, specify such that total count of the value of beginIndex and value of compNumber is less than the bytes count of variable specified in variableName. If the total count of value of beginIndex and value of compNumber is greater than the bytes count of variables specified in variableName, XpathFunctionException occurs.
- When you specify a non-numeric value in beginIndex and compNumber, XpathFunctionException occurs.
- If you specify a value that cannot exist as variable, in variableName, XpathFunctionException occurs.

(v) getHexString

Format

`csc:getHexString`(*'convertString'*,'*characterCode*')

Argument

convertString

Specify the character string to be converted to hexadecimal format.

characterCode

Specify the character code to be used for acquiring the byte data from the character string.

Notes

- When you specify blank character ( ) in convertString, blank character is returned as the process result.
- If you specify data other than character code corresponding to JavaVM of the environment, in which variable is used, in characterCode, XpathFunctionException occurs.

5.7 Scheduling comments

You can add description of the business process of activities on the business process editor, by using comments. Schedule a comment by selecting **Comment** from the pallet of the business process editor.

Method to start the edition

- Double click the Comments.
- When comment is in selected status, single click the comment.
- When comment is in selected status, click **F2** key.
- In context menu of comments, select **Settings**.

Method to end the edition

- Click the outer side of the comment being edited.
- Press **Ctrl** key + **Enter** key.

5.8 Saving Business Processes

When you are editing the contents of a business process in the Define Business Process window, you can save the contents as new contents or by overwriting the existing content during or after editing. The contents are saved in the repository.

The message format definition file specified during variable definition and the data transformation definition file generated during data transformation definition are saved simultaneously with the Define Business Process window.

The methods for saving business processes are described below.

Method 1

On the Eclipse menu bar, select **File** and then **Save**.

Method 2

On the Eclipse menu bar, select **File** and then **Save All**.

Method 3

In the Define Business Process window, press **Ctrl** + **S**.

If a business process has not been saved when you attempt to finish defining the business process by closing the Define Business Process window, the Save Resource dialog box appears. In this dialog box, you can save the business process that has not been saved.

! Important note

If invalid data has been entered, you might be unable to save the business process. In such a case, take action according to the displayed message.

5.9 Editing Business Processes

You can edit the contents of a saved business process. Open the Business Process Definition screen and edit the contents. The Business Process Definition screen opens when you choose and double-click the applicable business process from the service definition list in the tree view.

For details about how to define business process contents, see *5.3 Defining Business Process Contents*.

5.9.1 Modifying the definition information for business processes and activities

You can modify the definition information for business processes and activities.

! Important note

If definition information has been changed in the Properties view, apply the changes before performing other operations.

(1) Modifying the definition information for business processes

The following two methods are available to modify the definition information for business processes:

Method 1

1. From the services displayed in the service definition list in the tree view, select and double-click the applicable business process.

   The Define Business Process window for the selected business process opens.

2. Modify the contents.

Method 2

In the Properties view, you can modify the following definition information for business processes. To modify any other types of definition information, use Method 1.

- Business process name
- Service ID#
- Status persistence

#

To change a service ID, specify alphanumeric characters and underscores (_) of no more than eight bytes.

1. From the services displayed in the service definition list in the tree view, select the applicable business process.

   The properties related to the selected business process are displayed in the Properties view.

2. Select and change the desired items.

   Do not enter a single-byte or double-byte space before or after each entry item.

(2) Modifying the definition information for activities

The following two methods are available to modify the definition information for activities:

Method 1

1. From the services displayed in the service definition list in the tree view, select and double-click the applicable business process.

   The Define Business Process window opens.

2. Select the applicable activity on the canvas.
3. Double-click the activity, or right-click it and then select **Setting**.

   A dialog box for the activity opens.

4. Modify the contents.

Method 2

For activities other than the start and end activities, you can change the service name from the Properties view. To change any information other than service names, use Method 1.

To change the definition information for activities from the Properties view:

1. Select one of the activities on the canvas of the Define Business Process window.
   The properties related to the selected activity are displayed in the Properties view.
2. Double-click and change the desired items.
   Do not enter a single-byte or double-byte space before or after each entry item.

5.9.2 Modifying Activity Names

You can modify the names of all activities except start and end activities.

Activity names must be unique within each business process. Do not enter a single-byte or double-byte space before or after each entry item.

! Important note

If items such as activity names have been changed in the Properties view, apply the changes before performing other operations.

The following two methods are available to modify activity names:

Method 1

1. Select the applicable activity on the canvas.
2. With the activity selected, left-click or press the **F2** key.
3. Edit the activity name.
4. Press the **Enter** key or move the cursor off the activity name box.
   The activity name is modified.

If the modified activity name already exists within the business process or is left blank, an error occurs, and the activity name reverts to its original name.

Method 2

1. Select the applicable activity on the canvas.
   The properties related to the selected activity are displayed in the Properties view.
2. Select an activity name displayed in the Properties view and edit it.
3. Press the **Enter** key or move the cursor off the activity name box.

5.9.3 Changing a Running Business Process Definition

For changing the business process definition being operated according to the changes or enhancement in business, select the change method from the following, depending on the operation after change:

(i) When invoking the business process before change as a new process

Create and operate a new business process, as a business process after change. You must distinguish the service name and business process name in the business process before and after change.

(ii) When it is not possible to maintain uniqueness of the business process before change and correlation set

Create and operate a new business process, as a business process after change. You must distinguish the service name and business process name in the business process before and after change.

(iii) When all the processes belonging to the business process definition before change are complete

Delete all the process belonging to the business process before change and replace with the business process definition after change.

(iv) When you can change the service requester in cases other than (i)~(iii)

Create and operate a new business process, as a business process after change. You must distinguish the service name and business process name in the business process before and after change.

(v) When you cannot change the service requester in cases other than (i)~(iii)

Create the business processes for which version is upgraded and operate in parallel. You must change the service adapter according to the change contents after version upgradation.

Respective methods are as follows:

(1) Method to create and operate a new business process

In case of (i), (ii) and (iv)m create a new business process and operate the business process with service name and business process definition name after change. Procedure is as follows:

1. Export the repository information from the operating environment and import to the development environment.
2. Create a new business process based on the existing business process.
   Note You must distinguish the service name and business process name in the new and existing business process.
3. Perform packaging of the business process.
   For details on packaging, see "*7.2 Packaging*".
4. Perform deployment definition of the business process.
   For details on the deployment definition, see "*7.3 Defining Deployment of HCSC Components*".
5. Export the repository information that includes the business process created in step 2. And import the operating environment.
6. Deploy the business process from operating environment to execution environment.
   For details on the operations in the operating environment, see "*3.1.13 Deploying the business process*" in the "*Service Platform System Setup and Operation Guide*".

(2) Method of deleting the existing business process and replacing with latest business process

In case of (iii), delete the existing business process from the execution environment and replace it with the latest business process. Procedure is as follows:

1. Export the repository information from the operating environment and import to the development environment.
2. Delete the deployment information of the existing business process.
   For details on deleting the deployment information of business process, see "*7.3 Defining Deployment of HCSC Components*".
3. Change the existing business process definition, in the Business process definition screen.
4. Perform packaging of the business process.
   For packaging, see "*7.2 Packaging*".
5. Perform deployment definition of the business process.
   For details on deployment definition, see "*7.3 Defining Deployment of HCSC Components*".
6. In operating environment, check whether all the processes belonging to the existing business process definition are complete and delete the HCSC component deployed to the execution environment.
7. Export the repository information that includes the business process changed in step 2. And step 3. And import to the operating environment.
8. Deploy the business process from operating environment to execution environment.
   For operations of operating environment, see "*3.1.13 Deploying the business process*" in the "*Service Platform System Setup and Operation Guide*".

(3) Method of operating in parallel

In case of (v), operate the existing business process and business process, version of which is upgraded, in parallel. Before upgrading the version of the business process, check "*5.9.4 Upgrading version of business processes*".

1. Export the repository information from the operating environment, and import into the development environment.
2. Delete the deployment definition of the business process.

   For details on deleting the deployment definition of the business process, see *7.3 Defining Deployment of HCSC Components*.
3. Right click the business process in the tree view service definition list and choose **Upgrade**.
4. Edit the business process chooseed in step 3. in the Business Process Definition screen.
5. Package the business process.

   For details about packaging, see *7.2 Packaging*.
6. Create a deployment definition for the business process.

   For details about deployment definition, see *7.3 Defining Deployment of HCSC Components*.
7. Stop and delete the business process existing in the operating environment.
8. Export the repository information containing the business process changed in step 2. up to step 4. and import into the operating environment.
9. Deploy the business process from the operating environment to the execution environment.

   For operations in the operating environment, see "3.1.13 Deploying the business process" in the "Service Platform System Setup and Operation Guide".

5.9.4 Upgrading version of business processes

This section describes the operations that you can implement, operations that you cannot implement after upgrading the version of business process and precautions to be taken when upgrading the version.

(1) Operations that you can implement after the version upgradation

This table describes the operations that you can implement after upgrading the version of business process.

Table 5–16: TableOperations that you can implement after upgrading the version

#	Target activity	Operations	Remarks
1	Receive activity[#1]	Changing the activity name	-
2		Changing the operation name	In some cases, you require to change the operation name, depending on the changes after upgrading the version. For operation names that must be changed, see #4, #7, #15 and #18.
3			When you use the user-defined reception, you must add a new user reception.[#3][#6]
4		Changing the body assigned variable	When you change the body assigned variable specified in the reference destination of the correlation set, you must change the operation name.
5			When you use the user-defined reception, you must add a new user-defined reception.[#6]
6		Adding, changing or deleting the header assigned variable	When you use the SOAP reception, you must add a new user-defined reception.[#4][#6]
7		Adding, changing and deleting the assigned correlation set	When you add, change or delete a assigned correlation sets, you must change the operation name. Accordingly, when you use the user-defined reception, you must add a new user-defined reception.[#6]
8			Correlation set (pair of name and value) must be unique in all the versions.
9		Changing the communication model	When you use the user-defined reception, you must add a new user-defined reception.[#6]

#	Target activity	Operations	Remarks
10	Receive activity#1	Changing the instance generation	-
11		Adding and deleting an activity	-
12	Reply activity#1	Changing the activity name	-
13		Changing the operation name	You might have to change the operation name depending on the change contents after upgrading the version. For the operations that require change of operation name, see #4, #7, #15 and #18.
14			When you use the user-defined reception, you must add a new user-defined reception.#3#6
15		Changing the body assigned variable	When you change the body assigned variable specified at the correlation set reference destination, you must change the operation name.
16			When you use the user-defined reception, you must add a new user-defined reception.#6
17		Adding, changing and deleting the header assigned variable	When you use the SOAP reception, you must add a new user-defined reception.#4#6
18		Adding, changing and deleting the allocation correlation set	When you add, change or delete an allocation correlation set, you must change the operation name. accordingly, when you use the user-defined reception, you must add a new user-defined reception.#6
19			Correlation set (pair of name and value) must be unique in all the versions.
20		Changing and deleting the fault name	When you use SOAP reception, you must add a new user-defined reception.#5#6
21		Adding and deleting an activity	-
22	Other activities	Adding, changing and deleting an activity	When you add, change or delete the allocation correlation set in the invoke service activity, correlation set (pair of name and value) must be unique in all the versions.
23	User-defined reception#2	Adding the user-defined reception#7	You can add the user-defined reception to be used in latest version. You can also add the user-defined reception to be shared in latest and other versions.
24		Changing the user-defined reception#7	You can change the user-defined reception regardless of version.
25		Deleting the user-defined reception	You can delete the user-defined reception used only in the latest version.

Legend:

-: Corresponding description does not exist.

#1

Target version for performing operation, is the latest version. You cannot perform operations like reference or change in versions other than the latest version.

#2

You can reference the user-defined reception regardless of the version.

#3

This is not required when you change other operation names defined in the user-defined reception or add operation name to user-defined reception, by using the user-defined reception.

#4

This is not required when data validation function is inactive (when "IGNORE" is specified in the telegram-notfound-soapheader property of the HCSC server random definition file) and change does not have impact on operation.

#5

This is not required when changing the other operation name defined in the user-defined reception.

#6

To add a new user-defined reception, see "*(3)(b) Notes when using the user-defined reception*".

#7

You must define the user-defined reception by considering the definition contents of corresponding business process. Comtability with version other than latest version is checked in the validation of user-defined reception. For details on validation of the user-defined reception, see "2.10 Validating the user-defined reception" in the "*Service Platform Reception and Adapter Development Guide*".

(2) Operations that you cannot implement after upgrading a version

Following table describes the operation that you cannot implement after upgrading the version of business process.

Table 5–17: TableOperations that you cannot implement after upgrading a version

#	Target activity	Operation	Remarks
1	Business process#1	Changing the service ID	-
2		Changing the persistence of status	-
3		Changing the business process	-
4	User-defined reception#2	Adding the user-defined reception	You cannot add the user-defined reception used in versions other than the latest version.
5		Deleting the user-defined reception	You cannot delete the user-defined reception defined when upgrading the version.

Legend:

-: Corresponding description does not exist.

#1

Target version for performing operation, is the latest version. You cannot perform operations like reference or change in versions other than the latest version.

#2

You can reference the user-defined reception regardless of version.

When you upgrade a version of business process, you cannot use following functions in the business process having version other than latest version.

- Debugging the business process
- Invoking the latest process instance

(3) Points to be noted when upgrading the version of business process

When you upgrade a version of business process, you might not be able to execute the business process of version other than latest version, depending on the change contents after version upgradation.

Change the definition information after checking the following contents and verifying the impact of changes. It is recommended to acquire the version upgradation of the definition information, before performing the version upgradation.

(a) Notes when changing the business process definition in the latest version

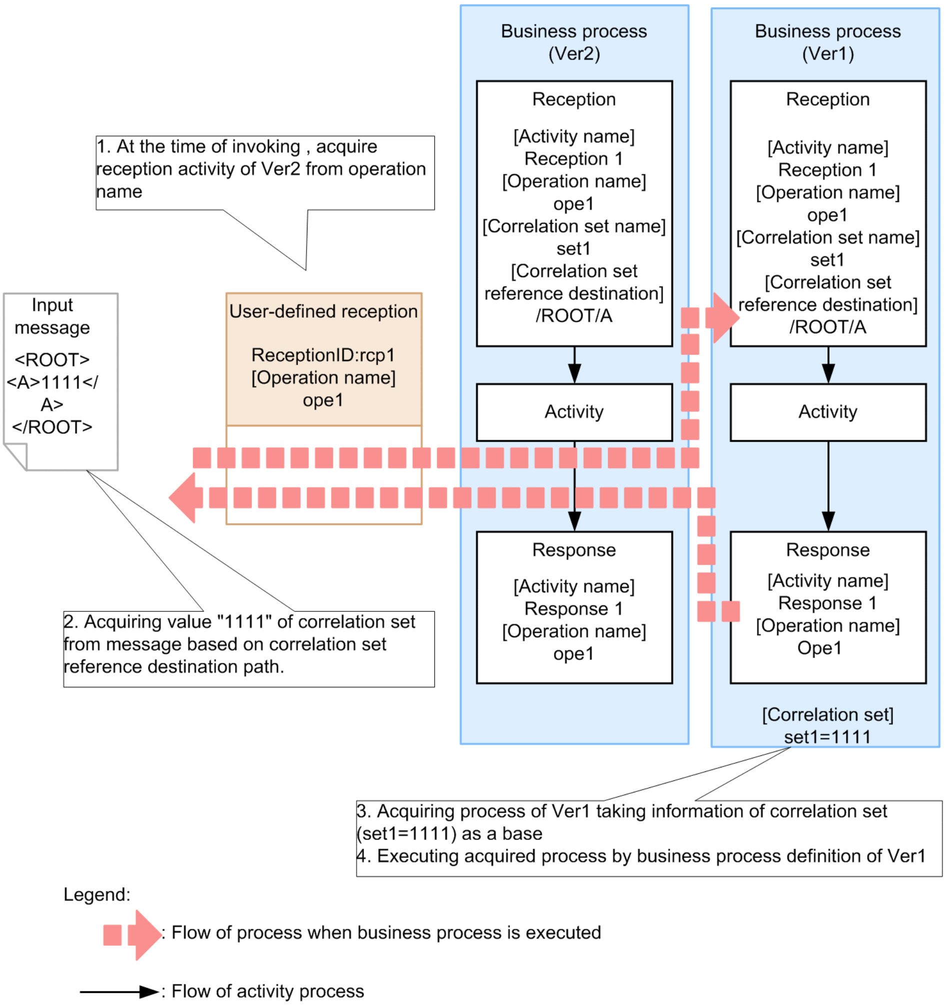
You can execute the business process of version other than latest version, with the following flow.

1. From the latest version, reference the business process of a version other than latest version, based on the operation specified at the time of invoking and acquire the reception activity for which the corresponding operation name has been defined.
2. Acquire the value of correlation set from the message, based on the correlation set acquisition position of reception activity.
3. Acquire the process being executed, based on the value of correlation set.
4. Compare the versions of acquired process sequentially from the latest version and if matching version is found, continue process with business process definition of the concerned version.

If you change the business process definition of the latest version after upgrading version of the business process in the environment, having a process instance for which versions other than latest version are not complete, change the business process definition, by considering the impact of above-mentioned process and also the changes done.

Following figure shows the flow of process when executing the business process in version other than latest version, after upgrading the version.

Figure 5–18: FigureFlow of the process when executing the business process of the version other than latest version, after upgrading the version


Business process (Ver2)
Reception
[Activity name] Reception 1
[Operation name] ope1
[Correlation set name] set1
[Correlation set reference destination] /ROOT/A
Activity
Response
[Activity name] Response 1
[Operation name] ope1
Business process (Ver1)
Reception
[Activity name] Reception 1
[Operation name] ope1
[Correlation set name] set1
[Correlation set reference destination] /ROOT/A
Activity
Response
[Activity name] Response 1
[Operation name] Ope1
[Correlation set] set1=1111
1. At the time of invoking , acquire reception activity of Ver2 from operation name
Input message
<ROOT> <A>1111</A> </ROOT>
User-defined reception
ReceptionID:rcp1
[Operation name] ope1
2. Acquiring value "1111" of correlation set from message based on correlation set reference destination path.
3. Acquiring process of Ver1 taking information of correlation set (set1=1111) as a base
4. Executing acquired process by business process definition of Ver1
Legend:
: Flow of process when business process is executed
: Flow of activity process

(b) Notes when using the user-defined reception

When you implement the following operations after upgrading version of the business process in which user-defined reception is used, you must retain the user-defined reception used in the versions other than latest version as it is and add a new user-defined reception to be used in the latest version.

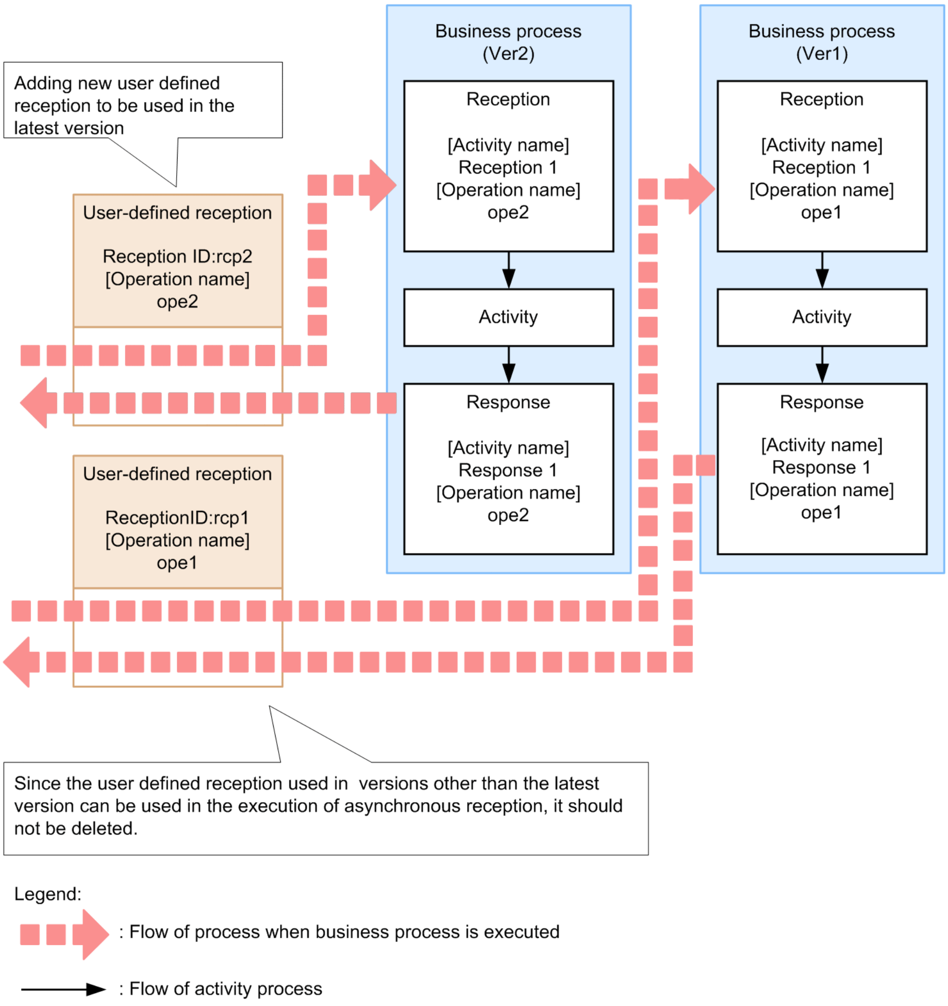
- Changing the operation name

- Changing the body assigned variable
- Adding, changing and deleting the header assigned variable
- Adding, changing and deleting the assigned correlation set
- Changing and deleting the fault name

However, you might not be required to add a new user-defined reception, even in the cases mentioned above. For details, see remarks and notes in "*Table5-16 Operations that you can implement after upgrading the version*".

Following figure shows the example of changing business process when it is required to add a new user-defined reception after upgrading the version.

Figure 5–19: FigureExample of change when it is required to add a new user-defined reception

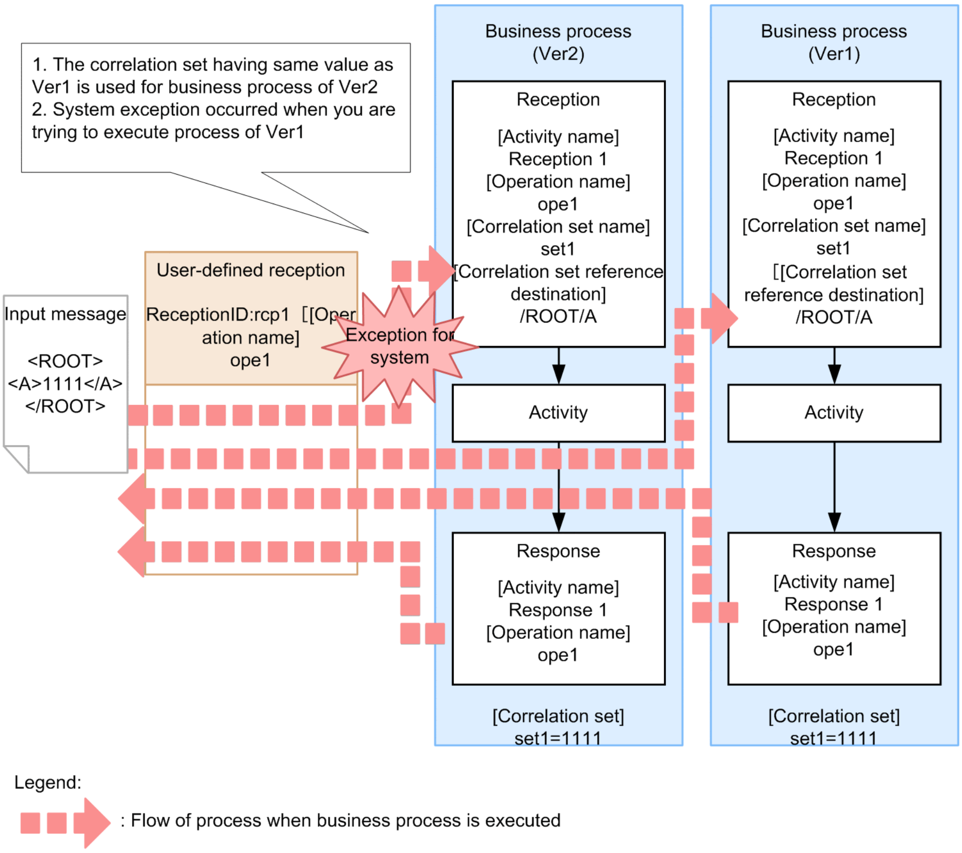

Business process (Ver2)
Business process (Ver1)
Adding new user defined reception to be used in the latest version
Reception
[Activity name] Reception 1
[Operation name] ope2
Reception
[Activity name] Reception 1
[Operation name] ope1
User-defined reception
Reception ID:rcp2
[Operation name] ope2
Activity
Activity
Response
[Activity name] Response 1
[Operation name] ope2
Response
[Activity name] Response 1
[Operation name] ope1
User-defined reception
ReceptionID:rcp1
[Operation name] ope1
Since the user defined reception used in versions other than the latest version can be used in the execution of asynchronous reception, it should not be deleted.
Legend:
: Flow of process when business process is executed
: Flow of activity process

(c) Notes related to uniqueness of correlation set

If uniqueness of correlation set (pair of name and value) is not maintained after upgrading the version, unexpected process might be invoked in the latest version and the version other than latest version.

Following figure shows the operations in case of specifying correlation set of value same as value before version upgdaration, after upgrading the version.

Figure 5–20: FigureWhen uniqueness of correlation set is not maintained (set up the correlation set in the reception)


1. The correlation set having same value as Ver1 is used for business process of Ver2
2. System exception occurred when you are trying to execute process of Ver1
Business process (Ver2)
Business process (Ver1)
Reception
[Activity name] Reception 1 [Operation name] ope1 [Correlation set name] set1 [Correlation set reference destination] /ROOT/A
Reception
[Activity name] Reception 1 [Operation name] ope1 [Correlation set name] set1 [[Correlation set reference destination] /ROOT/A
User-defined reception
ReceptionID:rcp1 [[Operation name] ope1
Input message
<ROOT> <A>1111</A> </ROOT>
Exception for system
Activity
Activity
Response
[Activity name] Response 1 [Operation name] ope1
Response
[Activity name] Response 1 [Operation name] ope1
[Correlation set] set1=1111
[Correlation set] set1=1111
Legend:
: Flow of process when business process is executed
: Flow of activity process

(d) Notes in case of making changes related to correlation set

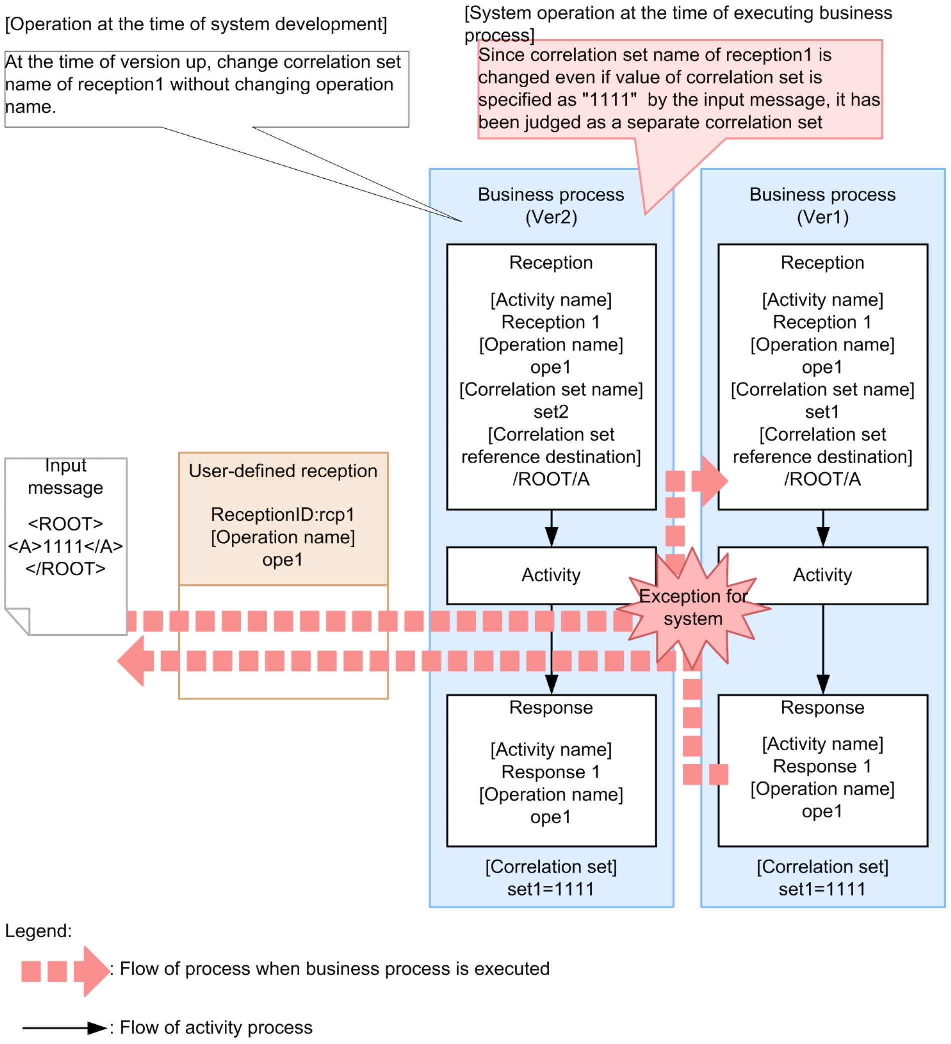
When you implement following operations after upgrading the version, value of correlation set might not match in latest version and version other than latest version.

- Changing the body assigned variable (part specification) specified in correlation set reference destination
- Adding, changing and deleting the assigned correlation set

In such cases, you might not be able to execute the process of version other than latest version. Therefore, you must change the operation name, when you make changes related to the correlation set.

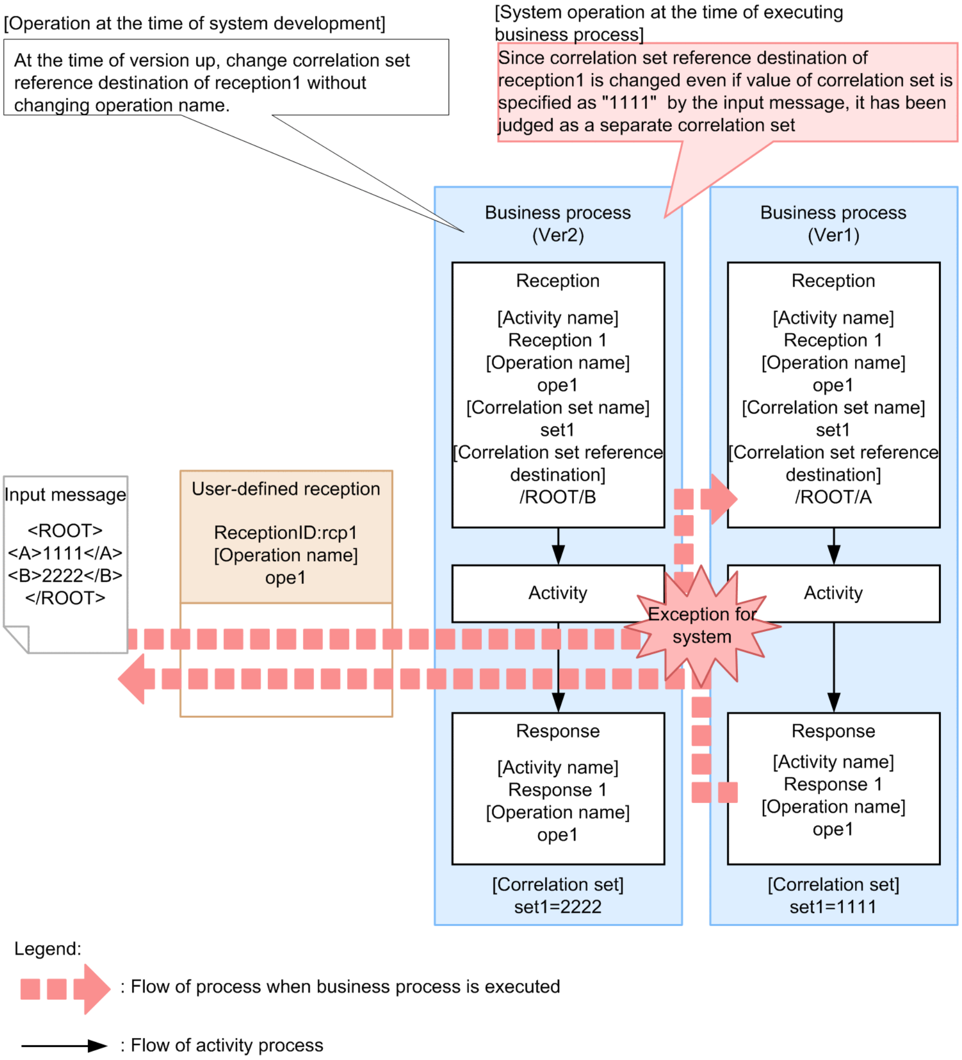
Following figure shows the operations when changing the correlation set after upgrading the version.

Figure 5–21: FigureWhen making changes related to correlation set (change the correlation set name)


[Operation at the time of system development]
At the time of version up, change correlation set name of reception1 without changing operation name.
[System operation at the time of executing business process]
Since correlation set name of reception1 is changed even if value of correlation set is specified as "1111" by the input message, it has been judged as a separate correlation set
Business process (Ver2)
Reception
[Activity name] Reception 1
[Operation name] ope1
[Correlation set name] set2
[Correlation set reference destination] /ROOT/A
Activity
Response
[Activity name] Response 1
[Operation name] ope1
[Correlation set] set1=1111
Business process (Ver1)
Reception
[Activity name] Reception 1
[Operation name] ope1
[Correlation set name] set1
[Correlation set reference destination] /ROOT/A
Activity
Response
[Activity name] Response 1
[Operation name] ope1
[Correlation set] set1=1111
Input message
<ROOT>
<A>1111</A>
</ROOT>
User-defined reception
ReceptionID:rcp1
[Operation name] ope1
Exception for system
Legend:
: Flow of process when business process is executed
: Flow of activity process

Following figure shows the operations when you change the correlation set reference destination, after upgrading the version.

Figure 5–22: FigureWhen making changes related to correlation set (change the correlation set reference destination)


[Operation at the time of system development]
At the time of version up, change correlation set reference destination of reception1 without changing operation name.
[System operation at the time of executing business process]
Since correlation set reference destination of reception1 is changed even if value of correlation set is specified as "1111" by the input message, it has been judged as a separate correlation set
Business process (Ver2)
Reception
[Activity name] Reception 1
[Operation name] ope1
[Correlation set name] set1
[Correlation set reference destination] /ROOT/B
Activity
Response
[Activity name] Response 1
[Operation name] ope1
[Correlation set] set1=2222
Business process (Ver1)
Reception
[Activity name] Reception 1
[Operation name] ope1
[Correlation set name] set1
[Correlation set reference destination] /ROOT/A
Activity
Response
[Activity name] Response 1
[Operation name] ope1
[Correlation set] set1=1111
Exception for system
Input message
<ROOT>
<A>1111</A>
<B>2222</B>
</ROOT>
User-defined reception
ReceptionID:rcp1
[Operation name]
ope1
Legend:
: Flow of process when business process is executed
: Flow of activity process

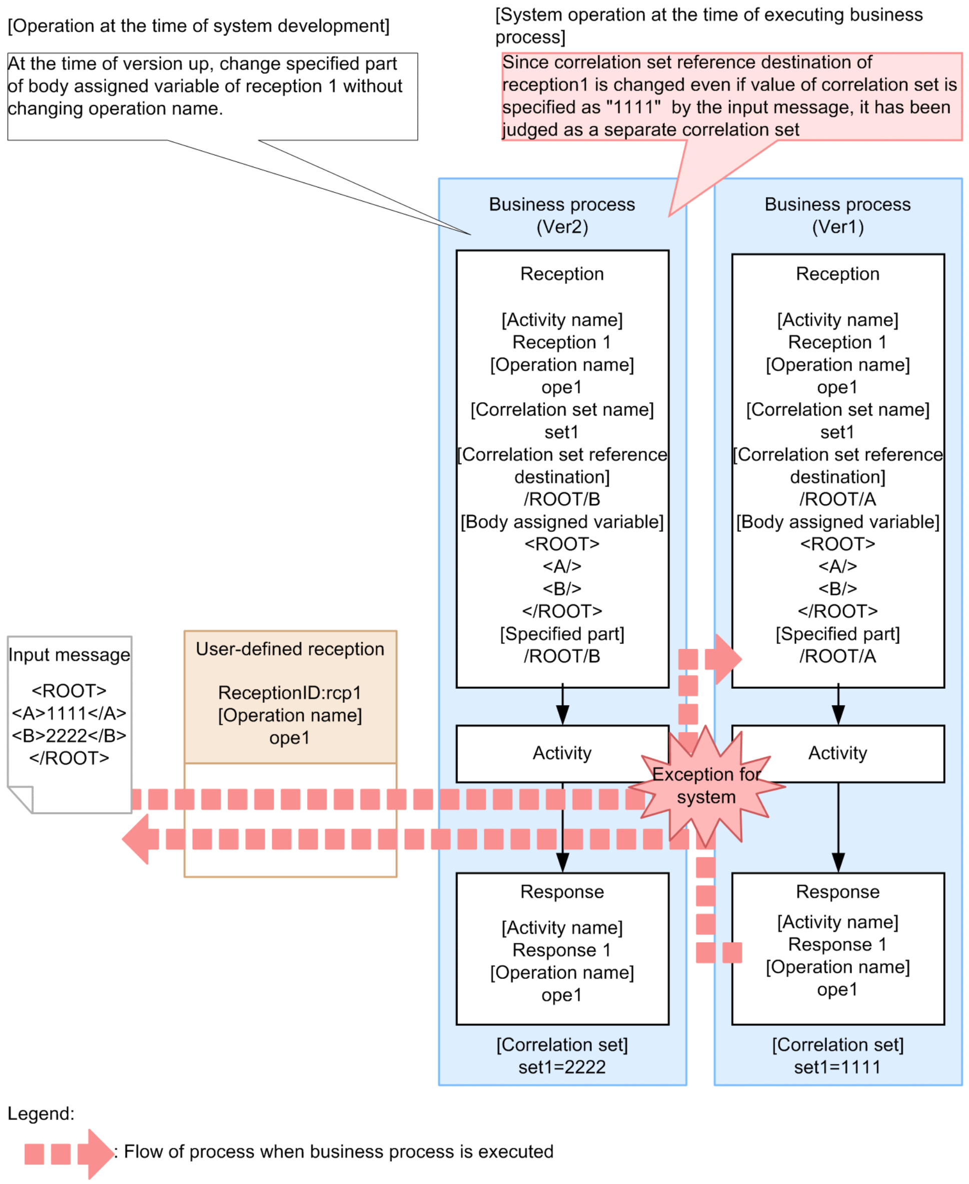
(e) Notes when making changes related to body assigned variables

After upgrading the version, when you make changes in the body assigned variable (part specification) specified in the correlation set reference destination, value of correlation set might not match in the latest version and version other than latest version. In such cases, you might not be able to execute the process of version other than latest version.

Accordingly, when you change the body assigned variable (part specification) specified in the correlation set reference destination, you must change the operation name.

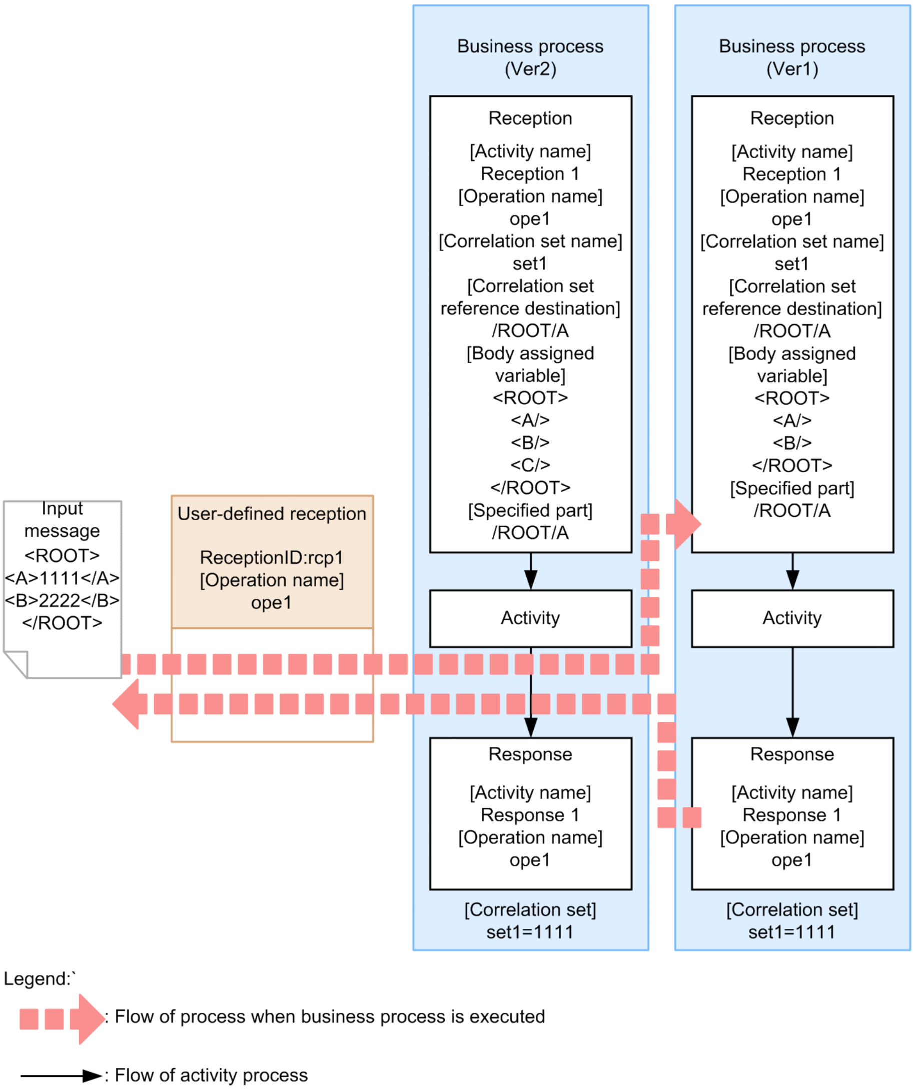
Following figure shows the operations in case of changing the body assigned variable (part specification) after upgrading the version.

Figure 5–23: FigureWhen making changes related to body assigned variable (change the part specification)



As shown in the following figure, when you change the XML schema of body assigned variable after upgrading the version, you can acquire the correlation set value and execute the business process other than that of latest version, if you are able to acquire same value in the latest version and the version other than latest.

Figure 5–24: FigureWhen making changes in the body assigned variables (change the XML schema of the variable)

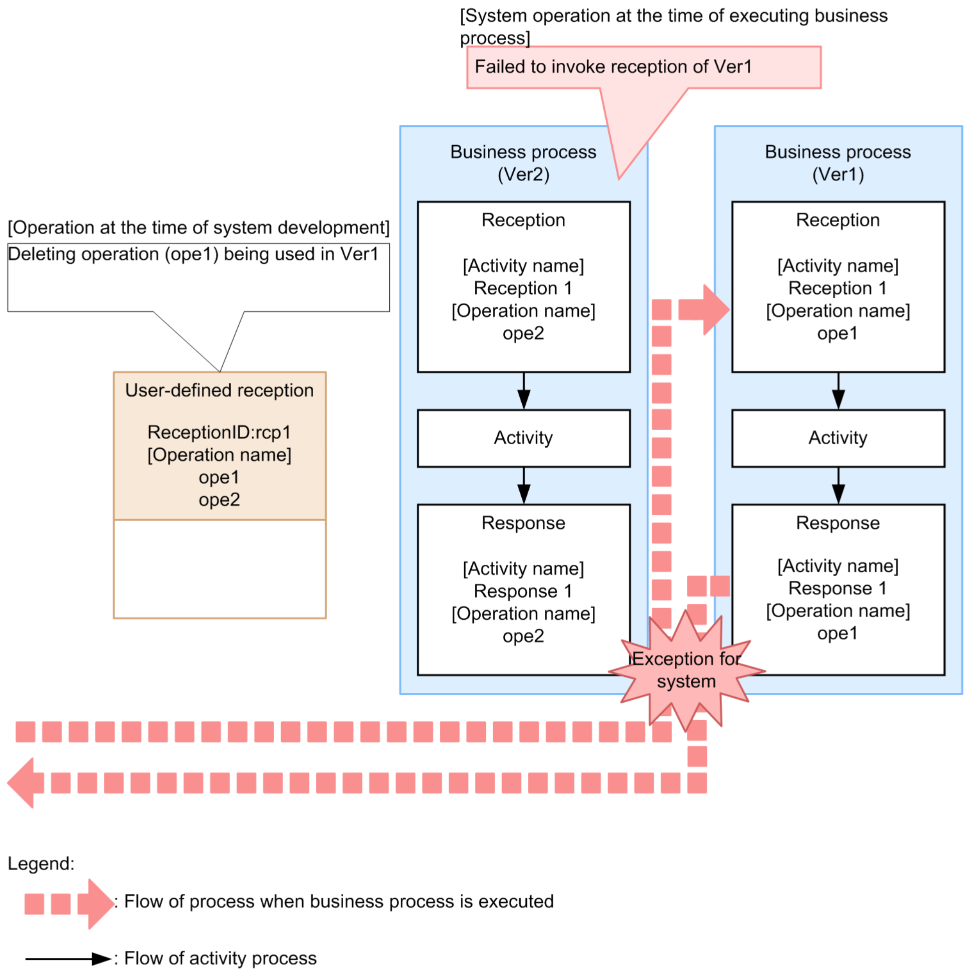

Business process (Ver2)
Reception
[Activity name] Reception 1
[Operation name] ope1
[Correlation set name] set1
[Correlation set reference destination] /ROOT/A
[Body assigned variable] <ROOT> <A/> <B/> <C/> </ROOT>
[Specified part] /ROOT/A
Activity
Response
[Activity name] Response 1
[Operation name] ope1
[Correlation set] set1=1111
Business process (Ver1)
Reception
[Activity name] Reception 1
[Operation name] ope1
[Correlation set name] set1
[Correlation set reference destination] /ROOT/A
[Body assigned variable] <ROOT> <A/> <B/> </ROOT>
[Specified part] /ROOT/A
Activity
Response
[Activity name] Response 1
[Operation name] ope1
[Correlation set] set1=1111
Input message <ROOT> <A>1111</A> <B>2222</B> </ROOT>
User-defined reception
ReceptionID:rcp1
[Operation name] ope1
Legend:`
: Flow of process when business process is executed
: Flow of activity process

(f) Notes when deleting operations of the user-defined reception

In the business process that uses the user-defined reception, if you delete the operation of the version other than latest version, after upgrading the version, you might not be able to execute the process of version other than latest version.

Following figure shows the operations in case of deleting the operation of user-defined reception, after upgrading the version.

Figure 5–25: FigureWhen deleting the operation of user-defined reception



5.10 Validating Business Processes

You can check the contents of a created business process for validity.

If the required business process definitions are missing, or if the definition relationship is invalid, the business process cannot operate normally. Therefore, before executing business processes in the execution environment, you must validate all business process definitions.

You validate whether all of the required items are present in the business processes you have created and whether their relationship is valid. You can perform validation at any time, as needed.

Validate the business processes as follows:

- Validation of the business process definition

  Validate the defined contents of the business process. Validate the mandatory items, whether the business process is structured, and perform other validations.
- Validation of the invoked service component

  Validate the compliance with service components invoked by the invoke service activity.
- Java validation

  Validate Java classes used in the invoke Java activity and the library used from Java classes.
- Validation of data transformation definition

  Validate the contents of the data transformation definition of data transformation activities.
- Validation of user-defined reception

  Validate the defined contents of the user-defined reception included in the business processes to be validated.

For details on the contents of business process definition, service component invocation, and Java validation, see *5.10.1 Validation Contents*.

For the contents of user-defined reception validation, see *8.7 Validating a User-Defined Reception*.

5.10.1 Validation Contents

In business process validation, the following items are validated:

- Required items
- Whether the defined business processes are structured
- Other items

If the number of errors during validation exceeds 100, the validation process is prematurely terminated. If this occurs, you must re-evaluate the created business processes.

Even if no error occurs during business process validation, errors may occur during business process execution in some cases. In such a case, collect a log and take the necessary action according to the message that is output to the log.

Note that business process validation essentially validates only the definition information, and does not guarantee that no error occurs during business process execution. Make sure to perform adequate testing of the business processes before placing them into actual operations.

(1) Validating required items

The activities of the business processes described in the Business Process Definition screen are validated as to whether the required items are defined. Optional items are also validated if they are specified. The following table shows the required items that are validated and the corrective actions to be taken.

Table 5–18: Required items validated, and corrective actions

Item No.	Validated content	Corrective action to be taken
1	Is the length of the business process names 64 bytes or less?	Take one of the following actions: • Shorten the business process names to 64 bytes or less. • Change the character format used.
2	Is the length of the names of the variables defined in the Variable-Correlation Set dialog box 64 bytes or less?	Take one of the following actions: • Shorten the variable names to 64 bytes or less. • Change the character format used.
3	If the variable type defined in the Variable-Correlation Set dialog box is message type, is a message format corresponding to the variable specified?	In the Variable-Correlation Set dialog box, specify a message format that corresponds to the message-type variables.
4	Is a part name specified for the variable defined in the Variable-Correlation Set dialog box?	Specify a part name for the variable in the Variable-Correlation Set dialog box.
5	Is an expression specified for the part name of the variable defined in the Variable-Correlation Set dialog box?	Specify an expression for the part name of the variable in the Variable-Correlation Set dialog box.
6	Is a type specified for the part name of the variable defined in the Variable-Correlation Set dialog box?	Specify a type for the part name of the variable in the Variable-Correlation Set dialog box.
7	Is the length of the name of the correlation set defined in the Variable-Correlation Set dialog box 64 bytes or less?	Take one of the following actions: • Shorten the correlation set name to 64 bytes or less. • Change the character format used.
8	Is at least one part name specified for the correlation set defined in the Variable-Correlation Set dialog box?	Specify at least one part name for the correlation set in the Variable-Correlation Set dialog box.
9	Is a variable name specified for the correlation set defined in the Variable-Correlation Set dialog box?	Specify a variable name for the correlation set in the Variable-Correlation Set dialog box.
10	Is a valid variable name specified for the correlation set defined in the Variable-Correlation Set dialog box?	Specify a valid variable name in the Variable-Correlation Set dialog box.
11	Is there a basic type part in the variable specified in the correlation set defined in the Variable-Correlation Set dialog box?	Specify a variable having a basic type part in the Variable-Correlation Set dialog box.
12	Is a part name specified for the correlation set defined in the Variable-Correlation Set dialog box?	Specify a part name for the correlation set in the Variable-Correlation Set dialog box.
13	Is a valid part name specified for the correlation set defined in the Variable-Correlation Set dialog box?	Specify a valid part name in the Variable-Correlation Set dialog box.
14	Is a valid variable specified for fault handling of activities?	Specify a valid variable in the Fault Handler dialog box.
15	Is a transition destination specified for fault handling of activities?	Specify a transition destination in the Fault Handler dialog box.
16	Is an allocated variable specified for fault handling of activities?	Specify an allocated variable in the Fault Handler dialog box.
17	Is a transition destination specified for fault handling of activities?	Specify a transition destination in the Fault Handler dialog box.
18	Is an operation name specified for a receive activity?	Specify an operation name for the receive activity.
19	Is the length of the activity names 64 bytes or less?	Take one of the following actions: • Shorten the activity names to 64 bytes or less.

Item No.	Validated content	Corrective action to be taken
19	Is the length of the activity names 64 bytes or less?	• Change the character format used.
20	Whether a valid header assigned variable has been specified in the receive activity	Specify a valid header assigned variable in the receive activity.
21	Whether a valid root element and namespace has been specified in the receive activity	Specify a valid root element and namespace in the receive activity.
22	Whether a valid body assigned variable has been specified in the receive activity	Specify a valid body assigned variable in the receive activity.
23	Whether a valid allocation correlation set has been specified in the receive activity	Specify a valid allocation correlation set in the receive activity.
24	Is a link name specified for a link connection?	Specify a link name for the link connection.
25	When yes is specified for the transition condition at the link destination of a link connection, is XPath specified?	Take one of the following actions for the link connection: • Specify a transition condition. • Choose no for the transition condition.
26	Is an operation name specified for a reply activity?	Specify an operation name for the reply activity.
27	Whether a valid header assigned variable has been specified in the reply activity.	Specify valid header assigned variable in the reply activity.
28	Whether valid root element and namespace has been specified in the reply activity.	Specify valid root element and namespace in the reply activity.
29	Whether valid body assigned variable has been specified in the reply activity.	Specify valid body assigned variable in the reply activity.
30	Whether a valid allocation correlation set has been specified in the reply activity.	Specify valid allocation correlation set variable in the reply activity.
31	Whether body assigned variable has been specified in the reply activity.	Specify body assigned variable in the reply activity.
32	Is a service name to be invoked specified for a invoke service activity?	Specify a service name to be invoked for the invoke service activity.
33	Whether header assigned variable for a valid request message has been specified in the invoke service activity.	Specify header assigned variable for a valid request message in the invoke service activity.
34	Whether root element and namespace for a valid request message has been specified in the invoke service activity.	Specify root element and namespace for a valid request message in the invoke service activity.
35	Whether header assigned variable for a valid response message has been specified in the invoke service activity.	Specify header assigned variable for a valid response message in the invoke service activity.
36	Whether root element and namespace for a valid response message has been specified in the invoke service activity.	Specify root element and namespace for a valid response message in the invoke service activity.
37	Whether header assigned variable for a valid request message has been specified in the invoke service activity.	Specify body assigned variable for a valid request message in the invoke service activity.
38	Whether header assigned variable for a valid reply message has been specified in the invoke service activity.	Specify body assigned variable for a valid response message in the invoke service activity.
39	Whether header assigned variable for a valid reply message in case of synchronous invoking has been specified in the invoke service activity.	Specify header assigned variable for a valid reply message in case of synchronous invoking, in the invoke service activity.

Item No.	Validated content	Corrective action to be taken
40	Is a valid correlation set specified for a invoke service activity?	Specify a valid correlation set for the invoke service activity.
41	Is at least one assign operation specified for an assign activity?	Specify at least one assign operation for the assign activity.
42	Is a copy source specified for an assign activity?	Specify a copy source for the assign activity.
43	Is a copy destination specified for an assign activity?	Specify a copy destination for the assign activity.
44	Is a valid copy source variable specified for an assign activity?	Specify a valid copy source variable for the assign activity.
45	Is a valid part name specified in the copy source for an assign activity?	Specify a valid part name in the copy source for the assign activity.
46	Is a source variable specified for a data transformation activity?	Specify a source variable for the data transformation activity.
47	Is a valid source variable specified for a data transformation activity?	Specify a valid source variable for the data transformation activity.
48	Is data transformation definition specified for a data transformation activity?	Specify data transformation definition for the data transformation activity.
49	Is a valid data transformation definition specified for the data transformation activity?	Specify a valid definition with reference to the message output during the validation.
50	Is Variable or Expression chooseed as the copy source type for an assign activity?	Specify Variable or Expression as the copy source type for the assign activity.
51	When a copy source variable is chooseed for an assign activity, is a variable name specified?	Specify a variable name for the copy source variable chooseed for the assign activity.
52	When a copy source expression is chooseed for an assign activity, is an expression specified?	Specify a copy source expression for the assign activity.
53	When a copy destination variable name is specified for an assign activity, is a valid variable specified?	Specify a valid copy destination variable for the assign activity.
54	Is a valid destination variable specified for a data transformation activity?	Specify a valid destination variable for the data transformation activity.
55	Is a valid part name specified in the copy destination for an assign activity?	Specify a valid part name in the copy destination for the assign activity.
56	Is a variable specified for the copy destination of an assign activity or the transformation destination of a data transformation activity?	Take one of the following actions: • Specify a copy destination variable for the assign activity. • Specify a transformation destination variable for the data transformation activity.
57	Is the length of the link name specified for a flow start activity 64 bytes or less?	Take one of the following actions for the link name defined for the flow start activity: • Shorten the link name to 64 bytes or less. • Change the character format used.
58	Is at least one switch condition specified for a switch activity?	Specify at least one switch condition for the switch activity.
59	Is there any transition destination that is not assigned to a conditional switch or default in a switch activity?	Assign the transition destination to a conditional switch or default in the switch activity.
60	Is a condition name specified for a switch activity?	Specify a condition name for the switch activity.
61	Is a switch condition specified for a switch activity?	Specify a switch condition for the switch activity.

Item No.	Validated content	Corrective action to be taken
62	Whether repetition conditions have been set in the while activity (only in case of condition specification method)	Set repetition conditions in the while activity.
63	Whether repetition list has been set in the while activity(only in case of list specification method)	Set repetition list in the while activity.
64	Whether repetition element variables have been set in the while activity(only in case of list specification method)	Set repetition element variables in the while activity.
65	Whether valid variable names have been specified in repetition element variables, in the while activity(only in case of list specification method)	Set valid repetition list in the while activity.
66	Is a Java class name specified for a Java invocation activity?	Specify a Java class name for the Java invocation activity.
67	Is a valid variable for argument specified for a Java invocation activity?	Specify a valid variable for argument for the Java invocation activity.
68	Is a valid variable for return value specified for a Java invocation activity?	Specify a valid variable for return value for the Java invocation activity.
69	Is a valid allocated variable specified for a throw activity?	Specify a valid allocated variable for the throw activity.
70	Is standby time set in the standby activity?	Set standby time in the standby activity.
71	Is a directory specified in the `lib` directory of the HCSCTE project?	Remove the directory from the `lib` directory of the HCSCTE project.
72	Is a file with any of the following names specified in the `lib` directory of the HCSCTE project: • `csbdef.jar` • `cscbp_ejb.jar` • `csbjava.jar`	Either delete the file or change the file name, and make sure that the files specified on the left are not specified in the `lib` directory of the HCSCTE project.
73	Whether variables to be validated have been specified in the validate activity	Specify the variables to be validated in the valid activity.
74	Whether valid variable names have been specified in the variables to be validated, in the valid activity	Specify valid variable names in the variables to be validated, in the validate activity.

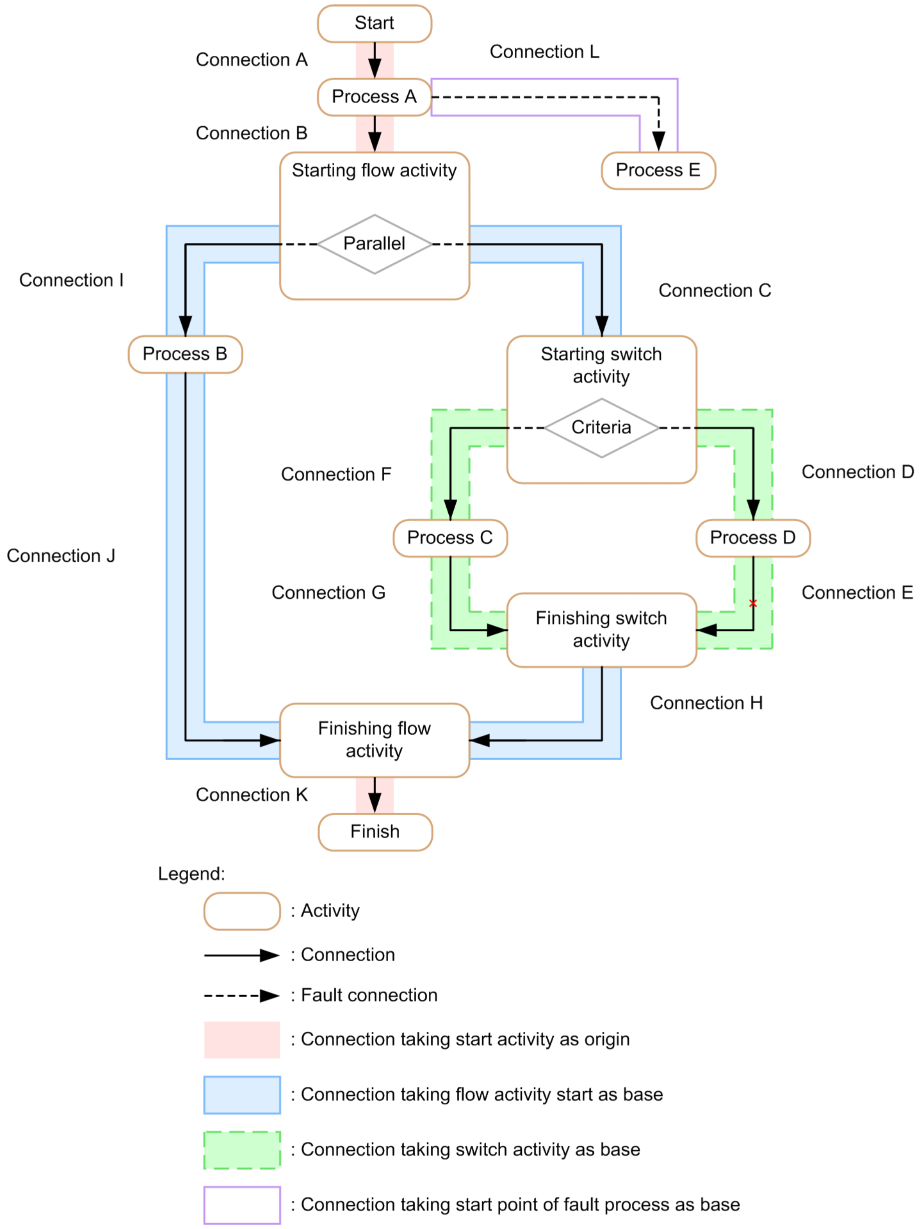
(2) Validating whether the defined business processes are structured

The structuring of business processes using connections is validated.

If an error occurs during the validation, the connection at the source of the connection that caused the error is also treated as an error.

In the example shown in the figure below, if Connection E is not present, even if normal connections are included for the switch activity and flow activities, connections that use the switch activity as their source, connections that use flow activities as their sources, and connections that use the start activity as their source are all treated as errors. However, if Process D is a throw activity, Connection E cannot be specified, and thus no error occurs.

Figure 5–26: Connection example



The following table shows the structuring contents of defined business processes that are validated and the corrective actions to be taken.

Table 5–19: Structuring contents of defined business processes that are validated, and corrective actions

Item No.	Classification	Validated content	Corrective action to be taken
1	Next connection destination	When the transition source is a start activity, is there a next connection destination?	Specify activities that are to be connected to the start activity.
2		When the transition source is a flow start activity, is there a next connection destination?	Specify activities that are to be connected to the flow start activity and link connections.
3		When the transition source is a switch start activity, is there a next connection destination?	Specify activities that are to be connected to the switch start activity.

Item No.	Classification	Validated content	Corrective action to be taken
4	Next connection destination	When the transition source is not any of the above activities, is there a next connection destination?	Specify activities that are to be connected to the transition source activity.
5	Flow process	Is a flow process connected to a switch end activity?	Connect the flow process to the corresponding flow end activity.
6		Are the connections shown below connected to a flow end activity? • Connection that uses a start activity as its source # • Connection that uses the start point of fault handling of an activity as its source #	Take one of the following actions: • Delete the flow end activity. • Change the flow end activity to a different activity.
7		Are there multiple flow end activities that are connected to the flow start activity?	Connect the flow start activity to one corresponding flow end activity.
8		Does a flow end activity receive connections from flow processes other than the corresponding ones?	Connect the corresponding flow processes to the flow end activity.
9		Is a flow process connected to an end activity?	Connect the flow process to a flow end activity.
10		Is the flow start activity connected directly to the flow end activity?	Specify at least one activity between the flow start activity and the corresponding flow end activity.
11	Switch process	Is a switch process connected to a flow end activity?	Connect the switch process to its corresponding switch end activity.
12		Are the connections shown below connected to a switch end process? • Connection that uses a start activity as its source # • Connection that uses the start point of fault handling of an activity as its source #	Take one of the following actions: • Delete the switch end activity. • Change the switch end activity to a different activity.
13		Are there multiple switch end activities that are connected to the switch start activity?	Connect the switch start activity to a single corresponding switch end activity.
14		Does a switch end activity receive connections from switch processes other than the corresponding ones?	Connect the corresponding switch process to the switch end activity.
15		Is a switch process connected to an end activity?	Connect the switch process to a switch end activity.
16		Is the switch start activity connected directly to the switch end activity?	Specify at least one activity between the switch activity and the corresponding switch end activity.
17	Fault handling	Is there an end activity in the middle of the processing of a connection that uses the start point of fault handling of an activity as the source?	Delete the end activity from the connection destination of the fault handling.

#

For details about connection that uses the start activity as the source and connection that uses the start point of fault handling of an activity as the source, see *Figure 5-26 Connection example*.

(3) Other items

The following table lists other items that are validated.

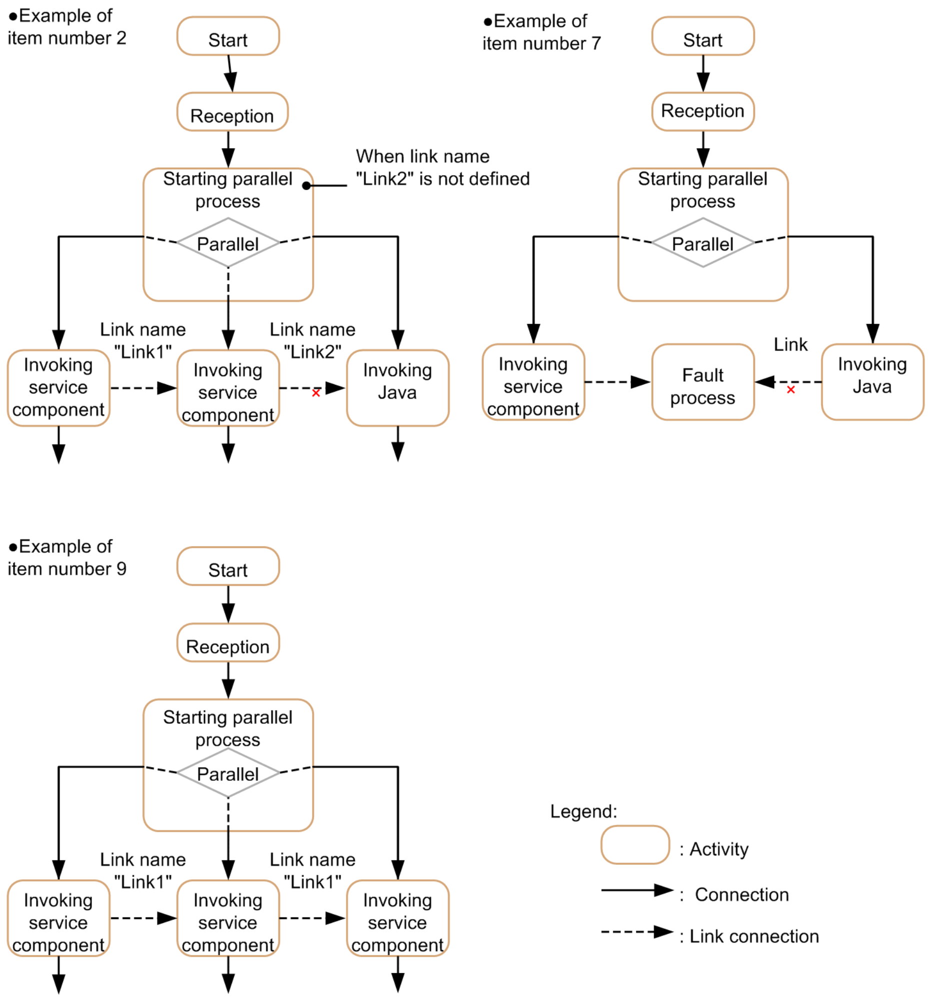
Table 5–20: Other items that are validated, and corrective actions

Item No.	Classificatio n	Validated content	Corrective action to be taken
1	Common to business processes	Is the basic activity that is executed at the start of a business process a receive activity that generates an instance?	Connect a receive activity that generates an instance next to the start activity.
2		Is a valid link name specified for a link connection?	Specify a link name at the link-source flow start activity.
3		Does the copy source of an assign activity have the same variable type as the copy destination?	In the assign activity, set the same variable type for the copy destination and the copy source.
4		Whether the same operation name has been used in multiple receive activities	Specify a unique operation name in each receive activity.
5		Do reply activities having the same operation name have the same variable type?	Specify the same allocated variable name for the reply activities having the same operation name.
6		Is there a single variable that corresponds to the fault names of the reply activities of the same operation? Example of incorrect specification: • Variables `variableX` and `variableY` correspond to fault name `faultA`. Example of correct specification: • Variables `variableX` and `variableY` correspond to fault names `faultA` and `faultB`, respectively. • Variable `variableX` corresponds to fault names `faultA` and `faultB`.	In the Fault Handler dialog box, specify a single allocated variable that corresponds to the fault names of the reply activities of the same operation.
7		Is a looping link connection specified?	Specify a link connection that does not loop.
8		Is a link connection that starts outside fault handling and ends inside it specified?	Delete the link connection that starts outside fault handling and ends inside it.
9		Is a link name used more than once?	Change the link names so that each link name is unique within a business process.
10		Is there a reply activity that corresponds to the operation of a synchronous receive activity?	Create a reply activity that corresponds to the operation of the synchronous receive activity.
11		Is there a receive activity that corresponds to the operation of a synchronous reply activity?	Take one of the following actions: • Create a receive activity that corresponds to the operation of the synchronous reply activity. • Change the operation name of the reply activity.
12		Do all receive activities that generate instances use the same correlation set?	Specify that all receive activities that generate instances use the same correlation set.
13		Is there a receive activity that generates an instance?	Take one of the following actions: • For receive activities, change Instance generation to `yes`. • Create a receive activity that generates an instance.

Item No.	Classificatio n	Validated content	Corrective action to be taken
14	Common to business processes	Is there an activity that cannot be reached even when connections are traced from a business process starting point?	Take one of the following actions: • Delete the activity that cannot be reached. • Specify connections so that the activity can be reached.
15		Is a fault variable sent out by a throw activity caught by the fault handling of a higher-order scope?	Take one of the following actions: • Specify an appropriate allocated variable for the fault handling. • Specify `catch-all` as an allocated variable for the fault handling.
16		Whether same link name has not been defined in multiple start flow activities	Specify link name such that it is unique in the business process
17		Whether the variable part fulfills the following conditions: • When type of part is XML type, part name should be unique in the business process • When type of part is other than XML type (numeric, string or boolean), part name should be unique between the variables having same message format • When the part name is same, type of part should also be the same	Set the part of variable such that all the conditions shown in the variable contents are fulfilled.
18		Whether the invalid activity has been defined in the scope activity for which [Commit only at the time of start and end of this scope] radio button is selected	Delete following activities defined in the scope activity for which [Commit only at the time of start and end of this scope] radio button is selected. • Receive activity • Reply activity • Standby activity
19	In case of persistence business process	Whether while activity of list specified format exist in the persistence business process	You cannot set while activity of list specified format, in the persistence process.
20	Non-persistent business processes	In a non-persistent business process, is there an activity that is executed after a reply activity?	Delete the activity that is executed after a reply activity.
21		Are there asynchronous receive activities and asynchronous service invocation activities in a non-persistent business process?	Take one of the following actions: • Change the asynchronous receive activities and asynchronous service invocation activities to synchronous. • Delete the asynchronous receive activities and asynchronous service invocation activities.
22		Does a standby activity exist in the non-persistent business process?	Delete the standby activity.

The following figure shows the statuses of the links that are the targets of the validation shown in Table 5-27 Other items that are validated, and corrective actions.

Figure 5–27: Status of the links for validation


●Example of item number 2
Start
Reception
Starting parallel process
Parallel
When link name "Link2" is not defined
Link name "Link1"
Link name "Link2"
Invoking service component
Invoking service component
Invoking Java
●Example of item number 7
Start
Reception
Starting parallel process
Parallel
Link
Invoking service component
Fault process
Invoking Java
●Example of item number 9
Start
Reception
Starting parallel process
Parallel
Link name "Link1"
Link name "Link1"
Invoking service component
Invoking service component
Invoking service component
Legend:
: Activity
: Connection
: Link connection

(4) Validation of the invoked service component

The service name and operation name of a service component allocated to the invoke service activity is validated.

The following table describes the validation contents and actions for the invoked service component:

Table 5–21: Validation contents and actions for the invoked service component

Item No.	Validated content	Corrective action to be taken
1	Does the service name assigned to the invoke service activity actually exist in the repository?	Assign a service name that actually exists in the repository to the invoke service activity.
2	If the service name assigned to the invoke service activity actually exists in the repository, does the operation name assigned to the activity exist in the invocation destination service component?	Assign an operation name that actually exists in the repository to the invoke service activity.
3	If the service name and operation name assigned to the invoke service activity actually exist in the repository, does the communication model specified in the activity match with the communication model of the invocation destination service component and operation?	Choose an appropriate service component and operation in the invoke service activity again.

(5) Java validation

The Java classes specified in the invoke Java activity are validated.

The following table describes the Java validation contents and actions:

Table 5–22: Java validation contents and actions

Item No.	Validated content	Corrective action to be taken
1	Does the class specified in Java class name of the invoke Java activity exist in the classes directory of the HCSCTE project?	Take one of the following actions: • Create the Java class from Java Edit. • Specify a class that exists in the classes directory of the HCSCTE project for the invoke Java activity.
2	Is a compilation error detected in the class# specified for the invoke Java activity?	Either eliminate the compilation error from the class specified for the invoke Java activity or specify a class in which a compilation error does not occur.

#

Only the classes specified for the Invoke Java activity are validated. The other classes referenced from these classes are not validated.

(6) Contents that cannot be validated

Even if no error occurs during business process validation, errors may occur during business process execution in some cases. In such a case, collect a log and take the necessary action according to the message that is output to the log. For details about the log output destination, see the contents about the Application Server log (system for executing J2EE applications) in the manual *Cosminexus Application Server Function Guide - Maintenance, Migration, and Compatibility*.

The following are examples in which errors do not occur during business process validation but do occur during business process execution:

- The number of instances generated by activities exceeds 32,762.
- A receive activity having the operation name specified by a service requester cannot be reached.
- A receive activity having the operation name specified by a service requester is in a standby state.
- The reply activity that corresponds to the receive activity having the operation name specified by a service requester cannot be reached.
- A basic activity other than a receive activity was executed before a receive activity that generates a process instance.
- A reply activity was executed when the service requester side was not waiting for a reply.
- The operation name of the receive activity specified by a service requester does not match the operation name of the reply activity returned as a reply to the service requester.
- The number of looping executions exceeds the limit.

5.10.2 Validation Method

To perform validation:

1. Right-click the service definition list in the tree view.
   The Service List pop-up menu opens.
2. On the pop-up menu, click Validation.
   The validation result is displayed in the console view.

Additionally, when packaging is executed, validation is automatically performed.

Note that if a business process and user-defined reception are not saved before validation, the **Save resource** dialog box appears and the definition can be saved. If multiple business processes and user-defined receptions are being edited, multiple dialog boxes appear. To save, click **Yes**. To save all in the future without displaying the confirmation dialog box, click **Yes to all** button.

5.10.3 Displaying the Validation Contents

A message indicating the result of validation is displayed in the console view. Make corrections according to the message if necessary.

The following table shows the types of messages that are displayed.

Table 5–23: Types of messages

Type	Explanation
Error	Displayed in either of the following cases: • The definition content is invalid. • Although the syntax for the definition is correct, the definition cannot be executed.
Warning	Displayed when a definition might cause an error during execution.
Information	Displays additional information.

5.11 Deleting Business Processes

You can delete business processes when you are upgrading their version.

You can delete business processes that have been defined.

(1) Deletion methods

The following two methods are available for deletion:

Method 1

1. Choose a business process from the service definition list in the tree view.
2. Click Delete.
   A deletion confirmation dialog box opens.
3. Click Yes.
   The specified business process is deleted.

Method 2

1. Choose and right-click a business process from the service definition list in the tree view.
   The **Service List** pop-up menu opens.
2. On the pop-up menu, click Delete.
   A deletion confirmation dialog box opens.
3. Click Yes.
   The specified business process is deleted.

! Important note

During version upgrading, you cannot delete just the latest version of a business process. All versions of a business process are deleted.

(2) Business processes that cannot be deleted

You cannot delete a business process for which deployment definition has already been performed.

6 Defining Data Transformation

This chapter explains the message format definition files and data transformation definition necessary for data transformation.

6.1 Files and Definitions Necessary for Data Transformation

To create files and definitions necessary for data transformation:

1. Create message format definition files.
   For details, see *6.2 Creating Message Format Definition Files*.
2. Define data transformation.
   For details, see *6.3 Defining Data Transformation*.

6.2 Creating Message Format Definition Files

This section describes the methods to create and set the message format definition files that form the basis of data transformation definitions. It also explains notes related to message format definition files and binary message formats.

(1) Creating message format definition files

Create a message format definition file that becomes the basis for defining data transformation. A message format definition file is required for the transformation-source and for the transformation-destination. For details about the types of message format definition files and creation methods, see *4. Creating Message Formats*.

For details about the message format definition file used for defining data transformation, see *2.6.5 Scoping of XML schema*.

(2) Setting message format definition files

Set up the created message format definition files in the Service Adapter Definition screen or the Business Process Definition screen. For details about the Service Adapter Definition screen and the Business Process Definition screen, see *5.3 Defining Adapter Contents* and *5.3 Defining Business Process Contents*, respectively.

(3) Notes on creating message format definition files

Format of a message format definition file

The message format definition file used for defining data transformation must satisfy the conditions described in *2.6.5 Scoping of XML schema*. For details on schema conditions, see *2.6.5 Scoping of XML schema*.

Elements not displayed on the window

For the elements annotation, appinfo, documentation, and notation, there is no tree display in the transformation-source schema tree viewer and the conversion destination schema tree viewer.

Window display of sequence element and choice element

The transformation-source and destination schema tree viewers display the sequence element or choice element as follows according to the contents of the XML schema to be defined.

- #(sequence)

  Displays the sequence element (compositor) wherein the occurrence count is fixed as once under the sequence element or choice element.
- #(choice)

  Displays the choice element (compositor) wherein the occurrence count is fixed as once under the sequence element or choice element.
- #anonymous

  Displays the sequence element, choice element or all element (compositor) wherein the occurrence count is not fixed as once.

(4) Points to be considered for binary message format for which a separator is set

When using binary message format for which separator is set, you must take note of following points.

- When "Till end of data" is defined in expression frequency and following 2 conditions are fulfilled, only 1 blank element is generated as a result of repetition.
  - Sibling element does not exist
  - Repetition end character is expressed immediately after intermediate delimitation character.
- When the binary data to be input includes byte data of value same as the separator character (including the cases when it is a part of configuration bytes of multi byte characters), analysis of binary data fails.
- Take note that even if escape character is not added, it is identified as data.

  Example: When character string same as start character is set immediately after start character

- You cannot use complex contents element in which bit column type simple contents element and size node is specified, in the binary message format definition, for which separator is set.
- Do not specify the following character code:
  - KEIS+EBCDIK
  - KEIS+EBCDIC
  - IBM_CODE+EBCDIC(LATIN)
  - IBM_CODE+EBCDIC(KANA)
  - JEF+EBCDIK
  - JEF+EBCDIC

(5) Points to be considered for message format of optional format (any format)

When you set optional format (any format) in the message format definition file, part of settings is ignored.

Following table describes the settings that are ignored.

Table 6–1: TableSettings ignored in case of optional format (any format)

Target component	Ignored settings
TP1/RPC reception	Independent definition file (cscurecptp1rpc.properties) • urecp-tp1rpc.dt-skip Independent definition file (csc_owncodeconvert.properties) • All settings
TP1 adapter	HCSC server runtime definition file (cscsvconfig.properties) • xmltelegram-maxcache-num • telegram-validation • xmltelegram-namespace-complement • telegram-undefined-character-code Independent definition file (csccustomadapter.properties) • custom-adapter.dt-skip Independent definition file (csc_owncodeconvert.properties) • All settings
HTTP adapter	
General custom adapter	

6.3 Defining Data Transformation

In the Data Transformation Definition screen, define data transformation by setting up and defining the transformation-source message format definition file and the transformation-destination message format definition file. The following definitions can be created in the Data Transformation Definition screen:

Data transformation definition

Data transformation definition refers to definition of the data transformation patterns between all types of formats in the service adapter and business process definitions.

A data transformation definition is configured from an XML schema file or a binary format definition file and the data transformation definition file.

Mapping definition

Mapping definition refers to information of the data transformation pattern of the data transformation definition.

The file exporting this information is referred to as the mapping definition file (`extension:.mdo`). The mapping definition file saves the edited results of the Data Transformation Definition screen.

The following subsection describes the procedure for defining data transformation:

6.3.1 Procedure for Defining Data Transformation

This subsection explains how to define data transformation in the Data Transformation Definition screen.

(1) Defining new data transformation

To define new data transformation:

Note that the procedure for defining data transformation varies according to the method displaying the Data Transformation Definition screen.

(a) Method for displaying from data transformation activity of the Adapter Definition screen or Business Process Definition screen

1. Open the Data Transformation Definition screen from the Service Adapter Definition screen or from a data transformation activity in the Business Process Definition screen.

   The Data Transformation Definition screen opens, and then the Root Element Chooseion dialog box opens.

   For details about the Data Transformation Definition screen and the **Root Element Chooseion** dialog box, see the manual *Cosminexus Service Platform Reference*.
2. Choose the root elements of all schema logical names.

   The OK button becomes enabled.
3. Click OK.

   The schemas of the selected root elements are displayed in a tree in the transformation-source schema tree viewer and the transformation-destination schema tree viewer.
4. Execute mapping.

   For details about mapping, see *6.4 Mapping*.
5. In the Data Transformation Definition screen, right-click an appropriate location in the transformation-source schema tree viewer, mapping viewer, or transformation-destination schema tree viewer, and choose Validation.

   Validation is executed.

! Important note

If you click **Cancel** in step 3, an error message appears. Click **OK** in the dialog box displaying the error message to reopen the **Choose root element** dialog box.

(b) To display from the Eclipse menu

1. In the Eclipse menu, choose **File**, **New** and **Others**.

The dialog box for choosing the wizard appears.

2. Choose **HCSCTE mapping definitions** and **Mapping definition file** and then click **Next**.
   The **New mapping definition file** dialog box appears.
3. Enter the directory for saving the Mapping Definition file and the file name and then click **Next**.
4. Choose the transformation-source and destination schemas and then click **End**.
   The Data Transformation Definition screen appears and then the **Choose root element** dialog box appears. For details about the Data Transformation Definition screen and the **Choose root element** dialog box, see *Cosminexus Service Platform Reference*.
5. Choose root elements of all schema logical names.
   You can now click **OK**.
6. Click OK.
   The tree of the schema of the selected root element appears in the transformation-source and destination schema tree viewers.
7. Map.
   For details about mapping, see *6.4 Mapping*.
8. Right click the applicable location in the Data Transformation Definition screen and then choose **Create data transformation definition**.
   The dialog box for specifying the directory for saving the data transformation definition and the file name appears.
9. Specify the directory for saving the data transformation definition and the file name and then click **Save**.
   Validation is executed and the data transformation definition is saved.

! Important note

If **Cancel** is clicked in Step 6., an error message appears. Click **OK** in the dialog box displaying the error message to reopen the **Choose root element** dialog box.

(2) Editing already defined data transformation

To edit already defined data transformation:

1. Open the Data Transformation Definition screen from the Service Adapter Definition screen or from a data transformation activity in the Business Process Definition screen.
   The Data Transformation Definition screen for already defined data transformation opens.
   For details about the Data Transformation Definition screen, see the manual *Cosminexus Service Platform Reference*.
2. Change the root element as needed.
   For details about how to change the root element, see *6.3.1(3) Changing the Root Element*.
3. Edit the mapping.
   For details about mapping, see *6.4 Mapping*.
4. In the Data Transformation Definition screen, right-click an appropriate location in the transformation-source schema tree viewer, mapping viewer, or transformation-destination schema tree viewer, and choose Validation.
   Validation is executed.

(3) Changing the Root Element

After you have displayed schemas in a tree view by selecting a root element, you can change the root element. The procedure for changing the root element is described below. Note that the method to change the root element varies according to the display method of the Data Transformation Definition screen.

(a) If the Data Transformation Definition screen is displayed from the data transformation activity of the Adapter Definition screen or Business Process Definition screen

1. Right-click a schema logical name in the transformation-source schema tree viewer or the transformation-destination schema tree viewer, and choose Choose Root Element.
   The Root Element Chooseion dialog box opens.

2. Change the root element of the schema logical name.
3. Click OK.

   The schemas of the changed root element are displayed as a tree in the transformation-source schema tree viewer and the transformation-destination schema tree viewer.

   If you executed mapping with the root element prior to the change, a message box opens, informing you that the content edited prior to the change will be discarded. To discard the content edited prior to the change and display the schemas of the newly selected root element, click **OK**.

(b) If the Data Transformation Definition screen is displayed from the Eclipse menu

1. Right click the schema logical name of the transformation-source and destination schema tree viewers and choose **Choose root element**.

   Setting transformation-source and destination schema files dialog box appears.
2. Choose the transformation-source and destination schemas.

   In the **Setting transformation-source and destination schema files** dialog box, the file specified previously is set. To retain the file specified previously, do not reset the file.
3. Click **OK**.

   The **Choose root element** dialog box appears.
4. Change the root element of the schema logical name.
5. Click **OK**.

   The transformation-source and destination schema tree viewers display trees of the changed root element schemas.

   If the status before change was mapped, a dialog box appears stating that contents edited before change are destroyed. To destroy contents edited before change and to open the tree of the new selected root element schema, click **OK**.

6.3.2 Procedure for defining changed message formats

If the message format changes, data transformation must be redefined. You can use a data transformation definition created earlier to redefine data transformation. A new definition is not required.

! Important note

If the message format changes when data transformation is being edited, once the Data Transformation Definition screen closes, reopen the Data Transformation Definition screen. If a new Data Transformation Definition screen appears when the Data Transformation Definition screen being edited is open, some of the changes in the message format might not be reflected.

The procedure for changing the message format and redefining data transformation is as follows:

1. Open the Data Transformation Definition screen from the data transformation activity of the Adapter Definition screen or the Business Process Definition screen.

   Before the Data Transformation Definition screen appears, a dialog box appears to confirm whether to reflect the changed message format. Click **OK** to open the next window.

   - If the root element is changed or deleted

     Before the Data Transformation Definition screen appears, the **Choose root element** dialog box appears. Choose a root element in the **Choose root element** dialog box and click **OK** to open the Data Transformation Definition screen. For the **Choose root element** dialog box, see *Cosminexus Service Platform Reference*.
   - If the root element is not to be changed or deleted

     The Data Transformation Definition screen appears.

   The displayed Data Transformation Definition screen reflects the changed message format#.
2. Map as and when required.

   For details about mapping, see *6.4 Mapping*.
3. Right click the applicable location of the data transformation-source schema tree viewer, mapping viewer or data transformation-destination schema tree viewer of the Data Transformation Definition screen and choose **Validate**.

   Validation is executed.

#

The Data Transformation Definition screen that copied mapped contents before changing the message format appears. Message formats before and after changes are compared and only the item with top priority similarity is copied. For copying mapping definitions and determining similarities, see *6.8 Copying Mapping Definitions*.

6.3.3 Points to be considered for data transformation definition

If you execute verification and packaging of data transformation definition when defining data transformation using schema file of huge message format, such as format in which element count is 2,000 or more, error occurs due to insufficient memory.

When error occurs, change the following options in <Eclipse installation directory>\eclipse\eclipse.ini file, restart Eclipse and execute again.

- -XmxNm(N is 1 or more integers (MB unit))
- -XX:MaxPermSize=Nm(N is 1 or more integers (MB unit))

Also, add the following options.

- -XssNm(N is 1 or more integers (MB unit))

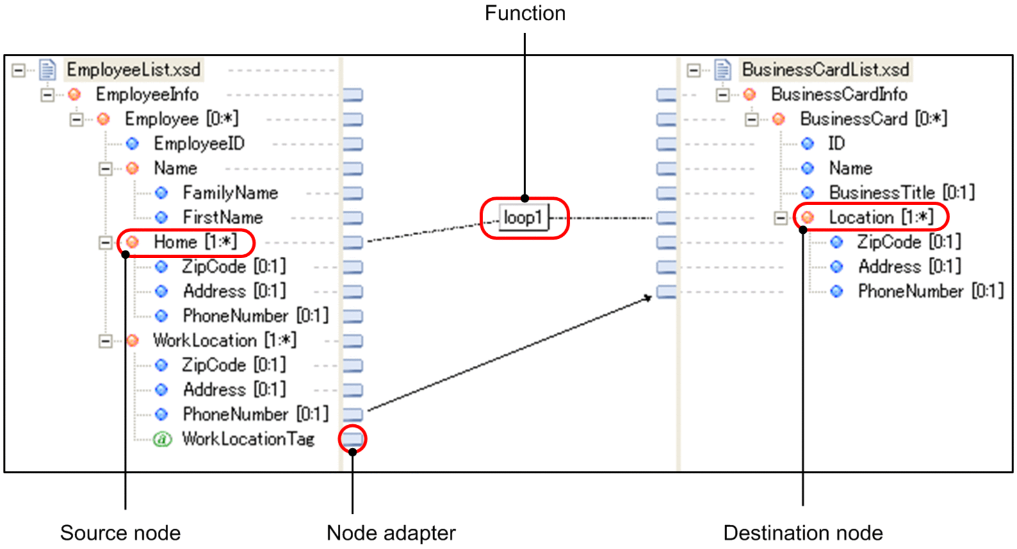
Do not specify following option:

- -Xverify:none

6.4 Mapping

Mapping means mapping a value at the transformation-source (mapping source) to a value at the transformation-destination (mapping destination). The following figure shows the Mapping screen.

Figure 6–1: Mapping screen



Mapping is largely divided into the following 4 types:

(1) Assigning transformation-source node values directly to transformation-destination nodes

Palette tools and dialog boxes might be used for the operation. For details about mapping methods, see *6.4.1 Assigning Transformation-source Node Values Directly to Transformation-destination Nodes*.

(2) Processing the transformation-source node values and mapping them to the transformation-destination node

Use the function for processing values of the transformation-source node.

Palette tools and dialog boxes might be used for the operation. For details about mapping methods, see *6.4.2 Processing the Transformation-source Node Values and Mapping Them to the Transformation-destination Node*.

(3) Specifying the scope of transformation-source and destination nodes and mapping automatically

Use dialog boxes for the operation. For details about mapping methods, see *6.4.3 Specifying the Scope of Transformation-source and Destination Nodes and Mapping Automatically*.

(4) Specifying a target from the element of the transformation-destination node and mapping automatically

Use dialog boxes for the operation. For details about mapping methods, see *6.4.4 Specifying a Target from the Element of the Transformation-destination Node and Mapping Automatically*.

For the method to cancel mapping, see *6.4.5 Canceling Mapping*.

For notes on mapping, see *6.4.10 Notes on Mapping*.

6.4.1 Assigning Transformation-source Node Values Directly to Transformation-destination Nodes

(1) Using the palette tool

To use the palette tool for mapping:

1. From the palette, choose Mapping.
2. Click the node adapter of the transformation-source node that becomes the mapping source.

For the mapping source, specify a mapping target# transformation-source node. If you try to specify a node that cannot be used as a mapping source, ⊘ is added to the cursor, preventing you from specifying the node.

3. Click the node adapter of the transformation-destination node that becomes the mapping destination.

   A mapping line ⟶ (assignment line) is set up. You can change the mapping line color. For details about how to change the mapping line color, see the manual *Cosminexus Service Platform Reference*.

   For the mapping destination, specify a mapping target# transformation-destination node that satisfies the conditions specified in *6.10.2 Correspondences Between Nodes and Functions That Can Be Mapped*. If you try to specify a node that cannot be used as a mapping destination, ⊘ is added to the cursor, preventing you from specifying the node.

#

For mapping targets, see *6.10.1 Mapping Targets and Non--Mapping Targets*.

(2) Using a dialog box

You use the Settings for Mapping Source dialog box for mapping. The mapping procedure that uses the Settings for Mapping Source dialog box is described below.

1. Right--click the transformation-destination node that becomes the mapping destination in the transformation-destination schema tree viewer, and choose Mapping source.

   The Settings for Mapping Source dialog box opens.
2. Click Add Node.

   The Node Chooseion dialog box opens.
3. Specify a transformation-source node that becomes the mapping source, and click OK.

   The path name of the node specified in Path/Function name is displayed. For details about the path name display format, see *6.4.6 Mapping Source Display Format*.

   For the mapping source, specify a mapping target# transformation-source node.
4. Click OK.

   A mapping line ⟶ (assignment line) is set up. You can change the mapping line color. For details about how to change the mapping line color, see the manual *Cosminexus Service Platform Reference*.

#

For mapping targets, see *6.10.1 Mapping Targets and Non--Mapping Targets*.

6.4.2 Processing the Transformation-source Node Values and Mapping Them to the Transformation-destination Node

To process the transformation-source node values and map them to the transformation-destination node, use the functions. For details about the functions, see *6.5 Using Functions to Process Values*.

(1) Using tools on the palette

To perform mapping by using tools on the palette:

1. From the palette, select the function to be used.
2. Click on an appropriate position in the mapping viewer.

   The function is deployed in the mapping viewer. You can move a deployed function by dragging and dropping it.
3. From the palette, select **Mapping**.
4. Click either the node adapter of the transformation-source node or a function that is to become the mapping source.

   For the mapping source, specify the following transformation-source node or function:

   - Transformation-source node

Transformation-source node used as the mapping target[#1]

- Function

  Function that does not have a mapping line set up on the output side

If you try to specify a node or function that cannot be used as a mapping source, ⊘ appears on the cursor, preventing you from specifying the node or function.

5. Click either the node adapter of the transformation-destination node or a function that is to become the mapping destination.

   A mapping line ⟶ (assignment line) is set up. If you are using a loop node function, a mapping line -·-·-·- (looping-compatible line) is set up. You can change the mapping line color. For details about how to change the mapping line color, see *1.12.5 Changing the Mapping Line Color* in the manual *uCosminexus Service Platform Reference Guide*.

   For the mapping destination, specify the following transformation-destination node or function:

   - Transformation-destination node

     A transformation-destination node that is the mapping target[#1] and satisfies the conditions specified in *6.10.2 Correspondences Between Nodes and Functions That Can Be Mapped*

   - Function

     A function that satisfies the conditions specified in *6.10.2 Correspondences Between Nodes and Functions That Can Be Mapped* and *6.10.3 Number of Mapping Lines That Can Be Connected*, and that does not form a closed route[#2] when functions are connected with each other

   If you try to specify a node or function that cannot be used as a mapping destination, ⊘ appears on the cursor, preventing you from specifying the node or function.

#1

For details about mapping targets, see *6.10.1 Mapping Targets and Non--Mapping Targets*.

#2

Mapping in which connections among functions form a closed route, such as mapping from Function A to Function B, from Function B to Function C, and then from Function C back to Function A

! Important note

If you have set up a mapping line before deploying functions, delete the mapping line, and then start again.

Reference note

When a function is deployed in the mapping viewer, a *function type* (such as `concat`) followed by a *serial number* is assigned as the function name. These serial numbers are assigned as integers to prevent function names from being duplicated. The first number is 1 and subsequent numbers are assigned in the order in which each function is mapped.

The name of each deployed function can be changed by either editing the function name directly or using a dialog box displayed by double-clicking the function name. For details about how to change function names, see *6.11 Editing function name directly*.

If a function is deleted, its name can be assigned to another function.

---

(2) Using a dialog box

As described below, the dialog box to be used for mapping differs depending on the types of mapping source and mapping destination used.

- If the mapping source is a transformation-source node and the mapping destination is a function:

  Use the dialog box for specifying a mapping destination function.

  If the mapping destination is a choose node function, see *6.5.17 Outputting Different Values According to Conditions*.

- If the mapping source and mapping destination are functions:

  Use the dialog box for specifying a mapping destination function.

If the mapping destination is a choose node function, see *6.5.17 Outputting Different Values According to Conditions*.

- If the mapping source is a function and the mapping destination is a transformation-destination node:
  Use the Settings for Mapping Source dialog box.

  If the mapping source is a loop node function or choose node function[#], you can also use the Loop Settings dialog box. For details about mapping using the Loop Settings dialog box, see *6.6.1 Mapping Using the Loop Settings Dialog Box*.

  #

  A choose node function that is connected to a loop node function or to nothing. If a choose node function is connected to a function other than a loop node function, you cannot use the Loop Settings dialog box.

(a) If the mapping source is a transformation-source node and the mapping destination is a function

To perform mapping when the mapping source is a transformation-source node and the mapping destination is a function:

1. From the palette, select the function to be used.
2. Click on an appropriate position in the mapping viewer.
   The function is deployed in the mapping viewer. You can move a deployed function by dragging and dropping it.
3. Use one of the following methods to open the dialog box for specifying a mapping destination function:
   - Right-click a mapping destination function, and then select **Setting**.
   - Double-click a mapping destination function.

   The dialog box for specifying a mapping destination function opens.
4. Click **Add Node** or **Select Node**.
   The Select Node dialog box opens.
5. Specify a transformation-source node that becomes the mapping source, and then click **OK**.
   The transformation-source node that is the mapping-source is set in **Input** (**Base path** in the case of a loop node function) in the dialog box for specifying a mapping destination function. The transformation-source node is displayed as a path name. For details about the path name display format, see *6.4.6 Mapping Source Display Format*.

   For the mapping source, specify the transformation-source node that is the mapping target[#] and satisfies the conditions specified in *6.10.2 Correspondences Between Nodes and Functions That Can Be Mapped*.
6. Click **OK**.
   A mapping line ⟶ (assignment line) is set up. If you are using a loop node function, a mapping line -·-·-·-· (looping-compatible line) is set up. You can change the mapping line color. For details about how to change the mapping line color, see *1.12.5 Changing the Mapping Line Color* in the manual *uCosminexus Service Platform Reference Guide*.

#

For details about mapping targets, see *6.10.1 Mapping Targets and Non--Mapping Targets*.

(b) If the mapping source and mapping destination are functions

To perform mapping when the mapping source and mapping destination are functions:

1. From the palette, select the function to be used.
2. Click on an appropriate position in the mapping viewer.
   The function is deployed in the mapping viewer. You can move a deployed function by dragging and dropping it.
3. Use one of the following methods to open the dialog box for specifying a mapping destination function:
   - Right-click a mapping destination function, and then select **Setting**.
   - Double-click a mapping destination function.

   The dialog box for specifying a mapping destination function opens.
4. Click **Add Function** or **Select Function**.

The Select Function dialog box opens.

5. Specify the function that is to become the mapping source, and then click **OK**.

   The mapping-source function is set in **Input** in the dialog box for specifying a mapping destination function. For a set constant function, a set constant value enclosed in single quotation marks (') is displayed. For a function that is not a set constant function, the name of the function is displayed.

   For the mapping source, specify a function that satisfies the conditions specified in *6.10.2 Correspondences Between Nodes and Functions That Can Be Mapped* and for which no mapping line is set up on the output side.

6. Click **OK**.

   A mapping line ⟶ (assignment line) is set up. If you are using a loop node function, a mapping line -·-·-·- (looping-compatible line) is set up. You can change the mapping line color. For details about how to change the mapping line color, see *1.12.5 Changing the Mapping Line Color* in the manual *uCosminexus Service Platform Reference Guide*.

(c) If the mapping source is a function and the mapping destination is a transformation-destination node

1. From the palette, select the function to be used.
2. Click on an appropriate position in the mapping viewer.

   The function is deployed in the mapping viewer. You can move a deployed function by dragging and dropping it.
3. Right-click the transformation-destination node that becomes the mapping destination in the transformation-destination schema tree viewer, and then select **Mapping source**.

   The Settings for Mapping Source dialog box opens.
4. Click **Add Function**.

   The Select Function dialog box opens.
5. Specify the function that is to become the mapping source, and then click **OK**.

   The specified function name is set in **Path/Function name**.

   For the mapping source, specify a function that satisfies the conditions specified in *6.10.2 Correspondences Between Nodes and Functions That Can Be Mapped* and for which no mapping line is set up on the output side. Note that because conditions are specified with a choose node function, you can specify a function for which a mapping line ⋯⋯> (condition line) is set up on the output side.
6. Click **OK**.

   A mapping line ⟶ (assignment line) is set up. You can change the mapping line color. For details about how to change the mapping line color, see *1.12.5 Changing the Mapping Line Color* in the manual *uCosminexus Service Platform Reference Guide*.

! Important note

If you have set up a mapping line before deploying functions, delete the mapping line, and then start again.

Reference note

When a function is deployed in the mapping viewer, a *function type* (such as `concat`) followed by a *serial number* is assigned as the function name. These serial numbers are assigned as integers to prevent function names from being duplicated. The first number is 1 and subsequent numbers are assigned in the order in which each function is mapped.

The name of each deployed function can be changed by either editing the function name directly or using a dialog box displayed by double-clicking the function name. For details about how to change function names, see *6.11 Editing function name directly*.

If a function is deleted, its name can be assigned to another function.

6.4.3 Specifying the Scope of Transformation-source and Destination Nodes and Mapping Automatically

Use the **Setting automatic mapping source** dialog box for specifying the scope of transformation-source and destination nodes and mapping automatically. The **Setting automatic mapping source** dialog box automatically maps elements with a high degree of similarity in the transformation-source and destination nodes if the scope of such nodes is specified. If a mapping target does not exist in the scope and if a mapping line is already connected in the transformation-destination node adapter, mapping is not executed. For determining similarities during mapping, see *6.4.9 Determining Similarities during Automatic Mapping*.

The mapping procedure using the **Setting automatic mapping source** dialog box is as follows:

1. Right click the element of the transformation-destination node that is the mapping destination of the transformation-destination schema tree viewer and choose **Automatic mapping**.

   Choose the top element in the scope for mapping in the elements of the transformation-destination node.

   The **Setting automatic mapping source** dialog box appears.

2. Click **Add node**.

   The **Choose node** dialog box appears.

3. Specify the transformation-source node element that is the mapping source and click **OK** button.

   Choose the top element in the scope for mapping in the elements of the transformation-source node.

   The path name of the selected element appears in **Path**. For details about the path name display format, see *6.4.6 Mapping Source Display Format*.

   To choose multiple elements, repeat step 3.

   Note that an error message is output in the following cases:

   - If the selected element is already selected
   - If the ascendant or descendant of the selected node is already selected

   After output of the error message, click OK, close the dialog box and then select the element again.

4. Click OK.

   If the mapping target exists in the specified scope, the mapping line ⟶ (assigned line) is set automatically.

   If a mapping target does not exist in the scope and if a mapping line is already connected in the transformation-destination node adapter, mapping is not executed.

Reference note

> You can change the color of the mapping line set. For details about how to change the color of the mapping line, see *Cosminexus Service Platform Reference*.

! Important note

> If multiple mapping targets exist, multiple mapping lines might be created in 1 transformation-source node because the most similar target is selected. If the mapped transformation-source node is not the target or if the mapping target is to be filtered, use the **Choose the automatic mapping candidate** dialog box or reduce the scope for automatic mapping. For details about how to map automatically using the **Choose the automatic mapping candidate** dialog box, see *6.4.4 Specifying a Target from the Element of the Transformation-destination Node and Mapping Automatically*.

6.4.4 Specifying a Target from the Element of the Transformation-destination Node and Mapping Automatically

Use the **Choose the automatic mapping candidate** dialog box for specifying a candidate from the element of the transformation-destination node and mapping automatically. If the target of the transformation-source node element is specified in the **Choose the automatic mapping candidate** dialog box, the selected candidate element is mapped and the mapping line (assigned line) is set automatically. Even if the mapping line is connected to the node adapter of the selected element, when a mapping line with a different dependent relationship in the same element can be connected, the specified candidate is mapped. For determining similarities during mapping, see *6.4.9 Determining Similarities during Automatic Mapping*.

The mapping procedure using the **Choose the automatic mapping candidate** dialog box is as follows:

1. Right click the element for mapping from the transformation-destination node that is the mapping destination of the transformation-destination schema tree viewer and choose **Choose the automatic mapping candidate**.

   If **Choose the automatic mapping candidate** cannot be selected, choose another candidate element because it means that the element cannot be mapped.

   The **Choose the automatic mapping candidate** dialog box appears.

2. Choose a mapping candidate from **List of automatic mapping candidates**.

   List of automatic mapping candidates displays the priority order of candidates from the top.

If a mapping candidate does not exist, the **List of automatic mapping candidates** does not display anything.

3. Click **OK**.

   The selected target element is mapped and the mapping line → (assigned line) is set automatically.

   Even if the mapping line is connected to the node adapter of the selected element, when a mapping line with a different dependent relationship in the same element can be connected, the specified candidate is mapped.

   Reference note

   > You can change the color of the mapping line set. For details about how to change the color of the mapping line, see *Cosminexus Service Platform Reference*.

6.4.5 Canceling Mapping

You can cancel mapping by deleting a mapping line or function.

(1) Deleting a mapping line

You can use one of the following methods to delete a mapping line:

Method 1

Right--click the mapping line to be deleted, and selectedelete.

Method 2

Choose the mapping line to be deleted, and press the Delete key.

Important note

> A user cannot delete condition lines. They are automatically deleted when the condition settings are cancelled.

(2) Deleting a function

You can use either of the following methods to delete a function:

Method 1

Right--click the function to be deleted, and selectedelete.

Method 2

Choose the function to be deleted and press the Delete key.

Important note

> If a function was specified as a condition (a function was specified for Condition in the Condition Settings dialog box), that function cannot be deleted.

6.4.6 Mapping Source Display Format

If a transformation-source node or function that is to become the mapping source is specified in the Settings for Mapping Source dialog box or the dialog box for specifying each function, the node or function is displayed as described below.

(1) If a transformation-source node is specified

The node path of a schema tree is displayed as described below.

(a) If a condition is specified for the transformation-source node

A condition enclosed in square brackets ([ ]) is displayed following the node name for which the condition is specified.

Example: `/aa/bb[position()='1']/cc`

If a function is specified as a condition (by specifying it in **Condition** in the Condition Settings dialog box), the function name enclosed in curly brackets ({}) is displayed.

Example: `/aa/bb[{length1}='5']`

If the values on the right side of the conditional expression contain ampersands (&) and apostrophes ('), these ampersands and apostrophes are displayed as entity references.

- Apostrophe ('): `&apos;`
- Ampersand (&): `&amp;`

Example:

When the left side of the conditional expression is `/root/input`, and the right side of the conditional expression is `a&'b'`

```
root/input = 'a&amp;&apos;b&apos;'
```

(b) If there are multiple schema logical names

$*schema-logical-name* is displayed at the beginning of the node path.

Example: `$source1/aa/bb/cc`

(c) If the transformation-source node is an attribute

@ is displayed before the node name.

Example: `/aa/bb/@cc`

(2) If a function is specified

The function name enclosed in curly brackets ({}) is displayed.

Example: {*function-name*}

6.4.7 Making Mapping Lines and Functions Easier to View

You can highlight only the mapping lines and functions associated with the selected node.

To highlight only the mapping lines and functions associated with the selected node:

1. Right-click on an appropriate position in the transformation-source schema tree viewer, mapping viewer, or transformation-destination schema tree viewer on the Data Transformation Definition screen.
   A pop-up menu is displayed.
2. Select **Highlight** from the pop-up menu.
   A highlight display menu is displayed.
3. Select either of the following menu options according to the usage.
   Only the related mapping lines and functions are highlighted according to the selected menu option.
   - **Highlight Relation to Transformation &Source Node**
   - **Highlight Relation to Transformation &Target Node**

   If you prefer not to highlight these mapping lines and functions, select **Disable Highlighting**.

The color brightness of highlighted mapping lines and functions differs according to the strength of the relation with the selected node. Highlighted mapping lines and functions are displayed in dark colors, whereas unhighlighted mapping lines and functions are displayed in light colors.

If you add mapping lines or functions while in highlight display mode, the display is refreshed when you reselect the node.

Reference note

If you cannot find a node that you want to select

To search for a transformation-source node or transformation-destination node:

1. Right-click the tree viewer, and then select **Search for Element Name**.
   The Search for Element Name dialog box opens.
2. Enter the character string to be searched in **Search**.
3. Select **Source node** or **Target node** in **Search for**, and then click **Search**.
   A list of node names containing the specified character string is displayed in the **Results list**.
4. Select the desired node name from the **Results list**.
   The selected node is also selected in the tree viewer.

6.4.8 Restricting mapping range

When creating data transformation definition file with the mapping definition editor, if XML schema (extension: xsd) has been specified in transformation destination schema, you can restrict the note to be mapped. You can use this function only when output file of transformation operation of file operation adapter is XML format.

Operation procedure is as follows:

1. Start the mapping definition editor.
   For details on how to start the mapping definition editor, see "*6.3.1(1)(b) To display from the Eclipse menu*".
2. Right click on elements (simple contents element or complex contents element) of transformation destination node handled by the Transformation destination schema viewer as records.
   Popup menu is displayed.
3. From pop up menu, select **Include grandchild node in mapping target**.
   Transformation destination nodes other than the set element and its grandchild elements are disabled.
   When mapping line is connected to the transformation destination node other than the set node and its grandchild node, confirmation dialog is displayed. If you click **Yes** button, mapping line connected to the transformation destination node is deleted and the transformation destination node other than the set element and is grandchild element are disabled.
   For disabled transformation destination node, error does not occur during verification, even if mapping line is not connection.

! Important note

- Specify the path of element to be handled as record, in the file operation adapter definition file. For details on the file operation adapter definition file, see "*File operation adapter definition file*" in "*Service Platform Reference Guide*".
- When you re-select the root element, settings of **Include grandchild node in mapping target** are reset.
- When you copy the mapping definition, status of setting **Include grandchild node in mapping target** is inherited to the mapping definition of copy destination.

6.4.9 Determining Similarities during Automatic Mapping

In automatic mapping, similarities between the schema elements of the transformation-source and destination nodes are determined and mapped. If many similar elements exist, they are mapped in priority order.

The following table shows the priority order for determining the similarities of schema elements of transformation-source nodes. Note that level 1 of the priority order is the highest and reduces in proportion to the value of the level.

Table 6–2: Priority order of similarities determined by automatic mapping

Priority order	Whole path		Terminal element	
	Name space	Element name	Name space	Element name
1	Y	Y	--	--
2	N	Y	--	--
3	--	C#	Y	Y

Priority order	Whole path		Terminal element	
	Name space	Element name	Name space	Element name
4	--	C#	N	Y
5	--	C#	N	C

(Legend)

Y: Matches.

C: Matches partially.

N: Does not match.

--: Not applicable.

#

Might not match depending on determination.

6.4.10 Notes on Mapping

After mapping is executed, you must confirm that the data transformation definition content is valid. When mapping, to avoid errors during validation, take the following points into consideration:

- Execute mapping according to the conditions specified in *6.10 Mapping Conditions*.
- Match the occurrence count of the transformation-source nodes at the mapping source to the occurrence count of the transformation-destination nodes at the mapping destination.
- Any complex content element whose minimum occurrence count is set to 2 or greater must be mapped, except when a node, whose minimum occurrence count is 0 occurs as the ancestor.
- Any simple content element (including the any element) whose minimum occurrence count is set to 1 or greater must be mapped.
- Avoid the existence of multiple any elements in the child element of an element with complex contents.
- When mapping a value to a child node of a `choice` element or to a descendant node, do not define mapping in which multiple child elements occur concurrently or in which no child element occurs.
- When mapping values to multiple child elements of a `choice` element or to a descendant node, make sure multiple child elements do not occur concurrently.
- When the minimum occurrence count of all child elements of a `choice` element is set to 1 or greater and the descendant element of one of the child elements is mapped, if you are using a choose node function for that child element, make sure multiple child elements do not occur concurrently.
- Do not define name spaces with URI of `xsl` and `xsi` specified in targetNamespace in the mapping--destination transformation-destination node.
- Even when an instance attribute (`xsi:type`, `xsi:schemaLocation`, or `xsi:noNamespaceSchemaLocation`) is defined for the mapping--source transformation-source node, it does not occur in the mapping--destination transformation-destination node.
- When the transformation-source node, in which the instance attribute `xsi:nil` is defined, is mapped in the transformation-destination node in which the `nillable` attribute is defined as `true`, the contents defined in the attribute `xsi:nil` of the transformation-source node will be applied to the transformation-destination node. In the same way, even when the transformation-source node, in which the instance attribute `xsi:nil` is defined, is mapped via the choose node function (`choose`), the contents defined in the attribute `xsi:nil` of the transformation-source node are applied to the transformation-destination node.
- Do not map from the transformation-source node with the nillable attribute defined to the transformation-destination node where the nillable attribute is not defined.
- When you change following message format while editing the mapping definition, message format before change is displayed. To reflect message format after change, close the target service and start mapping definition again.
  - Adding and removing global element
  - Changing global element other than root element

- When multiple namespace prefixes are defined for one namespace URL between schema of import destination or include destination, error occurs during verification (including packaging) of mapping definition. Use one type of namespace prefix for one namespace URI.
- When same namespace is defined in different namespace prefixes in message format schema file, root element selection dialog is displayed every time when starting the mapping definition. Do not specify two or more namespace prefixes for one namespace.
- Do not map the following #any element in data transformation definition.
    - #any element having #anonymous, #(sequence), or #(choice) in sibling elements
    - #any element existing under #(sequence) or #(choice) and having other #(sequence) or #(choice) in sibling element of #(sequence) or #(choice)
- When you specify output destination of node list type custom function to complex contents element, you cannot execute mapping to grandchild node having same dependency relation.
- Points to be considered for node list type custom function is specified in output value of selection function are as follows:
    - When mapping destination of Select function is complex contents element and other output value of Select function is "Output a value", you cannot set anything except node list type custom function.
    - When mapping destination of Select function is an attribute, you cannot set node list type custom function in the output value of Select function.

6.5 Using Functions to Process Values

To process the transformation-source node values and map them to the transformation-destination node, you use functions. This section explains the various functions to be used for various cases. For details about how to use a function for mapping, see *6.4.2 Processing the Transformation-source Node Values and Mapping Them to the Transformation-destination Node*.

The following table lists the functions.

Table 6–3: List of functions

Classificat ion	Function name	Explanation
String--based	Concatenate	Concatenates multiple strings.
	Acquire substring	Extracts a substring from a string.
	Acquire string length	Assigns a string character count to a mapping destination.
	Check string	Verifies that the specified string is contained in a string or that the string begins with the specified string, and assigns the logical value to a mapping destination.
	Trim node	Removes the leading and trailing spaces from a string. Also replaces consecutive spaces between strings with a single space.
Number--based	Convert number format	Converts the number format.
	Perform node operation	Computes (+, --, *, div, or mod) numbers.
	Round node	Rounds decimal digits (round off, round down, or round up).
	Sum up nodes	Sums up the node numbers of multiple node sets.
Bit operations	NOT operation	Execute NOT operation.
	Logical operation	Executes logical operation (AND, OR, XOR).
	Shift operation	Executes shift operation.
Node--based	Acquire node count	Assigns a node count to a mapping destination.
	Acquire node name	Assigns a node name to a mapping destination.
	Check node	Verifies that the specified transformation-source node exists and assigns the logical value to a mapping destination.
Control--based	Loop node	Maps looping.
	Choose node	Outputs different values according to conditions.
Other	Set constant	Assigns a specified value to a mapping destination.
	Replace value	Converts the mapping source value based on the specification of the conversion table and assigns value to the mapping destination.
	Radix Conversion	Executes basic transformation.
	Custom	Invoke a Java program created by the user.

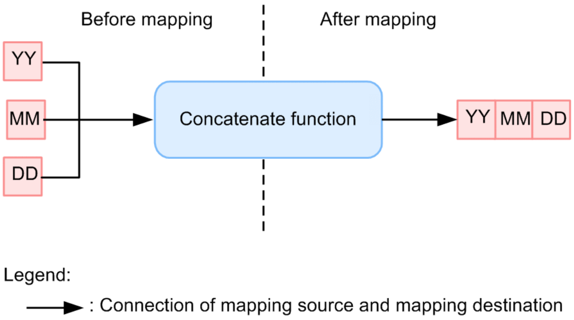
6.5.1 Concatenating Multiple Strings

Concatenates strings from multiple mapping sources into a single string.

(1) Function used

To concatenate strings from multiple mapping sources, you use the concatenate function. The following figure shows an example that uses the concatenate function.

Figure 6–2: Usage example of the concatenate function



(2) Setup procedure

To concatenate strings from multiple mapping sources into a single string:

1. From the palette, choose the concatenate function (concat), and deploy it in the mapping viewer.
2. Set up a mapping line.
3. Use one of the following methods to open the Concatenate dialog box.
   - Right--click the concatenate function, and choose **Setting**.
   - Double--click the concatenate function.

   The Concatenate dialog box opens.
   For details about the Concatenate dialog box, see the manual *Cosminexus Service Platform Overview*.
4. To add the mapping source of the string to be concatenated to Input, set up the following:
   - When the mapping source is a transformation-source node
     Clicking **Add Node** opens the Choose Node dialog box. In the Choose Node dialog box, choose the transformation-source node that becomes the mapping source.
   - When the mapping source is a function
     Clicking **Add Function** opens the Choose Function dialog box. In the Choose Function dialog box, choose the function that becomes the mapping source.

   If multiple mapping sources are specified in **Input**, the concatenation targets displayed in **Input** are concatenated sequentially starting at the top. The concatenation target displayed at the top is set up to the left of the concatenated string.
5. When a transformation-source node is specified in Input, to specify a condition for the transformation-source node and map it only when this condition is satisfied, click Set Node Condition.
   The Set Node Condition dialog box opens.
   For details about how to set a node condition, see *6.7 Specifying Node Conditions*. For details about the **Set Node Condition** dialog box, see the manual *Cosminexus Service Platform Reference*.
6. Click OK.

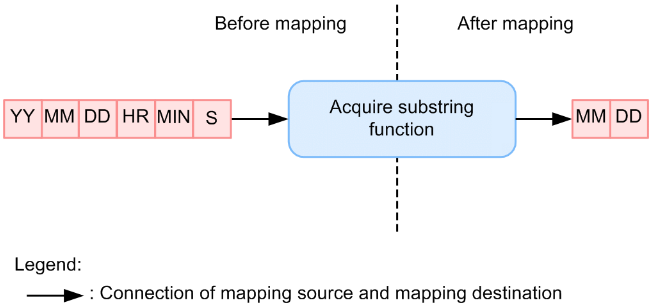
6.5.2 Extracting a Substring from a String

Extracts a substring from a string of a single mapping source.

(1) Function used

To extract a substring from a string of a single mapping source, you use the acquire substring function. The following figure shows an example that uses the acquire substring function.

Figure 6–3: Usage example of the acquire substring function


Before mapping
After mapping
YY
MM
DD
HR
MIN
S
Acquire substring function
MM
DD
Legend:
: Connection of mapping source and mapping destination

(2) Setup procedure

To extract a substring from a string of a single mapping source:

1. From the palette, choose the acquire substring function (substr), and deploy it in the mapping viewer.
2. Set up a mapping line.
3. Use one of the following methods to open the Acquire Substring dialog box.
   - Right--click the acquire substring function, and choose **Setting**.
   - Double--click the acquire substring function.

   The Acquire Substring dialog box opens.

   For details about the Acquire Substring dialog box, see the manual *Cosminexus Service Platform Overview*.
4. When a transformation-source node is specified in Input, to specify a condition for the transformation-source node and map it only when this condition is satisfied, click Set Node Condition.

   The Set Node Condition dialog box opens.

   For details about how to set a node condition, see *6.7 Specifying Node Conditions*. For details about the **Set Node Condition** dialog box, see the manual *Cosminexus Service Platform Reference*.
5. Specify the string to be extracted.
   - To specify the extraction range from the beginning of the string

     Choose the **Specify range from left side** radio button and specify in **Start position** the position of the start character of the string to be extracted. To specify the character count of the string to be extracted, specify a character count in **Count**. To extract a string from the start character specified in **Start position** to the end of the string, choose the **Acquire from start position to end of string** check box.
   - To specify the extraction range from the end of the string

     Choose the **Specify range from end** radio button and specify in **Start position** the position of the start character of the string to be extracted. To specify the character count of the string to be extracted, specify a character count in **Count**. To extract a string from the start character specified in **Start position** to the beginning of the string, choose the **Acquire from start position to beginning of string** check box.
   - To divide a string into two parts and specify either the beginning or ending part

     Choose the **Specify a substring** radio button and specify in Substring the character (string) to use for dividing the string in the mapping source. To extract the string that extends between the beginning character and the character (string) specified in Substring, specify Pre in Acquired Part. To extract the string that extends between the character (string) specified in Substring and the end, specify Post in Acquired Part.
6. Click OK.

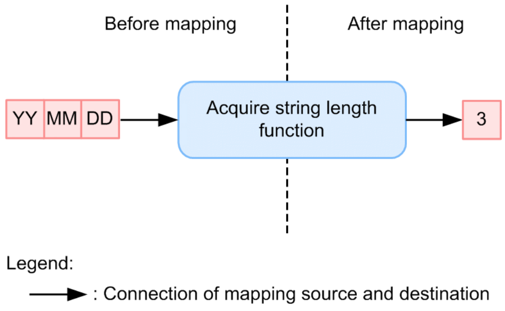
6.5.3 Assigning a String Character Count

Assigns the character count of a string of the mapping source to the mapping destination.

(1) Function used

To assign the character count of a string of the mapping source to the mapping destination, you use the acquire string length function. The following figure shows an example that uses the acquire string length function.

Figure 6–4: Usage example of the acquire string length function


Before mapping
After mapping
YY
MM
DD
Acquire string length function
3
Legend:
: Connection of mapping source and destination

(2) Setup procedure

To assign the character count of a string of the mapping source to the mapping destination:

1. From the palette, choose the acquire string length function (length), and deploy it in the mapping viewer.
2. Set up a mapping line.
3. Use one of the following methods to open the Acquire String dialog box:
   - Right--click the acquire string length function, and choose **Setting**.
   - Double--click the acquire string length function.

   The Acquire String dialog box opens.

   For details about the Acquire String dialog box, see the manual *Cosminexus Service Platform Overview*.
4. When a transformation-source node is specified in Input, to specify a condition for the transformation-source node and map it only when this condition is satisfied, click Set Node Condition.

   The Set Node Condition dialog box opens.

   For details about how to set a node condition, see *6.7 Specifying Node Conditions*. For details about the **Set Node Condition** dialog box, see the manual *Cosminexus Service Platform Reference*.
5. Click OK.

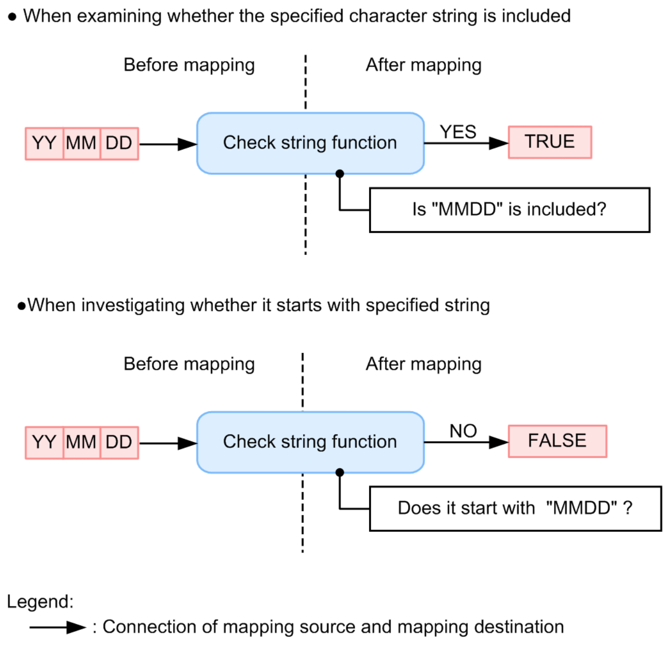
6.5.4 Verifying That the Specified String Is Present or That the String Begins with the Specified String

Verifies that the specified string is contained in the mapping source string or that the string begins with the specified string, and assigns the logical value to the mapping destination.

(1) Function used

To verify that the specified string is contained in the mapping source string or that the string begins with the specified string, and assign the logical value to the mapping destination, you use the check string function. The following figure shows an example that uses the check string function.

Figure 6–5: Usage example of the check string function


● When examining whether the specified character string is included
Before mapping
After mapping
YY MM DD
Check string function
YES
TRUE
Is "MMDD" is included?
●When investigating whether it starts with specified string
Before mapping
After mapping
YY MM DD
Check string function
NO
FALSE
Does it start with "MMDD" ?
Legend:
: Connection of mapping source and mapping destination

(2) Setup procedure

To check the string of the mapping source and assign the logical value to the mapping destination:

1. From the palette, choose the check string function (contain), and deploy it in the mapping viewer.
2. Set up a mapping line.
3. Use one of the following methods to open the Check String dialog box:
   - Right--click the check string function, and choose **Setting**.
   - Double--click the check string function.

   The Check String dialog box opens.
   For details about the Check String dialog box, see the manual *Cosminexus Service Platform Overview*.
4. When a transformation-source node is specified in Input, to specify a condition for the transformation-source node and map it only when this condition is satisfied, click Set Node Condition.
   The Set Node Condition dialog box opens.
   For details about how to set a node condition, see *6.7 Specifying Node Conditions*. For details about the **Set Node Condition** dialog box, see the manual *Cosminexus Service Platform Reference*.
5. In Check type, choose one of the following:
   - To verify that the string specified in **String** is included
     Choose Include specified string.
   - To verify that the string begins with the string specified in **String**
     Choose Start with specified string.
6. In String, specify the string to be checked.
7. Click OK.

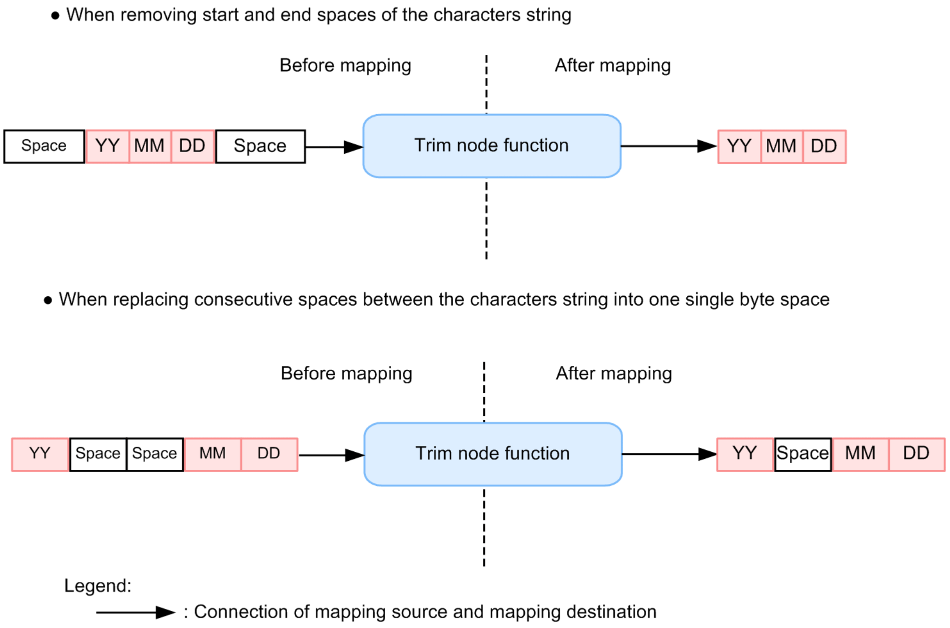
6.5.5 Removing Spaces from a String

Removes the leading and trailing spaces (single--byte space, tab, carriage return, and line feed) from a string of the mapping source. Also replaces consecutive spaces between strings with a single space.

(1) Function used

To remove spaces from a string, you use the trim node function. The following figure shows an example that uses the trim node function.

Figure 6–6: Usage example of the trim node function


● When removing start and end spaces of the characters string
Before mapping
After mapping
Space
YY
MM
DD
Space
Trim node function
YY
MM
DD
● When replacing consecutive spaces between the characters string into one single byte space
Before mapping
After mapping
YY
Space
Space
MM
DD
Trim node function
YY
Space
MM
DD
Legend:
: Connection of mapping source and mapping destination

(2) Setup procedure

To remove spaces:

1. From the palette, choose the trim node function (trim), and deploy it in the mapping viewer.
2. Set up a mapping line.
3. Use one of the following methods to open Trim Node dialog box:
   - Right--click the trim node function, and choose **Setting**.
   - Double--click the trim node function.

   The Trim Node dialog box opens.
   For details about the Trim Node dialog box , see the manual *Cosminexus Service Platform Overview*.
4. When a transformation-source node is specified in Input, to specify a condition for the transformation-source node and map it only when this condition is satisfied, click Set Node Condition.
   The Set Node Condition dialog box opens.
   For details about how to set a node condition, see *6.7 Specifying Node Conditions*. For details about the **Set Node Condition** dialog box, see the manual *Cosminexus Service Platform Reference*.
5. Click OK.

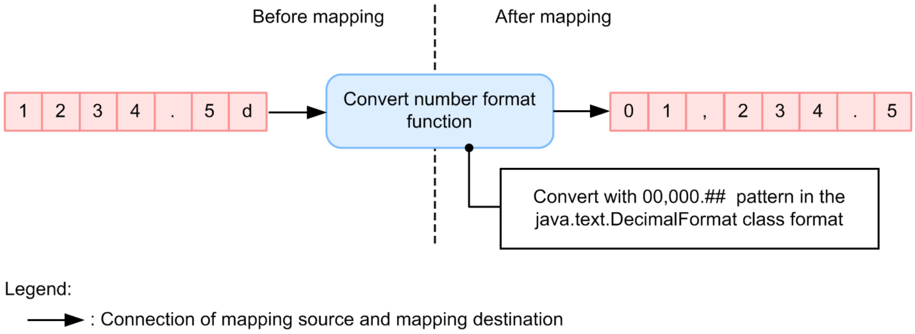
6.5.6 Converting the Number Format

Converts the format of the number at the mapping source using the format pattern of the `java.text.DecimalFormat` class. You can also change the decimal separator or grouping separator to be specified in the pattern. For details about the format pattern of the `java.text.DecimalFormat` class, see the documentation related to Java API.

(1) Function used

To convert the format of the number at the mapping source using the format pattern of the `java.text.DecimalFormat` class, you use the convert number format function. The following figure shows an example that uses the convert number format function.

Figure 6–7: Usage example of the convert number format function


Before mapping
After mapping
1 2 3 4 . 5 d
Convert number format function
0 1 , 2 3 4 . 5
Convert with 00,000.## pattern in the java.text.DecimalFormat class format
Legend:
: Connection of mapping source and mapping destination

(2) Setup procedure

To convert the format of the number at the mapping source, using the format pattern of the `java.text.DecimalFormat` class:

1. From the palette, choose the convert number format function (format), and deploy it in the mapping viewer.
2. Set up a mapping line.
3. Use one of the following methods to open the Convert Number Format dialog box:
   - Right--click the convert number format function, and choose **Setting**.
   - Double--click the convert number format function.

   The Convert Number Format dialog box opens.

   For details about the **Convert Number Format** dialog box, see the manual *Cosminexus Service Platform Reference*.
4. When a transformation-source node is specified in Input, to specify a condition for the transformation-source node and map it only when this condition is satisfied, click Set Node Condition.

   The Set Node Condition dialog box opens.

   For details about how to set a node condition, see *6.7 Specifying Node Conditions*. For details about the **Set Node Condition** dialog box, see the manual *Cosminexus Service Platform Reference*.
5. In Pattern, specify the pattern string of the java.text.DecimalFormat class format after conversion.
6. To change the decimal separator or grouping separator to be specified in the pattern, choose the Change Symbols check box and specify the following:
   - To change the decimal separator

     In Decimal Separator, specify the symbol for the decimal separator after the change. If the specification is omitted, the default symbol is used.
   - To change the grouping separator

     In Grouping Separator, specify the symbol for the grouping separator after the change. If the specification is omitted, the default symbol is used.
7. Click OK.

6.5.7 Computing Numbers

Performs the specified operation on two mapping source numbers. The following types of operations can be specified:

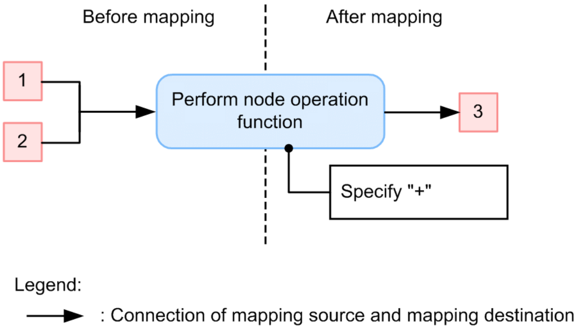
- + (addition)
- -- (subtraction)

- * (multiplication)
- / (division)
- % (remainder)

(1) Function used

To perform the specified operation on two mapping source numbers, you use the perform node operation function. The following figure shows an example that uses the perform node operation function.

Figure 6–8: Usage example of the perform node operation function


Before mapping
After mapping
1
2
Perform node operation function
3
Specify "+"
Legend:
: Connection of mapping source and mapping destination

(2) Setup procedure

To perform the specified operation on two mapping source numbers:

1. From the palette, choose the perform node operation function (calc), and deploy it in the mapping viewer.
2. Set up a mapping line.
3. Use one of the following methods to open the Perform Node Operation dialog box.
    - Right--click the perform node operation function, and choose **Setting**.
    - Double--click the perform node operation function.

   The Perform Node Operation dialog box opens.
   For details about the Perform Node Operation dialog box, see the manual *Cosminexus Service Platform Overview*.
4. In Operation, specify the operation to be executed.
5. When a transformation-source node is specified in Input 1 or Input 2, to specify a condition for the transformation-source node and map it only when this condition is satisfied, click Set Node Condition.
   The Set Node Condition dialog box opens.
   For details about how to set a node condition, see *6.7 Specifying Node Conditions*. For details about the **Set Node Condition** dialog box, see the manual *Cosminexus Service Platform Reference*.
6. Click OK.

6.5.8 Rounding Decimal Digits

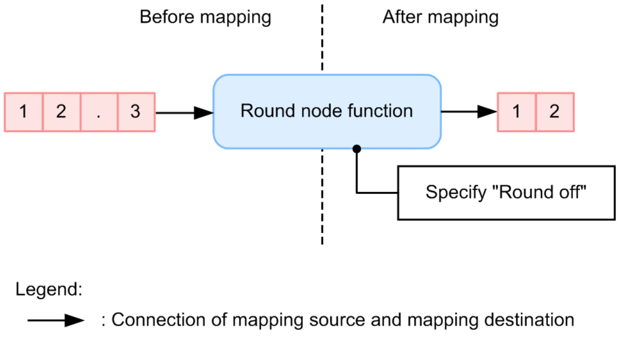
Rounds the decimal digits when the mapping source number is a decimal number. The following types of operations can be specified:

- Round off
- Round down
- Round up

(1) Function used

To round the decimal digits, you use the round node function. The following figure shows an example that uses the round node function.

Figure 6–9: Usage example of the round node function



(2) Setup procedure

To round the decimal digits:

1. From the palette, choose the round node function (round), and deploy it in the mapping viewer.
2. Set up a mapping line.
3. Use one of the following methods to open the Round Node dialog box.
   - Right--click the round node function, and choose **Setting**.
   - Double--click the round node function.

   The Round Node dialog box opens.
   For details about the Round Node dialog box, see the manual *Cosminexus Service Platform Overview*.
4. To specify a condition for the transformation-source node specified in Input and map it only when this condition is satisfied, click Set Node Condition.
   The Set Node Condition dialog box opens.
   For details about how to set a node condition, see *6.7 Specifying Node Conditions*. For details about the **Set Node Condition** dialog box, see the manual *Cosminexus Service Platform Reference*.
5. In Rounding type, specify the rounding operation to be executed.
6. Click OK.

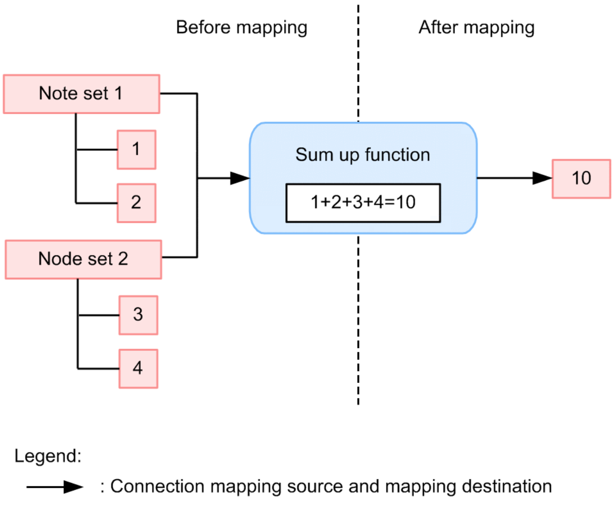
6.5.9 Summing Up the Node Numbers of Multiple Node Sets

Specifies the node sets of multiple mapping sources and sums up the node numbers included in these node sets.

(1) Function used

To sum up the node numbers of multiple node sets, you use the sum up nodes function. The following figure shows an example that uses the sum up nodes function.

Figure 6–10: Usage example of the sum up nodes function



(2) Setup procedure

To sum up the node numbers of multiple node sets:

1. From the palette, choose the sum up nodes function (sum), and deploy it in the mapping viewer.
2. Set up a mapping line.
3. Use one of the following methods to open the Sum Up Nodes dialog box.
    - Right--click the sum up nodes function, and choose **Setting**.
    - Double--click the sum up nodes function.

   The Sum Up Nodes dialog box opens.
   For details about the Sum Up Nodes dialog box, see the manual *Cosminexus Service Platform Overview*.
4. To add the numbers of the mapping sources to be summed up to Input, click Add Node.
   The Node Chooseion dialog box opens.
5. Specify the node sets that become the mapping sources.
   The numbers of the mapping sources to be summed up are set in **Input**.
6. To specify a condition for the transformation-source node specified in Input and map it only when this condition is satisfied, click Set Node Condition.
   The Set Node Condition dialog box opens.
   For details about how to set a node condition, see *6.7 Specifying Node Conditions*. For details about the **Set Node Condition** dialog box, see the manual *Cosminexus Service Platform Reference*.
7. Click OK.

6.5.10 Using NOT operation

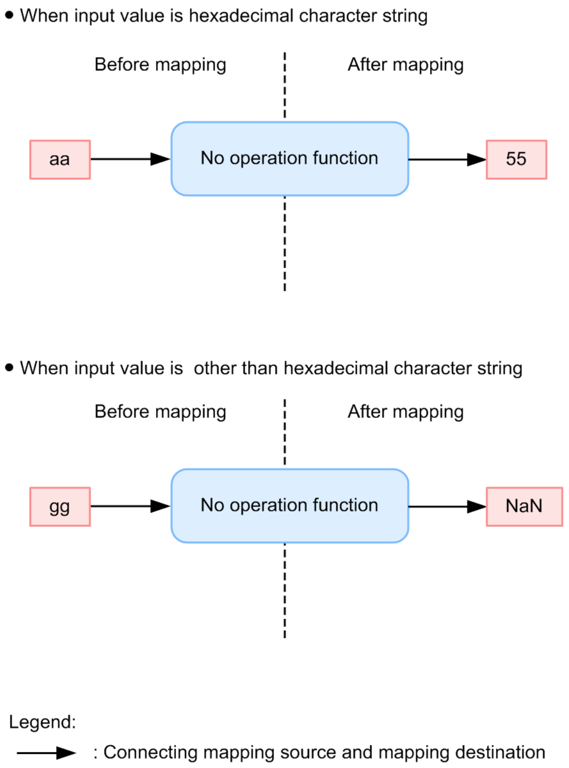
Execute NOT operation by considering the input value as hexadecimal character string. If input value is other than hexadecimal character string, error value (NaN) is returned. If input value exceeds 64 bit, round the bits more than 64 bits.

At the time of transformation source node, function is converted to character string at the time of executing transformation.

(1) Function used

Use NOT operation functionto execute NOT operation. Following figure shows the usage example:

Figure 6–11: FigureNOT operation function usage example


● When input value is hexadecimal character string
Before mapping
After mapping
aa
No operation function
55
● When input value is other than hexadecimal character string
Before mapping
After mapping
gg
No operation function
NaN
Legend:
: Connecting mapping source and mapping destination

(2) Setting procedure

Procedure for setting up the NOT operation is as follows:

1. Select NOT operation function (not) from the pallet and deploy to the Mapping viewer.
2. Set the mapping line.
3. Display NOT operation dialog with any of the following methods:
   - Right click the NOT operation function and select **Settings**.
   - Double click the NOT operation function.

   NOT operation dialog is displayed.

For details on [NOT operation] dialog, see "*1.6.25 NOT Operation dialog* " in "*Service Platform Reference Guide*".

1. To specify input value of NOT operation in **Input**, click **Select node** button.
   Select node dialog is displayed.
2. Specify node set, which is input value.
   Input value of NOT operation is set in **Input**.
3. If you want to execute mapping only when you set conditions in input value specified in **Input** and those conditions are fulfilled, click **Specify conditions** button.
   Node conditions setting dialog is displayed.
   For details on how to specify node conditions, see "*6.7 Specifying Node Conditions*". For details on Specify node conditions dialog, see '*1.6.9 Set Node Condition Dialog*" in "*Service Platform Reference Guide*".
4. When you want to specify a function of negation target in **Input**, click **Select function** button.
   Function selection dialog is displayed.

For details on Function selection dialog, see "*1.6.7 Select function dialog*" in "*Service Platform Reference Guide*".

1. Click [OK] button.

6.5.11 Using logical operation

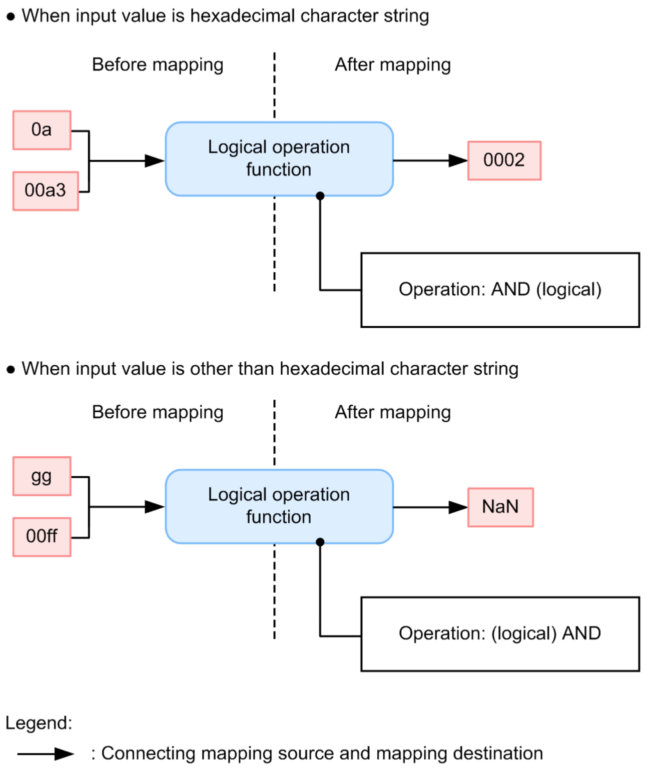
Execute logical operation by considering the input value as hexadecimal character string. You can specify "AND" (logical total), "OR" (logical sum), "XOR" (exclusive logical sum) as operation type. When input value is other than hexadecimal character string, error value (NaN) is returned. When input value exceeds 64 bit, round the bits more than 64 bits.

Process the logical operation function, with big Endian. If digit count of 2 input values differs, perform operation according to the greater digit count. Transformation source node and function are converted to character string at the time of executing transformation.

(1) Function used

To execute the logical operation, use Logical operation function. Following figure shows the usage example.

Figure 6–12: FigureLogical operation function usage example


● When input value is hexadecimal character string
Before mapping
After mapping
0a
00a3
Logical operation function
0002
Operation: AND (logical)
● When input value is other than hexadecimal character string
Before mapping
After mapping
gg
00ff
Logical operation function
NaN
Operation: (logical) AND
Legend:
: Connecting mapping source and mapping destination

(2) Setting procedure

Procedure for setting up the logical operation is as follows:

1. Select logical operation logical function (bitop) from the pallet and deploy to the Mapping viewer.
2. Set up the mapping line.
3. Display Logical operation dialog with either of the following methods:
   - Right click the logical operation function and select **Settings**.
   - Double click the logical operation function.

   Logical operation dialog is displayed.
   For details on Logical operations dialog, see "*1.6.26 Logical operation dialog*" in "*Service Platform Reference Guide*".
4. To specify input value of logical operation in **Input 1**, click **Select node** button of **Input 1**.
   Node selection dialog is displayed.

5. Specify node set, which is an input value.

   Input value of logical operation is set in **Input 1**.
6. To specify input value of logical function in **Input 2**, click **Select node** button of **Input 2**.

   Node selection dialog is displayed.
7. Specify node set, which is input value.

   Input value of logical operation is set in **Input 2**.
8. If you want to execute mapping only when you set conditions in input value specified in **Input 1** or **Input 2** and that conditions are fulfilled, click **Specify node conditions** button.

   Specify node conditions dialog is displayed.

   For details on how to specify node conditions, see "*6.7 Specifying Node Conditions*". Also, for details on Specify node condition dialog, see "*1.6.9 Set Node Condition Dialog* " in "*Service Platform Reference Guide*".
9. When you want to specify logical operation target function in **Input 1** or **Input 2**, click **Select function** button.

   Function selection dialog is displayed.

   For details on Function selection dialog, see "*1.6.7 Select function dialog*" in "*Service Platform Reference Guide*".
10. Specify operation method from **Operations**.
11. Click **OK** button.

6.5.12 Using shift operation

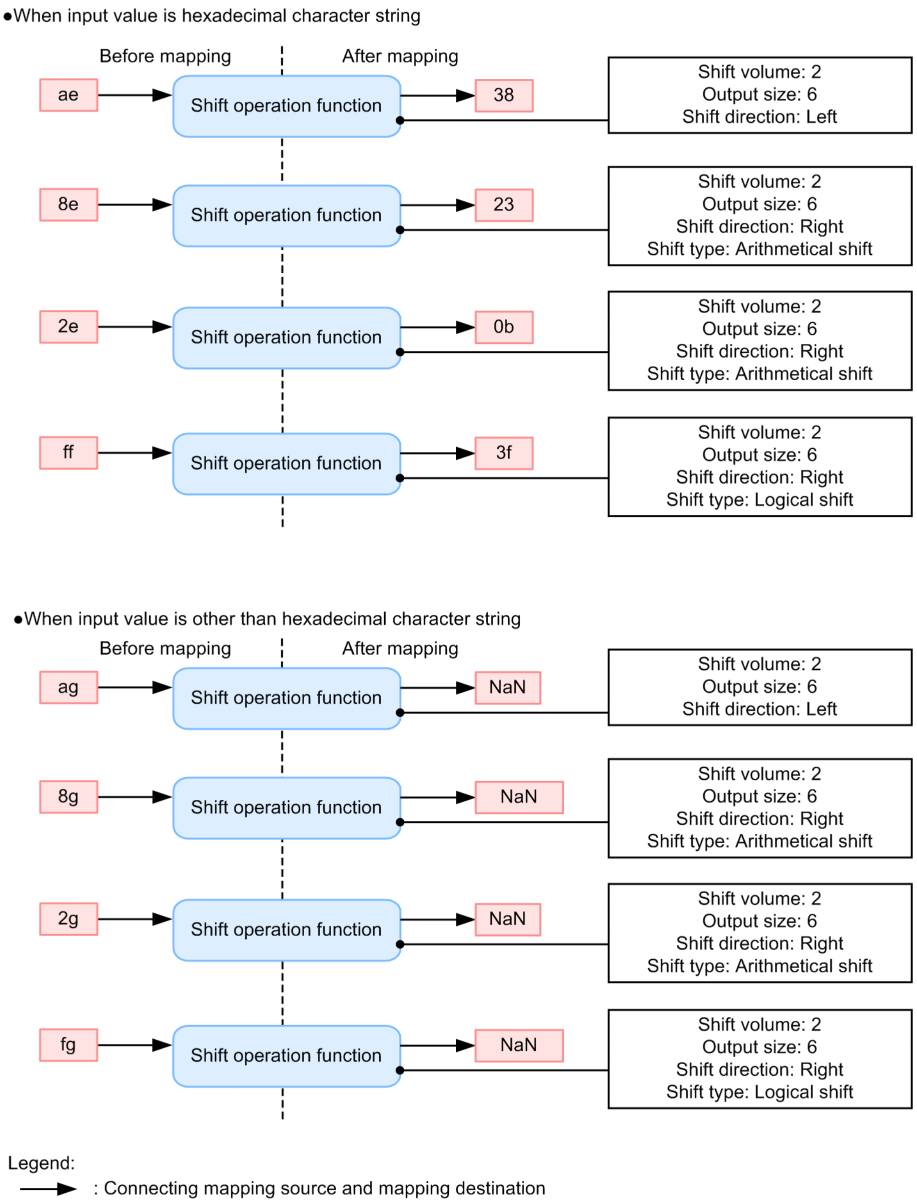
Execute shift function by considering input value as hexadecimal character string. Shift operation, obtains and returns the result of shift operation specified in shift volume, shift direction and shift type, from child bits, equal to the size specified in output size. If the input value is other than hexadecimal character string, the error value (NaN) is returned. If input value exceeds 64 bit, round the bits more than 64 bits.

Process the shift operation function, with Big Endian.

(1) Function used

To execute Shift operation, use Shift operation function. Following figure shows the usage example.

Figure 6–13: FigureUsage example of Shift operation function



(2) Setting procedure

Procedure for setting Shift operation is as follows:

1. Select Shift operation function (shift) from pallet and deploy to the mapping viewer.
2. Set the mapping line.
3. Display Shift operation dialog with either of the following methods.
    - Right click Shift operation function and select **Settings**.
    - Double click the Shift operation function.

   Shift operation dialog is displayed.
   For details on Shift operation dialog see "*1.6.27 Shift operation dialog*" in "*Service Platform Reference Guide*".
4. To specify input value of shift operation in **Input** click **Select node** button.
   Node selection dialog is displayed.
5. Specify node set, which is input value.

Input value of shift operation is set in **Input**.

6. When you want to execute mapping only when you set conditions in input value specified in **Input** and that condition is fulfilled, click **Specify node conditions** button.

   Specify node conditions dialog is displayed.

   For details on how to specify node conditions, see "*6.7 Specifying Node Conditions*". For details on Specify node conditions dialog, see "*1.6.9 Set Node Condition Dialog*" in "*Service Platform Reference Guide*".

7. When you want to specify Shift operation target function in **Input**, click **Select function** button.

   Function selection dialog is displayed.

   For details on Function selection dialog, see "*1.6.7 Select function dialog*" in "*Service Platform Reference Guide*".

8. Specify shift volume.
9. Specify output size.
10. Specify shift direction.
11. Specify shift types.
12. Click **OK** button.

(3) Execution example

Following table describes execution example:

Input#1	Shift volume	Output size	Shift direction	Shift type	Output result#1#2#3
76(0111 0110)	3	8	Left	Calculation	b0(1011 0**000**)
			Right	Calculation	0e(**000**0 1110)
			Right	Logic	0e(**000**0 1110)
89(1000 1001)			Left	Calculation	48(0100 1**000**)
			Right	Calculation	f1(**111**1 0001)
			Right	Logic	11(**000**1 0001)
76(0111 0110)	2	6	Left	Calculation	18(0001 1000) [d8(1101 10**00**)]
			Right	Calculation	1d(0001 1101) [1d(**00**01 1101)]
			Right	Logic	1d(0001 1101) [1d(**00**01 1101)]
89(1000 1001)			Left	Calculation	24(0010 0100) [24(0010 01**00**)]
			Right	Calculation	22(0010 0010) [e2(**11**10 0010)]
			Right	Logic	22(0010 0010) [22(**00**10 0010)]

Note#1

Contents of ( ) are decimal integer character string expression.

Note#2

Contents of [ ] is value before rounding up with output bits count.

Note#3

Bold letters are supplementary bits.

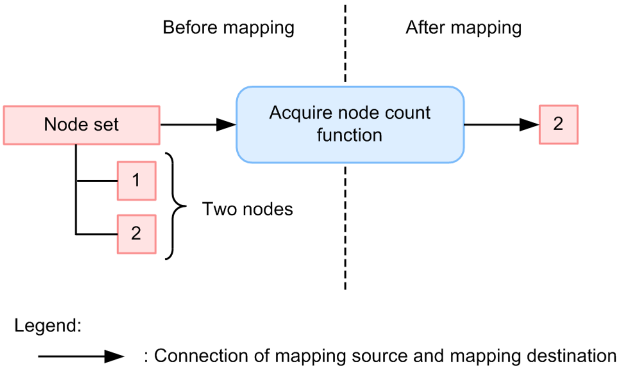
6.5.13 Assigning a Node Count

Assigns the node count of a mapping source node set to a mapping destination.

(1) Function used

To assign the node count of a mapping source node set to a mapping destination, you use the acquire node count function. The following figure shows an example that uses the acquire node count function.

Figure 6–14: Usage example of the acquire node count function


Before mapping
After mapping
Node set
Acquire node count function
2
1
2
Two nodes
Legend:
: Connection of mapping source and mapping destination

(2) Setup procedure

To assign the node count of a mapping source node set to a mapping destination:

1. From the palette, choose the acquire node count function (count), and deploy it in the mapping viewer.
2. Set up a mapping line.
3. Use one of the following methods to open the Acquire Node Count dialog box.
    - Right--click the acquire node count function, and choose **Setting**.
    - Double--click the acquire node count function.

    The Acquire Node Count dialog box opens.
    For details about the Acquire Node Count dialog box, see the manual *Cosminexus Service Platform Overview*.
4. To specify a condition for the transformation-source node specified in Input and map it only when this condition is satisfied, click Set Node Condition.
    The Set Node Condition dialog box opens.
    For details about how to set a node condition, see *6.7 Specifying Node Conditions*. For details about the **Set Node Condition** dialog box, see the manual *Cosminexus Service Platform Reference*.
5. Click OK.

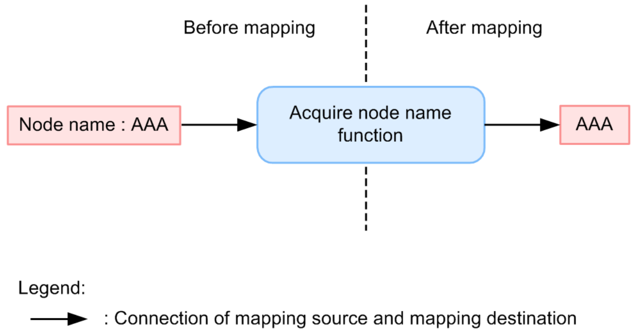
6.5.14 Assigning a Node Name

Assigns the node name of a mapping source to a mapping destination. If the instance of the mapping source (transformation-source node) does not exist, an empty string is assigned.

(1) Function used

To assign the node name of a mapping source to a mapping destination, you use the acquire node name function. The following figure shows an example that uses the acquire node name function.

Figure 6–15: Usage example of the acquire node name function


Before mapping
After mapping
Node name : AAA
Acquire node name function
AAA
Legend:
: Connection of mapping source and mapping destination

(2) Setup procedure

To assign the node name of a mapping source to a mapping destination:

1. From the palette, choose the acquire node name function (name), and deploy it in the mapping viewer.
2. Set up a mapping line.
3. Use one of the following methods to open the Acquire Node Name dialog box.
   - Right--click the acquire node name function, and choose **Setting**.
   - Double--click the acquire node name function.

   The Acquire Node Name dialog box opens.
   For details about the Acquire Node Name dialog box, see the manual *Cosminexus Service Platform Overview*.
4. To specify a condition for the transformation-source node specified in Input and map it only when this condition is satisfied, click Set Node Condition.
   The Set Node Condition dialog box opens.
   For details about how to set a node condition, see *6.7 Specifying Node Conditions*. For details about the **Set Node Condition** dialog box, see the manual *Cosminexus Service Platform Reference*.
5. Click OK.

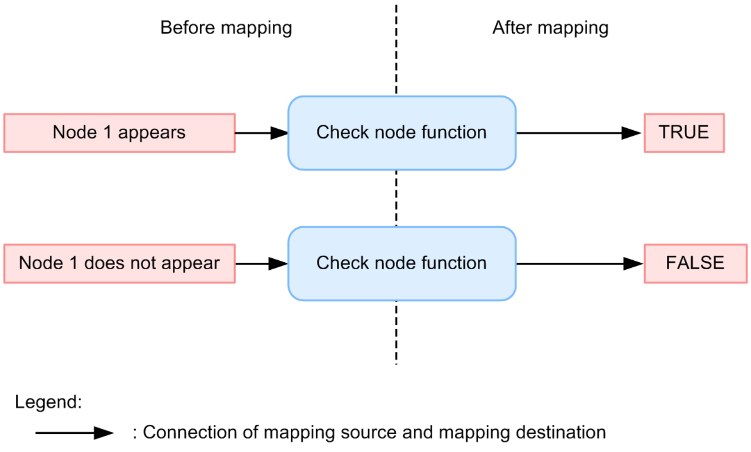
6.5.15 Verifying That a Node Exists

Verifies that the transformation-source node of the specified mapping source exists and assigns the logical value to a mapping destination.

(1) Function used

To verify that the transformation-source node of the specified mapping source exists and assign the logical value to a mapping destination, you use the check node function. The following figure shows an example that uses the check node function.

Figure 6–16: Usage example of the check node function



(2) Setup procedure

To verify that the transformation-source node of the specified mapping source exists:

1. From the palette, choose the check node function (exist), and deploy it in the mapping viewer.
2. Set up a mapping line.
3. Use one of the following methods to open the Check Node dialog box.
   - Right--click the check node function, and choose **Setting**.
   - Double--click the check node function.

   The Check Node dialog box opens.
   For details about the Check Node dialog box, see the manual *Cosminexus Service Platform Overview*.
4. To specify a condition for the transformation-source node specified in Input and map it only when this condition is satisfied, click Set Node Condition.
   The Set Node Condition dialog box opens.
   For details about how to set a node condition, see *6.7 Specifying Node Conditions*. For details about the **Set Node Condition** dialog box, see the manual *Cosminexus Service Platform Reference*.
5. Click OK.

6.5.16 Mapping Looping

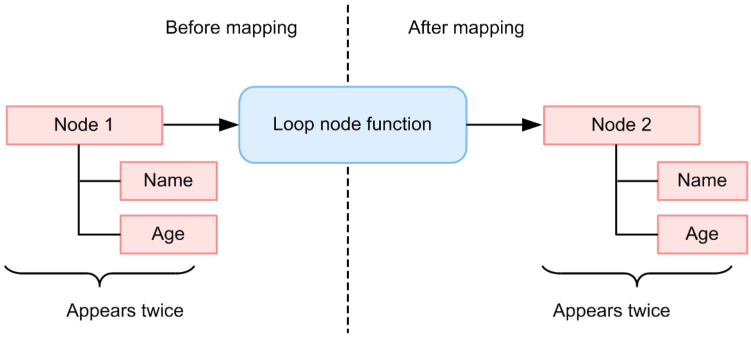
Repeatedly maps a transformation-source node that occurs more than once to a transformation-destination node that occurs more than once as the mapping source. Can also sort the instances of a transformation-source node that occurs more than once.

This subsection explains the basic method for mapping looping. For details about synthesizing loops and looping dependent targets, see *6.6 Specifying Looping*.

(1) Function used

To repeatedly map a transformation-source node that occurs more than once to a transformation-destination node that occurs more than once as the mapping source, you use the loop node function. The following figure shows an example that uses the loop node function.

Figure 6–17: Usage example of the loop node function



Legend:

⟶ : Connection of mapping source and mapping destination

(2) Setup procedure

To repeatedly map a transformation-source node that occurs more than once to a transformation-destination node that occurs more than once as the mapping source:

1. From the palette, choose the loop node function (loop), and deploy it in the mapping viewer.
2. Set up a mapping line.
3. Use one of the following methods to open the Loop Node dialog box.
   - Right--click the loop node function, and choose **Setting**.
   - Double--click the loop node function.

   The Loop Node dialog box opens.
   For details about the **Loop Node** dialog box, see the manual *Cosminexus Service Platform Reference*.
4. To sort the instances of a transformation-source node, click Add.
   The Add/Edit Sort Condition dialog box opens.
   For details about the **Add and Edit Sort Condition** dialog box, see the manual *Cosminexus Service Platform Reference*.
5. Click Choose Node.
   The Node Chooseion dialog box opens. In the Choose Node dialog box, specify the transformation-source node of the instances to be sorted.
6. Specify the following items:
   - **Order**
     Choose **ascending** or **descending**.
   - **Language**
     Choose **ja**, **en**, or **auto**.
   - **Data type**
     Choose **text** or **numeric**.
   - **Case order**
     Choose **Upper case** or **Lower case**.
7. Click OK.
   The content specified in the Add/Edit Sort Condition dialog box is set up in **Sort conditions** in the Loop Node dialog box.
8. To add keys to Sort conditions, repeat steps 4 through 7.
   You can set up a maximum of eight keys.
9. To change the order in Sort conditions, click Up or Down.
   Sorting occurs sequentially, in the order of the conditions specified in **Sort conditions**.

For example, if date is specified at the top, followed by name, instances are first sorted according to date. If the same dates are found within the sorted result, they are further sorted by name.

10. Click OK.

! Important note

- Do not define a nested loop for the same transformation-destination node. That is, do not serially connect the loop node function to the same transformation-destination node.
- Do not specify multiple assignment lines of the same looping dependent target to the same transformation-destination node. For details about looping dependent targets, see *6.6.3 Mapping Looping Dependent Targets*.
- Do not specify the looping--compatible line and assignment line of the same looping dependent target concurrently to the same transformation-destination node. For details about looping dependent targets, see *6.6.3 Mapping Looping Dependent Targets*.
- When you specify a loop for a transformation-destination node, specify a looping dependent target for the descendant node of the transformation-destination node. For details about looping dependent targets, see *6.6.3 Mapping Looping Dependent Targets*.

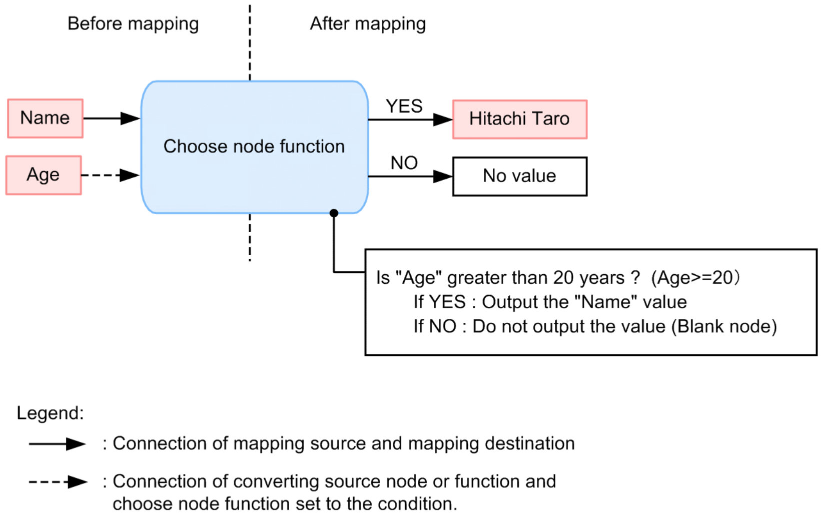
6.5.17 Outputting Different Values According to Conditions

Specifies a condition and outputs different values depending on whether the condition is satisfied.

(1) Function used

To specify a condition and output different values depending on whether the condition is satisfied, you use the choose node function. The following figure shows an example that uses the choose node function.

Figure 6–18: Usage example of the choose node function


Before mapping
After mapping
Name
Age
Choose node function
YES
NO
Hitachi Taro
No value
Is "Age" greater than 20 years ? (Age>=20)
If YES : Output the "Name" value
If NO : Do not output the value (Blank node)
Legend:
: Connection of mapping source and mapping destination
: Connection of converting source node or function and choose node function set to the condition.

(2) Setup procedure

To specify a condition and output different values depending on whether the condition is satisfied:

1. From the palette, choose the choose node function (choose), and deploy it in the mapping viewer.
2. Set up a mapping line.
   To set up a mapping line by specifying a mapping source in the Choose Node dialog box, instead of using the palette tool, execute Steps 7 through 11.
3. Use one of the following methods to open the Choose Node dialog box.
   - Right--click the choose node function, and choose **Setting**.
   - Double--click the choose node function.

The Choose Node dialog box opens.

If you specified a mapping line in step 2, a mapping source is specified in Output of Condition & output value.

For details about the Choose Node dialog box, see the manual *Cosminexus Service Platform Overview*.

4. Use one of the following methods to open the Add/Edit Condition dialog box.
    - When a mapping source is specified in Output of Condition & output value
      Choose the mapping source specified in Output of Condition & output value, and click **Edit**.
    - When a mapping source is not specified in Output of Condition & output value
      Click **Add**.

    For details about the Add/Edit Condition dialog box, see the manual *Cosminexus Service Platform Overview*.
5. Click Set Condition.
   The Condition Settings dialog box opens.
6. Specify a condition.
   For details about how to set up the condition in the **Condition Settings** dialog box, see the manual *Cosminexus Service Platform Reference*.
7. Specify the following in the When condition is true column:
    - To output the transformation-destination node
      Choose the Output the node check box.
    - To not output the transformation-destination node
      Choose the Do not output the node check box.

    If the mapping source has already been specified in Output of Condition & output value, the mapping source is displayed in Value. If you do not wish to change the mapping source, proceed to step 10.
8. If the Output the node check box was selected in step 7, specify the following in the Output value column:
    - To output a value to the transformation-destination node
      Choose the Output the node check box.
    - To not output a value to the transformation-destination node
      Choose the Empty check box.
9. If the Output the node check box was selected in step 8, specify the following for Value:
    - To output the value of the transformation-source node
      Clicking **Choose Node** opens the Choose Node dialog box. In the Choose Node dialog box, choose the transformation-source node whose value is to be output.
    - To output the value of the function
      Clicking **Choose Function** opens the Choose Function dialog box. In the Choose Function dialog box, choose the function whose value is to be output.
10. When a transformation-source node is specified in Value, to specify a condition for the transformation-source node and map it only when this condition is satisfied, click Set Node Condition.
    The Set Node Condition dialog box opens.
    For details about how to set a node condition, see *6.7 Specifying Node Conditions*. For details about the **Set Node Condition** dialog box, see the manual *Cosminexus Service Platform Reference*.
11. Click OK.
    The Add/Edit Condition dialog box closes.
12. To add or edit a condition, repeat steps 4 through 11.
    If you specify multiple conditions and mapping sources in Condition & output value, condition evaluation is sequentially performed, starting with the condition displayed at the top.
13. In the When no condition matches column of the Choose Node dialog box, specify the mapping source that is to be used when none of the conditions are satisfied.
    The setting method is the same as that used for the **When condition is true** column of the Add/Edit Condition dialog box.
14. Click OK.

If you specify a function or a transformation-source node that is not for the mapping source inside the condition specified in the Choose Node dialog box, a mapping line ⇢ (condition line) is set up between the transformation-source node or function inside that condition and the choose node function.

! Important note

- When mapping a value to a transformation-destination node of complex content, do not choose Output the value and specify Value. When mapping a value to a transformation-destination node that is not of complex content, you must specify Value if you choose Output the value.
- When mapping a value to a `choice` element all of whose child elements' minimum occurrence count is set to 1 or greater, do not specify Empty.
- When mapping a value to an element (other than a `choice` element) that has a child element whose minimum occurrence count is set to 1 or greater, do not specify Empty.
- When mapping a value to a node whose occurrence count is set to 1 or greater, do not specify Do not output the node.
- If you specify even a single loop node function for an output value, specify the loop node function for all other output values. If you specify an item other than a loop node function for an output value, specify an item other than a loop node function for all other output values.

6.5.18 Assigning a Specified Value

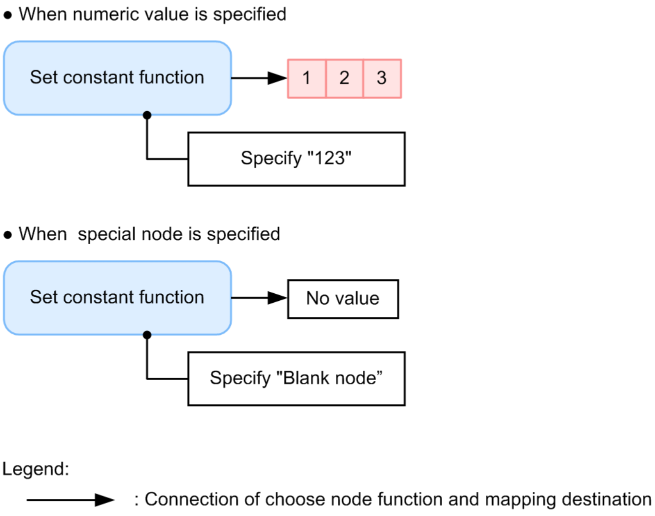
Assigns a specified value to a mapping destination. The following value types can be specified:

- Character string
- Number
- Boolean (`true` or `false`)
- Special node (empty node or no node output)

(1) Function used

To assign a specified value to a mapping destination, you use the set constant function. The value specified in the set constant function can be assigned to multiple mapping destinations. The set constant function does not have a mapping source. Therefore, the value of the set constant function is assigned as is to the mapping destination. The following figure shows an example that uses the set constant function.

Figure 6–19: Usage example of the set constant function


● When numeric value is specified
Set constant function
1
2
3
Specify "123"
● When special node is specified
Set constant function
No value
Specify "Blank node”
Legend:
: Connection of choose node function and mapping destination

(2) Setup procedure

To assign a specified value to a mapping destination:

1. From the palette, choose the set constant function (const), and deploy it in the mapping viewer.
2. Set up a mapping line.
3. Use one of the following methods to open the Set Constant dialog box:
   - Right--click the set constant function, and choose **Setting**.
   - Double--click the set constant function.

   The Set Constant dialog box opens.

   For details about the Set Constant dialog box, see the manual *Cosminexus Service Platform Overview*.
4. Specify the value to be assigned to the mapping destination.
   - To specify a string
     Choose the **String** radio button and specify the value to be assigned in **Value**.
   - To specify a number
     Choose the **Number** radio button and specify the value to be assigned in **Value**.
   - To specify a logical value
     Choose the **Logical Value** radio button, and then choose **Positive** or **Negative** in the **Logical Value** column.
   - To specify a special node
     Choose the **Special Node** radio button, and click **Do not output the node** or **Empty node in the Value column**.
5. Click OK.

! Important note

- When specifying a special node, specify a transformation-destination node for the mapping destination.
- When specifying a value other than a special node, specify a node other than a complex-content node for the mapping destination.
- If you select **do not output the node for a special node**, specify a transformation-destination node whose minimum occurrence count is 0 for the mapping destination.
- When mapping a value to a `choice` element all of whose child elements' minimum occurrence counts are set to 1 or greater, specify a node that is not an empty node (special node).
- When mapping a value to an element (other than a `choice` element) that has a child element whose minimum occurrence count is set to 1 or greater, specify a node that is not an empty node (special node).

6.5.19 Converting a Value with the Conversion Table

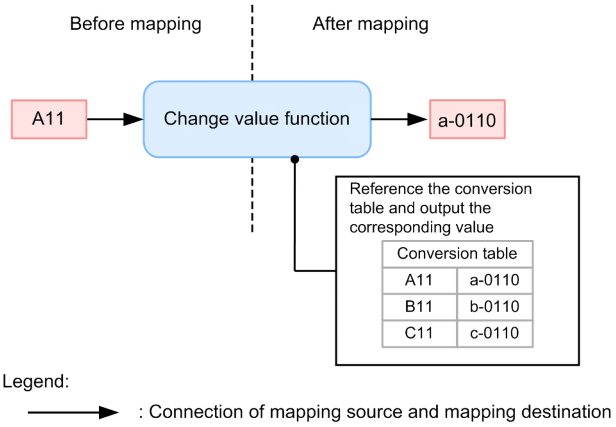
Use the conversion table to convert the mapping source value and assign this value to the mapping destination.

(1) Function used

To convert the mapping source value and assign it to the mapping destination with the conversion table, use the Replace Value Function.

The following figure shows an example wherein the replace value function is used:

Figure 6–20: Usage example of the replace value function


Before mapping
After mapping
A11
Change value function
a-0110
Reference the conversion table and output the corresponding value
Conversion table
A11
a-0110
B11
b-0110
C11
c-0110
Legend:
: Connection of mapping source and mapping destination

(2) Setup procedure

To convert the mapping source value and assign this value to the mapping destination using a conversion table, first create and save the conversion table. If you specify the information of the created conversion table in a system property file (`usrconf.properties`), you can use the conversion table. After the conversion table is available for use, define the replace value function.

(a) Creating and saving a conversion table

This sub--section explains how to create and save a conversion table.

Tip

The conversion table that is created is not to be packaged. Therefore, the conversion table can be edited anytime even after the actual operations start in the execution environment. The edited contents are reflected when the system is next started.

Creating a conversion table

Create the conversion table in a csv file. The format of the csv file is as follows:

```
"A01","B01"
"A02","B02"
"A03","B03"
       :
```

Substitute value considering `A`*nn* and `B`*nn* coded in a same line as a combination pair. Enclose the string to be substituted in double quotations (`"`), and code. If the string to be substituted contains double quotation, you can use the double quotation as an escape character. If characters outside the scope of escape are escaped, the escaped characters are ignored and subsequent characters are processed.

Note that you can use the following character codes in the conversion table:

- MS932
- UTF8
- UTF16 (big endian or little endian)

The following table describes the characters to be used to code the conversion table:

Table 6–4: Characters to be used in the conversion table

Target	Characters to be used
Start character	Start character is not required.
End character	One of the following linefeed code: • LF (0x0A) • CR+LF (0x0D0A) If the executing OS is UNIX, use LF and for Windows, use CR+LF.

Target	Characters to be used
End character	Note: LF: Linefeed CR: Carriage return
Intervening separation character	Use commas (,).

! Important note

- The data coded in the conversion table is handled as a string. Therefore, if a numeric value is entered in the replace value function, it is converted into a string and processed.
- You cannot specify an empty element for the value to be converted entered.
- If you want to enter an attribute value, the input value is handled with a normalized value (substituting a tab with a space, continuous spaces with a single space, and so on). Therefore, the search key value coded in the conversion table needs to be considered as a normalized value.

Tip

You can specify whether to convert from A*nn* to B*nn* or from B*nn* to A*nn*, when you define the Replace Value Function.

Saving the conversion table

Save the created conversion table in any location.

(b) Registering in the system property file (usrconf.properties)

After you create a conversion table, register the created conversion table in usrconf.properties. For details about how to register a conversion table, see the contents related to the registration of a conversion table in the manual *Cosminexus Service Platform System Setup and Operation Guide*.

(c) Defining Replace Value Function

The following is the procedure for using the replace value function to convert the mapping source value and assigning the value to the mapping destination:

1. From the palette, choose Replace Value function (replace) and deploy in the mapping viewer.
2. Set up a mapping line.
3. Open the Replace Value dialog box with one of the following methods:
   - Right click on **Replace Value** function and choose **Setting**.
   - Double click on **Replace Value** function.

   The Replace Value dialog box opens.

   For details about the Replace Value dialog box, see the manual *Cosminexus Service Platform Overview*.
4. When a transformation-source node is specified in Input, to specify a condition for the transformation-source node and map it only when this condition is satisfied, click Set Node Condition.

   The Set Node Condition dialog box opens.

   For details about how to set a node condition, see *6.7 Specifying Node Conditions*. For details about the **Set Node Condition** dialog box, see the manual *Cosminexus Service Platform Reference*.
5. In Conversion table ID, specify the conversion table name coded in the system property file (usrconf.properties).
6. In Specify Reference Key, choose either the left or right row of the conversion table as the source for changing the value.
7. In Operation when search fails, choose the system operation for the case where the input value from the mapping source does not exist in the conversion table.

   To output the default value and assign value to the mapping destination

   Choose the **Substitute default value** radio button and enter the string to be used as the default value in Value.

   To treat as a conversion error

   Choose the **conversion error** with a radio button.
8. Click OK.

6.5.20 Performing basic number transformation

Consider the input value as basic number specified in input basic number and transform the input value to basic number specified in output basic number. If input value is other than basic number specified in input basic number, error value (NaN) is returned. If input value exceeds 64 bits, round the bits more than 64 bits.

Process the radix conversion function with Big Endian.

Specifications of basic number transformation are as follows:

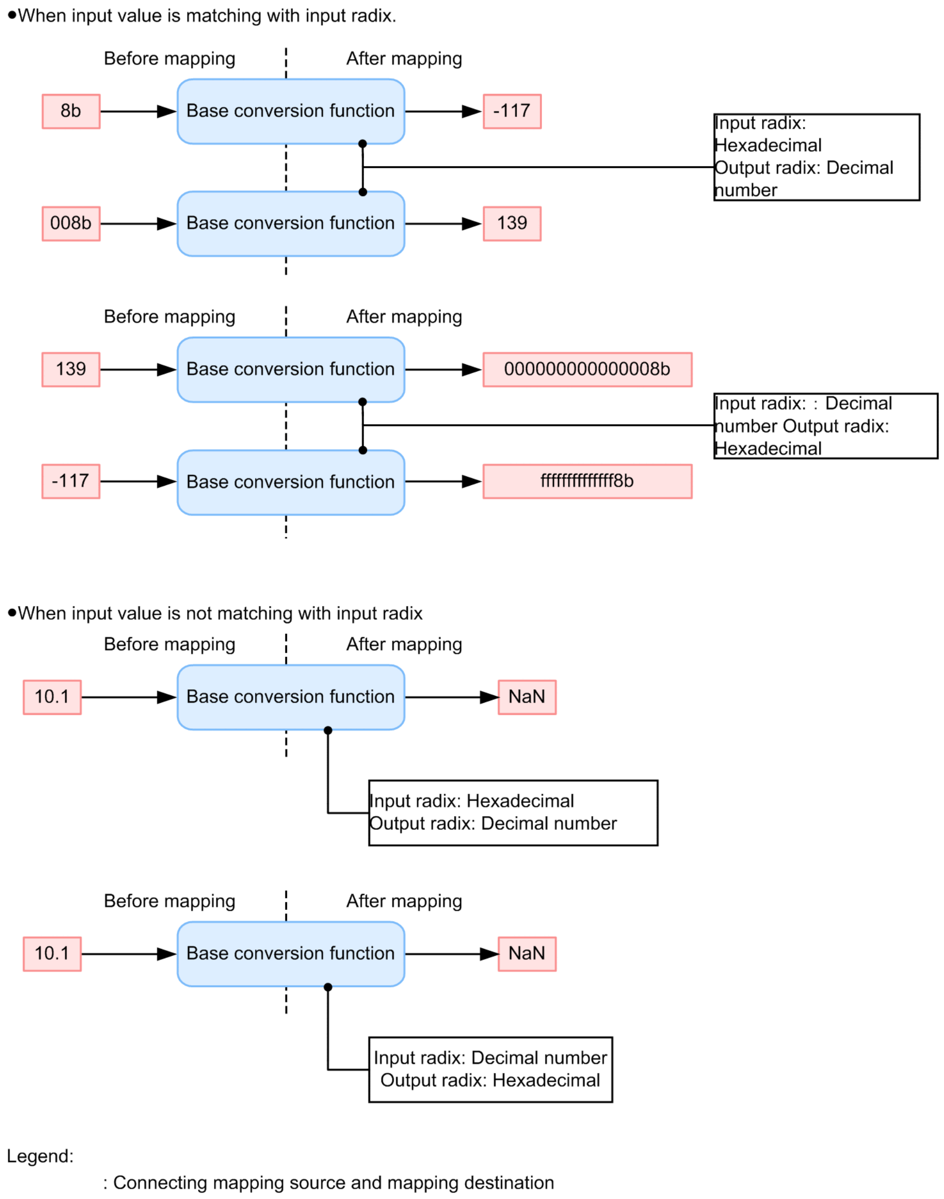
- When input basic number is decimal number, perform basic number transformation as 64 bit.
- When output basic number is decimal number, consider the maximum bit of input data extended to 4 bit unit, as encoded bit.
- When input basic count and output basic count is not decimal number, set the basic number to bit length of transformation source. If input basic count is binary number, expand the data that does not fulfill 4 bit, such that it becomes 4 bit unit.
- Format of input value of input basic number is as follows:

Input basic number	Format
Binary number	[01]+
Decimal	[+-]?[0-9]+
Hexadecimal	[0-9a-fA-F]+

(1) Function used

To transform input value to basic number specified in output basic number, use Radix conversion function. Following figure shows the usage example.

Figure 6–21: FigureUsage example of radix conversion function



(2) Setting procedure

Procedure for transforming the input value to basic number specified in output basic number is as follows:

1. Select radix conversion function (radix) from pallet and deploy to the mapping viewer.
2. Set the mapping line.
3. Display Radix Conversion dialog with either of the following method:
   - Right click the Radix conversion function and select **Settings**.
   - Double click the radix conversion function.

   Radix Conversion dialog is displayed.

   For details on Radix Conversion dialog, see "*1.6.36 Base conversion dialog*" in "*Service Platform Reference Guide*".
4. To specify input value of basic number transformation in **Input**, click **Select node** button.

   Node selection dialog is displayed.

5. Specify node set, which is input value.
   Input value of basic number transformation is set in **Input**.
6. When you want to execute mapping only when you set conditions in input value specified in **Input** and that condition is fulfilled, click **Specify node conditions** button.
   Specify node conditions dialog is displayed.
   For details on how to specify node conditions, see "*6.7 Specifying Node Conditions*". For details on the Specify node conditions dialog, see " *Set Node Condition Dialog* " in "*Service Platform Reference Guide*"
7. When you want to specify basic number transformation target function in **Input**, click **Select function** button.
   Function selection dialog is displayed.
   For details on Function selection dialog, see "*1.6.7 Select function dialog*" in "*Service Platform Reference Guide*".
8. Specify input basic number.
9. Specify output basic number.
10. Click **OK** button.

(3) Execution example

Following table describes the execution example:

Input	Basic number transformation	Result
1101	Binary numberdecimal	-3
0100	Binary numberdecimal	4
1101	Binary numberHexadecimal	d
11	DecimalBinary number	**0000 0000 0000 0000 0000 0000 0000 0000** **0000 0000 0000 0000 0000 0000 0000** 1011[#1]
11	DecimalHexadecimal	**00 00 00 00 00 00 00 0**b[#1]
11	HexadecimalBinary number	0001 0001
11	HexadecimalDecimal	17
8b	HexadecimalDecimal	-117
1102	Binary numberDecimal	NaN[#2]
110f	DecimalHexadecimal	NaN[#2]

Note#1

Bold letters are supplementary bits.

Note#2

As you cannot convert the input value to basic number specified in input basic number, result becomes "NaN".

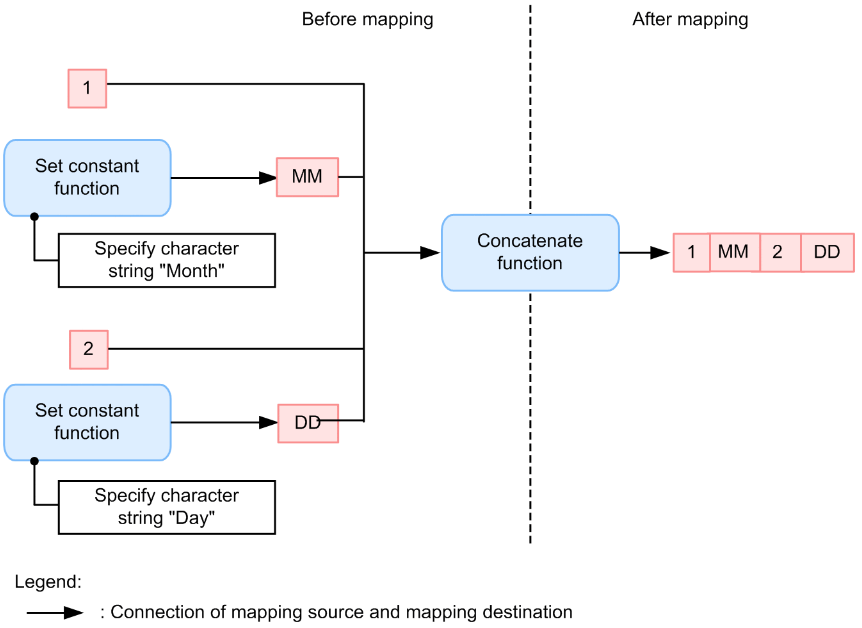
6.5.21 Assigning a Value to a Transformation-source Node Value

Adds a value to a transformation-source node value and assigns it to the mapping destination.

(1) Function used

To add a value to a transformation-source node value and assign it to the mapping destination, you use the concatenate function and the set constant function in combination. The following figure shows an example that uses the concatenate function and the set constant function in combination.

Figure 6–22: Usage example of a combination of concatenate function and set constant function


Before mapping
After mapping
1
Set constant function
MM
Specify character string "Month"
2
Set constant function
DD
Specify character string "Day"
Concatenate function
1
MM
2
DD
Legend:
: Connection of mapping source and mapping destination

(2) Setup procedure

To add a value to a transformation-source node value and assign it to the mapping destination:

1. From the palette, choose the set constant function (const), and deploy it in the mapping viewer.
2. Use the set constant function to specify a value.
   For details about how to specify a value, see *6.5.18 Assigning a Specified Value*.
3. From the palette, choose the concatenate function (concat), and deploy it in the mapping viewer.
4. Use the concatenate function to concatenate the value specified by the set constant function with the transformation-source node value.
   For details about the concatenation method, see *6.5.1 Concatenating Multiple Strings*.

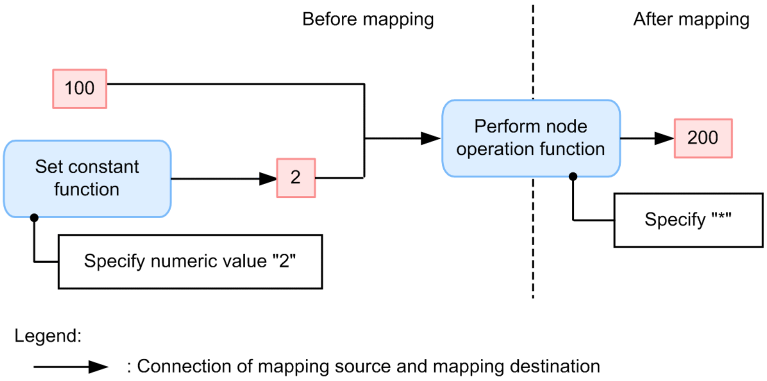
6.5.22 Doubling a Transformation-source Node Value

Doubles a transformation-source node value and assigns it to the mapping destination.

(1) Function used

To double a transformation-source node value and assign it to the mapping destination, you use the perform node operation function and the set constant function in combination. The following figure shows an example that uses the perform node operation function and the set constant function in combination.

Figure 6–23: Usage example of a combination of perform node operation function and set constant function


Before mapping
After mapping
100
Set constant function
2
Specify numeric value "2"
Perform node operation function
200
Specify "*"
Legend:
: Connection of mapping source and mapping destination

(2) Setup procedure

To double a transformation-source node value and assign it to the mapping destination:

1. From the palette, choose the set constant function (const), and deploy it in the mapping viewer.
2. Use the set constant function to specify a value.
   For details about how to specify a value, see *6.5.18 Assigning a Specified Value*.
3. From the palette, choose the perform node operation function (calc), and deploy it in the mapping viewer.
4. Use the perform node operation function to calculate the value specified by the set constant function and the transformation-source node value.
   For details about the computation method, see *6.5.7 Computing Numbers*.

6.5.23 Invoke a Java program created by the user

Invoke a Java program created by the user and use as a function. You can use this for processes other than those in other functions provided by uCosminexus Service Architect.

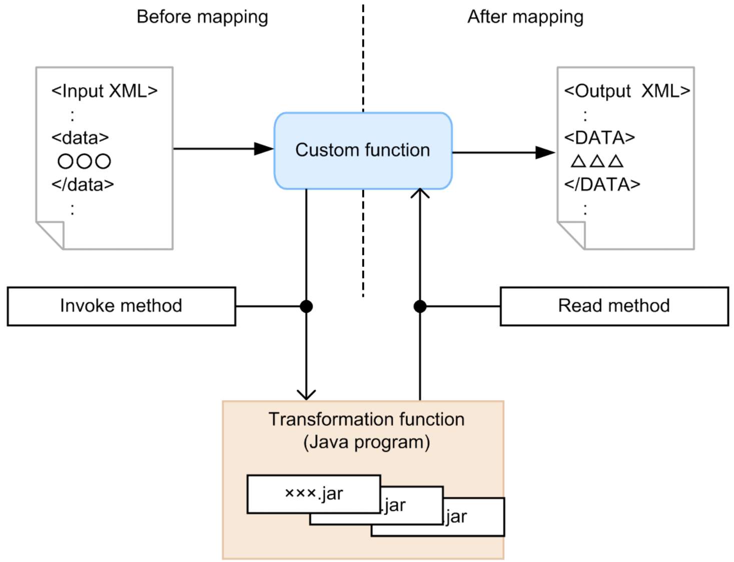
For details about how to create Java programs for invocation, see *6.9 Creating Java programs to be used in the custom function*.

(1) Function used

Use the custom function to invoke a Java program created by the user.

The following figure shows how to invoke a Java program from the custom function. Note that in the `custom` function, invoking a Java program is called as *transformation function*.

Figure 6–24: Invoking a Java program from the custom function (transformation function)



(2) Setting procedure

The following procedure describes how to invoke a Java program created by the user:

1. In the Eclipse menu, choose **Window** and **Settings**.
   The **Settings** dialog box appears.
2. In the left of the dialog box, choose **HCSC--Definer** and **Data transformation**.
   The settings of the Data Transformation Definition screen appear on the right side of the dialog box.
3. Click the Custom function tab.
   The Custom function tab of the **Settings** dialog box appears.
4. Specify the transformation function definition file and click **OK** button.
5. Choose the custom function from the palette (custom) and deploy it in the mapping viewer.
6. Set the mapping line.
   The number of mapping lines to be set is the same as for the argument of the invoked Java program.
7. Open the **Custom** dialog box by either of the following methods:
   - Right click the custom function and choose **Settings**.
   - Double--click the custom function.
8. Click **Choose transformation function**.
   The **Choose transformation function** dialog box appears.
9. Choose the method to be invoked from the custom function and click **OK** button.
   The **Choose transformation function** dialog box closes and the **Custom** dialog box appears.
10. Specify the input value corresponding to **Argument name** in **Input value**.
    The specified input value is entered in the argument of the Java program.
11. Click **OK** button.

Notes

- The input value of the argument is not set automatically if the mapping line is set in the mapping viewer after setting the transformation function in the **Custom** dialog box. Specify again in Input value the input value corresponding to **Argument name** in the **Custom** dialog box.
- Invoke the custom function from the information of the method selected in the **Choose transformation function** dialog box. Even if the contents of the transformation function definition file change after selecting the method in the **Choose transformation function** dialog box, the change is not reflected in information of the method for

invoking from the custom function. To reflect the changes of the transformation function definition file to the custom function, select the transformation function again.

- If the following setting contents do not match, an error occurs while executing the custom function but not while validating the business process. Setting contents must always match.
    - Setting contents of the transformation function definition file
    - Setting contents of the custom function
    - Contents of the packaged jar file
- To use multiple transformation function definition files, switch the transformation function definition files in the **Settings** dialog box and set the custom function. Switching the transformation function definition files does not affect a set custom function.

6.6 Specifying Looping

In mapping that uses the loop node function, you can synthesize loops and change the looping dependent target.

6.6.1 Mapping Using the Loop Settings Dialog Box

For mapping from the loop node function or the choose node function# to a transformation-destination node, you can also use the Loop Settings dialog box.

The procedure for mapping using the Loop Settings dialog box is described below.

1. From the palette, choose the loop node function or the choose node function.
2. Click an appropriate location in the mapping viewer.
   The function is deployed in the mapping viewer. You can move the deployed function by dragging it.
3. In the transformation-destination schema tree viewer, right--click the transformation-destination node that becomes the mapping destination, and choose Loop Node.
   The Loop Settings dialog box opens.
4. Click Add Function.
   The Choose Function dialog box opens.
5. Specify the loop node function that becomes the mapping source or the choose node function, and click OK.
   The specified function name is set up in Function name.
6. Click OK.
   A mapping line -·-·-·- (looping--compatible line) is set up. You can change the mapping line color. For details about how to change the mapping line color, see the manual *Cosminexus Service Platform Reference*.

#

A choose node function that is connected to a loop node function or to nothing. If a function is connected to a function other than a loop node function, you cannot use the Loop Settings dialog box.

! Important note

When a loop is specified for a transformation-source node, a descendant node of that transformation-source node also becomes the looping target and occurs at the transformation-destination node.

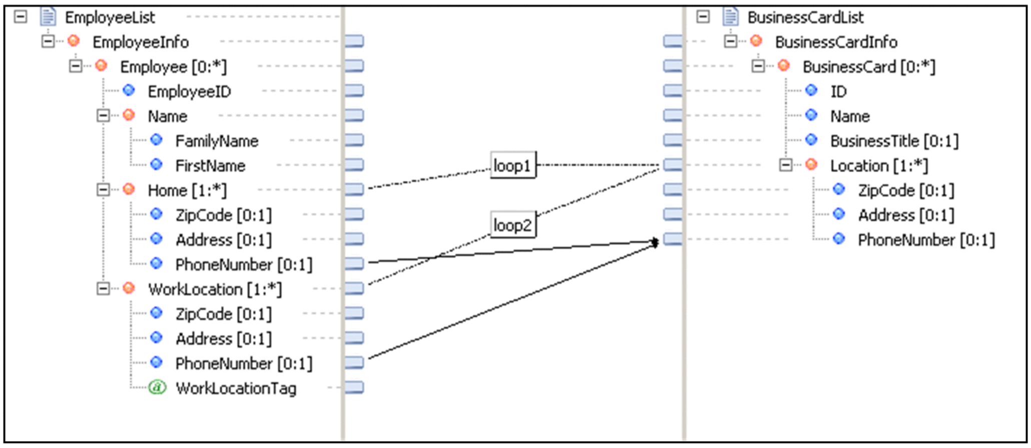
However, if an absolute path is specified for the mapping source in the Settings for Mapping Source dialog box, the value of the node specified by the absolute path is always assigned. If a relative path is displayed for the mapping source in the Settings for Mapping Source dialog box, a descendant node of the transformation-source node for which looping is specified becomes the looping target, and the value of the node corresponding to looping is assigned.

For details about how to display transformation-source node paths, see *6.6.6 Displaying the Path of a Transformation-source Node for Which a Looping Dependent Target Is Specified*.

6.6.2 Synthesizing Loops

You can synthesize a loop of multiple transformation-source nodes and map them to a single transformation-destination node. The following figure shows a window in which loops have been synthesized.

Figure 6–25: Window with synthesized loops



When you synthesize loops, you can use the Loop Settings dialog box to specify the order in which transformation-source nodes occur at the transformation-destination node. In the case shown in Figure 6-25, you can decide whether the transformation-source node (home) corresponding to loop1 or the transformation-source node (WorkLocation) corresponding to loop2 will occur first at the transformation-destination node. Note that a descendant node of a transformation-source node corresponding to a loop also becomes a looping target and occurs at the transformation-destination node.

To specify the order in which transformation-source nodes are to be assigned to the transformation-destination node:

1. In the transformation-destination schema tree viewer, right-click the transformation-destination node that becomes the mapping destination, and choose Loop Node.

   The Loop Settings dialog box opens.

2. Rearrange the function names displayed under Loop to the order in which they are to occur at the transformation-destination node.

   Transformation-source nodes sequentially occur at the transformation-destination node, beginning with the one corresponding to the function name displayed at the top.

   - To move up in order

     Choose the function name to be moved up, and click **Up**.

   - To move down in order

     Choose the function name to be moved down, and click **Down**.

3. Click OK.

6.6.3 Mapping Looping Dependent Targets

When you use the loop node function to map a transformation-source node to a transformation-destination node, that loop node function is automatically mapped as a looping target of a descendant node's mapping source. Also, deleting the loop node function and mapping lines cancel the looping of a node. In such a case, the loop node function set up in the ancestor node of the node, for which looping is cancelled, will be automatically mapped as a looping target of the mapping source in the descendant node. This is called a looping dependent target.

A looping dependent target is set up in the following cases:

- When the loop node function is used to specify a loop for a node, and its descendant node is mapped later

  The loop node function used by a node automatically becomes the looping dependent target of the descendant node's mapping source.

- When a node is mapped, and the loop node function is used to specify a loop for its ancestor node later

  The loop node function used by the ancestor node becomes the looping dependent target of its descendant node's mapping source.

- When the looping of a node, for which looping is set up in the ancestor node, will be cancelled

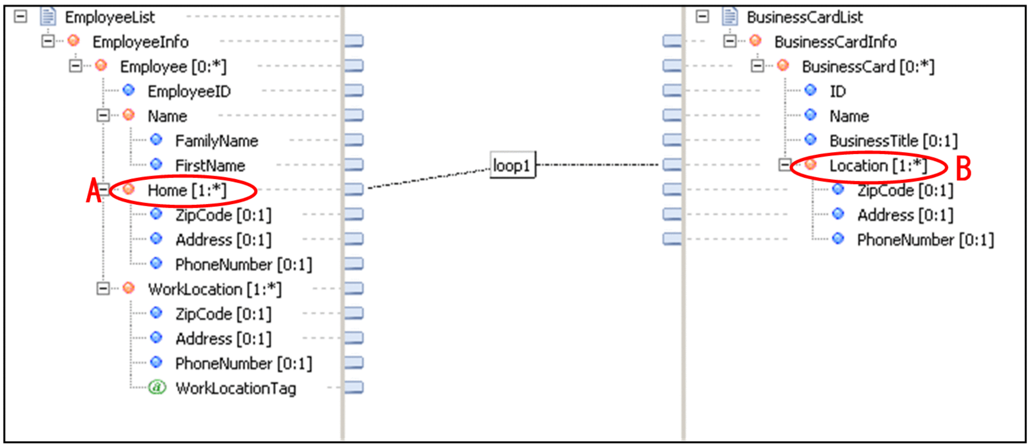
The loop node function set up in the ancestor node becomes the looping dependent target of the mapping source in the descendant node of the node for which looping is cancelled.

The timing at which a looping dependent target is mapped in each case is explained below.

(1) When a loop is specified for a node and its descendant node is mapped later

This subsection explains the timing at which the looping dependent target is mapped when a loop is specified for a node and its descendant node is mapped later.

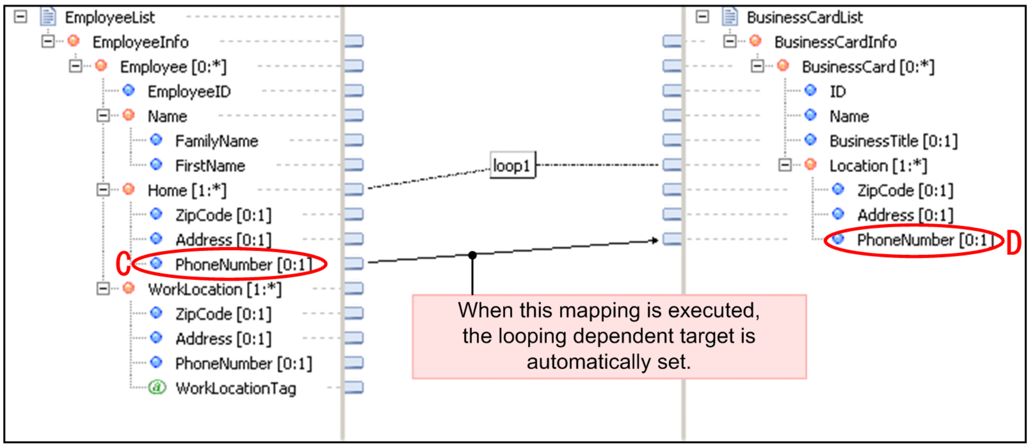
Figure 6–26: Mapping when the loop node function is used



First, the loop node function is used to map a transformation-source node A to a transformation-destination node B.

Next, node C, which is a descendant node of transformation-source node A, is mapped to node D, which is a descendant node of transformation-source node B. The following figure shows the timing at which the looping dependent target is mapped when no function is used for the descendant node.

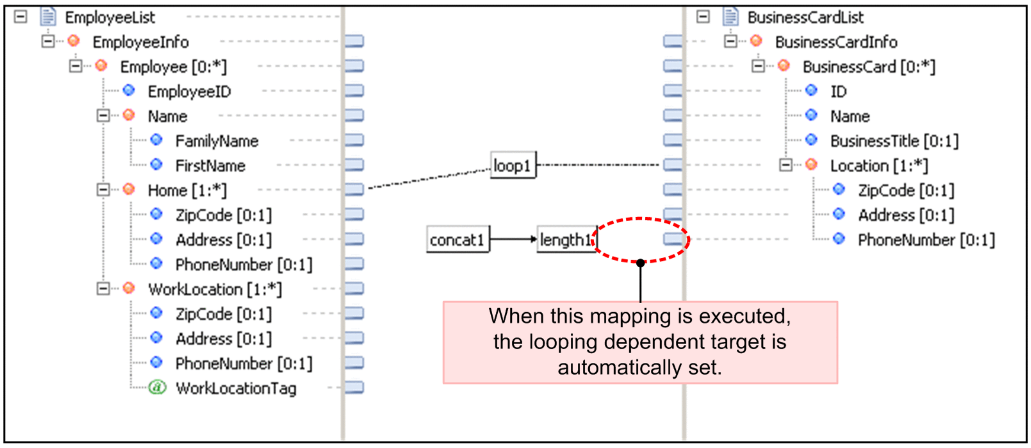
Figure 6–27: Timing at which the looping dependent target is mapped (When no function is used for the descendant node)



When node D is connected, a looping dependent target is automatically set up for the descendant node. In Figure 6-27, loop1 is set up for the looping dependent target of descendant node D's mapping source.

If multiple functions are used for mapping a descendant transformation-source node to the transformation-destination node, a looping dependent target is automatically set up when the transformation-destination node is connected. The following figure shows the timing at which the looping dependent target is mapped when multiple functions are used for the descendant node.

Figure 6–28: Timing at which the looping dependent target is mapped (When multiple functions are used for the descendant node)

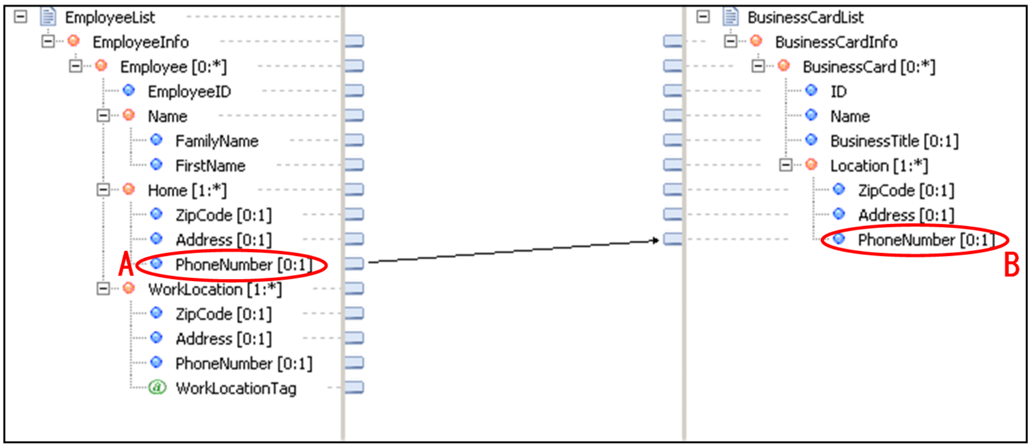


When this mapping is executed, the looping dependent target is automatically set.

(2) When a node is mapped and a loop is then specified for its ancestor node

This subsection explains the timing at which the looping dependent target is mapped when a node is mapped, and a loop is then specified for its ancestor node.

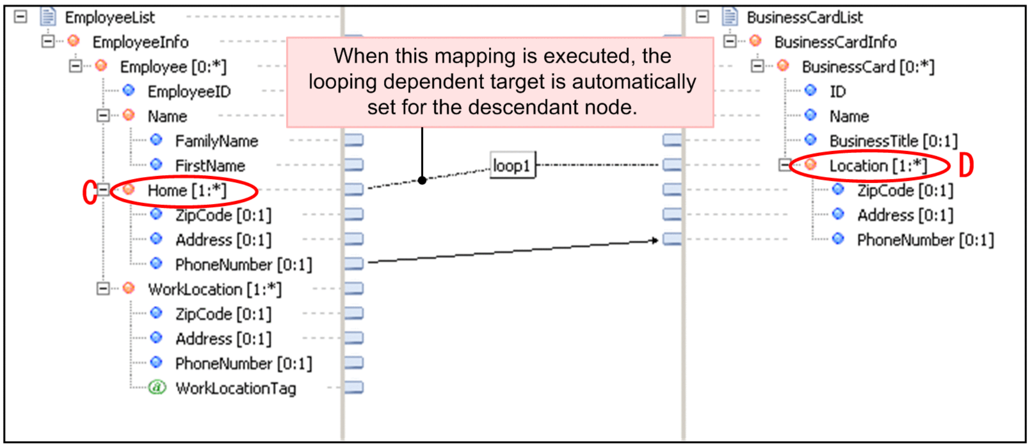
Figure 6–29: Mapping when the loop node function is not used



First, a transformation-source node A is mapped to a transformation-destination node B.

Next, node C, which is a descendant node of transformation-source node A, is mapped to node D, which is a descendant node of transformation-source node B. The following figure shows the timing at which the looping dependent target is mapped when the ancestor node is mapped later.

Figure 6–30: Timing at which the looping dependent target is mapped (When the ancestor node is mapped later)



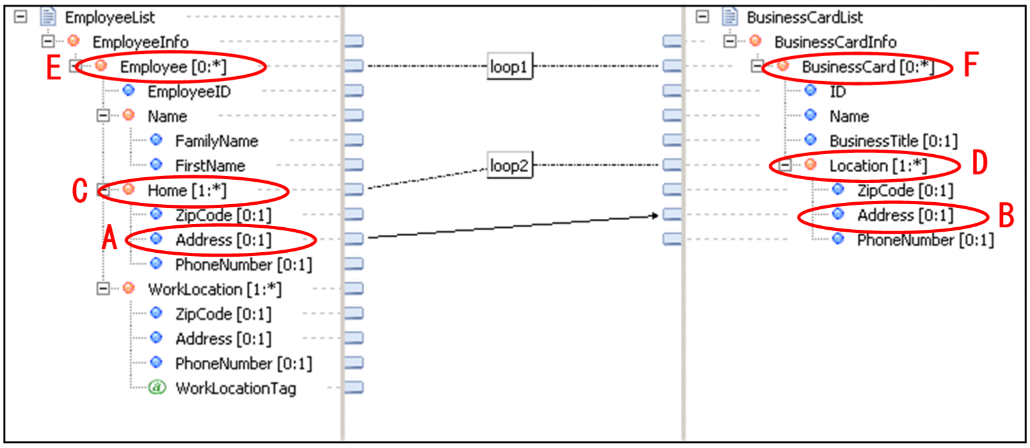
When this mapping is executed, the looping dependent target is automatically set for the descendant node.

When node C is mapped to node D, the looping dependent target is automatically set up for the descendant node. In Figure 6-29, the looping dependent target loop1 is set up for the descendant node B.

(3) When the looping of a node for which looping is set up in the ancestor node is cancelled

The timing at which the looping dependent target is mapped when the looping of a node for which looping is set up in the ancestor node is cancelled is as follows:

Figure 6–31: Mapping when the loop node function is used and when the loop node function is not used

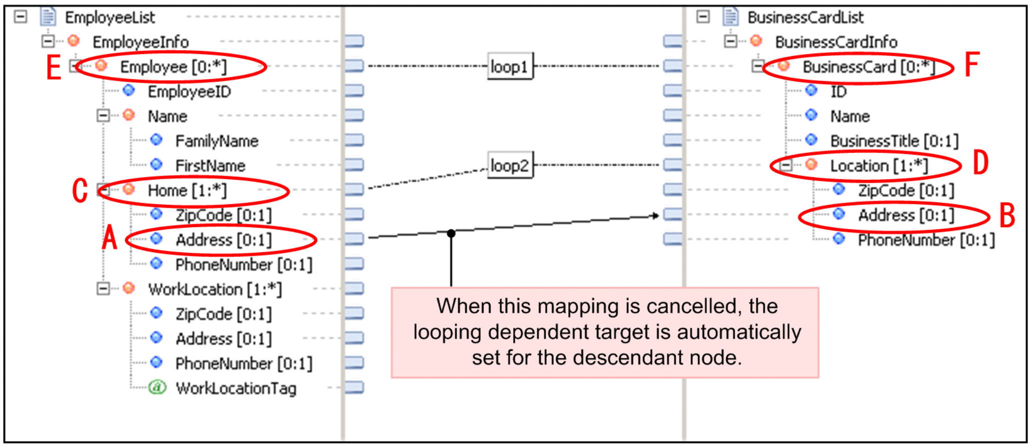


First, a transformation-source node A is mapped to a transformation-destination node B.

Next, the loop node function is used, and node C, which is a descendant node of transformation-source node A, is mapped to node D, which is a descendant node of transformation-source node B

After this, the loop node function is used, and the transformation-source node E that is the ancestor node of the transformation-source node A and transformation-source node C is mapped to the transformation-destination node F that is the ancestor node of the transformation-destination node B and transformation-destination node D.

Figure 6–32: Timing at which the looping dependent is mapped (When mapping is cancelled)



When this mapping is cancelled, the looping dependent target is automatically set for the descendant node.

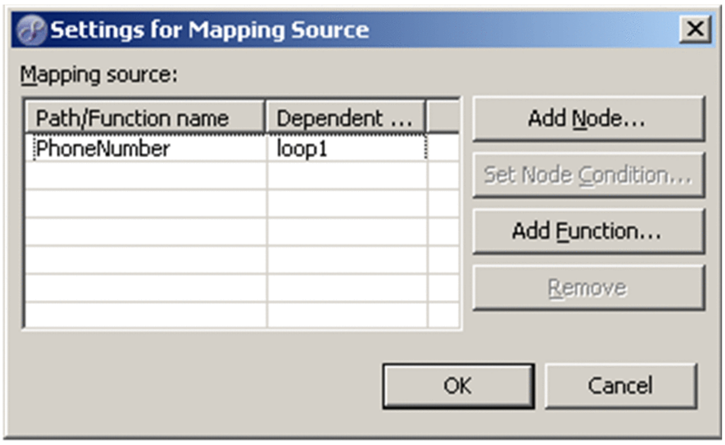
When the mapping of node C to node D is cancelled, the looping dependent target is set up automatically for the descendant node of the node in which the mapping is cancelled. In Figure 6-31, the looping dependent target loop1 is set up for the descendant node B.

6.6.4 Checking Looping Dependent Targets

To check the looping dependent targets that were automatically set up:

1. In the schema tree viewer, right--click the descendant node of the transformation-destination node for which a loop was set up, and click Mapping source.

   The Settings for Mapping Source dialog box opens.



Settings for Mapping Source

Mapping source:

Path/Function name	Dependent ...
PhoneNumber	loop1

   The relative path from the transformation-source node for which the loop node function was mapped is displayed in Path/Function name. For details about how to display path names, see *6.6.6 Displaying the Path of a Transformation-source Node for Which a Looping Dependent Target Is Specified*. The function name that becomes the looping dependent target is displayed in Dependent target.

6.6.5 Changing Looping Dependent Targets

When multiple transformation-source nodes are mapped to a single transformation-destination node, you can change the looping dependent targets that were automatically set up for the transformation-source nodes.

To change looping dependent targets:

1. In the schema tree viewer, right--click the descendant node of the transformation-destination node for which a loop was set up, and click Mapping source.

The Settings for Mapping Source dialog box opens.

2. Change Dependent target in Mapping source.

   The path of the changed dependent target becomes an absolute path. For details about how to display path names, see *6.6.6 Displaying the Path of a Transformation-source Node for Which a Looping Dependent Target Is Specified*.

3. Click OK.

6.6.6 Displaying the Path of a Transformation-source Node for Which a Looping Dependent Target Is Specified

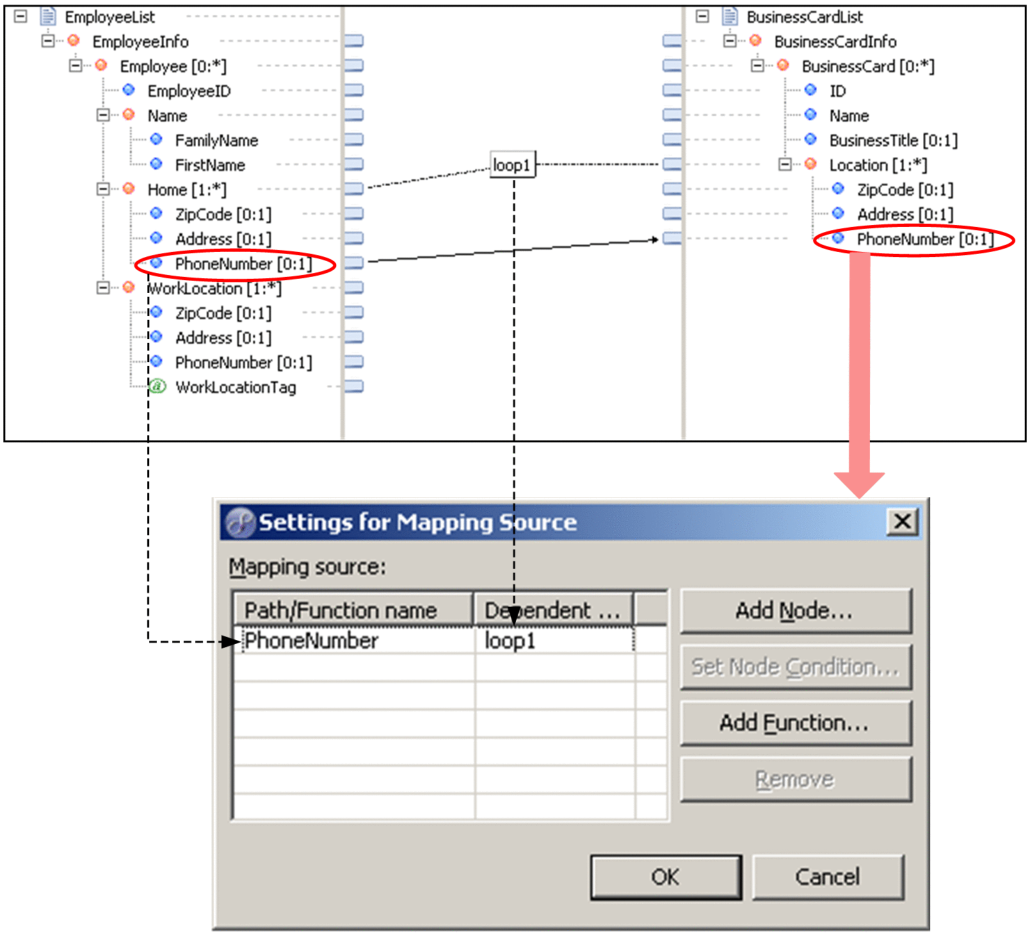
In the Settings for Mapping Source dialog box, a relative path or absolute path is displayed in Path/Function name. The following subsections explain the circumstances under which a relative path or absolute path is displayed:

(1) Relative path

When a looping dependent target is specified for the descendant transformation-source node to which the loop node function is mapped, a relative path from the transformation-source node for which the loop node function is mapped is displayed.

The following figure shows an example.

Figure 6–33: Example of a relative path



Path/Function name	Dependent ...
PhoneNumber	loop1

When the loop node function loop1 is mapped to the home node, the looping dependent target loop1 is set up for the PhoneNumber node, which is a descendant node of the home node.

In this case, if you display the Settings for Mapping Source dialog box from the PhoneNumber node after mapping, a relative path from the home node is displayed in Path/Function name.

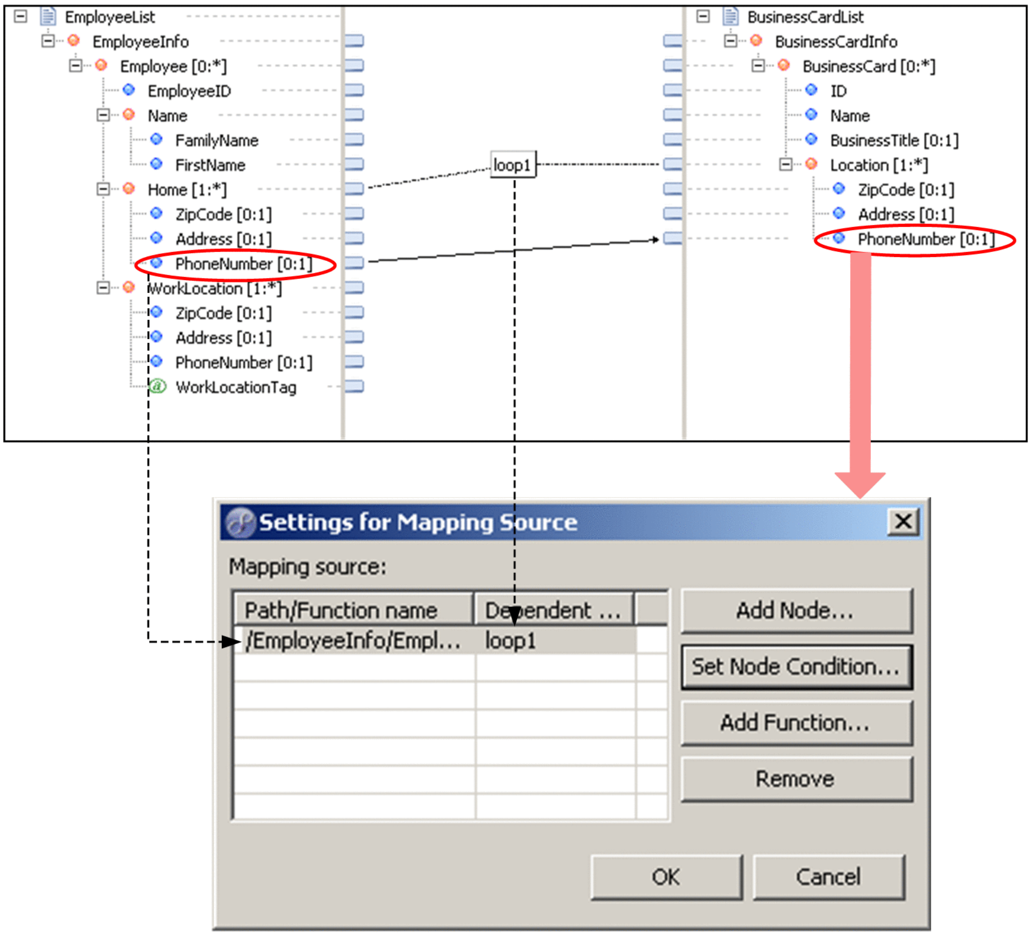
(2) Absolute path

In the following cases, an absolute path is displayed:

- When a node condition is set up for the transformation-source node to which the loop node function of the looping dependent target is mapped, or to the ancestor transformation-source node of that transformation-source node

  The following figure shows an example.

  Figure 6–34: Example 1 of an absolute path



Path/Function name	Dependent ...
/EmployeeInfo/Empl...	loop1

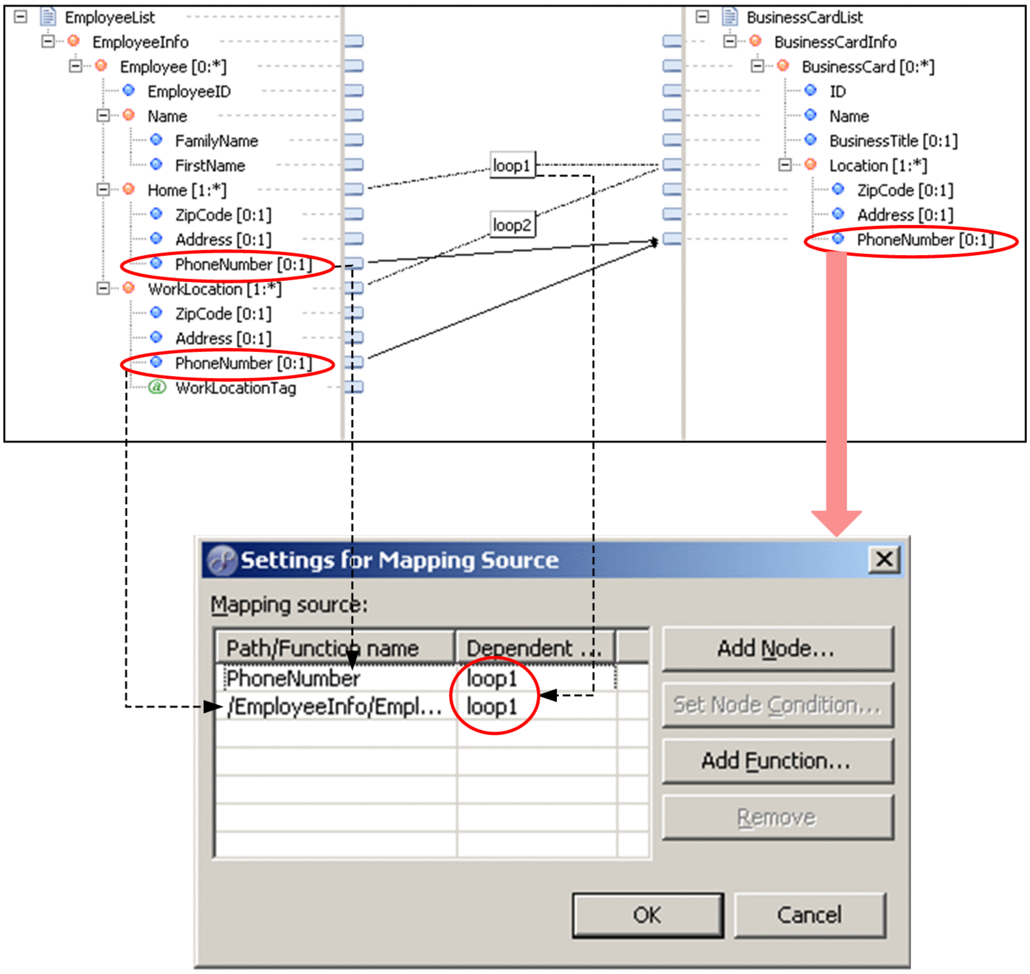
When the loop node function loop1 is mapped to the home node, the looping dependent target loop1 is set up for the PhoneNumber node, which is a descendant node of the home node. Note that a node condition is specified for the home node.

In this case, if you display the Settings for Mapping Source dialog box from the PhoneNumber node after mapping, the absolute path is displayed in Path/Function name.

- When a transformation-source node to which a looping dependent target is specified is not a descendant node of the transformation-source node to which the looping node function of the looping dependent target is mapped

  The following figure shows an example.

Figure 6–35: Example 2 of an absolute path (Before changing the dependent target)



Mapping source:

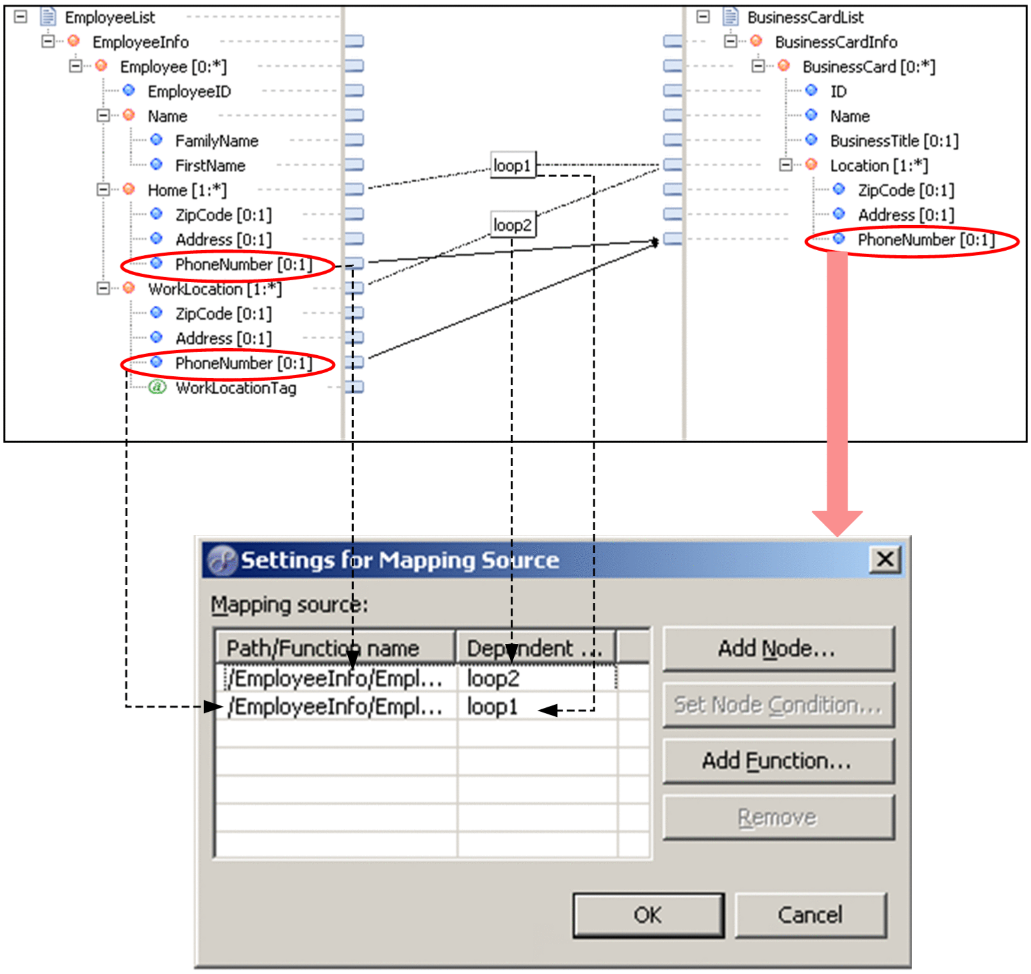
Path/Function name	Dependent ...
PhoneNumber	loop1
/EmployeeInfo/Empl...	loop1

First, the loop node function loop1 is mapped to the home node, and then the loop node function loop2 is mapped to the WorkLocation node. As a result, for both the home node's descendant PhoneNumber node and the WorkLocation node's descendant PhoneNumber node, the loop node function loop1, which was mapped first, is set up as the looping dependent target.

In this case, if you display the Settings for Mapping Source dialog box from the PhoneNumber node after mapping, the following is displayed in Path/Function name:

- The descendant PhoneNumber node of the home node (to which loop1 is mapped)
  A relative path from the home node is displayed.
- The descendant PhoneNumber node of the WorkLocation node (to which loop2 is mapped)
  The absolute path is displayed.

Figure 6–36: Example 2 of absolute path (After changing the dependent target)



Mapping source:

Path/Function name	Dependent ...
/EmployeeInfo/Empl...	loop2
/EmployeeInfo/Empl...	loop1

The dependent target for the home node's descendant phone number node is specified as loop2, which is the loop node function of the WorkLocation node.

In this case, if you display the Settings for Mapping Source dialog box from the PhoneNumber node after mapping, the display in Path/Function name changes as follows.

- The descendant PhoneNumber node of the home node (to which loop1 is mapped)

  The relative path from the home node changes to the absolute path.

6.6.7 Relating repeat process of each element by setting up the linkage path

For relating each element not depending on repetition, with the element depending on repetition and transform the same, you must set linkage path in Repeat function.

This section describes the method for setting up linkage path in Repeat function, displaying linkage path and transformation example using linkage path.

(1) Setting up the linkage path

You can link the repeat process of each element at transformation source by setting up the linkage path in Repeat function.

A path that is target of repeat linkage must fulfill the following conditions:

- Transformation destination node of mapping line depends on the corresponding Repeat function.

- Path of the transformation source node of mapping line includes the linkage path specified in Repeat function.

> **!** Important note
>
> You cannot add paths corresponding to following cases, in linkage path. If you specify, error message is displayed.
>
> - If you specify a path that is ancestor node or grandchild node of the standard path of same Repeat function
> - If you specify multiple ancestor nodes or grandchild nodes of linkage path of same Repeat function

Set up the linkage path by the following procedure.

1. Display Repeat dialog, with either of the following methods.
   - Right click Repeat function and select **Settings**.
   - Double click the Repeat function.

   For details on Repeat dialog, see "*1.6.5 Loop Settings Dialog* " in "*Service Platform Reference Guide*".
2. Click **Add node** button of **Linkage path**.
   Node selection dialog is displayed.
3. Select elements to be linked and click **OK** button.
   Path of the selected elements is displayed in **Linkage path**. For specifying multiple linkage paths, repeat steps. 2~3.
4. Click **OK** button of Repeat dialog.
   Linkage path is set up.

(2) Displaying linkage path

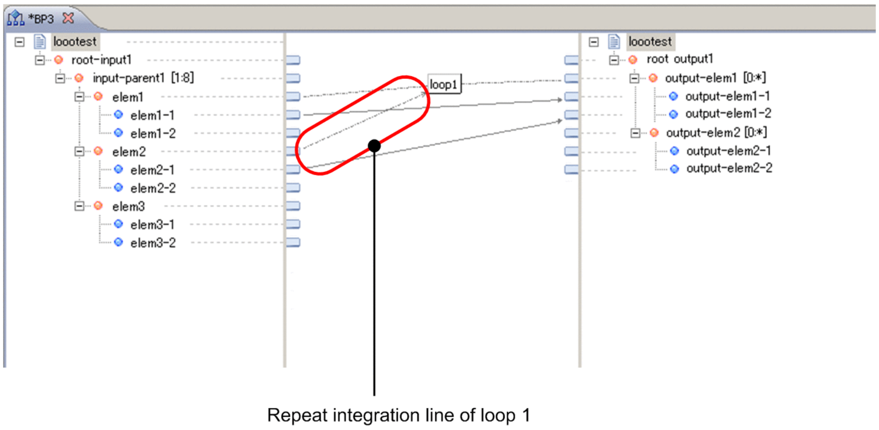
The set linkage path is displayed in the Mapping viewer and Property viewer.

In the Mapping viewer, Mapping line ⇢ (Repeat linkage line) is displayed from linkage path to Repeat function. Color of Mapping line is same as Repeat support line. For details on how to change the color of mapping line, see "*1.12.5 Changing the Mapping Line Color* " in "*Service Platform Reference Guide*".

You can cancel the mapping (linkage) by deleting Repeat linkage line from Mapping viewer. For details on how to cancel mapping, see "*6.4.5 Canceling Mapping*".

Following figure shows the display example of repeat linkage line.

Figure 6–37: FigureDisplay example of repeat linkage line

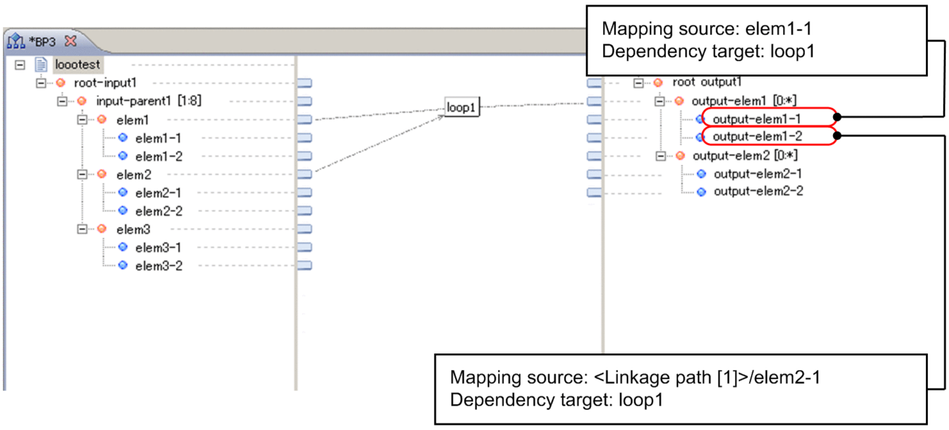


(3) Transformation example using the linkage path

Following figure shows the transformation example of Repeat function using linkage path.

- When mapping the value linked with Repeat function

Figure 6–38: FigureTranformation example 1 using the linkage path



Mapping source: elem1-1
Dependency target: loop1

Mapping source: <Linkage path [1]>/elem2-1
Dependency target: loop1

- Input data

```
<?xml version="1.0" encoding="UTF-8"?>
<root-input1>
   <input-parent1>
      <elem1>
         <elem1-1>あいうえお</elem1-1>
         <elem1-2>かきくけこ</elem1-2>
      </elem1>
      <elem2>
         <elem2-1>さしすせそ</elem2-1>
         <elem2-2>たちつてと</elem2-2>
      </elem2>
      <elem3>
         <elem3-1>なにぬねの</elem3-1>
         <elem3-2>はひふへほ</elem3-2>
      </elem3>
   </input-parent1>
   <input-parent1>
      <elem1>
         <elem1-1>アイウエオ</elem1-1>
         <elem1-2>カキクケコ</elem1-2>
      </elem1>
      <elem2>
         <elem2-1>サシスセソ</elem2-1>
         <elem2-2>タチツテト</elem2-2>
      </elem2>
      <elem3>
         <elem3-1>ナニヌネノ</elem3-1>
         <elem3-2>ハヒフヘホ</elem3-2>
      </elem3>
   </input-parent1>
</root-input1>
```

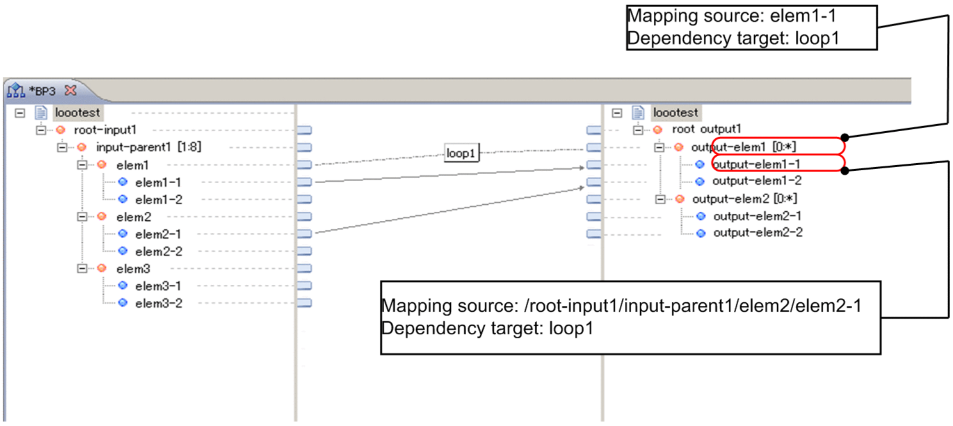
- Output data

```
<?xml version="1.0" encoding="UTF-8"?>
<root-output1>
   <output-elem1>
      <output-elem1-1>あいうえお</output-elem1-1>
      <output-elem1-2>さしすせそ</output-elem1-2>
   </output-elem1>
   <output-elem1>
      <output-elem1-1>アイウエオ</output-elem1-1>
      <output-elem1-2>サシスセソ</output-elem1-2>
   </output-elem1>
</root-output1>
```

Acquiring elem2-1
of position linked
with loop1

- When mapping a value that is not linked with Repeat function

Figure 6–39: FigureTransformation example 2 using linkage path



Mapping source: elem1-1
Dependency target: loop1

Mapping source: /root-input1/input-parent1/elem2/elem2-1
Dependency target: loop1

●Input data

```
<?xml version="1.0" encoding="UTF-8"?>
<root-input1>
  <input-parent1>
    <elem1>
      <elem1-1>あいうえお</elem1-1>
      <elem1-2>かきくけこ</elem1-2>
    </elem1>
    <elem2>
      <elem2-1>さしすせそ</elem2-1>
      <elem2-2>たちつてと</elem2-2>
    </elem2>
    <elem3>
      <elem3-1>なにぬねの</elem3-1>
      <elem3-2>はひふへほ</elem3-2>
    </elem3>
  </input-parent1>
  <input-parent1>
    <elem1>
      <elem1-1>アイウエオ</elem1-1>
      <elem1-2>カキクケコ</elem1-2>
    </elem1>
    <elem2>
      <elem2-1>サシスセソ</elem2-1>
      <elem2-2>タチツテト</elem2-2>
    </elem2>
    <elem3>
      <elem3-1>ナニヌネノ</elem3-1>
      <elem3-2>ハヒフヘホ</elem3-2>
    </elem3>
  </input-parent1>
</root-input1>
```

● Output data

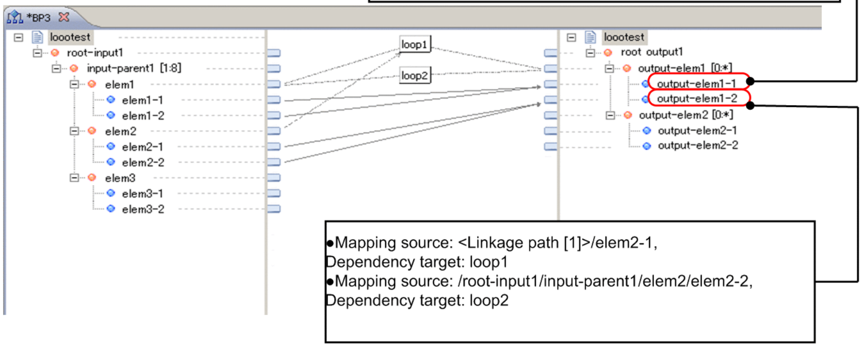
```
<?xml version="1.0" encoding="UTF-8"?>
<root-output1>
  <output-elem1>
    <output-elem1-1>あいうえお</output-
elem1-1>
    <output-elem1-2>さしすせそ</output-
elem1-2>
  </output-elem1>
  <output-elem1>
    <output-elem1-1>アイウエオ</output-
elem1-1>
    <output-elem1-2>さしすせそ</output-
elem1-2>
  </output-elem1>
</root-output1>
```

Always acquiring first elem2-1, as it is not integrated

- When referencing value linked with multiple Repeat functions

Figure 6–40: FigureTransformation example 3 using linkage path

●Mapping source: elem1-1, Dependency target: loop1
●Mapping source: elem1-2, Dependency target: loop2



●Mapping source: <Linkage path [1]>/elem2-1,
Dependency target: loop1
●Mapping source: /root-input1/input-parent1/elem2/elem2-2,
Dependency target: loop2

- Input data

```
<?xml version="1.0" encoding="UTF-8"?>
<root-input1>
    <input-parent1>
        <elem1>
            <elem1-1>あいうえお</elem1-1>
            <elem1-2>かきくけこ</elem1-2>
        </elem1>
        <elem2>
            <elem2-1>さしすせそ</elem2-1>
            <elem2-2>たちつてと</elem2-2>
        </elem2>
        <elem3>
            <elem3-1>なにぬねの</elem3-1>
            <elem3-2>はひふへほ</elem3-2>
        </elem3>
    </input-parent1>
    <input-parent1>
        <elem1>
            <elem1-1>アイウエオ</elem1-1>
            <elem1-2>カキクケコ</elem1-2>
        </elem1>
        <elem2>
            <elem2-1>サシスセソ</elem2-1>
            <elem2-2>タチツテト</elem2-2>
        </elem2>
        <elem3>
            <elem3-1>ナニヌネノ</elem3-1>
            <elem3-2>ハヒフヘホ</elem3-2>
        </elem3>
    </input-parent1>
</root-input1>
```

- Output data

```
<?xml version="1.0" encoding="UTF-8"?>
<root-output1>
    <output-elem1>

        <output-elem1-1>あいうえお</output-elem1-1>
        <output-elem1-2>さしすせそ</output-elem1-2>
    </output-elem1>

    <output-elem1>
        <output-elem1-1>アイウエオ</output-elem1-1>
        <output-elem1-2>サシスセソ</output-elem1-2>

   </output-elem1>
    <output-elem1>

        <output-elem1-1>かきくけこ</output-elem1-1>
        <output-elem1-2>たちつてと</output-elem1-2>
    </output-elem1>
    <output-elem1>

        <output-elem1-1>カキクケコ</output-elem1-1>
        <output-elem1-2>たちつてと</output-elem1-2>
    </output-elem1>
</root-output1>
```

Acquiring elem2-1 of position linked with loop1

Always acquiring first elem2-2, as it is not integrated with loop2

6.7 Specifying Node Conditions

You can specify a condition for the transformation-source node and map it only when this condition is satisfied. Specifying a condition for the transformation-source node is called node condition setting. As described below, the method for starting node condition setting differs depending on the mapping destination.

- When the mapping destination is a transformation-destination node

  Start node condition setting from the Settings for Mapping Source dialog box. The Settings for Mapping Source dialog box opens when you right--click the transformation-destination node and choose Mapping source.
- When the mapping destination is a function

  Start node condition setting from the dialog box for setting up each function. You can use one of the following methods to open the dialog box for setting up each function:
  - Right--click each function, and choose **Setting**.
  - Double--click each function.

To set up node conditions:

1. In the Settings for Mapping Source dialog box or the dialog box for setting up each function, choose the transformation-source node for which to specify a node condition.
2. Click Set Node Condition.

   The Set Node Condition dialog box opens.

   The Set Node Condition dialog box displays the transformation-source node selected in the Settings for Mapping Source dialog box and its ancestor node. You can specify node conditions for all nodes displayed in the Set Node Condition dialog box.
3. Choose the node for which to specify a node condition, and click Set Condition.

   The Condition Settings dialog box opens.
4. Specify a condition.

   From the following radio buttons, choose the condition to be specified in Condition and specify details for that condition.
   - **Node or function to return the boolean value** radio button
   - **Condition format** radio button
   - **Logical operation of two conditions** radio button

   The specified condition is displayed in Condition to be generated.

   To specify a transformation-source node or function in Condition, specify the following transformation-source node or function:
   - Transformation-source node

     Node that is a mapping target# and that has a simple content or attribute
   - Function

     Function that is not a control function and for which no mapping line is specified on the output side
5. To specify a condition that is opposite in content from the condition displayed in Condition to be generated, choose the Negate the condition check box.
6. Click OK.

   The condition specified in the Condition Settings dialog box is displayed in Condition in the Set Node Condition dialog box.
7. To specify node conditions for other nodes, repeat steps 3. to 6.
8. In the Set Node Condition dialog box, click OK.

   The node conditions are set up.

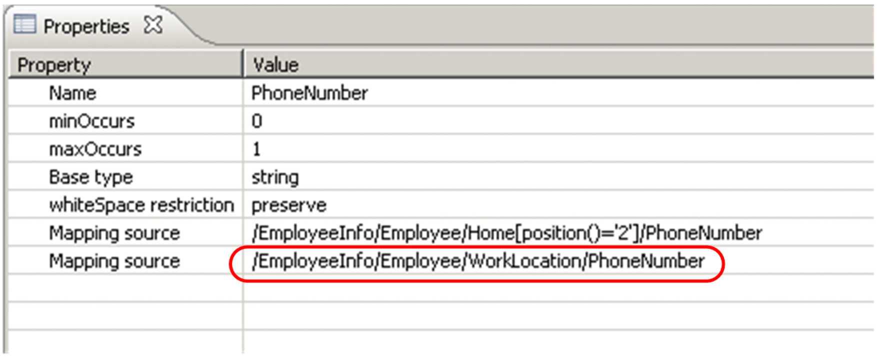
   If you specify a transformation-source node or function that is not a mapping source inside a condition specified in the Condition Settings dialog box, a mapping line ⇢ (condition line) is set up between the transformation-source node or function inside that condition and the mapping destination.

#

For mapping targets, see *6.10.1 Mapping Targets and Non--Mapping Targets*.

Tip

When you choose a transformation-destination node for which a condition was specified, the specified node condition is displayed in Mapping source in the property area. A node condition enclosed inside square brackets "[ ]" is displayed following the node name for which the condition was specified. When the node (specified in Condition of the Condition Settings dialog box) is a descendant node of the node (specified in the Set Node Condition dialog box) for which the condition was specified, a relative path from the node for which the condition was specified is displayed. In all other cases, the absolute path is displayed.



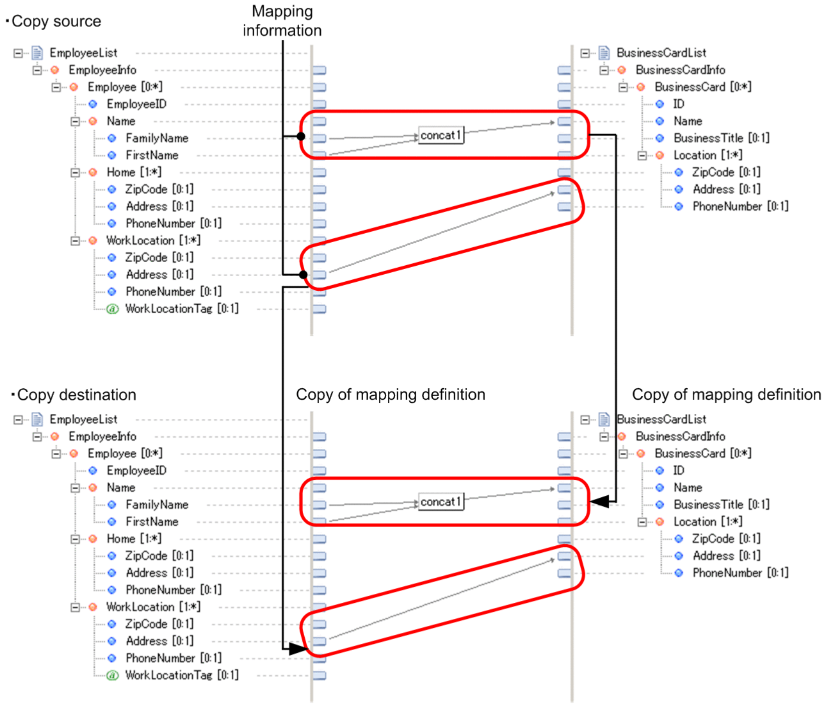
Property	Value
Name	PhoneNumber
minOccurs	0
maxOccurs	1
Base type	string
whiteSpace restriction	preserve
Mapping source	/EmployeeInfo/Employee/Home[position()='2']/PhoneNumber
Mapping source	/EmployeeInfo/Employee/WorkLocation/PhoneNumber

6.8 Copying Mapping Definitions

You can save the contents mapped in the Data Transformation Definition screen as mapping definitions. You can copy and use the saved mapping definitions when creating other data transformation definitions.

The following figure shows copying of mapping definitions.

Figure 6–41: Copying of mapping definitions



Mapping definitions are copied for each piece of mapping information. Mapping information refers to a transformation-destination node, a transformation-source node that can be obtained by tracing the mapping lines from the transformation-destination node, and functions and mapping lines.

You can copy mapping definitions even if the schema type used when creating a mapping definition file for the copy source (XSD file or FDX file) differs from the schema type of the copy destination.

There are two types of mapping definition copying method: structure matching copy mapping, and element name matching copy mapping. Structure matching copy mapping can be used when the element names and schema structures of the copy source and destination match. Therefore, if the node to which mapping definitions are to be copied already has mapping lines connected, mapping definitions cannot be copied. Element name matching copy mapping can be used when the element names of the copy source and destination are similar. Therefore, unlike structure matching copy mapping, mapping definitions can be copied even if the schema structures of the copy source and destination are different. For these copying methods, see *6.8.4 Copying mapping definitions*.

6.8.1 Flow of copying mapping definitions

The flow of copying mapping definitions is as follows:

1. Save the contents mapped in the Data Transformation Definition screen as mapping definitions.
   For details about how to save mapping definitions, see *6.8.2 Saving mapping definitions*.
2. Register the saved mapping definition as the copy source.
   For details about how to register mapping definitions, see *6.8.3 Registering mapping definitions*.
3. Specifying the transformation-source and destination schemas and open the Data Transformation Definition screen that is the copy destination.

For details about how to open the Data Transformation Definition screen, see *6.3.1 Procedure for Defining Data Transformation* . Note that mapping must not be carried out if the Data Transformation Definition screen is open.

4. Copy the mapping definition.

   Schema information of the mapping definitions of the copy source and destination are compared and if the elements are similar, mapping definitions are copied automatically. For conditions for copying mapping definitions, see *6.8.5 Determining similarities*.

   For details about how to copy mapping definitions, see *6.8.4 Copying mapping definitions*.

6.8.2 Saving mapping definitions

The following points describes how to save the mapping definition:

1. Right click the applicable location of the transformation-source schema tree viewer, mapping viewer or transformation-destination schema tree viewer in the Data Transformation Definition screen and choose **Save mapping definition**.

   The **Save mapping definition** dialog box appears.

   Note that the directory appearing first in the **Save mapping definition** dialog box can be set in any location in the **Settings** dialog box. For details see the contents related to the **Settings** dialog box in *Cosminexus Service Platform Reference*.
2. Specify the file for saving the mapping definition (extension:.mdo)
3. Click **OK** button.

   The mapping definition is saved.

6.8.3 Registering mapping definitions

This section describes how to register the mapping definition. Only one mapping definition can be saved for 1 Eclipse. A registered mapping definition can be shared by all Data Transformation Definition screens. Note that if Eclipse closes, the registered mapping definition is destroyed.

1. Right click the applicable location of the transformation-source schema tree viewer, mapping viewer or transformation-destination schema tree viewer in the Data Transformation Definition screen and choose **Register mapping definition**.

   The **Register mapping definition** dialog box appears.
2. Click **Choose** and specify the mapping definition file to be registered as the copy source (extension:.mdo).

   Note that the directory appearing when **Choose** is clicked box can be set in any location in the **Settings** dialog box. For details see the contents related to the **Settings** dialog box in *Cosminexus Service Platform Reference*.
3. Click **OK** button.

   The mapping definition is registered.

6.8.4 Copying mapping definitions

This section describes a method to copy the mapping definition.

Reference note

When you copy the mapping definition, name same as copy source function is set in the copy destination function. If the function with same name already exists in mapping definition editor of copy destination, name to which serial number is added at the end, is set.

(1) Structure matching copy mapping

Structure matching copy mapping is the copy method that can be used when not only the element name of copy destination and copy source, but the schema structure matches. Method to copy mapping definition with structure matching copy mapping is as follows:

1. In the Transformation destination schema tree viewer of the Data transformation definition screen, right click Copy contents element and select [Copy mapping]-[structure matching].

   The **Set copy mapping source** dialog box appears.
2. Click Add node.

   Node chooseion dialog appears.
3. Choose the element to be copied and click **OK** button.

   The path of the selected node is added in **Path**. If multiple schemas exist in the transformation-source, choose multiple elements to be copied.
4. Click **OK** button.

   The mapping definition is copied.

(2) Element name matching copy mapping

Element name matching copy mapping is the copy method that can be used when element names of copy source and copy destination match. Method to copy the mapping definition with element name matching copy mapping is as follows:

1. In the Transformation destination schema tree viewer of the Data transformation definition screen, right click copy contents element and select [Copy mapping]-[Element name matching].

   Copy mapping range setting dialog is displayed.
2. Set copy source and copy destination information.

   Set copy source information in **Copy mapping source** and copy destination information in **Copy mapping destination**.

   Select node to be set in each item by clicking **Select node** button next to the item. The selected node is displayed in each item.
3. Click **OK** button.

   Mapping definition is copied.

! Important note

In the following cases, error is displayed and mapping file is not copied.

- When mapping definition file of copy source (extension:.mdo) has not been registered
- When mapping definition file of copy source (extension:.mdo) cannot be read
- When registered mapping definition file of copy source (extension:.mdo) is blank

6.8.5 Determining similarities

If mapping definitions are copied, the schema element of the mapping definition of the copy source and destination are compared and if the elements are similar, they are copied. If multiple elements are similar, they are copied in the priority order from the top. Priority according to which mapping is copied, is explained here.

! Important note

Functions that fulfill the following condition are not copied regardless of priority.

- Functions that do not have transformation source node corresponding to sort conditions, among the Repeat functions for which sort condition is set
- Functions that do not have transformation source node corresponding to linkage path, among the Repeat function for which linkage path is set
- Functions that do not have transformation source node corresponding to conditions, among the Select function for which condition is set

(1) Structure matching copy mapping

Following table describes the priority according to which mapping is copied when you copy mapping definition with the structure matching copy mapping.

Table 6–5: Priority order of copying mapping definitions

Priority order	Schema logical name	New path	Element name
1	Y	Y	Y
2	--	Y	Y

Legend:
Y: Matches.
--: Does not match.

(2) Element name matching copy mapping

Following table describes the priority according to which mapping is copied when you copy mapping definition with the element name matching copy mapping.

Table 6–6: TablePriority according to which mapping definition is copied (element name matching copy mapping)

Priority	Element		Path	
	Namespace	Element name	Namespace	Element name
1	N	N	Y	Y
2	N	N	N	Y
3	Y	Y	N	A
4	N	Y	N	A
5	N	A	N	A

(Legend)
Y: Matches
A: Partially matches
N: Does not match

In element name matching copy mapping, you must set such that conflict does not occur in the dependency relation of copied elements. Copy the transformation destination node for which dependency relation is set, by conforming to the following conditions.

- When transformation destination node which is dependency target has been copied, copy such that it is included in grandchild of transformation destination node of dependency target.
- If you cannot copy the transformation destination node that is dependency target, search the transformation destination node having higher similarity level than transformation target node that is dependency target and copy such that it is included in grandchild of that node. If you cannot find a transformation destination node having higher similarity level, copy as no dependency relation.

Also, if node condition has been set in the following value, copy such that ancestor node for which conditions are set become the ancestor node of the transformation sauce node that is connection source between transformation destination node and function.

- Input value of conversion destination node
- Input value of function
- Linkage path of repeat function

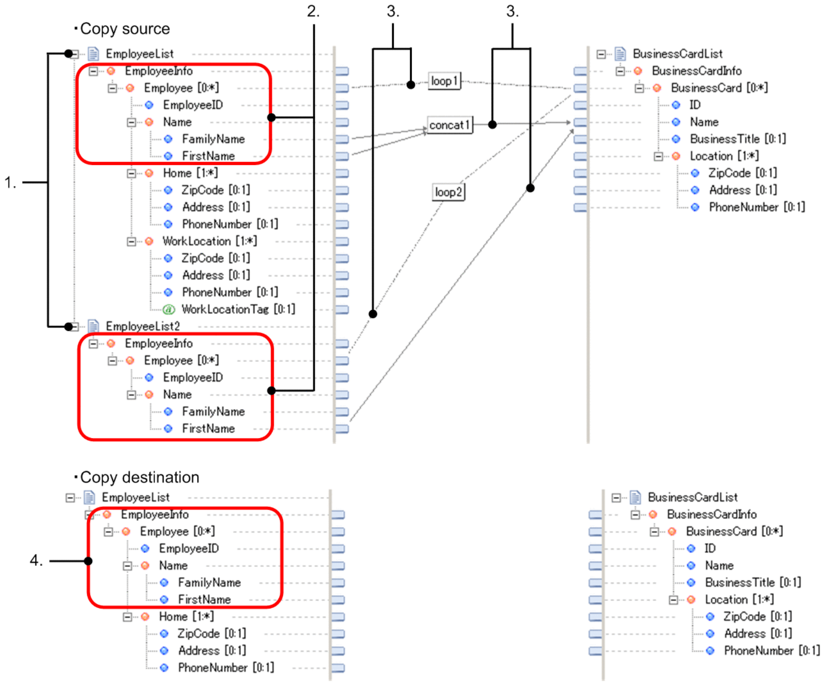
6.8.6 Notes on copying mapping definitions

Planned copying might not be executed depending on the schema contents of the copy source and destination. This section describes the notes on copying mapping definitions.

(1) If multiple targets are to be copied

If all the conditions from 1. to 4. are applicable, copy targets are considered to be multiple targets.

Figure 6–42: Example of copy source and copy destination if multiple targets are to be copied

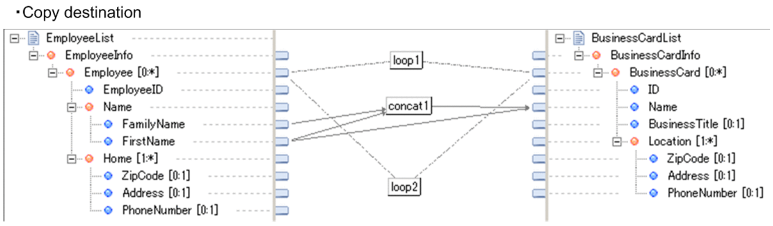


Conditions

1. Multiple schemas are registered in the transformation-source schema of the copy source mapping definition
2. A node with the same path exists in the transformation-source schema of the copy source mapping definition
3. Mapping information includes the node of 2. in the copy source mapping definition
4. A node with the same path as 2. exists in 1 transformation-source schema of the copy destination

The following figure shows the copy result for Figure 6-42.

Figure 6–43: Copy result if multiple targets are to be copied

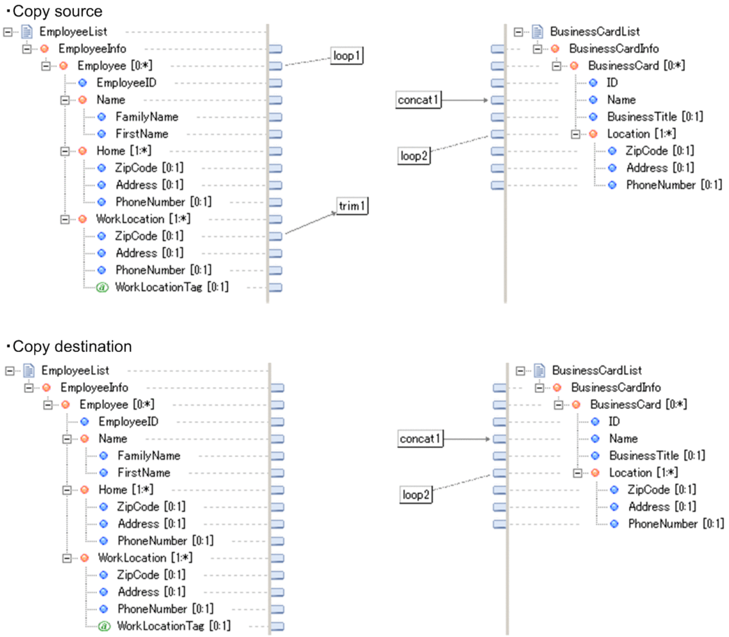


(2) If the mapping definition of the copy source is incomplete

If the transformation-source and destination nodes are not connected and the copy source mapping definition is incomplete, only mapping of the transformation-destination node is copied and mapping of the transformation-source node is not copied.

The following figure shows an example of copying the mapping definition if the mapping definition of the copy source is incomplete.

Figure 6–44: Copying a mapping definition if the copy source mapping definition is incomplete



・Copy source

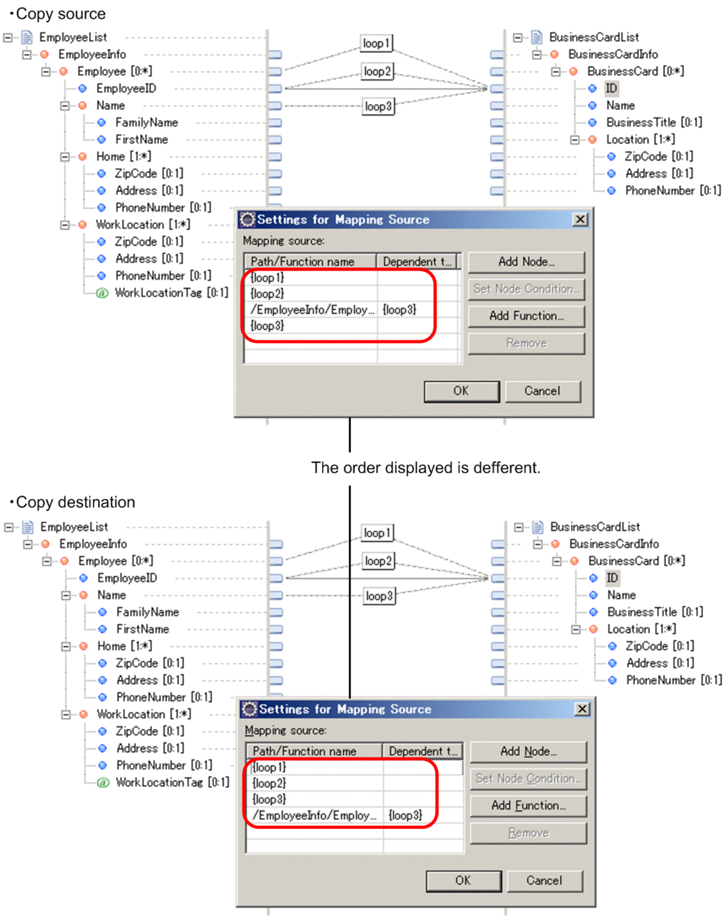
・Copy destination

(3) If multiple loop node functions are connected to 1 element

If the mapping definition copy source is connected to multiple loop node functions in 1 simple content element or complex content element and if the mapping definition is copied, the loop node function is copied first. At this stage, if the **Set mapping source** dialog box appears, the display order of **Path or function name** in the copy destination might be different from the copy source. This does not affect the copy result of the mapping definition.

The following figure shows an example of copying mapping definitions if multiple loop node functions are connected to 1 element.

Figure 6–45: Copying mapping definitions if multiple loop node functions are connected to 1 element



Copy source — Settings for Mapping Source

Mapping source:

Path/Function name	Dependent t...
{loop1}	
{loop2}	
/EmployeeInfo/Employ...	{loop3}
{loop3}	

Copy destination — Settings for Mapping Source

Mapping source:

Path/Function name	Dependent t...
{loop1}	
{loop2}	
{loop3}	
/EmployeeInfo/Employ...	{loop3}

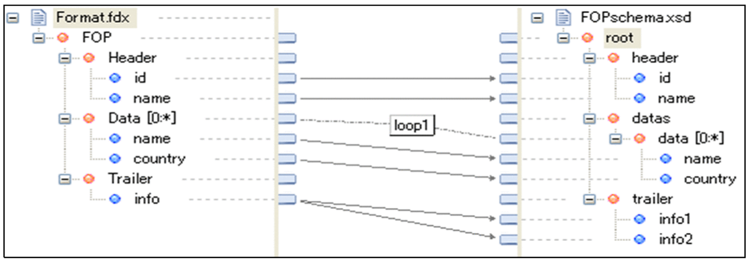
(4) When mapping target is restricted

When element mapped in mapping definition file of copy source cannot be mapped in mapping definition file of copy destination, mapping of that element is not copied.

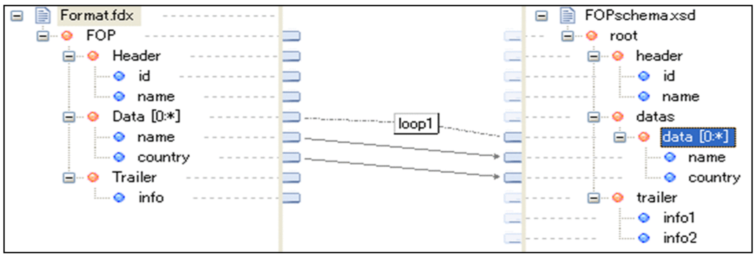
For details on the mapping target restrictions, see "*6.4.8 Restricting mapping range*".

Figure 6–46: FigureCopy of mapping when mapping target is restricted

- Copy source



- Copy destination



6.9 Creating Java programs to be used in the custom function

You can use a Java program as the function using the custom function. You can use this for processes other than those in other functions provided by uCosminexus Service Architect.

Custom function has following types:

- Character string type Custom function

  This is Custom function having argument and return value as character string type.

Type of argument	java.lang.String
Type of return value	java.lang.String

- Node list type custom function

  This is Custom function having argument and return value as node list type. In argument, you can pass nodes of simple contents element, complex contents element, any element or any Attribute. Also, you can return node directly, as return value.

Type of argument	java.lang.Object
Type of return value	org.w3c.dom.NodeList

Data type that is actually passed to argument differs depending on the contents of following arguments:

Table 6–7: TableArgument definition contents

Contents of argument	Actual type of argument
Simple contents element/complex contents element	org.w3c.dom.NodeList
Attribute	org.w3c.dom.NodeList
any attribute	org.w3c.dom.NodeList
anyAttribute attribute	org.w3c.dom.NodeList
Character string type Custom function	java.lang.String
Node list type custom function	org.w3c.dom.NodeList
Other functions (excluding Repeat function and Select function)	java.lang.String

This section describes the method of creating Java program invoked from Custom function. In Custom function, invoked Java program is referred as transformation function. For details on how to invoke a Java program, see "*6.5.23 Invoke a Java program created by the user*".

Create the transformation function by the following procedure:

1. Creating the transformation function definition file

   Create the transformation function definition file defining the configuration of the transformation function. For details about how to create the transformation function definition file, see *6.9.1 Creating the Transformation Function Definition File*.

2. Creating the Java form file

   Create the Java form file to be used when coding the Java program on the basis of the created transformation function definition file. For details about how to create the Java form file, see *6.9.2 Creating the Java form file*.

3. Coding, building and debugging Java programs

   Code, build and debug Java programs using Java development tools. For details about how to code, build and debug Java programs, see *6.9.4 Coding, building and debugging Java programs*.

4. Packaging Java programs

   Package the Java program created using building tools in a jar file. For details about how to package Java programs, see *6.9.5 Packaging Java programs*.

Important note

- Memory requirement (stack memory) increases according the number of arguments in the invoked Java program. If there are many arguments, stack overflow error might occur. Aim for a maximum of 10 arguments.
- Resource secured at the time of invoking Java program and maintained even when the process ends. Therefore, when high load is applied on entire system, OutOfMemoryError might occur due to causes like insufficient Java heap or insufficient Perm heap. Due to this, you must implement process to release resources properly or error processing like roll back, if OutOfMemoryError occurs.

6.9.1 Creating the Transformation Function Definition File

The transformation function definition file defines the configuration of the transformation function invoked from the custom function. Use the file to

- Create the Java form file
- Define the custom function

The following is a descriptive example of the transformation function definition file. The slanting part is data specified by the user.

```
<?xml version="1.0" encoding="UTF-8"?>
<func:customFunc xmlns:func="http://www.hitachi.co.jp/soft/xml/cosminexus/cscdt/functions">
 <func:jar name="sample.jar">
  <func:package name="jp.co.Hitachi.soft.sample">
   <func:class name="SampleClass">
    <func:method name="sampleFunction">
     <func:comment>Description of sampleFunction</func:comment>
     <func:arguments>
      <func:argument name="arg1">
       <func:comment>Description of arg1</func:comment>
      </func:argument>
      <func:argument name="arg2">
       <func:comment>Description of arg2</func:comment>
      </func:argument>
      <func:argument name="arg3">
       <func:comment>Description of arg3</func:comment>
      </func:argument>
     </func:arguments>
    </func:method>
    <func:method name="sampleStringFunction">
     <func:comment>Description of sampleStringFunction</func:comment>
     <func:arguments>
      <func:argument name="arg1" type="String">
       <func:comment>Description of arg1</func:comment>
      </func:argument>
     </func:arguments>
     <func:return type="String"/>
    </func:method>
    <func:method name="sampleNodeListFunction">
     <func:comment>Description of sampleNodeListFunction</func:comment>
     <func:arguments>
      <func:argument name="arg1" type="Object">
       <func:comment>Description of arg1</func:comment>
      </func:argument>
     </func:arguments>
     <func:return type="NodeList"/>
    </func:method>
   </func:class>
  </func:package>
 </func:jar>
</func:customFunc>
```

The user creates the transformation function definition file using the XML editor. The following points describes how to create the transformation function definition file using the XML editor of Eclipse.

1. In the Eclipse menu, choose **File**, **New** and **Others**.
   The **New** dialog box appears.
2. Choose **XML** and **XML (basic template)** and then click **Next**.
   The **XML file** page appears.

3. Specify the directory for saving the transformation function definition file and the file name and then click **Next**.

   The **Create the XML file** from the following page appears.

4. Choose the **Create the SML file from XML schema or file** radio button and then click **Next**.

   The **Choose XML schema or file** page appears.

5. Click **Import file**.

   The **Import** dialog box appears.

6. Specify the following directory in **From the following directories**.

   ```
   uCosminexus Service Architect installation directory\CSCTE\resources
   \customfunc
   ```

7. Check the **customfunc** check box and the **customfunction_XMLSchema.xsd** check box.

8. Specify the folder for importing the schema file in **Import--destination folder**.

9. Click **End**.

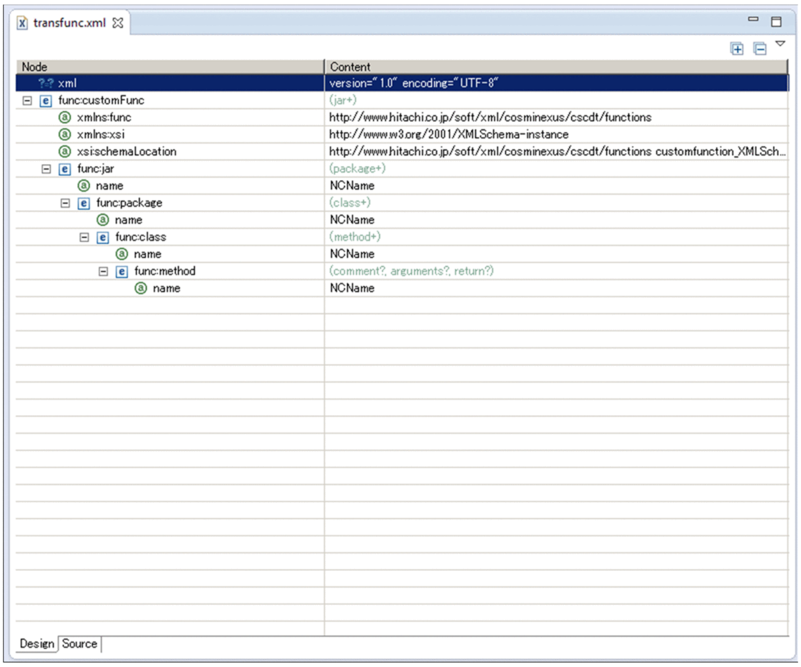
   The **Choose XML schema or file** page reappears.

10. Click **Next**.

    The **Choose root element** page appears.

11. Specify **customFunc** in **Root element** and then click **End**.

    The form of the transformation function definition file is created.



Node	Content
?-? xml	version="1.0" encoding="UTF-8"
func:customFunc	(jar+)
xmlns:func	http://www.hitachi.co.jp/soft/xml/cosminexus/cscdt/functions
xmlns:xsi	http://www.w3.org/2001/XMLSchema-instance
xsi:schemaLocation	http://www.hitachi.co.jp/soft/xml/cosminexus/cscdt/functions customfunction_XMLSch...
func:jar	(package+)
name	NCName
func:package	(class+)
name	NCName
func:class	(method+)
name	NCName
func:method	(comment?, arguments?, return?)
name	NCName

12. Add the required elements in the form of the transformation function definition file and create the file.

    The following tabs are used in the transformation function definition file.

Table 6–8: List of tabs used in the transformation function definition file

Item no.	Element or attribute name	Contents	Type	Number of items		Number of characters	
				Minimum	Maximum	Minimum	Maximum
1	customFunc	Transformation function definition	Element	1	1	--	--
2	jar	jar file information	Element	1	255	--	--
3	name	jar file name	Attribute	1	1	1	100
4	package	Package information	Element	1	255	--	--

Item no.	Element or attribute name	Contents	Type	Number of items		Number of characters	
				Minimum	Maximum	Minimum	Maximum
5	name	Package name	Attribute	1	1	1	255
6	class	Class information	Element	1	255	--	--
7	name	Class name	Attribute	1	1	1	100
8	method	Method information	Element	1	255	--	--
9	name	Method name	Attribute	1	1	1	100
10	comment	Method comment	Element	0	1	0	1,024
11	arguments	Argument information	Element	0	1	--	--
12	argument	Argument	Element	1	255	--	--
13	name	Argument name	Attribute	1	1	1	100
14	Type	Argument type (String or Object)	Attribute	0	1	-	-
15	comment	Argument comment	Element	0	1	0	1,024
16	return	Return value	Element	0	1	-	-
17	type	Return value type (String or NodeList)	Attribute	1	1	-	-

(Legend)

--:There is no limit on the number of characters.

Note the following when creating the transformation function definition file:

- Do not use the same class name and method name.
- Do not specify String in the class name.
- An unintended result such as Java syntax error etc might occur depending on the contents because the comment is embedded as it is in 1 line of the specified character string. If an unintended result occurs, modify the Java file directly.

6.9.2 Creating the Java form file

The Java form file is the form of the Java program describing the configuration of the package, class and method (including argument) for developing the transformation function. The Java form file is created automatically by uCosminexus Service Architect on the basis of the transformation function definition file.

The following is an example of the Java form file. The slanting part is data entered from the transformation function definition file.

Example of not having NodeList type in Method definition is as follows:

```
/**
 * Java Template
 */

package jp.co.Hitachi.soft.sample;

public class SampleClass {

    /**
     * Description related to the custom function...
     *
     * @param arg1
     *  Description related to the selected argument...
     * @param arg2
```

```
      *  Description related to the selected argument...
      * @param arg3
      *  Description related to the selected argument...
      *
      * @return
      */
    public static String SampleFunction(String arg1, String arg2, String arg3) {

        // TODO Add Java Code Here
        return "";
    }
}
```

Example of having NodeList type in Method definition is as follows:

```
/**
 * Java Template
 */

package jp.co.Hitachi.soft.cscdt.sample;

import org.w3c.dom.NodeList;

public class SampleFunctionClass {

    /**
     * Custom function related description...
     *
     * @param arg1
     *  Selected argument related description...
     *
     * @return
     */
    public static String SampleStringFunction(String arg1) {

        // TODO Add Java Code Here
        return "";
    }

    /**
     * Custom function related description...
     *
     * @param arg1
     *  Selected argument related description...
     *
     * @return
     */
    public static NodeList SampleNodeListFunction(Object arg1) {

        // TODO Add Java Code Here
        return null;
    }
}
```

Add "import org.w3c.dom.NodeList;" when 1 or more Method definitions having return value of NodeLit type in Class definition exist. However, do not add when all types of return value within Method definition is of String type.

Implement the process in `// TODO Add Java Code Here` and create the Java program.

The following procedure shows how to create the Java form file.

1. In the Eclipse menu, choose **File**, **New** and **Java project**.
   The **New Java project** dialog box appears.
2. Specify the project name and then click **End**.
   The Java project is created in the Eclipse workspace.
3. Move the transformation function definition file created in *6.9.1 Creating the Transformation Function Definition File* to the following location:
   `Directory of created Java projects\src`
4. In the Eclipse menu, choose **File** and **New**.
   The Eclipse screen is updated.
5. Right click the transformation function definition file in the package and explorer view and choose **HCSC--Definer** and **Create the Java form file**.

The Java form file output wizard appears.

6. Specify the output location and the character code set of the output file of the Java form file and then click **Next**.

   You can choose the output character code from **MS932**, **UTF--8** and **UTF--16**. Note that in **UTF--16**, the BOM control code (0xFEFF) is added.

7. Check the check box of the output class in the Java form file and then click **End**.

   The Java form file output wizard closes and the Java form file is created in the specified location.

6.9.3 Referring to external jar from transformation function

In Custom function, you can handle multiple user-defined jar file (external jar), in addition to transformation function jar file.

When you want to refer external jar from transformation function jar file, add external jar to usrconf.cfg (option definition file for java application). Description of setting contents is as follows:

Storage destination of usrconf.cfg

```
<Service platform installation directory>\CC\server\usrconf\ejb\server name
\usrconf.cfg
```

Description format of usrconf.cfg

```
add.class.path=<external jar name>#
```

# Set the name of external jar in full path.

Specification example

```
add.class.path=C:\usrFunc\usrFunc.jar
```

In above-mentioned specification example, storage destination of external jar is specified in "C:\usrFunc \usrFunc.jar". It is referred at the time of executing transformation function jar file.

For details on usrconf.cfg, see "*14.2 usrconf.cfg (Option definition file for Java application)*" in "*Application Server Definition Reference Guide*".

6.9.4 Coding, building and debugging Java programs

Code, build and debug the Java program on the basis of the created Java form file. Use Java development tools such as Eclipse etc for coding, building and debugging Java programs.

The following table describes specifications of the Java program interface.

Table 6–9: Java program interface specifications

Item no.	Interface	Contents
1	Method	Access modifier is `public` and static method (static statement). Cannot overload.
2	Argument	Can have more than 0 arguments. It is input parameter of String type or Object type.
3	Function value	Set function value of String type or NodeList type. Null cannot be returned.
4	Exception	If an exception is thrown, data transformation process fails.

Note the following when creating a Java program:

- If a jar file requiring external reference exists, add the class path in `usrconf.cfg` (operation definition file for java applications).
- The custom function operates by multiple threads. Implement the Java program by making it thread--safe.
- Do not implement a Java program that accesses external resources.

6.9.5 Packaging Java programs

Package the coded, built and debugged Java program in a jar file. Use build tools such as Ant etc for packaging Java programs.

Save the packaged jar file in the following location. If the HCSC server is used by the cluster configuration, save in the respective environments.

```
Cosminexus installation directory\CSC\userlib\customfunc
```

jar file stored in above-mentioned directory is shared in all the HCSC servers on the same machine.

If you specify file path in usrconf.properties (user property file for J2EE server), you can change the storage destination of jar file in HCSC server unit. In this case, file path specified in usrconf.properties is given priority over the above-mentioned directory.

If the jar file is saved in the HCSC server, restart the J2EE server.

For details about how to save the jar file in the HCSC server, see the contents related to saving the jar file used for data transformation in *Cosminexus Service Platform System Setup and Operation Guide*.

6.10 Mapping Conditions

Depending on the combination of the mapping source (transformation-source node or function), mapping destination (transformation-destination node or function), and mapping line types, mapping may not be possible in some cases. This section explains the conditions that enable mapping.

6.10.1 Mapping Targets and Non--Mapping Targets

Some elements (nodes) can be specified as mapping targets, and others cannot. You cannot specify elements that cannot be made mapping targets for the transformation-source node or transformation-destination node. The following table shows elements that cannot be made mapping targets.

Table 6–10: Elements that cannot be made mapping targets

Item	Content
Element (compositor) whose occurrence count is not fixed to 1 and whose name is not specified	The following child elements (compositors) that are descendent elements under a `complexType` element whose occurrence count is not fixed to 1 and whose name is not specified are excluded from mapping: • `Sequence`[#1] • `choice` • `all` • These elements (compositors) appear as #anonymous[#2].
Recursive structure	If a recursive structure is defined, only the recursive--start element is displayed, and the start element is excluded from mapping.
Mixed content	When `true` is specified for the `mixed` attribute, which has a mixed content, the inter--element text is excluded from mapping. Note that the inter--element is not displayed.
Alternate group	An element in which `true` is specified for the `abstract` attribute is excluded from mapping. Furthermore, because the specification of the `SubstitutionGroup` attribute is ignored, an element for which the `SubstitutionGroup` attribute is specified is not displayed in the schema tree viewer.
`any` element and `anyAttribute` element	`any` element and `anyAttribute` element are excluded from mapping.
`union` element or `list` element that is nested in two or more stages	`union` element or `list` element (the attribute of the `union` element or `list` element is further defined) is excluded from mapping.
Simple--content or attribute transformation-destination node for which `fixed` is specified	Simple--content or attribute transformation-destination node for which `fixed` is specified is excluded from mapping.

#1

Exclude if only one simple content element exists under the element (compositor).

#2

For details about #anonymous, see *6.2(3) Notes on creating message format definition files*.

6.10.2 Correspondences Between Nodes and Functions That Can Be Mapped

When mapping, you must take into consideration the correspondences among the transformation-source nodes, transformation-destination nodes, and functions that can be mapped. This section describes the mapping of mapping possible transformation source nodes, transformation destination nodes and functions.

Table 6–11: Mapping from a transformation-source node to a transformation-destination node

Mapping source	Mapping destination		
Transformation-source node	Transformation-destination node		
	Simple content#1#2	Complex content#1	Attribute#1#3
Simple content#1#2	Y	--	Y
Complex content#1	--	--	--
Attribute#1#3	Y	--	Y

Legend:

Y: Can be mapped.

--: Cannot be mapped.

#1

For details about the simple contents, complex contents, and attributes, see the transformation-source schema tree viewer, mapping viewer, andtransformation-destination schema tree viewer in the manual *Cosminexus Service Platform Reference*.

#2

The any element can only be mapped with another any element.

#3

The anyAttribute attribute can only be mapped with another anyAttribute attribute.

Table 6–12: Mapping from a transformation-source node to a function

Mapping source	Mapping destination																				
Transformation-source node	Function																				
	String--based					Number--based				Bits group			Node--based			Control--based		Other			
	Concatenate	Acquire substring	Acquire string length	Check string	Trim node	Convert number format	Perform node operation	Round node	Sum up nodes	Negation operation	Logical operation	Shift operation	Acquire node count	Acquire node name	Check node	Loop node	Choose node	Set constant	Custom	Replace value	Radix conversion#1
Simple content#2#3	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	--	Y	Y	Y
Complex content#2	--	--	--	--	--	--	--	--	--	--	--	--	Y	Y	Y	Y	--	--	--	--	--
Attribute#2 #4	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	--	Y	--	Y	Y	Y

Legend:

Y: Can be mapped.

--: Cannot be mapped.

#1

Input basic numbers are hexadecimal, decimal or binary numbers.

#2

For details about the simple contents, complex contents, and attributes, see the transformation-source schema tree viewer, mapping viewer, and transformation-destination schema tree viewer in the manual *Cosminexus Service Platform Reference*.

#3

You can map any element with any element or Node list type custom function. You can perform mapping by adding Node list type custom function between any element and any element. You cannot perform mapping by adding function between any element and Node list type custom function.

#4

You can perform mapping of anyAttribute attribute with anyAttribute attribute or Node list type custom function. You cannot perform mapping by adding function between anyAttribute attribute and anyAttribute attribute. Also, you cannot perform mapping by adding function between anyAttribute attribute and Node list type custom function.

Table 6–13: Mapping from a function to a transformation-destination node

Mapping source		Mapping destination		
Function		Transformation-destination node		
		Simple content#1#2	Complex content#1	Attribute#1#3
String--based	Concatenate	Y	--	Y
	Acquire substring	Y	--	Y
	Acquire string length	Y	--	Y
	Check string	Y	--	Y
	Trim node	Y	--	Y
Number--based	Convert number format	Y	--	Y
	Perform node operation	Y	--	Y
	Round node	Y	--	Y
	Sum up nodes	Y	--	Y
Bit group	NOT operation	Y	--	Y
	Logical operation	Y	--	Y
	Shift operation	Y	--	Y
Node--based	Acquire node count	Y	--	Y
	Acquire node name	Y	--	Y
	Check node	Y	--	Y
Control--based	Loop node	Y	Y	--
	Choose node	Y	Y	Y
Other	Set constant	Y	Y	Y
	Custom	Y	--	Y
	Replace value	Y	--	Y
	Radix conversion#4	Y	--	Y

Legend:

Y: Can be mapped.

--: Cannot be mapped.

#1

For details about the simple contents, complex contents, and attributes, see the transformation-source schema tree viewer, mapping viewer, and transformation-destination schema tree viewer in the manual *Cosminexus Service Platform Reference*.

#2

You can map any element with any element or Node list type custom function. You can perform mapping by adding Node list type custom function between any element and any element. You cannot perform mapping by adding function between any element and Node list type custom function.

#3

The anyAttribute attribute can only be mapped with another anyAttribute attribute. 2 anyAttribute attributes with a function entered cannot be mapped.

#4

Output basic number is hexadecimal, decimal or binary number.

Table 6–14: Mapping a function to another function (mapping destination is character string group, numeric value group and bit group)

Mapping source$^{\#1}$		Mapping destination											
Function		Function											
		String--based					Number--based				Bit group		
		Concatenate	Acquire substring	Acquire string length	Check string	Trim node	Convert number format	Perform node operation	Round node	Sum up nodes	Negation operation	Logical operation	Shift operation
String--based	Concatenate	Y	Y	Y	Y	Y	Y	Y	Y	--	Y	Y	Y
	Acquire substring	Y	Y	Y	Y	Y	Y	Y	Y	--	Y	Y	Y
	Acquire string length	Y	Y	Y	Y	Y	Y	Y	Y	--	--	--	--
	Check string	Y	Y	Y	Y	Y	Y	Y	Y	--	--	--	--
	Trim node	Y	Y	Y	Y	Y	Y	Y	Y	--	Y	Y	Y
Number--based	Convert number format	Y	Y	Y	Y	Y	Y	Y	Y	--	--	--	--
	Perform node operation	Y	Y	Y	Y	Y	Y	Y	Y	--	--	--	--
	Round node	Y	Y	Y	Y	Y	Y	Y	Y	--	--	--	--
	Sum up nodes	Y	Y	Y	Y	Y	Y	Y	Y	--	--	--	--
Bit group	NOT operation	Y	Y	Y	Y	Y	--	--	--	--	Y	Y	Y
	Logical operation	Y	Y	Y	Y	Y	--	--	--	--	Y	Y	Y
	Shift operation	Y	Y	Y	Y	Y	--	--	--	--	Y	Y	Y
Node--based	Acquire node count	Y	Y	Y	Y	Y	Y	Y	Y	--	--	--	--
	Acquire node name	Y	Y	Y	Y	Y	Y	Y	Y	--	--	--	--
	Check node	Y	Y	Y	Y	Y	Y	Y	Y	--	--	--	--
Control--based	Loop node	--	--	--	--	--	--	--	--	--	--	--	--
	Choose node	--	--	--	--	--	--	--	--	--	--	--	--
Other	Replace value	Y	Y	Y	Y	Y	Y	Y	Y	--	Y	Y	Y

Mapping source#1		Mapping destination											
Function		Function											
		String--based					Number--based				Bit group		
		Concatenate	Acquire substring	Acquire string length	Check string	Trim node	Convert number format	Perform node operation	Round node	Sum up nodes	Negation operation	Logical operation	Shift operation
Other	Radix conversion (output basic number: binary number)	Y	Y	Y	Y	Y	--	--	--	--	--	--	--
	Radix conversion (output basic number: decimal)	Y	Y	Y	Y	Y	Y	Y	Y	--	--	--	--
	Radix conversion (output basic number: hexadecimal)	Y	Y	Y	Y	Y	--	--	--	--	Y	Y	Y
	Custom (character string type)	Y	Y	Y	Y	Y	Y	Y	Y	--	Y	Y	Y
	Custom (node list type)	--	--	--	--	--	--	--	--	--	--	--	--
	Set constant	Y	Y	Y	Y	Y	Y	Y	Y	--	Y#2	Y#2	Y#2

Legend:

Y: Can be mapped.

--: Cannot be mapped.

#1

If you specify even a single loop node function as the mapping source function, specify the loop node function for all other mapping sources.

If you specify an item other than a loop node function for the mapping source function, specify an item other than a loop node function for all other mapping sources.

#2

In case of type other than character string type, error occurs in verification.

Table 6–15: Mapping a function to another function (mapping destination is node group, control group and others)

Mapping source		Mapping destination											
Function		Function											
		Node--based			Control--based		Other						
		Acquire node count	Acquire node name	Check node	Loop node#2	Choose node#3	Replace value	Radix conversion (Input basic number :binary number)	Radix conversion (input basic number :decimal)	Radix conversion (input basic number :hexadecimal)	Custom (character string type)	Custom (node list type)	Set constant#4
String--	Concatenate	--	--	--	--	Y	Y	Y	Y	Y	Y	Y	--

Mapping source		Mapping destination											
Function		Function											
		Node--based			Control--based		Other						
		Acquire node count	Acquire node name	Check node	Loop node#2	Choose node#3	Replace value	Radix conversion (Input basic number :binary number)	Radix conversion (input basic number :decimal)	Radix conversion (input basic number :hexadecimal)	Custom (character string type)	Custom (node list type)	Set constant#4
based	Acquire substring	--	--	--	--	Y	Y	Y	Y	Y	Y	Y	--
	Acquire string length	--	--	--	--	Y	Y	--	Y	--	Y	Y	--
	Check string	--	--	--	--	Y	Y	--	--	--	Y	Y	--
	Trim node	--	--	--	--	Y	Y	Y	Y	Y	Y	Y	--
Number--based	Convert number format	--	--	--	--	Y	Y	--	--	--	Y	Y	--
	Perform node operation	--	--	--	--	Y	Y	--	Y	--	Y	Y	--
	Round node	--	--	--	--	Y	Y	--	Y	--	Y	Y	--
	Sum up nodes	--	--	--	--	Y	Y	--	Y	--	Y	Y	--
Bit group	NOT operation	--	--	--	--	Y	Y	--	--	Y	Y	Y	--
	Logical operation	--	--	--	--	Y	Y	--	--	Y	Y	Y	--
	Shift operation	--	--	--	--	Y	Y	--	--	Y	Y	Y	--
Node--based	Acquire node count	--	--	--	--	Y	Y	--	Y	--	Y	Y	--
	Acquire node name	--	--	--	--	Y	Y	--	--	--	Y	Y	--
	Check node	--	--	--	--	Y	Y	--	--	--	Y	Y	--
Control--based	Loop node	--	--	--	--	Y	--	--	--	--	--	--	--
	Choose node	--	--	--	--	--	--	--	--	--	--	--	--
Other	Replace value	--	--	--	--	Y	Y	Y	Y	Y	Y	Y	--
	Radix conversion (output basic number: binary number)	--	--	--	--	Y	Y	Y	--	--	Y	Y	--
	Radix conversion (output basic number: decimal)	--	--	--	--	Y	Y	--	Y	--	Y	Y	--
	Radix conversion (output basic number: hexadecimal)	--	--	--	--	Y	Y	--	--	Y	Y	Y	--

Mapping source		Mapping destination											
		Function											
		Node--based			Control--based		Other						
Function		Acquire node count	Acquire node name	Check node	Loop node#2	Choose node#3	Replace value	Radix conversion (Input basic number :binary number)	Radix conversion (input basic number :decimal)	Radix conversion (input basic number :hexadecimal)	Custom (character string type)	Custom (node list type)	Set constant#4
Other	Custom (character string type)	--	--	--	--	Y	Y	Y	Y	Y	Y	Y	--
	Custom (node list type)	--	--	--	--	Y#5	--	--	--	--	--	Y	--
	Set constant	--	--	--	--	Y	Y	Y	Y#6	Y#7	Y#6	Y	--

(Legend)

Y: Mapping is possible.

--: Mapping is not possible.

#1

If you specify even 1 Repeat function in mapping source function, Specify Repeat function even for all other mapping sources.

If you specify function other than Repeat function in mapping source function, specify function other than Repeat function for other mapping sources as well.

#2

There are following restrictions in the Repeat function of output side.

Connection is not possible when input of Repeat function is linkage path itself or ancestor node and grandchild node in that Repeat function.

#3

There are following restrictions in Select function of output side.

When Repeat is already connected in input of Select function, you cannot connect function other than Repeat function in input.

When function other than Repeat function is already connected in input of Select function, you cannot connect Repeat in input.

#4

There are following restrictions to the Set constant function.

When the Set constant function has already been connected in input of transformation destination node, Concatenate function, Perform node operation function, Logical operation function, Select function, Custom function, you cannot connect the equivalent Set constant function in input.

#5

When output destination of Select function is attribute, validation error occurs.

#6

When type is other than character string type, validation error occurs.

#7

When type is other than numeric value type and character string type, validation error occurs.

6.10.3 Number of Mapping Lines That Can Be Connected

The number of mapping lines that can be used to connect transformation-source nodes, transformation-destination nodes, and functions is predetermined.

The following table lists the number of mapping lines that can be connected as outputs.

Table 6–16: Number of mapping lines that can be connected as outputs

Node or function	Number of mapping lines that can be connected
Transformation-source node	Multiple[#1]
All functions	1[#2]

#1

There is no upper limit.

#2

For the set constant function, you can connect as many mapping lines as required.

The following table lists the number of mapping lines that can be connected as inputs.

Table 6–17: Number of mapping lines that can be connected as inputs

Node or function		Number of mapping lines that can be connected
Transformation-destination node		Multiple[#1]
String-based	Concatenate	Multiple[#1]
	Acquire substring	1
	Acquire string length	1
	Check string	1
	Trim node	1
Number-based	Convert number format	1
	Perform node operation	2
	Round node	1
	Sum up nodes	Multiple[#1]
Bit-based	NOT operation	1
	Logical operation	2
	Shift operation	1
Node-based	Acquire node count	1
	Acquire node name	1
	Check node	1
Control-based	Loop node	1
	Choose node	Multiple[#1]
Other	Set constant	0
	Custom	Multiple[#2]
	Replace value	1
	Radix Conversion	1

#1

There is no upper limit.

#2

You can enter the same number as the argument of the corresponding transformation function. You can also enter 0.

6.11 Editing function name directly

You can change the function name defined in mapping definition, to any name by editing directly.

Function name is automatically attached with "function type + serial number" and only number is separated when there are many functions of same time. By changing this function name to any name, you can discriminate only by name, even if there are large numbers of functions having same type.

6.11.1 Method of editing function name

You can change function name with two methods such as method of directly editing the function scheduled to the Mapping viewer and a method of using dialog. Procedure in case of using each method is as follows:

(1) When directly editing function scheduled to the Mapping viewer

Procedure for directly selecting function and editing function name in the Mapping viewer is as follows:

1. Select function for editing function name in the Mapping viewer.
2. Left click the selected function or press **F2** key.
   Function name changes to configurable status.
3. Enter a function name.
   Specify a function name in NCName. You can specify characters in the range of 1~64.

(2) When using a dialog

Procedure for editing the function name by using the Setting dialog of each function is as follows:

1. Display a dialog for setting the dialog by either of the following method:
   - Right click the function and select **Settings**.
   - Double click the function.

   Dialog for setting the function is displayed.
   For details on dialog of each function, see "*1.6 Dialogs Related to Data Transformation*" in "*Service Platform Reference Guide*".
2. Enter any function name.
   Specify function name with NCName. You can specify characters in the range of 1~64 characters.
3. Click **OK** button.

6.11.2 Displaying function name after edition

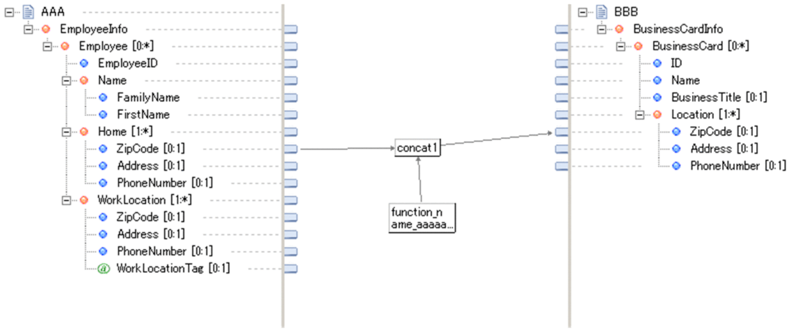
(1) Function name displayed in the Mapping viewer of the Data transformation definition screen

When you edit the function name, it is displayed as follows:

- When function name has 1~10 characters
  Entire function name is displayed.
- When function name has 11~20 characters
  Entire function is displayed by entering linefeed after 10 characters.
- When function name has 21~64 characters
  Function name is displayed till first 19 characters by entering linefeed after 10 characters and 20 characters onwards are abbreviated as "...".

Following figure shows the display example of function name after edition.

Figure 6–47: FigureDisplay example of function name after edition (in case of the Data transformation definition screen)

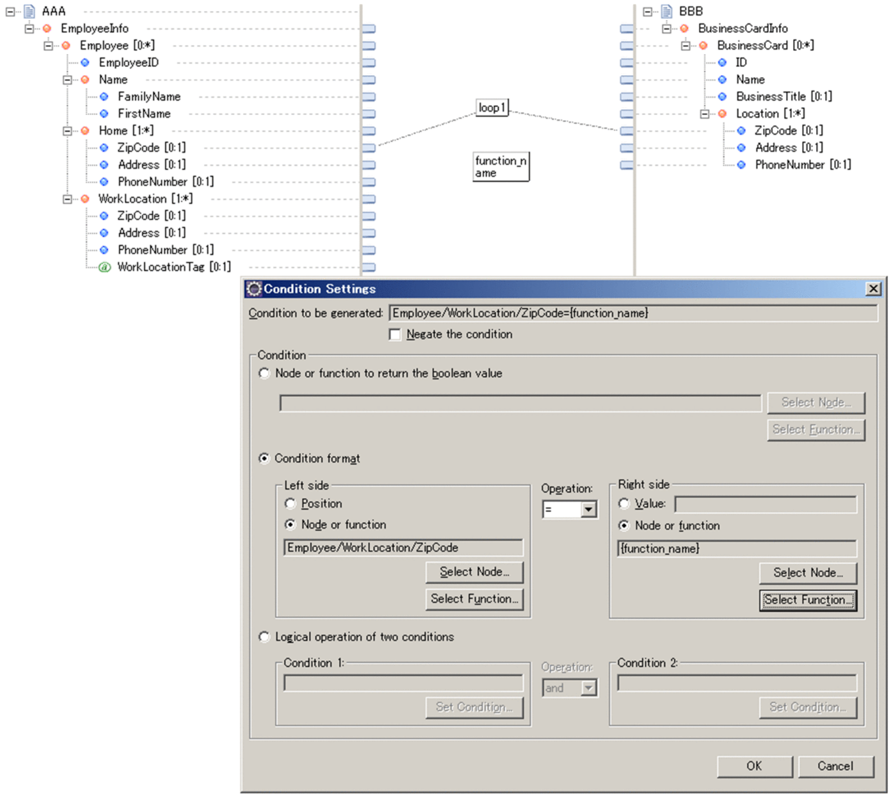


(2) Function name displayed in the dialog

When you edit function name, it is displayed in node conditions of Condition settings dialog, by enclosing in curly brackets ({ }).

Following figure shows the display example of function name after edition:

Figure 6–48: FigureDisplay example of function name after edition (in case of Conditions setting dialog)



Condition to be generated: Employee/WorkLocation/ZipCode={function_name}

□ Negate the condition

Condition

○ Node or function to return the boolean value

● Condition format

Left side: ○ Position ● Node or function — Employee/WorkLocation/ZipCode

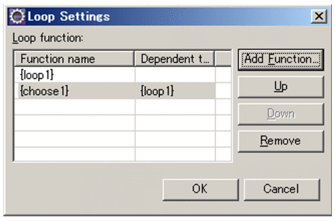
Operation: =

Right side: ○ Value: ● Node or function — {function_name}

○ Logical operation of two conditions

Condition 1: | Operation: and | Condition 2:

Figure 6–49: FigureDisplay example of function name after edition (in case of Repeat settings dialog)



Loop function:

Function name	Dependent t...
{loop1}	
{choose1}	{loop1}

6.12 Importing mapping definition using Excel

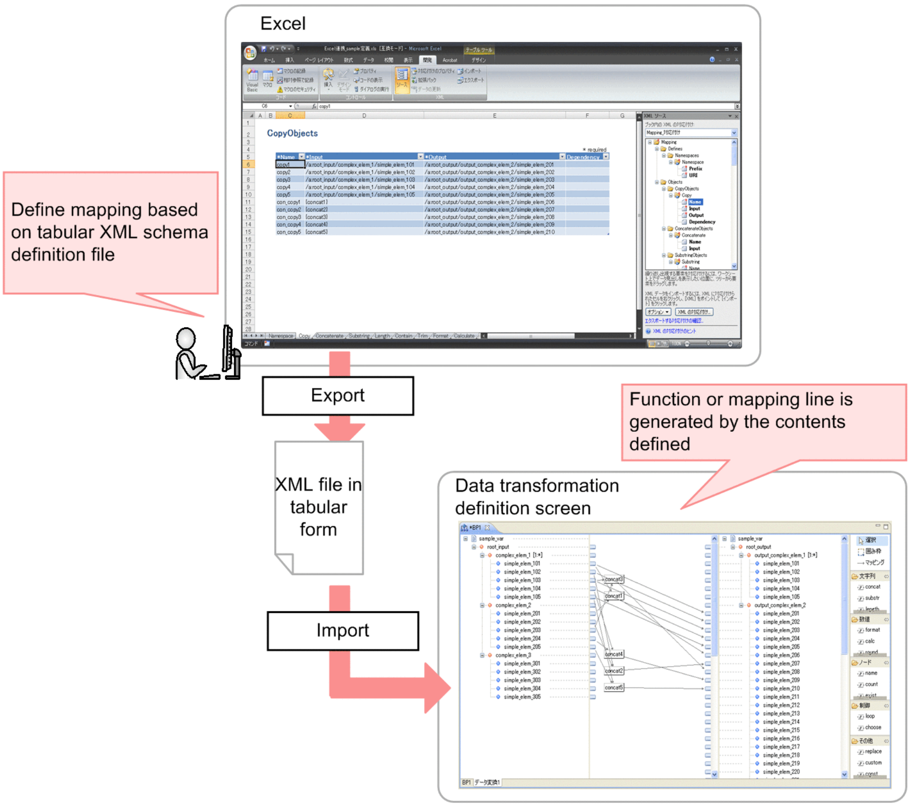
In Service Platform, you can import the file in which combination (mapping) of transformation source and transformation destination is defined in Excel, to the development environment.

When you want to perform similar mapping in multiple data transformation definitions, you can effectively use one mapping definition by using mapping definition created in Excel. Also, as you can check the mapping definition contents in Excel, you can reduce the mistakes at the time of definition in case of multiple transformation sources and transformation destinations and thus improve the development effectiveness.

To import mapping definition, export the contents mapped by defining in Excel, based on table format XML schema definition file. After that, import the data transformation definition file of development environment. In Data transformation definition screen, function or mapping line is generated according to the imported table format XML file.

Following figure shows the flow of importing the mapping definition.

Figure 6–50: FigureFlow of importing the mapping definition


Excel
Define mapping based on tabular XML schema definition file
Export
XML file in tabular form
Import
Function or mapping line is generated by the contents defined
Data transformation definition screen

You can directly create XML file according to table format XML schema definition file and import to the Data ran formation definition file.

In Service Platform, Excel of the following version is recommended.

- Microsoft Office Excel 2007

Even when you create file by using any other tool, you can import if table format XML file is appropriate for table format XML schema definition file.

This section further describes procedure for creating table format XML file and for importing the mapping definition.

6.12.1 Creating table format XML file

Method for creating table format XML file has two types such as type of using a template file in which each element of table format XML schema definition file has already been mapped to Excel and a type of mapping element individually based on the table format XML schema definition file.

Reference note

Following has already been set up in the template file.

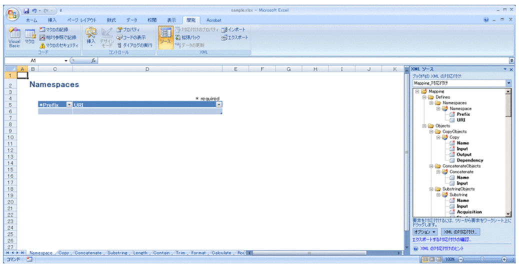
- Element has been mapped by diving sheets for each object.
- In mandatory items, asterisk (*) are added to beginning of item name. For mandatory items, you must set up value in all the lines to be defined. In case of blank row, error occurs.
- If value of item is restricted to fixed value (items for which value can be entered only as "y" or "n"), data input rules are set and you can select value from the list.

Creation procedure in respective cases is as follows:

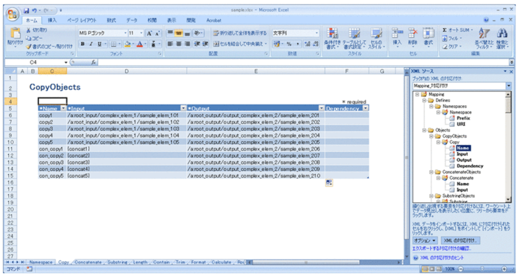
(1) When using a template file

1. Copy a template file (dt_import_template.xlsx) and save with any name.

   The location of storing the template file (dt_import_template.xlsx) is "<Service Platform installation directory>\CSCTE\resources\dt_import".
2. Start Microsoft Office Excel 2007 and open the copied file.



3. Define the function or mapping line in the cell in which mapping of XML element has been performed.


CopyObjects

* required

*Name	*Input	*Output	Dependency
copy1	/a:root_input/complex_elem_1/sample_elem_101	/a:root_output/output_complex_elem_2/sample_elem_201	
copy2	/a:root_input/complex_elem_1/sample_elem_102	/a:root_output/output_complex_elem_2/sample_elem_202	
copy3	/a:root_input/complex_elem_1/sample_elem_103	/a:root_output/output_complex_elem_2/sample_elem_203	
copy4	/a:root_input/complex_elem_1/sample_elem_104	/a:root_output/output_complex_elem_2/sample_elem_204	
copy5	/a:root_input/complex_elem_1/sample_elem_105	/a:root_output/output_complex_elem_2/sample_elem_205	
con_copy1	[concat1]	/a:root_output/output_complex_elem_2/sample_elem_206	
con_copy2	[concat2]	/a:root_output/output_complex_elem_2/sample_elem_207	
con_copy3	[concat3]	/a:root_output/output_complex_elem_2/sample_elem_208	
con_copy4	[concat4]	/a:root_output/output_complex_elem_2/sample_elem_209	
con_copy5	[concat5]	/a:root_output/output_complex_elem_2/sample_elem_210	

   For details on the definition method, see "*6.12.2 Setting up the mapping definition*".
4. Click **Export** button of **Development** tab and save the file in XML format.

   The definition contents are exported as the table format XML file.

(2) When mapping the element based on table format XML schema definition file

1. Start Microsoft Office Excel 2007.
2. Select **Source** button of **Development** tab.

   XML source work window is displayed.
3. Click **XML mapping...** button of XML source work window.

   XML mapping dialog is displayed.

4. Click **Add** button, to add table format XML schema definition file (dt_import_mapping.xsd).

   Storage destination of table format XML schema definition file (dt_import_mapping.xsd) is "<Service Platform installation directory>\CSCTE\resources\dt_import".

   Schema is displayed in tree format in XML source work window.

5. Drag and drop the element part of "***Objects" or "Namespaces" to the cell. #

   Data of the element part that is Dragged and dropped becomes configurable.

   #

   To map element, divide sheet for each object (such as Copy or Concatenate). If you set up multiple elements in each sheet, you cannot import the mapping definition properly.

6. Define function or mapping line in the cell in which mapping of XML element is performed.

   For details on the definition method, see "*6.12.2 Setting up the mapping definition*".

7. Click **Export** button of **Development** tab and save the file in XML format.

   Definition contents are exported as table format XML file.

6.12.2 Setting up the mapping definition

To define the mapping in Excel, set up values of the transformation source node or transformation destination node to each object, based on table format XML schema definition file.

This section describes items that you can define in input and output values of a mapping definition and method to set up each item.

(1) Items that you can define in input and output value

Set up path or function name of transformation source node in the Data transformation definition screen, in Input of objects such as CopyObjects(mapping to transformation destination node) or ConcatenateObjects (Concatenate function). Set the path of transformation destination node of Data transformation definition screen in Output of CopyObjects (mapping to transformation destination node). You can specify node conditions, when specifying transformation source node in the input value.

Following table describes nodes and functions that you can define in each item of mapping definition and possibility of setting up conditions to transformation source node.

Table 6–18: TableDefinition items of input and output value (node)

Definition items of input and output value		Node that can be set										Possibility of setting conditions to transformation source node
		Transformation source node					Transformation destination node					
Object	Item	Simplecontents	Complexcontexts	Attributes	anyelement	anyAttribute attribute	Simplecontents	Complexcontents	Attribute	anyelement	anyAttribute attribute	
CopyObjects (mapping to transformation destination node)	Input	Y	Y	Y	Y	Y	-	-	-	-	-	Y
	Outp ut	-	-	-	-	-	Y	Y	Y	Y	Y	-

Definition items of input and output value		Node that can be set										Possibility of setting conditions to transformation source node
Object	Item	Transformation source node					Transformation destination node					
		Simplecontents	Complexcontexts	Attributes	anyelement	anyAttribute attribute	Simplecontents	Complexcontents	Attribute	anyelement	anyAttribute attribute	
CopyObjects (mapping to transformation destination node)	Dependency	N	N	N	N	N	-	-	-	-	-	-
ConcatenateObjects (Concatenate function)	Input	Y	N	Y	N	N	-	-	-	-	-	Y
SubstringObjects (Acquire substring function)	Input	Y	N	Y	N	N	-	-	-	-	-	Y
LengthObjects (Acquire string length function)	Input	Y	N	Y	N	N	-	-	-	-	-	Y
ContainObjects (Check string function)	Input	Y	N	Y	N	N	-	-	-	-	-	Y
TrimObjects (Trim node function)	Input	Y	N	Y	N	N	-	-	-	-	-	Y
FormatObjects (Convert number format function)	Input	Y	N	Y	N	N	-	-	-	-	-	Y
CalculateObjects (Perform node operation function)	Input 1	Y	N	Y	N	N	-	-	-	-	-	Y
	Input 2	Y	N	Y	N	N	-	-	-	-	-	Y
RoundObjects (Round node function)	Input	Y	N	Y	N	N	-	-	-	-	-	Y
SumObjects (Sum up nodes function)	Input	Y	N	Y	N	N	-	-	-	-	-	Y
NotObjects (NOT operation function)	Input	Y	N	Y	N	N	-	-	-	-	-	Y
BitOpObjects (Logical operation function)	Input 1	Y	N	Y	N	N	-	-	-	-	-	Y
	Input 2	Y	N	Y	N	N	-	-	-	-	-	Y
ShiftObjects (Shift operation function)	Input	Y	N	Y	N	N	-	-	-	-	-	Y
NameObjects (Acquire node name function)	Input	Y	Y	Y	N	N	-	-	-	-	-	Y
CountObjects (Acquire node count function)	Input	Y	Y	Y	N	N	-	-	-	-	-	Y
ExistObjects (Check node function)	Input	Y	Y	Y	N	N	-	-	-	-	-	Y
LoopObjects (Repeat function)	Input	Y	Y	N	N	N	-	-	-	-	-	Y
	RelationalPath	Y	Y	N	N	N	-	-	-	-	-	Y
	SortKey	Y	Y	Y	N	N	-	-	-	-	-	N

Definition items of input and output value		Node that can be set										Possibility of setting conditions to transformation source node
Object	Item	Transformation source node					Transformation destination node					
		Simplecontents	Complexcontexts	Attributes	anyelement	anyAttribute attribute	Simplecontents	Complexcontents	Attribute	anyelement	anyAttribute attribute	
ChooseObjects (Select function)	Condition	Y	N	Y	N	N	-	-	-	-	-	-
	OutputValue	Y	N	Y	N	N	-	-	-	-	-	Y
ReplaceObjects (Replace value function)	Input	Y	N	Y	N	N	-	-	-	-	-	Y
RadixObjects (Radix conversion function)	Input	Y	N	Y	N	N	-	-	-	-	-	Y
CustomObjects (Set custom function)	Input	Y	Y	Y	Y	Y	-	-	-	-	-	Y
Setting up conditions to transformation source node		Y	N	Y	N	N	-	-	-	-	-	-

(Legend)

Y: Definition is possible.

N: Definition is not possible.

-: Not applicable

Table 6–19: TableDefinition items of input and output value (function)

Definition items of input and output value		Configurable functions																				
Object	Item	Concatenate	Acquiresubstring	Acquirestringlength	Checkstring	Trimnode	Convertnumberformat	Performnodeoperation	Roundnode	Sumupnodes	Negationoperation	Logicaloperation	Shiftoperation	Acquirenodename	Acquirenodecount	Checknode	Repeat	Select	Replacevalue	Transformbasicnumber	Custom	Constant
CopyObjects (mapping to transformation destination node)	Input	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y
	Output	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
	Dependency	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	Y	N	N	N	N	N
Concatenate Objects (Concatenate function)	Input	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	N	Y	Y	Y	Y
Substring Objects (Acquire substring function)	Input	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	N	Y	Y	Y	Y
Length Objects (Acquire string length function)	Input	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	N	Y	Y	Y	Y
Contain Objects (Check string function)	Input	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	N	Y	Y	Y	Y
TrimObjects (Trim node function)	Input	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	N	Y	Y	Y	Y
Format Objects (Convert number format function)	Input	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	N	N	Y	Y	Y	N	N	Y	Y	Y	Y
Calculate	Input1	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	N	N	Y	Y	Y	N	N	Y	Y	Y	Y

Definition items of input and output value		Configurable functions																				
Object	Item	Concatenate	Acquire substring	Acquire string length	Check string	Trim node	Convert number format	Perform node operation	Round node	Sum up nodes	Negation operation	Logical operation	Shift operation	Acquire node name	Acquire node count	Check node	Repeat	Select	Replace value	Transform basic number	Custom	Constant
Objects (Perform node operation function)	Input2	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	N	N	Y	Y	Y	N	N	Y	Y	Y	Y
Round Objects (Round Node function)	Input	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	N	N	Y	Y	Y	N	N	Y	Y	Y	Y
SumObjects (Sum up Nodes Function)	Input	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
NotObjects (NOT operation function)	Input	Y	Y	N	N	Y	N	N	N	N	Y	Y	Y	N	N	N	N	N	Y	Y	Y	Y
BitOp Objects (Logical operation function)	Input1	Y	Y	N	N	Y	N	N	N	N	Y	Y	Y	N	N	N	N	N	Y	Y	Y	Y
	Input2	Y	Y	N	N	Y	N	N	N	N	Y	Y	Y	N	N	N	N	N	Y	Y	Y	Y
ShiftObjects (Shift operation function)	Input	Y	Y	N	N	Y	N	N	N	N	Y	Y	Y	N	N	N	N	N	Y	Y	Y	Y
Name Objects (Acquire Node	Input	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N

Definition items of input and output value		Configurable functions																				
Object	Item	Concatenate	Acquiresubstring	Acquirestringlength	Checkstring	Trimnode	Convertnumberformat	Performnodeoperation	Roundnode	Sumnodes	Negationoperation	Logicaloperation	Shiftoperation	Acquirenodename	Acquirenodecount	Checknode	Repeat	Select	Replacevalue	Transformbasicnumber	Custom	Constant
Name function)	Input	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
Count Objects (Acquire node count function)	Input	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
ExistObjects (Check node function)	Input	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
LoopObjects (Repeat function)	Input	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
	Relational Path	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
	SortKey	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N	N
Choose Objects (Select Function)	Condition	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	N	Y	Y	Y	Y
	Output Value	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	Y	Y	Y	Y
Replace Objects (Replace Value function)	Input	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	N	Y	Y	Y	Y
RadixObjects	Input	Y	Y	Y	N	Y	N	Y	Y	Y	Y	Y	Y	N	Y	N	N	N	Y	Y	Y	Y

Definition items of input and output value		Configurable functions																				
Object	Item	Concatenate	Acquire substring	Acquire string length	Check string	Trim node	Convert number format	Perform node operation	Round node	Sum up nodes	Negation operation	Logical operation	Shift operation	Acquire node name	Acquire node count	Check node	Repeat	Select	Replace value	Transform basic number	Custom	Constant
(Transform basic number function)	Input	Y	Y	Y	N	Y	N	Y	Y	Y	Y	Y	Y	N	Y	N	N	N	Y	Y	Y	Y
Custom Objects (Custom function)	Input	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	N	Y	Y	Y	Y
Setting up conditions to transformation source node		Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	N	N	Y	Y	Y	Y

(Legend)

Y: Definition is possible.

N: Definition is not possible.

-: Not applicable

For details on definition items and definition example of each object, see "*6.13 Definition details of table format XML schema definition file*".

(2) Syntax rules

Following table describes BNF mapped to input and output value of each object.

Table 6–20: TableBNF mapped to input and output value of each object

Object/item	Mapped BNF
Copy/Input	Input path or function with conditions
Concatenate/Input	
Substring/Input	
Length/Input	

Object/item	Mapped BNF
Contain/Input	Input path or function with conditions
Trim/Input	
Format/Input	
Calculate/Input1	
Calculate/Input2	
Round/Input	
Not/Input	
BitOp/Input1	
BitOp/Input2	
Shift/Input	
Choose/OutputValue	
Radix/Input	
Custom/Input	
Sum/Input	Input path with conditions
Name/Input	
Count/Input	
Exist/Input	
Loop/Input	
Loop/RelationalPath	
Copy/Output	Output path
Copy/Dependency	Function
Loop/SortKey	Input path
Choose/Condition	Conditions#

#

You cannot use the item relative path.

Following table describes the syntax rules of these BNF.

Table 6–21: TableSyntax rules of BNF

BNF	Syntax rules of BNF
Input path	::= [Variable] ('/' Node)+
Path with conditions	::= [Variable] ('/' Step)+
Function	::= '{' NCName '}'
Output path	::= [Variable] ('/' Node)+
Conditions	::= Basic method | ('not(' Condition ')') | '(' Condition ')'
Variable	::= '$' NCName
Steps	::= Node ['[' Condition ']']

BNF	Syntax rules of BNF
Node	::= (['@'] [Namespace prefix ':'] NCName) ｜ '#any' ｜ '#anyAttribute'
Namespace prefix	::= NCName
Basic format	::= Input path | Input relative path | Function | Condition expression | Complex condition expression
Condition expression	::= ('position()' | Input path | Input relative path | Function) Operator (Value | Function | Input path | Input relative path)
Input relative path	::= '.' | Node ('/' node)*
Complex conditions expression	::= or expression
or expression	::= and expression ['or' and expression]
and expression	::= Condition ['and' condition]
Operator	::= '=' | '!=' | '>' | '<' | '>=' | '<='
Value#	::= "'" [^']+ "'"

#

You can specify a value with minimum 1 and maximum 1,024 characters. However, actual reference expression is handled as one character.

(3) Setting up the transformation source node

When setting up the transformation source node in an object, specify the path of transformation source node in Input, with absolute path. Specification in relative path is not supported. For details on how to set up the node path, see "*(7) Setting up the node path*".

Also, you can specify the node conditions in transformation source node depending on the definition items. For details on how to specify node conditions, see "*(6) Setting up node conditions*".

(4) Setting up the transformation destination node

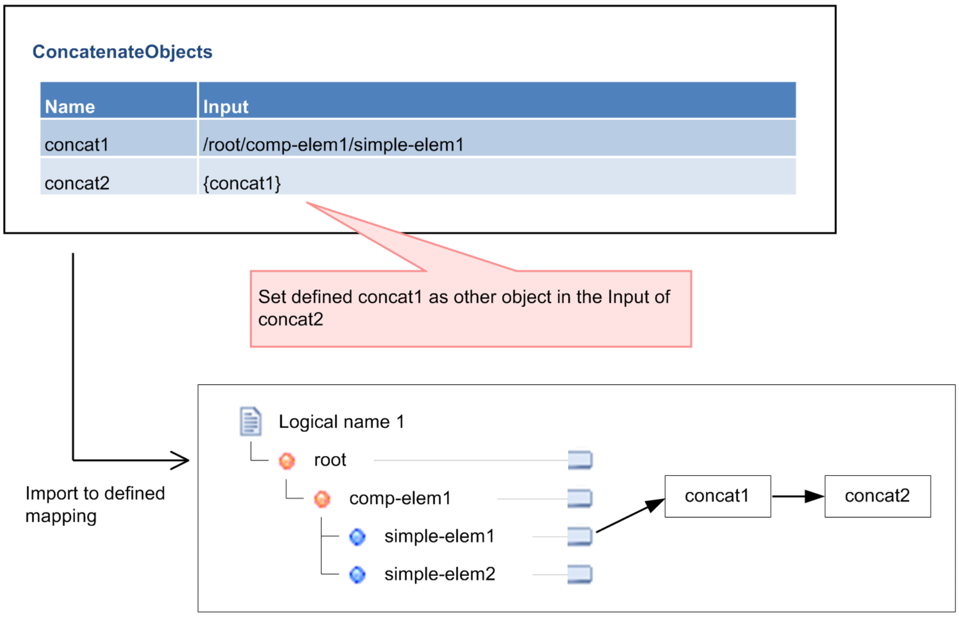
To set up the transformation destination node, specify path of transformation destination node in Output of CopyObjects (mapping to transformation destination node). For details on setting up the node path, see "*(7) Setting up the node path*".

(5) Setting up a function

In function, specify a function name defined in other objects. Specify function name in the part enclosed in curly brackets ({ }).

Following figure shows the definition example.

Figure 6–51: FigureDefinition example (setting up a function)



ConcatenateObjects

Name	Input
concat1	/root/comp-elem1/simple-elem1
concat2	{concat1}

Set defined concat1 as other object in the Input of concat2

Import to defined mapping

(6) Setting up node conditions

When you set up transformation source node in the items in which node conditions can be specified, you can specify node conditions in each element of transformation source node.

For the types of transformation source node or function for which you can specify node conditions, see *Table6-18* and *Table 6-19*.

Specify node conditions in the part enclosed in square bracket ([ ]) at the end of each item.

Specify node conditions by either of the following methods:

Table 6–22: Table Methods for specifying node conditions

#	Method of setting node conditions	Outline
1	Node o function that returns boolean value	When you specify the conversion source node or function, returns boolean value depending on their existence.
2	Condition expression	Returns boolean value for the conditions specified in the format of "<left edge><operation><right edge>" condition. Specify transformation source node or function in the left edge and right edge.
3	Logical total or logical sum of 2 conditions	"<Condition 1><Operation><Condition 2>" Returns boolean value for the conditions specified by combining the conditions in above-mentioned #1 and #2 with and or or.

Specify path of transformation source node of node conditions setting, by either of absolute path, or relative path from condition setting target node. For details on node path specification, see "*(7) Setting up the node path*".

However, when you use the grandchild node of condition settings target node, in conditions, it is referred as relative path after importing to data transformation definition screen, regardless of relative path specification.

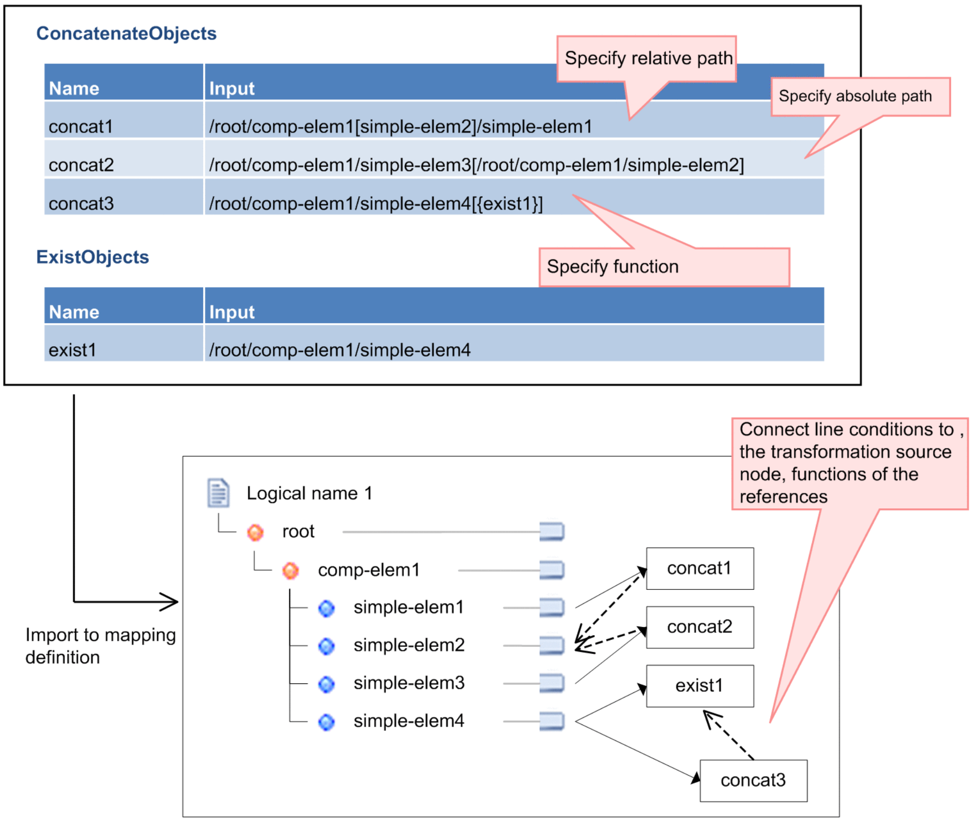
For details on how to specify function, see "*(5) Setting up a function*".

(a) Node or function that returns boolean value

When you specify transformation source node or function, boolean value is returned depending on their existence.

Following figure shows the definition example of setting existence of node or function in the condition.

Figure 6–52: FigureDefinition example (node condition settings 1)



ConcatenateObjects

Name	Input
concat1	/root/comp-elem1[simple-elem2]/simple-elem1
concat2	/root/comp-elem1/simple-elem3[/root/comp-elem1/simple-elem2]
concat3	/root/comp-elem1/simple-elem4[{exist1}]

ExistObjects

Name	Input
exist1	/root/comp-elem1/simple-elem4

(b) Condition expression

Boolean value is returned for the condition expression specified in "<Left edge><Operation><right edge>" format. Following table describes the values that you can specify in left edge, operation and right edge:

Table 6–23: TableValues that you can specify in each item of condition expression

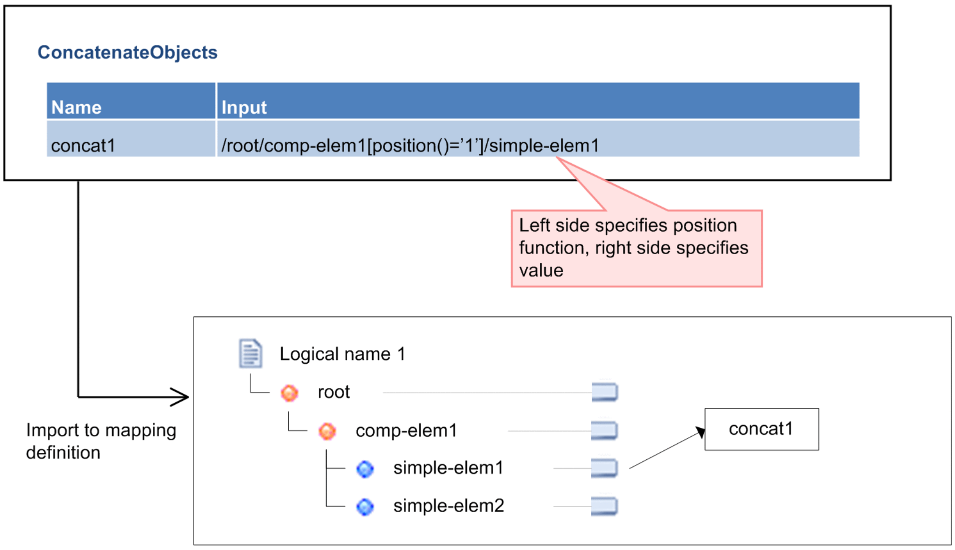
Item of condition expression	Value that you can specify
Left edge	You can specify either of the following. • Position function to be specified in "position()" • Transformation source node that you can specify in condition settings You can specify transformation source node in relative path as well.# • Other functions that you can specify in condition settings
Operation	You can specify either of "=", "!=", ">", "<", ">=", "<=".
Right edge	You can specify either of the following: • Value enclosed in apostrophe (') (you cannot specify space). When value includes apostrophe (') or ampersand (&), use with entity reference expressions such as apostrophe as "&apos;" and ampersand as "&amp;", as shown in the following example:. (Example) In case of "a & 'b'", define as "a &amp; &apos;b&apos;". • Transformation source node that you can specify in condition settings You can specify transformation source node even in relative path# • Other functions that can be specified in conditions settings

Note#

For details on how to specify a relative path, see "*(7)(c) Specifying a relative path*".

Following figure shows the definition example when setting the condition expression:

Figure 6–53: FigureDefinition example (Node condition settings 2)



ConcatenateObjects

Name	Input
concat1	/root/comp-elem1[position()=’1’]/simple-elem1

Left side specifies position function, right side specifies value

Import to mapping definition

(c) Logical total and logical sum of two conditions

Boolean value is returned in logical total or logical sum, in which two conditions are combined with and or or operation, in the "<Condition1><Operation><Condition2>" format. Specify either of the following in <Condition>:

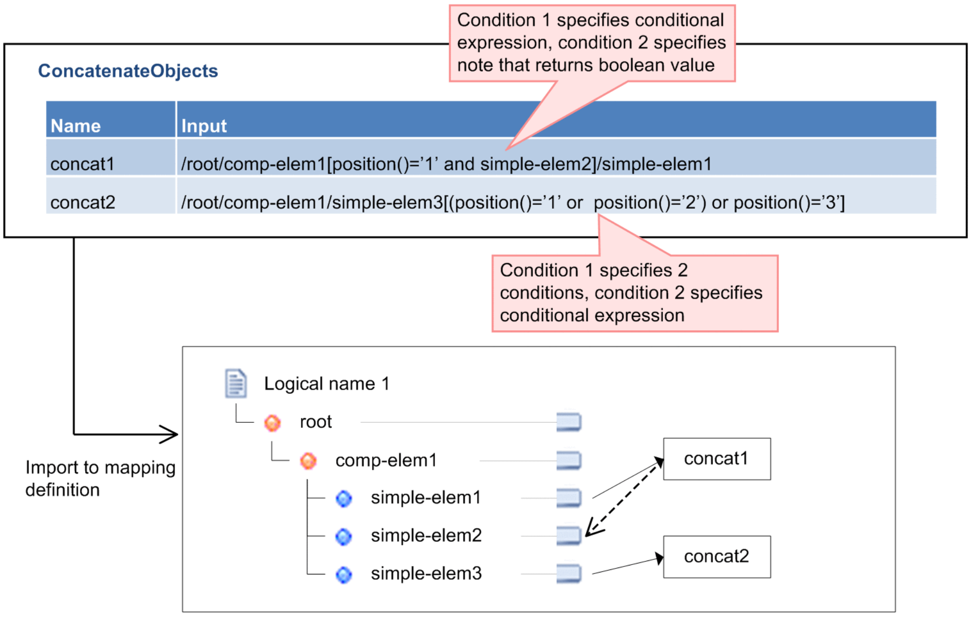
- Node or function that returns boolean value
- Condition expression
- Logical total or logical sum of two conditions

If condition has multiple "logical total or logical sum of two conditions", logical total is given priority. However, if you enclose leading and trailing part of "logical product and logical sum of two conditions" in bracket, that logical expression is given priority.

Also, you can deny these conditions. For details, see "*(d) Denying conditions*".

Following figure shows the definition example when logical product or logical sum of two conditions is set in the condition:

Figure 6–54: FigureDefinition example (node condition settings 3)



ConcatenateObjects

Name	Input
concat1	/root/comp-elem1[position()='1' and simple-elem2]/simple-elem1
concat2	/root/comp-elem1/simple-elem3[(position()='1' or position()='2') or position()='3']

Condition 1 specifies conditional expression, condition 2 specifies note that returns boolean value

Condition 1 specifies 2 conditions, condition 2 specifies conditional expression

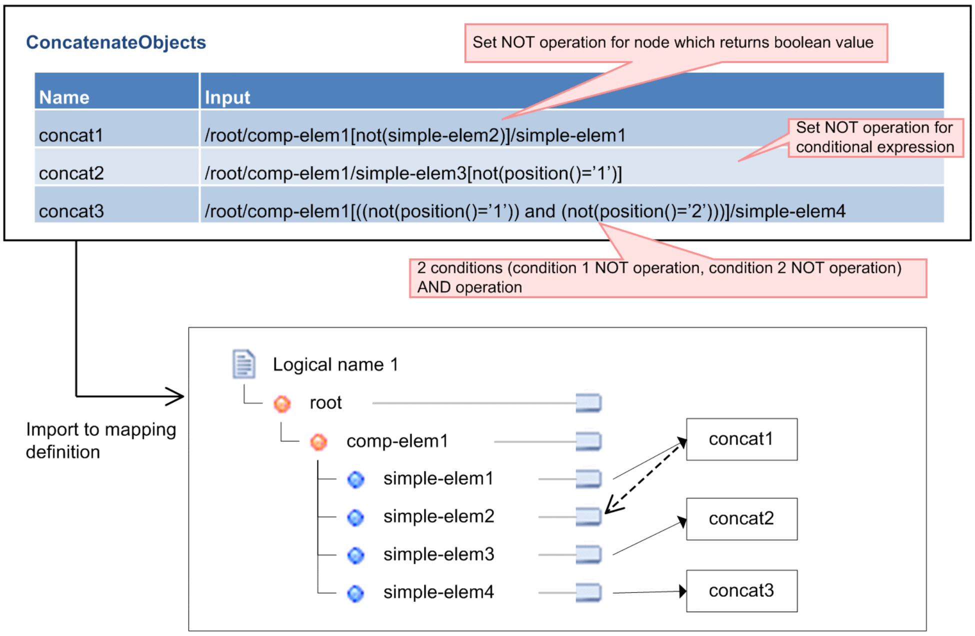
Import to mapping definition

(d) Denying conditions

You can deny conditions in the format of "not (<condition>)".

Following figure shows the definition example when denying the condition:

Figure 6–55: FigureDefinition example (node condition settings 4)



ConcatenateObjects

Name	Input
concat1	/root/comp-elem1[not(simple-elem2)]/simple-elem1
concat2	/root/comp-elem1/simple-elem3[not(position()='1')]
concat3	/root/comp-elem1[((not(position()='1')) and (not(position()='2')))]/simple-elem4

Set NOT operation for node which returns boolean value

Set NOT operation for conditional expression

2 conditions (condition 1 NOT operation, condition 2 NOT operation) AND operation

Import to mapping definition

(7) Setting up the node path

This section describes method of specifying logical name and namespace in case of specifying the path of the transformation source node or transformation destination node.

Description of the method to specify the relative path of transformation source node in node conditions specification is as follows:

(a) Specifying a logical name

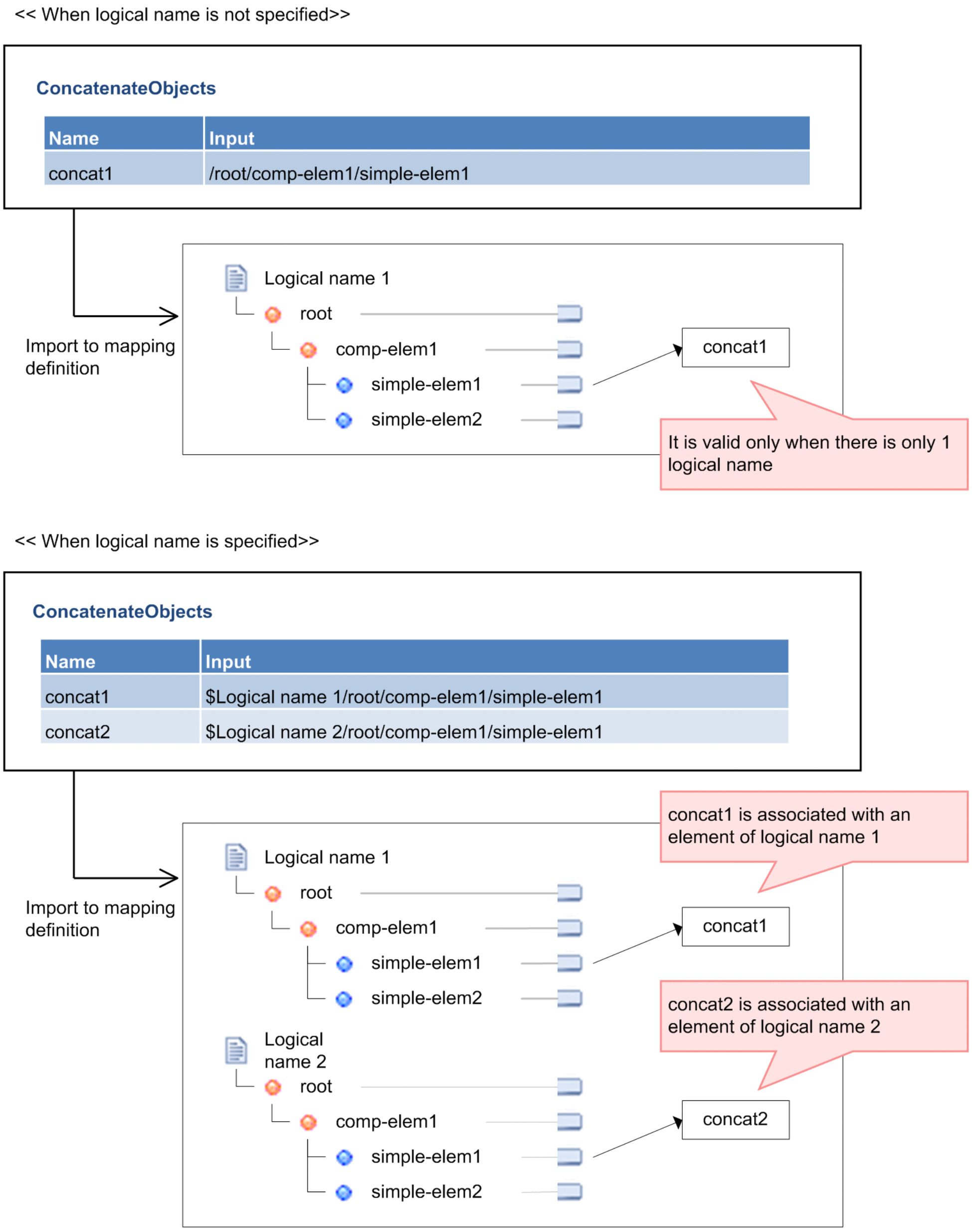
Specify a logical name of transformation source node and transformation destination node, before root element in ""$"+logical name" format.

When there are multiple transformation source nodes of the Data transformation definition screen, you must specify the logical name of transformation source node. If there is not even 1 transformation source node, specifying the logical name is optional.

Specification of the logical name is optional in the transformation destination node.

Following figure shows the definition example:

Figure 6–56: FigureDefinition example (node path settings 1)



<< When logical name is not specified>>

ConcatenateObjects

Name	Input
concat1	/root/comp-elem1/simple-elem1

Import to mapping definition
Logical name 1
root
comp-elem1
simple-elem1
simple-elem2
concat1
It is valid only when there is only 1 logical name

<< When logical name is specified>>

ConcatenateObjects

Name	Input
concat1	$Logical name 1/root/comp-elem1/simple-elem1
concat2	$Logical name 2/root/comp-elem1/simple-elem1

Import to mapping definition
concat1 is associated with an element of logical name 1
Logical name 1
root
comp-elem1
simple-elem1
simple-elem2
concat1
concat2 is associated with an element of logical name 2
Logical name 2
root
comp-elem1
simple-elem1
simple-elem2
concat2

(b) Specifying a namespace

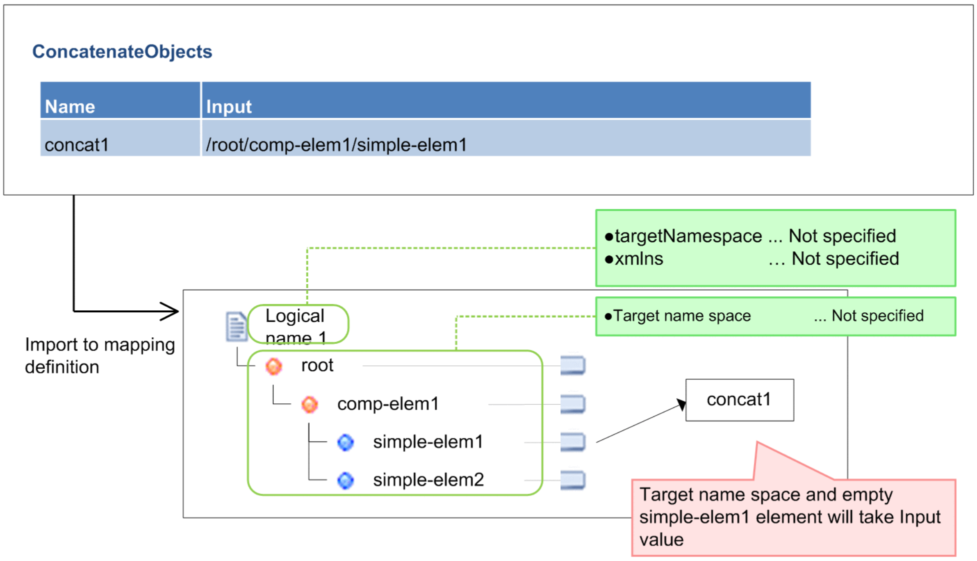
In each element name in the path of transformation source node and transformation destination node, you can use namespace prefix defined in namespace. For details on how to define the namespace prefix, see "*6.13.1 Namespaces (namespace information)*".

Specify namespace prefix in the "Namespace prefix+":"+element name" format.

Following figure shows the definition example:

Figure 6–57: FigureDefinition example (node path settings 2)

<<When name space is not specified>>

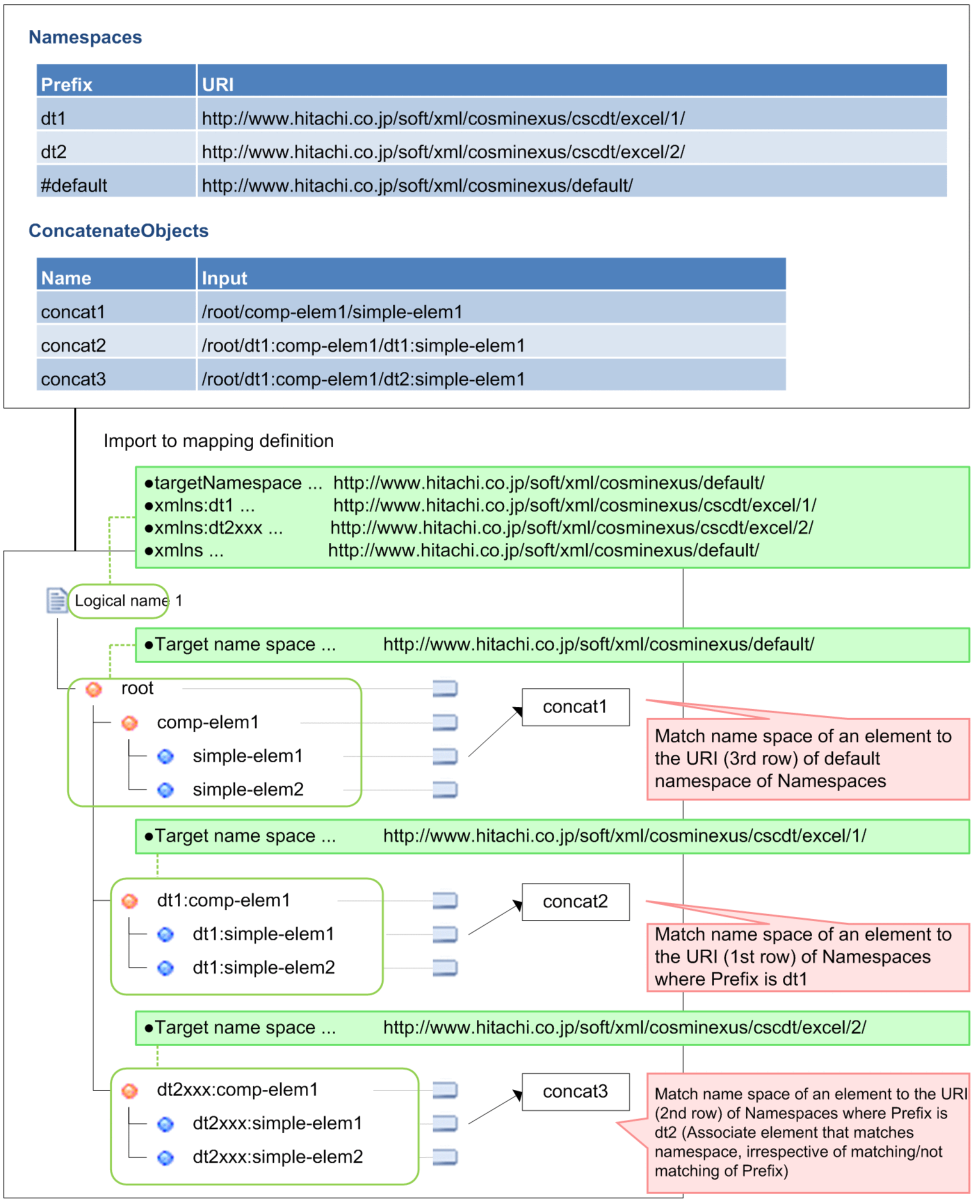


ConcatenateObjects

Name	Input
concat1	/root/comp-elem1/simple-elem1

●targetNamespace ... Not specified
●xmlns … Not specified
●Target name space ... Not specified
Import to mapping definition
Logical name 1
root
comp-elem1
simple-elem1
simple-elem2
concat1
Target name space and empty simple-elem1 element will take Input value

<<When name space is specified>>



Namespaces

Prefix	URI
dt1	http://www.hitachi.co.jp/soft/xml/cosminexus/cscdt/excel/1/
dt2	http://www.hitachi.co.jp/soft/xml/cosminexus/cscdt/excel/2/
#default	http://www.hitachi.co.jp/soft/xml/cosminexus/default/

ConcatenateObjects

Name	Input
concat1	/root/comp-elem1/simple-elem1
concat2	/root/dt1:comp-elem1/dt1:simple-elem1
concat3	/root/dt1:comp-elem1/dt2:simple-elem1

Import to mapping definition

●targetNamespace ... http://www.hitachi.co.jp/soft/xml/cosminexus/default/
●xmlns:dt1 ... http://www.hitachi.co.jp/soft/xml/cosminexus/cscdt/excel/1/
●xmlns:dt2xxx ... http://www.hitachi.co.jp/soft/xml/cosminexus/cscdt/excel/2/
●xmlns ... http://www.hitachi.co.jp/soft/xml/cosminexus/default/

Logical name 1
●Target name space ... http://www.hitachi.co.jp/soft/xml/cosminexus/default/
root
comp-elem1
simple-elem1
simple-elem2
concat1
●Target name space ... http://www.hitachi.co.jp/soft/xml/cosminexus/cscdt/excel/1/
dt1:comp-elem1
dt1:simple-elem1
dt1:simple-elem2
concat2
●Target name space ... http://www.hitachi.co.jp/soft/xml/cosminexus/cscdt/excel/2/
dt2xxx:comp-elem1
dt2xxx:simple-elem1
dt2xxx:simple-elem2
concat3

Match name space of an element to the URI (3rd row) of default namespace of Namespaces

Match name space of an element to the URI (1st row) of Namespaces where Prefix is dt1

Match name space of an element to the URI (2nd row) of Namespaces where Prefix is dt2 (Associate element that matches namespace, irrespective of matching/not matching of Prefix)

(c) Specifying a relative path

When specifying the node conditions, you can specify path of transformation source node in relative path from conditions setting target node. You can specify relative path for the nodes described in the following table. You cannot use any other description method (".." or "//").

Table 6–24: TableMethod of specifying a relative path

Node to be specified in relative path	Specification method
Condition settings target node	Specify period (.). Specification example When path of transformation source node is "/root/comp-elem1/simple-elem1" and condition setting target is "simple-elem1", specify "/root/comp-elem1/simple-elem1[.]" to set existence of "simple-elem1" as condition.

Node to be specified in relative path	Specification method
Grandchild node of condition settings target node	Specify path from child node, of the condition settings target node. Specification example When path of transformation source node is "/root/comp-elem1/simple-elem1" and condition setting target is "comp-elem1", specify "/root/comp-elem1[simple-elem1]/simple-elem1", to set existence of "simple-elem1" as conditions. When condition setting target is "root", specify "/root[comp-elem1/simple-elem1]/comp-elem1/simple-elem1" to set existence of "simple-elem1" as conditions.

6.12.3 Importing mapping definition

Procedure for importing the table format XML file to the Data transformation definition screen is as follows:

1. Select transformation source schema as well as transformation destination schema and start the Data transformation definition screen.
2. Right click on the Mapping viewer of the Data transformation definition screen and select **Import mapping definition**.
   Import mapping definition screen is displayed.
3. Click **Browse** button.
   Specify the table format XM file dialog is displayed.
4. Select the XML file to be imported and click **Open** button.
   The selected XML file is set and Specify the table format XML file dialog closes.
5. Click **Finish** button.
   If function or mapping line has already been defined in the Data transformation definition screen, confirmation message is displayed. If no problem is found, click **Yes** button.
   Function and mapping line defined in Excel are reflected in the Data transformation definition screen.

6.12.4 Points to be considered when using mapping definition using Excel

Points to be considered when defining the mapping in Excel are as follows:

- Input value might be replaced with another value, due to the Auto format function of Excel. Therefore, it is recommended to set the display format of **Cell format settings** to **Character string**.
- Value that you enter in Excel is case sensitive.
- If you do a definition that cannot be set on the Data transformation definition screen, such as referring to the same function from multiple input values, error occurs during import.
- Default value at the time of scheduling the function on the Data transformation definition screen is set in inactive items on the Data transformation definition screen.
- Cell in which XML element mapping is done, serves as the table of Excel. For table, see the Help of Excel.
- When you delete unnecessary lines in the table in which XML element mapping is done, delete row instead of making the cell blank. If row is not deleted, even if those are empty cells, those are exported as blank cell. Accordingly, if it mandatory item, error occurs at the time of importing.
- When you use template file of old version and new definition, you must implement either of the following work.
  - Acquire the template file again.
    For details on procedure, see "*6.12.1(1) When using a template file*".
  - Use template file of the old version as it is and perform re-mapping of table format XML schema definition file.
    For details on procedure, see "*6.12.1(2) When mapping the element based on table format XML schema definition file*".

6.13 Definition details of table format XML schema definition file

This section describes the contents of the table format XML schema definition file used in mapping definition of Excel.

Following table describes the definitions list of the table format XML schema definition file.

Table 6–25: TableDefinitions list of table format XML schema definition file

Major classification	Intermediate classification	Description
Defines	-	Defines the pre-requisite information for defining an object.
	Namespaces	Defines the namespace information used in definition of path that shows the transformation source node or transformation destination node on the Data transformation definition screen.
Objects#	-	Defines function or mapping line.
	CopyObjects	Defines a mapping line to the transformation destination node.
	ConcatenateObjects	Defines the Concatenate function.
	SubstringObjects	Defines the Acquire substring function.
	LengthObjects	Defines the Acquire string length function.
	ContainObjects	Defines the Check string function.
	TrimObjects	Defines the Trim node function.
	FormatObjects	Defines the Convert number format function.
	CalculateObjects	Defines the Perform node operation function.
	RoundObjects	Defines the Round node function.
	SumObjects	Defines he Sum up nodes function.
	NotObjects	Defines the NOT operation function.
	BitOpObjects	Defines the Logical operation function.
	ShiftObjects	Defines the Shift operation function.
	NameObjects	Defines the Acquire node name function.
	CountObjects	Defines the Acquire node count function.
	ExistObjects	Defines the Check node function.
	LoopObjects	Defines the Repeat function.
	ChooseObjects	Defines the Select function.
	ReplaceObjects	Defines the Replace value function.
	RadixObjects	Defines the Radix conversion function.
	CustomObjects	Defines the Custom function.
	ConstantObjects	Defines the Set constant function.

(Legend)

-: Not applicable

#

Definition range of CopyObjects and other than CopyObjects differs as follows:

CopyObjects:

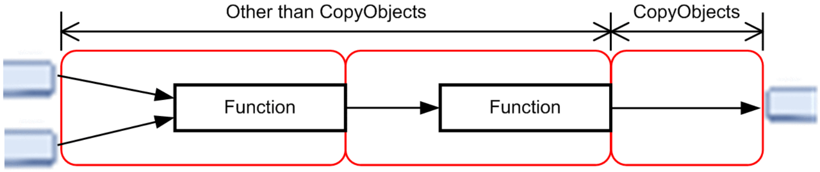
Definition range is the mapping line from the transformation source node or function to the transformation destination node.

Other than CopyObjects:

Definition range is the mapping line connected to function unit or transformation source node/input side function.

Following figure shows the definition range of CopyObjects and other than CopyObjects:

Figure 6–58: FigureDefinition range of CopyObjects and other than CopyObjects



Further sections describe the details of each definition.

6.13.1 Namespaces (namespace information)

Define the namespace information to be used in the definition of the path that shows the transformation source node or transformation destination node on the mapping definition editor, such as Input items of the object. If the Prefix item defined in Namespaces has been used as namespace prefix described in Input item of the object, determine the namespace to which that namespace prefix belongs, with the information defined in Namespaces.

Definition having Prefix item as "#default" shows the default namespace (namespace to belong to when namespace prefix does not exist). When not having definition where Prefix item is "#default", default namespace is considered as blank.

When URI item is blank, namespace of element to which corresponding namespace prefix is attached is considered as blank.

(1) Definition item

Following table shows the definition item:

Table 6–26: TableNamespaces definition

Item name	Definition contents	Value that you can define	Additional description
Prefix	Namespace prefix	Value of NCName type or "#default"	This is mandatory item. You cannot define same prefix in duplication. In case of "#default" Set the default namespace.
URI	Namespace	Value of anyURI type	-

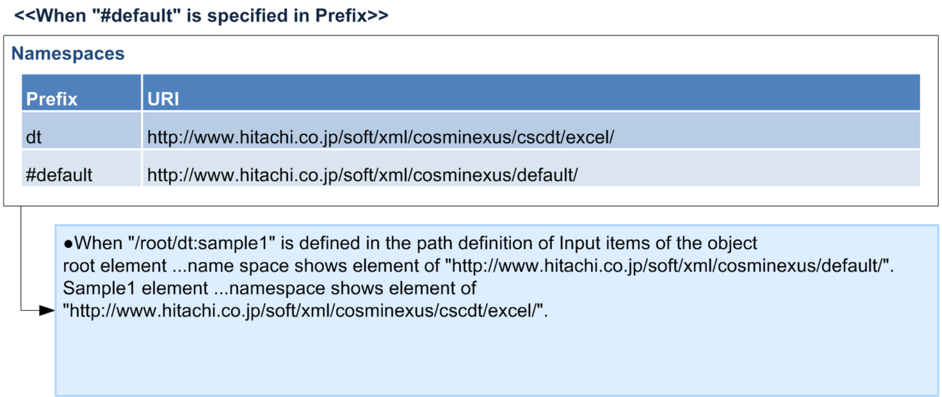
(Legend)

-: Corresponding item does not exist.

(2) Definition example

Following figure shows the definition example:

Figure 6–59: FigureNamespaces definition example1



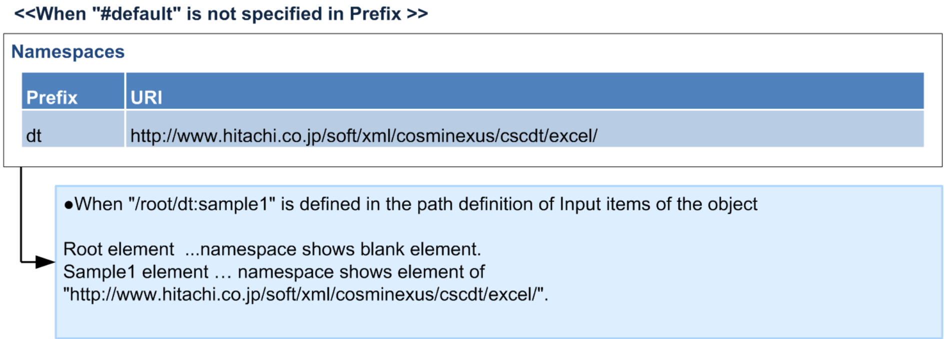
<<When "#default" is specified in Prefix>>

Namespaces

Prefix	URI
dt	http://www.hitachi.co.jp/soft/xml/cosminexus/cscdt/excel/
#default	http://www.hitachi.co.jp/soft/xml/cosminexus/default/

●When "/root/dt:sample1" is defined in the path definition of Input items of the object
root element ...name space shows element of "http://www.hitachi.co.jp/soft/xml/cosminexus/default/".
Sample1 element ...namespace shows element of
"http://www.hitachi.co.jp/soft/xml/cosminexus/cscdt/excel/".

Figure 6–60: FigureNamespaces definition example2



<<When "#default" is not specified in Prefix >>

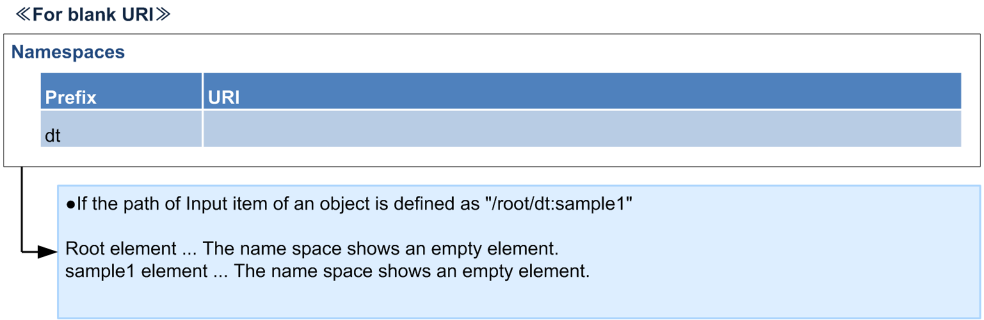
Namespaces

Prefix	URI
dt	http://www.hitachi.co.jp/soft/xml/cosminexus/cscdt/excel/

●When "/root/dt:sample1" is defined in the path definition of Input items of the object

Root element ...namespace shows blank element.
Sample1 element … namespace shows element of
"http://www.hitachi.co.jp/soft/xml/cosminexus/cscdt/excel/".

Figure 6–61: FigureNamespaces definition example3



≪For blank URI≫

Namespaces

Prefix	URI
dt	

●If the path of Input item of an object is defined as "/root/dt:sample1"

Root element ... The name space shows an empty element.
sample1 element ... The name space shows an empty element.

6.13.2 CopyObjects (mapping to transformation destination node)

Defines the mapping line from function or transformation source node defined in other object to the transformation destination node.

(1) Definition item

Following table describes definition items:

Table 6–27: TableCopyObjects definition

Item name	Definition contents	Value that you can define	Additional description
Name	Mapping name	Value of NCName type	This is mandatory item. In Objects, you cannot define same name in duplication.

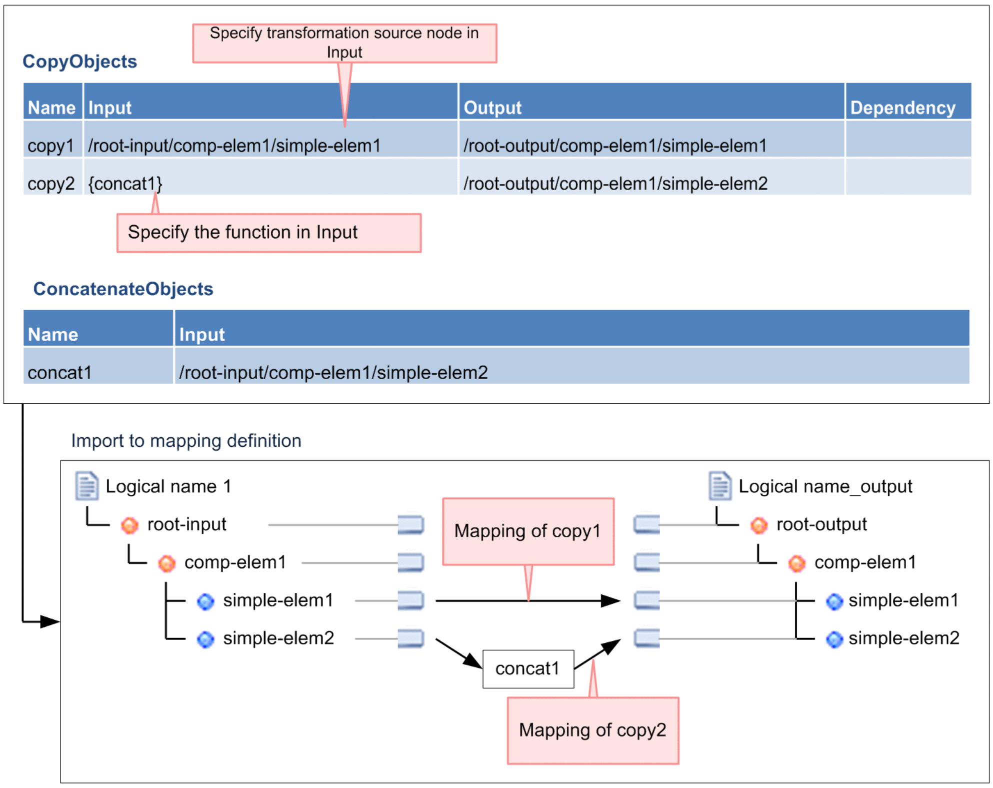
Item name	Definition contents	Value that you can define	Additional description
Name	Mapping name	Value of NCName type	This item is not used in the mapping definition.
Input	Input	See "*6.12.2 Setting up the mapping definition*"	This is mandatory item.
Output	Output	See "*6.12.2 Setting up the mapping definition*"	This is mandatory item.
Dependency	Dependency target	See "*6.12.2 Setting up the mapping definition*"	This item is mandatory when multiple LoopObjects that can be set as dependency target have been defined. When there is 1 LoopObjects that can be set as dependency target, set that LoopObjects as dependency target. For the definition of LoopObjects, see "*6.13.18 LoopObjects (Repeat function)*".

(2) Definition example

Following figure shows the definition example:

Figure 6–62: FigureCopyObjects definition example1

≪If there is no dependable LoopObjects≫



Specify transformation source node in Input

CopyObjects

Name	Input	Output	Dependency
copy1	/root-input/comp-elem1/simple-elem1	/root-output/comp-elem1/simple-elem1	
copy2	{concat1}	/root-output/comp-elem1/simple-elem2	

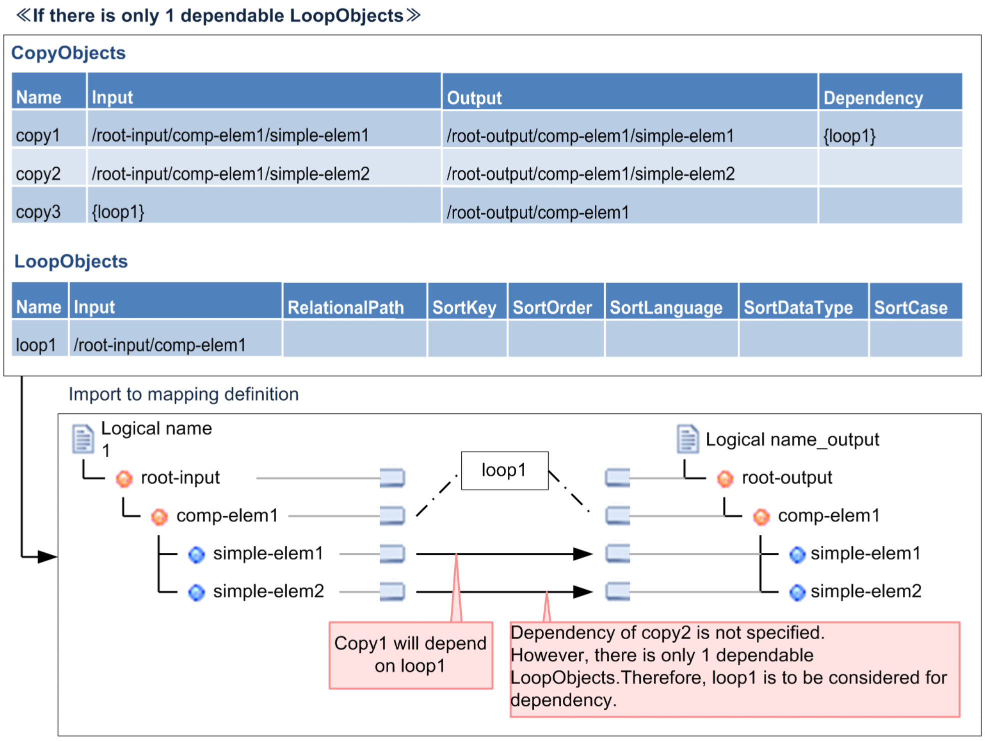
Specify the function in Input

ConcatenateObjects

Name	Input
concat1	/root-input/comp-elem1/simple-elem2

Import to mapping definition

Figure 6–63: FigureCopyObjects definition example2



≪If there is only 1 dependable LoopObjects≫

CopyObjects

Name	Input	Output	Dependency
copy1	/root-input/comp-elem1/simple-elem1	/root-output/comp-elem1/simple-elem1	{loop1}
copy2	/root-input/comp-elem1/simple-elem2	/root-output/comp-elem1/simple-elem2	
copy3	{loop1}	/root-output/comp-elem1	

LoopObjects

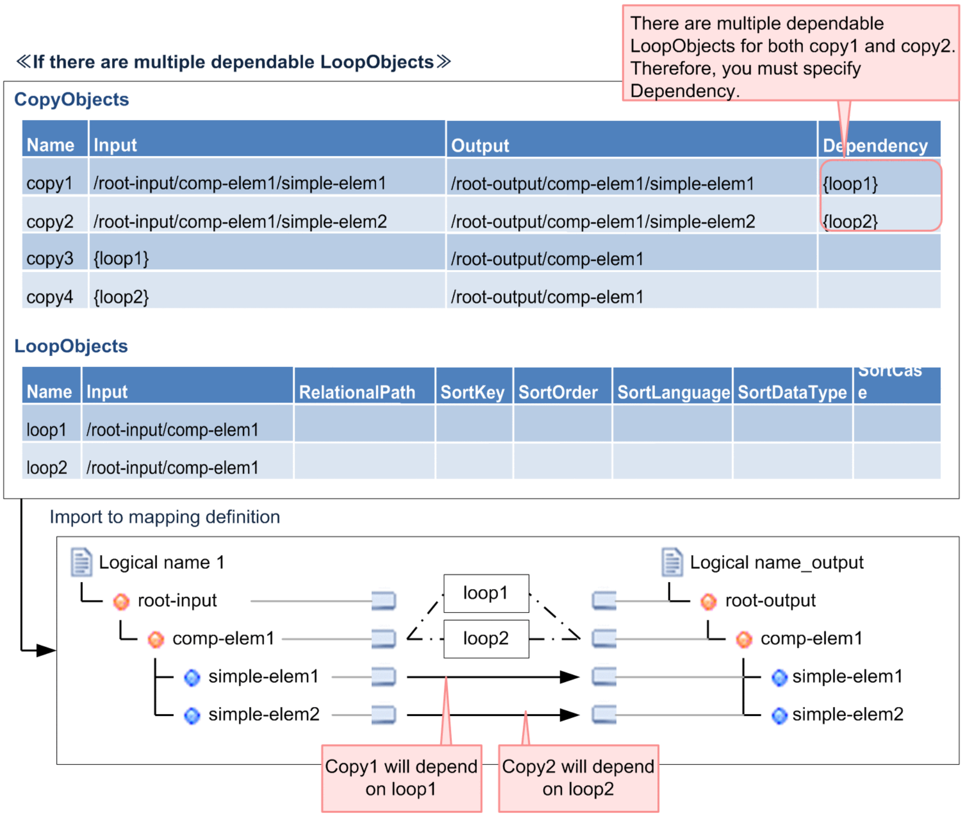
Name	Input	RelationalPath	SortKey	SortOrder	SortLanguage	SortDataType	SortCase
loop1	/root-input/comp-elem1						

Import to mapping definition

Copy1 will depend on loop1

Dependency of copy2 is not specified. However, there is only 1 dependable LoopObjects.Therefore, loop1 is to be considered for dependency.

Figure 6–64: FigureCopyObjects definition example3



There are multiple dependable LoopObjects for both copy1 and copy2. Therefore, you must specify Dependency.

≪If there are multiple dependable LoopObjects≫

CopyObjects

Name	Input	Output	Dependency
copy1	/root-input/comp-elem1/simple-elem1	/root-output/comp-elem1/simple-elem1	{loop1}
copy2	/root-input/comp-elem1/simple-elem2	/root-output/comp-elem1/simple-elem2	{loop2}
copy3	{loop1}	/root-output/comp-elem1	
copy4	{loop2}	/root-output/comp-elem1	

LoopObjects

Name	Input	RelationalPath	SortKey	SortOrder	SortLanguage	SortDataType	SortCase
loop1	/root-input/comp-elem1						
loop2	/root-input/comp-elem1						

Import to mapping definition

Copy1 will depend on loop1

Copy2 will depend on loop2

6.13.3 ConcatenateObjects (Concatenate function)

Defines Concatenate function (concat).

(1) Definition items

Following table shows the definition items:

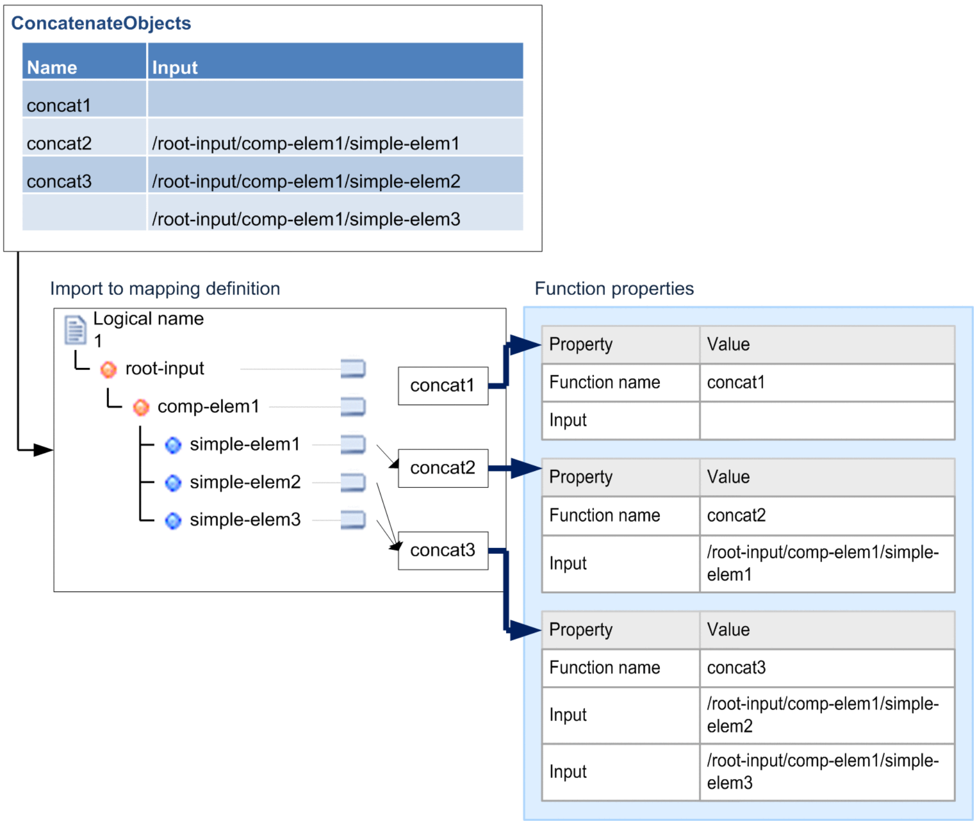
Table 6–28: TableConcatenateObjects definition items

Item name	Definition contents	Value that you can define	Additional description
Name	Function name	Value of NCName type	For details on the values that you can define, see "*6.13.24 Objects for which you can define the same Name element in multiple rows*".
Input	Input	See "*6.12.2 Setting up the mapping definition*".	You can define this item for multiple items in the same Name. Set value in the list in the defined order.

(2) Definition example

Following figure shows the definition example:

Figure 6–65: FigureConcatenateObjects definition example



ConcatenateObjects

Name	Input
concat1	
concat2	/root-input/comp-elem1/simple-elem1
concat3	/root-input/comp-elem1/simple-elem2
	/root-input/comp-elem1/simple-elem3

Property	Value
Function name	concat1
Input	

Property	Value
Function name	concat2
Input	/root-input/comp-elem1/simple-elem1

Property	Value
Function name	concat3
Input	/root-input/comp-elem1/simple-elem2
Input	/root-input/comp-elem1/simple-elem3

6.13.4 SubstringObjects (Acquire substring function)

Acquire substring function (substr).

(1) Definition items

Following table describes the definition items.

Table 6–29: TableSubstringObjects definition items

Item name	Definition contents	Value that you can define	Additional description
Name	Function name	Value of NCName type	This is mandatory item. You cannot define same Name in Objects, in duplication.
Input	Input	See "*6.12.2 Setting up the mapping definition*"	-
Acquisition	Specification item	Either of "left", "right", "division"	This is mandatory item. In case of "left" As specification method, set "Specify range from the beginning" In case of "right" As specification method, set "Specify range from end" In case of "division" As specification method, set "Specify division character string"
Start	Start position	Integer of 1~2,147,483,647	This is mandatory item when acquisition is "left" or "right". In other cases, setting value is not incorporated. Surrogate pair is handled as 2 characters.
Count	Characters count	Integer of 0~2,147,483,647 or space	When Acquisition is "left" or "right" and Count is blank, set "all after start position". When Acquisition is other than "left" and "right", setting value is not incorporated. Surrogate pair is handled as 2 characters.
String	Division character string	Character string less than 1,024 characters (you cannot define linefeed)	This is mandatory item when Acquisition is "division". In other cases, setting value is not incorporated.
Part	Acquisition position	"pre" or "post"	This is mandatory item when Acquisition is "division". In other cases, setting value is not incorporated. In case of "pre" Set "Previous" as acquisition part. In case of "post" Set "Post" as acquisition part.

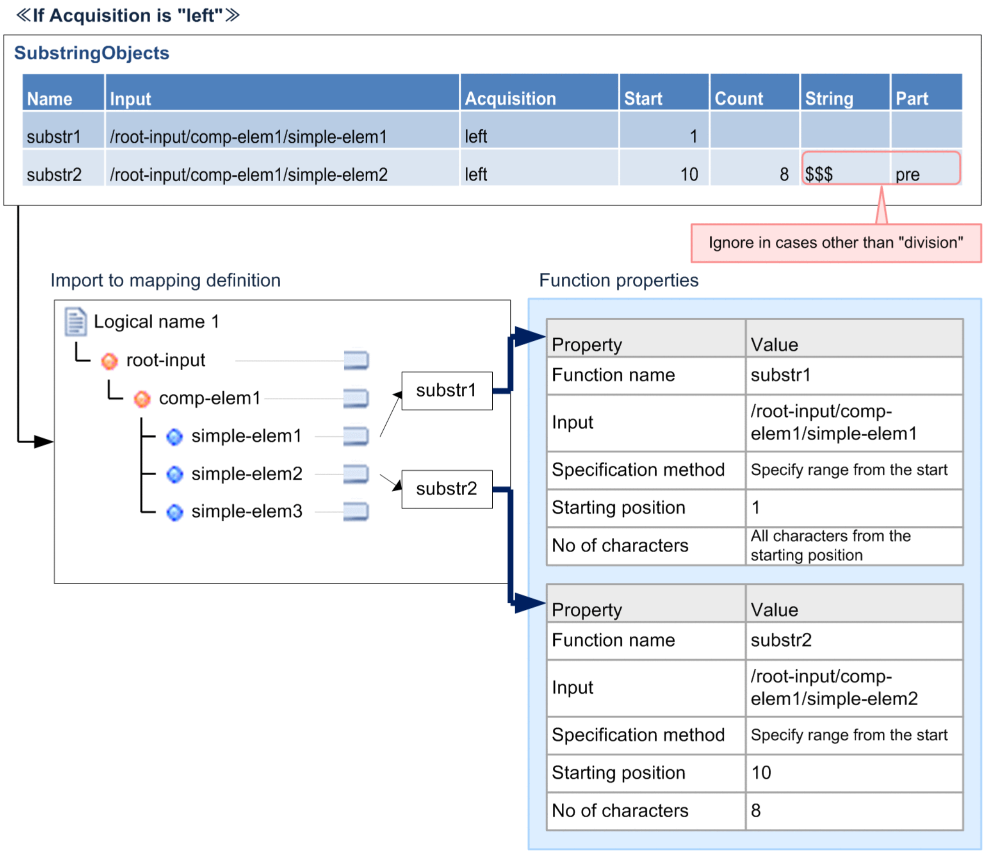
(Legend)

-: Corresponding item does not exist.

(2) Definition example

Following figure shows the definition example:

Figure 6–66: FigureSubstringObjects definition example1



≪If Acquisition is "left"≫

SubstringObjects

Name	Input	Acquisition	Start	Count	String	Part
substr1	/root-input/comp-elem1/simple-elem1	left	1			
substr2	/root-input/comp-elem1/simple-elem2	left	10	8	$$$	pre

Ignore in cases other than "division"

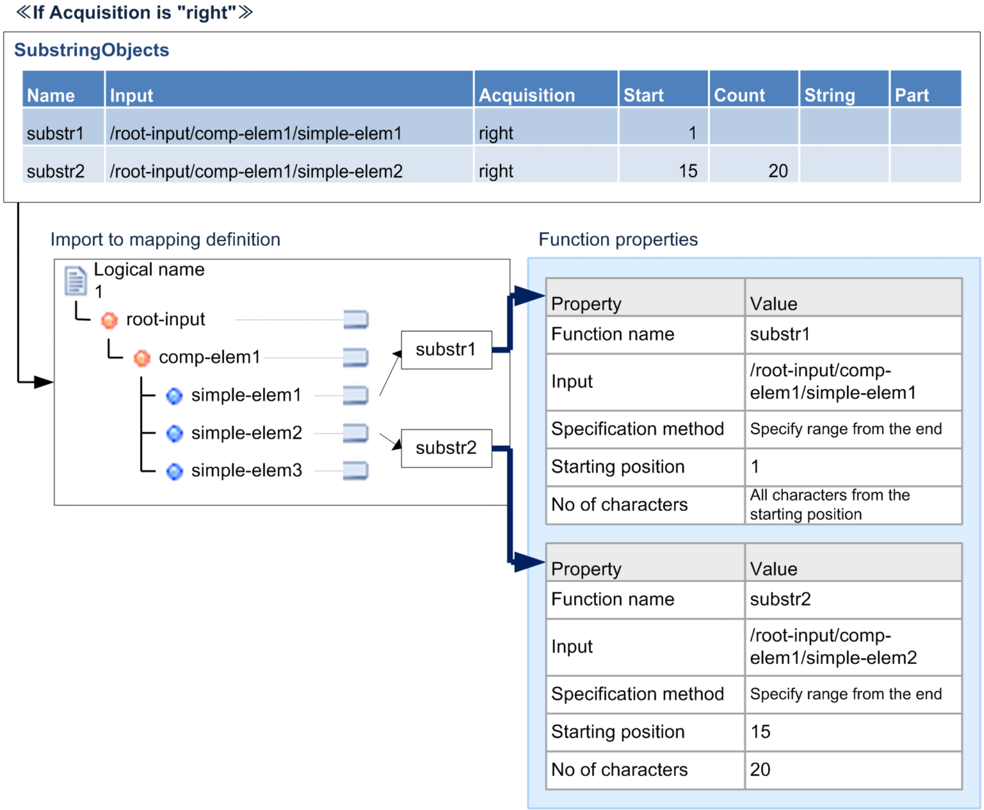
Import to mapping definition

Function properties

Property	Value
Function name	substr1
Input	/root-input/comp-elem1/simple-elem1
Specification method	Specify range from the start
Starting position	1
No of characters	All characters from the starting position

Property	Value
Function name	substr2
Input	/root-input/comp-elem1/simple-elem2
Specification method	Specify range from the start
Starting position	10
No of characters	8

Figure 6–67: FigureSubstringObjects definition example2



≪If Acquisition is "right"≫

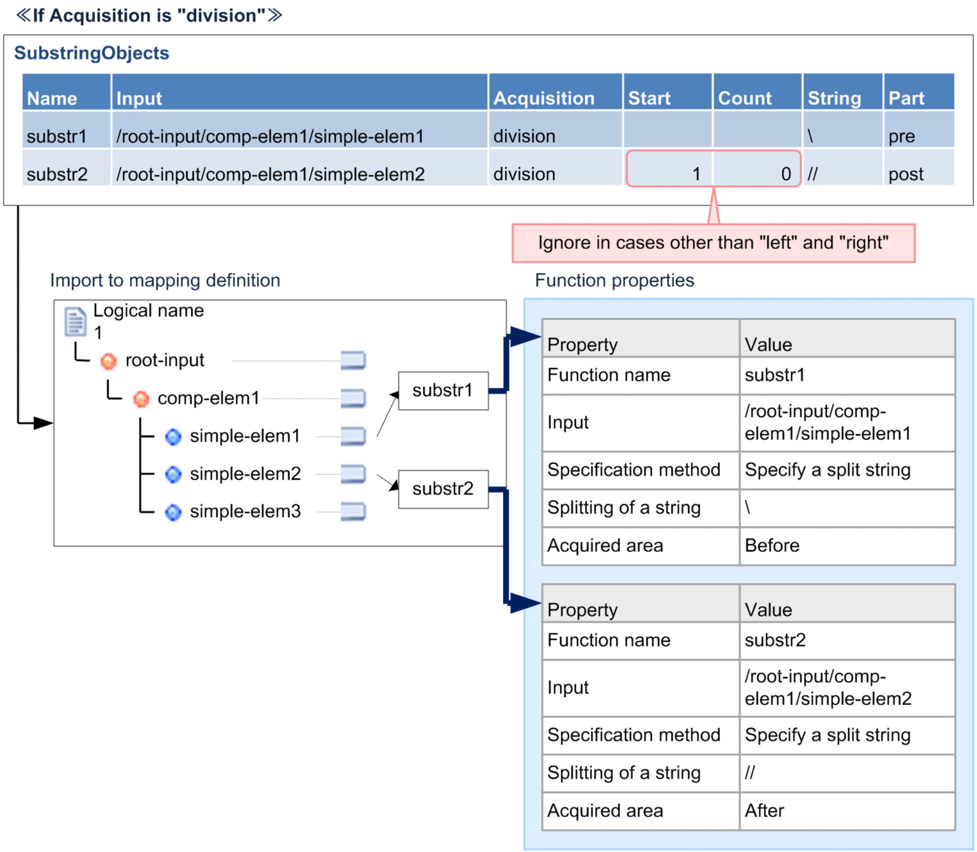
SubstringObjects

Name	Input	Acquisition	Start	Count	String	Part
substr1	/root-input/comp-elem1/simple-elem1	right	1			
substr2	/root-input/comp-elem1/simple-elem2	right	15	20		

Property	Value
Function name	substr1
Input	/root-input/comp-elem1/simple-elem1
Specification method	Specify range from the end
Starting position	1
No of characters	All characters from the starting position

Property	Value
Function name	substr2
Input	/root-input/comp-elem1/simple-elem2
Specification method	Specify range from the end
Starting position	15
No of characters	20

Figure 6–68: FigureSubstringObjects definition example3



≪If Acquisition is "division"≫

SubstringObjects

Name	Input	Acquisition	Start	Count	String	Part
substr1	/root-input/comp-elem1/simple-elem1	division			\	pre
substr2	/root-input/comp-elem1/simple-elem2	division	1	0	//	post

Ignore in cases other than "left" and "right"

Import to mapping definition

Function properties

Property	Value
Function name	substr1
Input	/root-input/comp-elem1/simple-elem1
Specification method	Specify a split string
Splitting of a string	\
Acquired area	Before

Property	Value
Function name	substr2
Input	/root-input/comp-elem1/simple-elem2
Specification method	Specify a split string
Splitting of a string	//
Acquired area	After

6.13.5 LengthObjects (Acquire string length function)

This section defines the Acquire string length function (length).

(1) Definition item

Following table describes definition items.

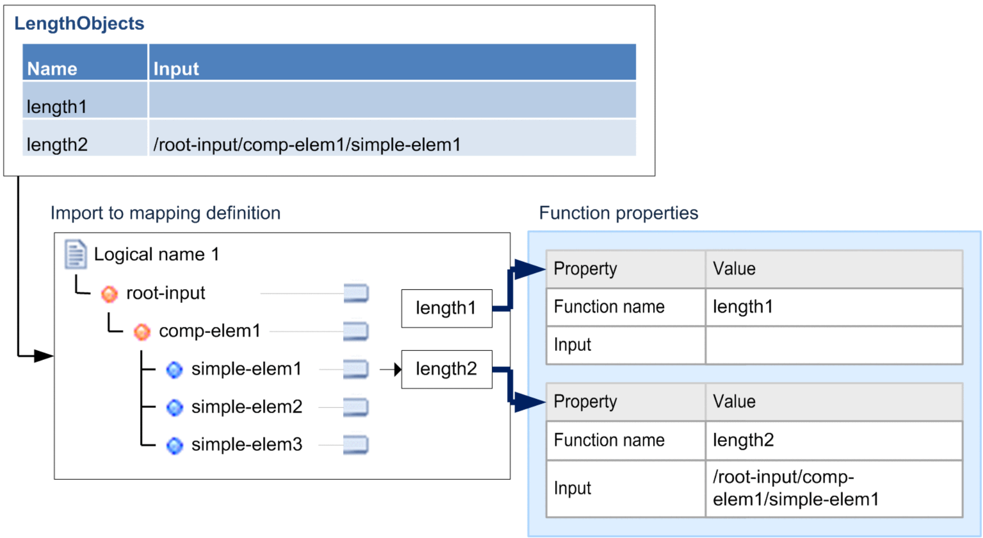
Table 6–30: TableLengthObjects definition item

Item name	Definition contents	Value that you can define	Additional description
Name	Function name	Value of NCName type	This is mandatory item. You cannot define same Name in Objects, in duplication.
Input	Input	See "*6.12.2 Setting up the mapping definition*"	Surrogate pair is handled as 2 characters.

(2) Definition example

Following figure shows the definition example:

Figure 6–69: FigureLengthObjects definition example



6.13.6 ContainObjects(Check string function)

This section defines the Check string function (contain).

(1) Definition item

Following table describes the definition items.

Table 6–31: TableContainObjects definition items

Item name	Definition contents	Value that you can define	Additional description
Name	Function name	Value of NCName type	This is mandatory item. You cannot define the Name in Objects, in duplication.
Input	Input	See "*6.12.2 Setting up the mapping definition*"	-
Check	Check type	"include" or "start"	This is mandatory item. In case of "include" As check type, set up the "Include the specified character string". In case of "start" As check type, set up "Start from the specified character string".
String	Check target character string	Character string having less than 1,024 characters (you cannot define the linefeed code)	-

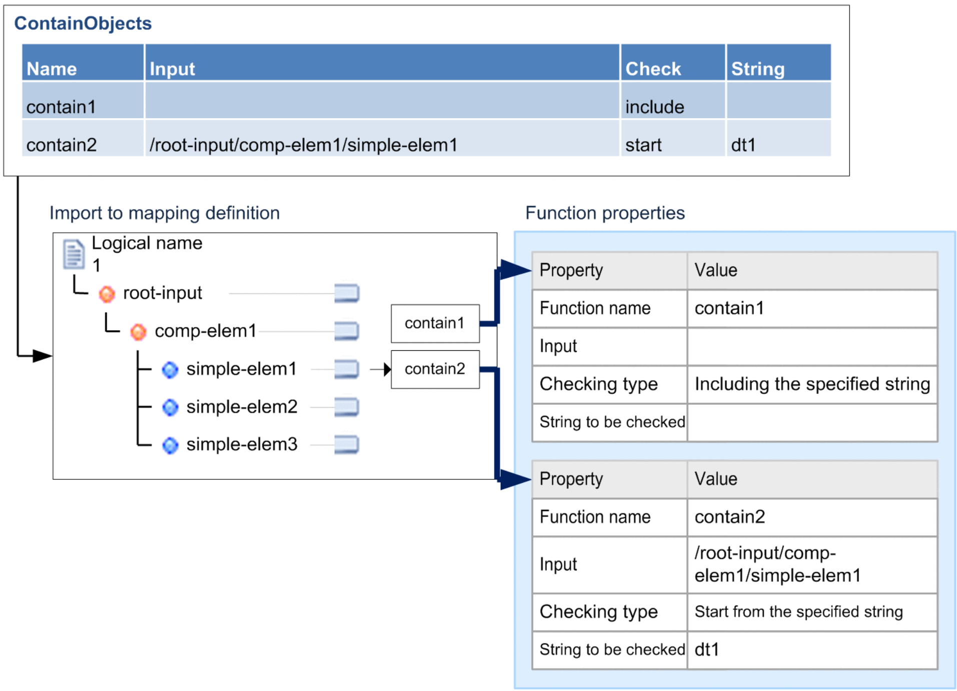
(Legend)

-: Corresponding item does not exist.

(2) Definition example

Following figure shows the definition example.

Figure 6–70: FigureContainObjects definition example



ContainObjects

Name	Input	Check	String
contain1		include	
contain2	/root-input/comp-elem1/simple-elem1	start	dt1

Property	Value
Function name	contain1
Input	
Checking type	Including the specified string
String to be checked	

Property	Value
Function name	contain2
Input	/root-input/comp-elem1/simple-elem1
Checking type	Start from the specified string
String to be checked	dt1

6.13.7 TrimObjects (Trim node function)

This section defines the Trim node function (trim).

(1) Definition items

Following table describes definition items:

Table 6–32: TableTrimObjects definition items

Item name	Definition contents	Value that you can define	Additional description
Name	Function name	Value of NCName	This is mandatory item. You cannot define the Name in Objects, in duplication.
Input	Input	See "*6.12.2 Setting up the mapping definition*"	-

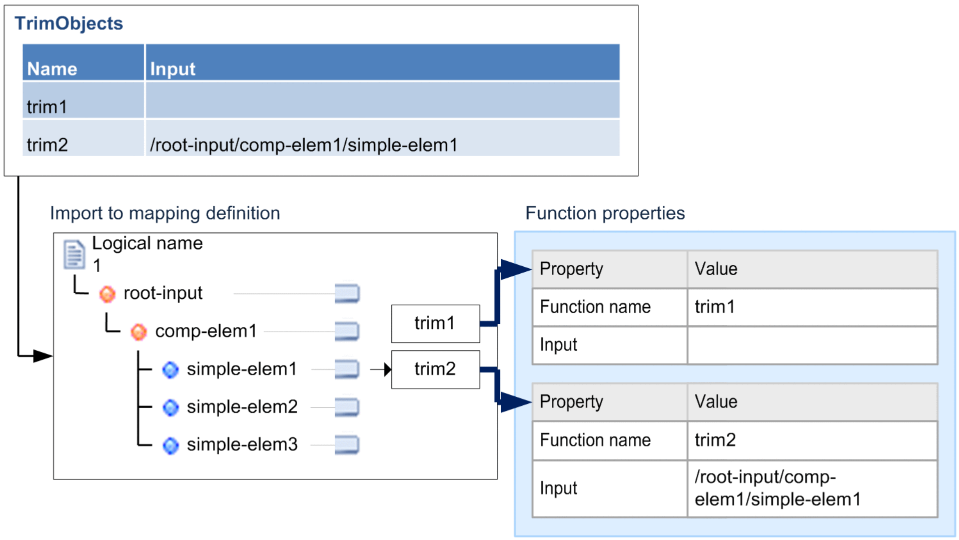
(Legend)

-: Corresponding item does not exist.

(2) Definition example

Following figure shows the definition example:

Figure 6–71: FigureTrimObjectsDefinition example



TrimObjects

Name	Input
trim1	
trim2	/root-input/comp-elem1/simple-elem1

Function properties

Property	Value
Function name	trim1
Input	

Property	Value
Function name	trim2
Input	/root-input/comp-elem1/simple-elem1

6.13.8 FormatObjects(Convert number format function)

This section defines Convert number format function (format).

(1) Definition items

Following table describes definition items:

Table 6–33: TableFormatObjects definition items

Item name	Definition contents	Value that you can define	Additional description
Name	Function name	Value of NCName	This is mandatory item. You cannot define the Name in Objects, in duplication.
Input	Input	See "*6.12.2 Setting up the mapping definition*"	-
Pattern	Pattern	• Pattern character string of the format of java.text.DecimalFormat class after transformation • Less than 1,024 characters (You cannot define linefeed code)	-
ChangeSymbol	Change in symbol	"y" or "n"	This is mandatory item. In case of "y" Set up "Exists" for change in symbol. In case of "n" Set up "Does not exist" for Change in symbol
DecimalSeparator	Characters below decimal point	1 character	This is mandatory item when ChangeSymbol is "y". In other cases, setting value is not incorporated.
GroupingSeparator	Digit separation character	1 character	This is mandatory item when ChangeSymbol is "y". In other cases, setting value is not incorporated.

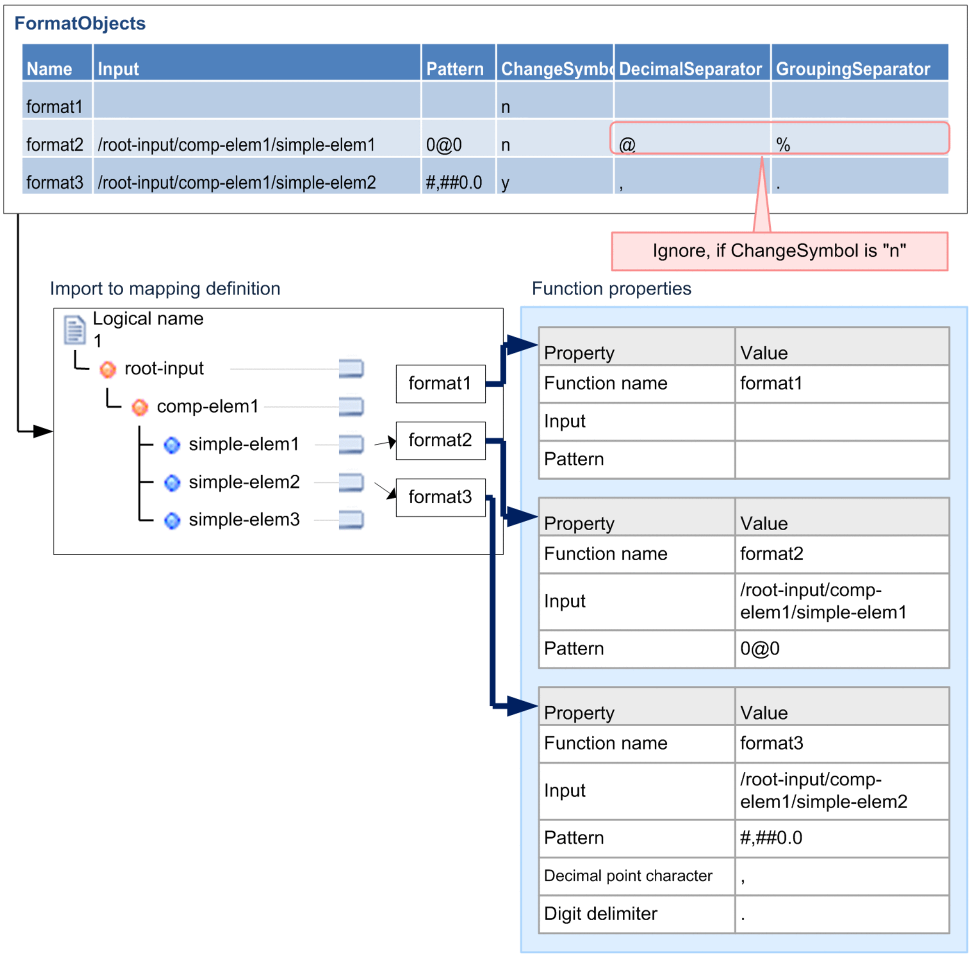
(Legend)

-: Corresponding item does not exist.

(2) Definition example

Following figure shows the definition example:

Figure 6–72: FigureFormatObjectsDefinition example



FormatObjects

Name	Input	Pattern	ChangeSymbo	DecimalSeparator	GroupingSeparator
format1			n		
format2	/root-input/comp-elem1/simple-elem1	0@0	n	@	%
format3	/root-input/comp-elem1/simple-elem2	#,##0.0	y	,	.

Ignore, if ChangeSymbol is "n"

Import to mapping definition

Function properties

Property	Value
Function name	format1
Input	
Pattern	

Property	Value
Function name	format2
Input	/root-input/comp-elem1/simple-elem1
Pattern	0@0

Property	Value
Function name	format3
Input	/root-input/comp-elem1/simple-elem2
Pattern	#,##0.0
Decimal point character	,
Digit delimiter	.

6.13.9 CalculateObjects(Perform node operation function)

This section defines the Perform node operation function (calc).

(1) Definition item

Following table describes definition items:

Table 6–34: TableCalculateObjectsDefinition item

Item name	Definition contents	Value that you can define	Additional description
Name	Function name	Value of NCName type	This is mandatory item. You cannot define the Name in Objects, in duplication.
Input1	Input 1	See "*6.12.2 Setting up the mapping definition*"	-
Operation	Operation	Either of "+", "-", "*", "/", or "%"	This is mandatory item.

Item name	Definition contents	Value that you can define	Additional description
Input2	Input 2	See "*6.12.2 Setting up the mapping definition*"	-

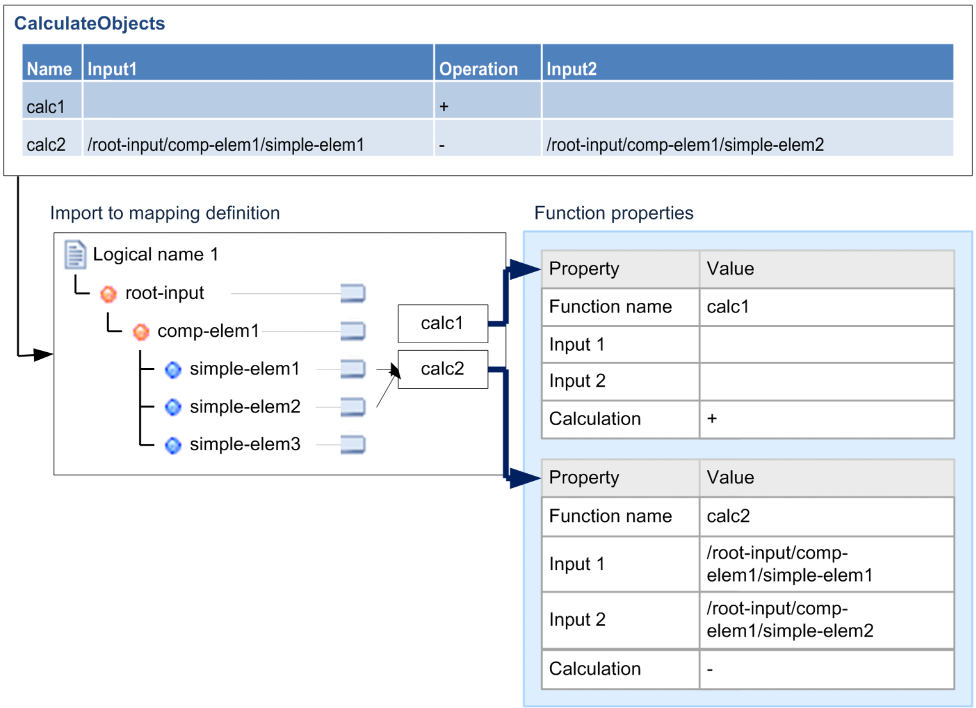
(Legend)

-: Corresponding item does not exist.

(2) Definition example

Following figure shows the definition example:

Figure 6–73: FigureCalculateObjectsDefinition example



CalculateObjects

Name	Input1	Operation	Input2
calc1		+	
calc2	/root-input/comp-elem1/simple-elem1	-	/root-input/comp-elem1/simple-elem2

Property	Value
Function name	calc1
Input 1	
Input 2	
Calculation	+

Property	Value
Function name	calc2
Input 1	/root-input/comp-elem1/simple-elem1
Input 2	/root-input/comp-elem1/simple-elem2
Calculation	-

6.13.10 RoundObjects(Round node function)

This section defines the Round node function (round).

(1) Definition item

Following table describes definition items:

Table 6–35: TableRoundObjectsDefinition item

Item name	Definition contents	Value that you can define	Additional description
Name	Function name	Value of NCName type	This is mandatory item. You cannot define the Name in Objects, in duplication.
Input	Input	See "*6.12.2 Setting up the mapping definition*"	-
Type	Round node types	Either of "off", "down", or "up"	This is mandatory item. In case of "off" Set up "Round" as Round node type.

Item name	Definition contents	Value that you can define	Additional description
Type	Round node types	Either of "off", "down", or "up"	In case of "down" Set up "Round down" as Round node type. In case of "up" Set up "Round up" as Round node type.

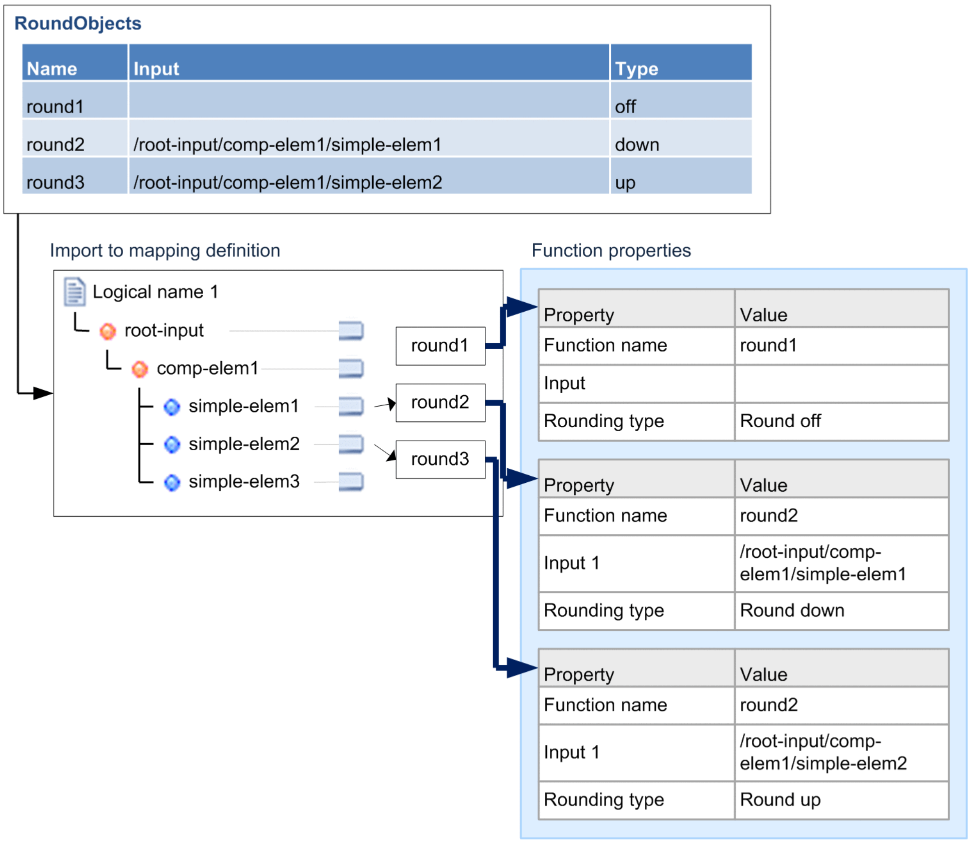
(Legend)

-: Corresponding item does not exist.

(2) Definition example

Following figure shows the definition example:

Figure 6–74: FigureRoundObjectsDefinition example



RoundObjects

Name	Input	Type
round1		off
round2	/root-input/comp-elem1/simple-elem1	down
round3	/root-input/comp-elem1/simple-elem2	up

Property	Value
Function name	round1
Input	
Rounding type	Round off

Property	Value
Function name	round2
Input 1	/root-input/comp-elem1/simple-elem1
Rounding type	Round down

Property	Value
Function name	round2
Input 1	/root-input/comp-elem1/simple-elem2
Rounding type	Round up

6.13.11 SumObjects(Sum up nodes function)

This section defines the Sum up nodes function (sum).

(1) Definition item

Following table describes definition items:

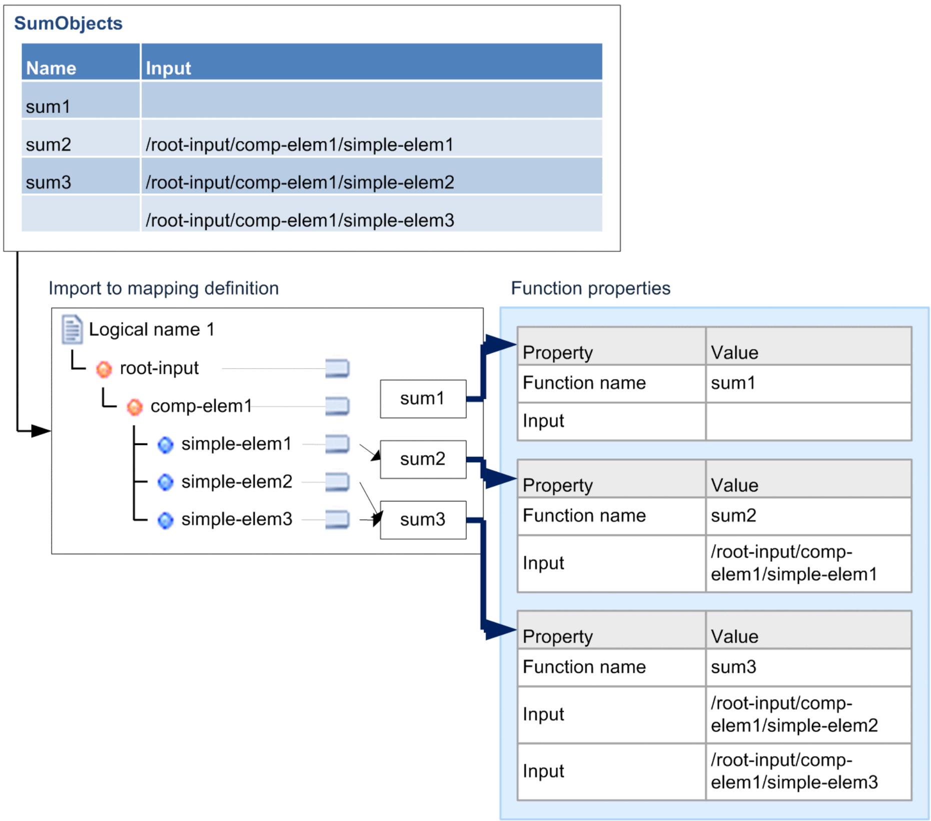
Table 6–36: TableSumObjectsDefinition item

Item name	Definition contents	Value that you can define	Additional description
Name	Function name	Value of NCName	For details on the values that you can define, see "*6.13.24 Objects for which you can define the same Name element in multiple rows*".
Input	Input	See "*6.12.2 Setting up the mapping definition*"	You can perform multiple definitions in same Name. However, you cannot specify same Input for multiple times in the same Name. Sets values in the list, in the defined order.

(2) Definition example

Following figure shows the definition example:

Figure 6–75: FigureSumObjectsDefinition example



SumObjects

Name	Input
sum1	
sum2	/root-input/comp-elem1/simple-elem1
sum3	/root-input/comp-elem1/simple-elem2
	/root-input/comp-elem1/simple-elem3

Function properties

Property	Value
Function name	sum1
Input	

Property	Value
Function name	sum2
Input	/root-input/comp-elem1/simple-elem1

Property	Value
Function name	sum3
Input	/root-input/comp-elem1/simple-elem2
Input	/root-input/comp-elem1/simple-elem3

6.13.12 NotObjects(NOT operation function)

This section defines the NOT operation function (not).

(1) Definition item

Following table describes definition items:

Table 6–37: TableNotObjectsDefinition item

Item name	Definition contents	Value that you can define	Additional description
Name	Function name	Value of NCName type	This is mandatory item. You cannot define the Name in Objects, in duplication.
Input	Input	See "*6.12.2 Setting up the mapping definition*"	-

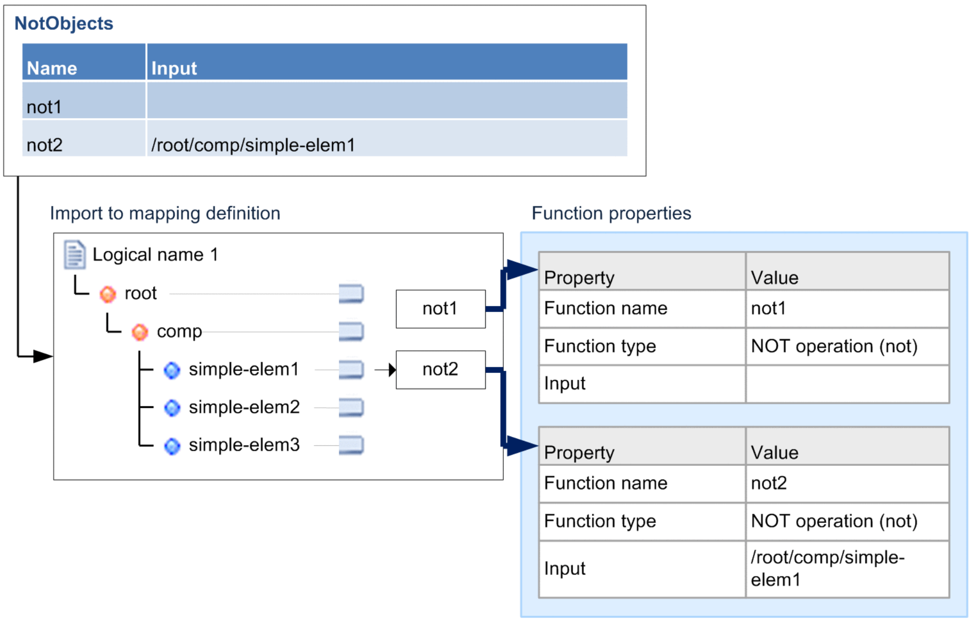
(Legend)

-: Corresponding item does not exist.

(2) Definition example

Following figure shows the definition example:

Figure 6–76: FigureNotObjectsDefinition example



NotObjects

Name	Input
not1	
not2	/root/comp/simple-elem1

Property	Value
Function name	not1
Function type	NOT operation (not)
Input	

Property	Value
Function name	not2
Function type	NOT operation (not)
Input	/root/comp/simple-elem1

6.13.13 BitOpObjects(Logical operation function)

This section defines the Logical operation function (bitop).

(1) Definition item

Following table describes definition items:

Table 6–38: TableBitOpObjectsDefinition item

Item name	Definition contents	Value that you can define	Additional description
Name	Function name	Value of NCName type	This is mandatory item. You cannot define the Name in Objects, in duplication.
Input1	Input 1	See "*6.12.2 Setting up the mapping definition*".	
Input2	Input 2	See "*6.12.2 Setting up the mapping definition*"	-

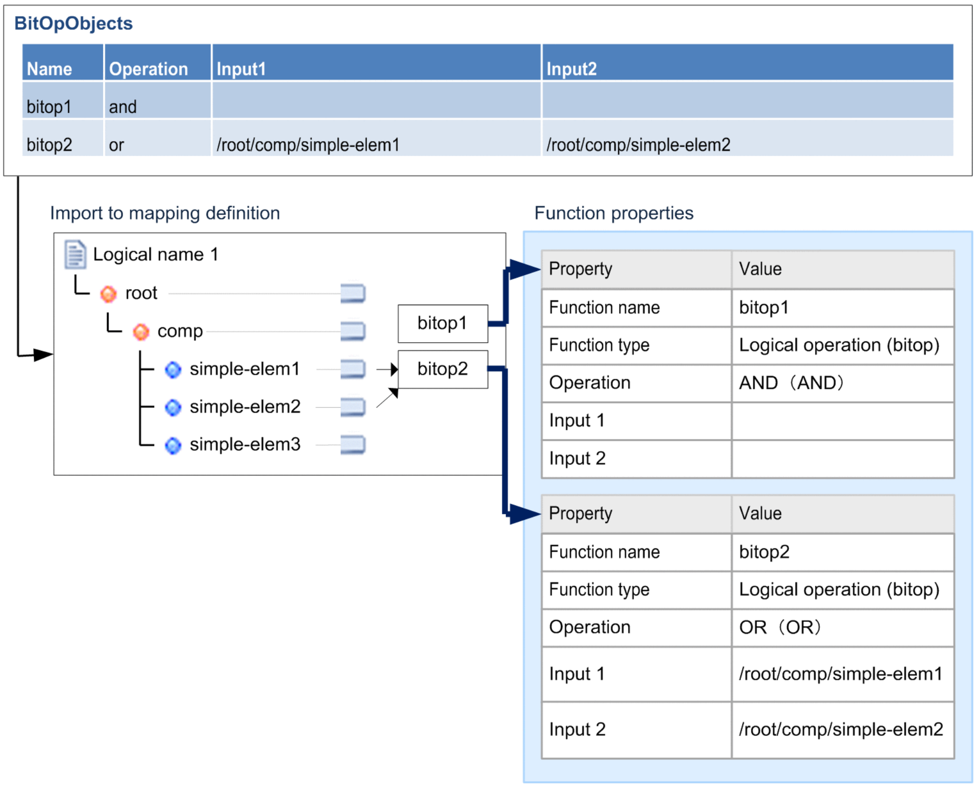
(Legend)

-: Corresponding item does not exist.

(2) Definition example

Following figure shows the definition example:

Figure 6–77: FigureBitOpObjectsDefinition example



BitOpObjects

Name	Operation	Input1	Input2
bitop1	and		
bitop2	or	/root/comp/simple-elem1	/root/comp/simple-elem2

Property	Value
Function name	bitop1
Function type	Logical operation (bitop)
Operation	AND（AND）
Input 1	
Input 2	

Property	Value
Function name	bitop2
Function type	Logical operation (bitop)
Operation	OR（OR）
Input 1	/root/comp/simple-elem1
Input 2	/root/comp/simple-elem2

6.13.14 ShiftObjects(Shift operation function)

This section defines the Shift operation function (shift).

(1) Definition item

Following table describes definition items:

Table 6–39: TableShiftObjectsDefinition item

Item name	Definition contents	Value that you can define	Additional description
Name	Function name	Value of NCName type	This is mandatory item. You cannot define the Name in Objects, in duplication.
Input	Input	See "*6.12.2 Setting up the mapping definition*"	-
ShiftAmount	Shift volume (bit)	Numeric value in the range of 0~64	This is mandatory item.
OutputSize	Output size (bit)	Numeric value in the range of 1~64	This is mandatory item.
ShiftDirection	Shift direction	"left" or "right"	This is mandatory item.

Item name	Definition contents	Value that you can define	Additional description
ShiftType	Shift type	"arithmetic" or "logical"	This is mandatory item when ShiftDirection is "right". In other cases, setting value is not incorporated. "arithmetic" is the arithmetic shift. "logical" is the logical shift.

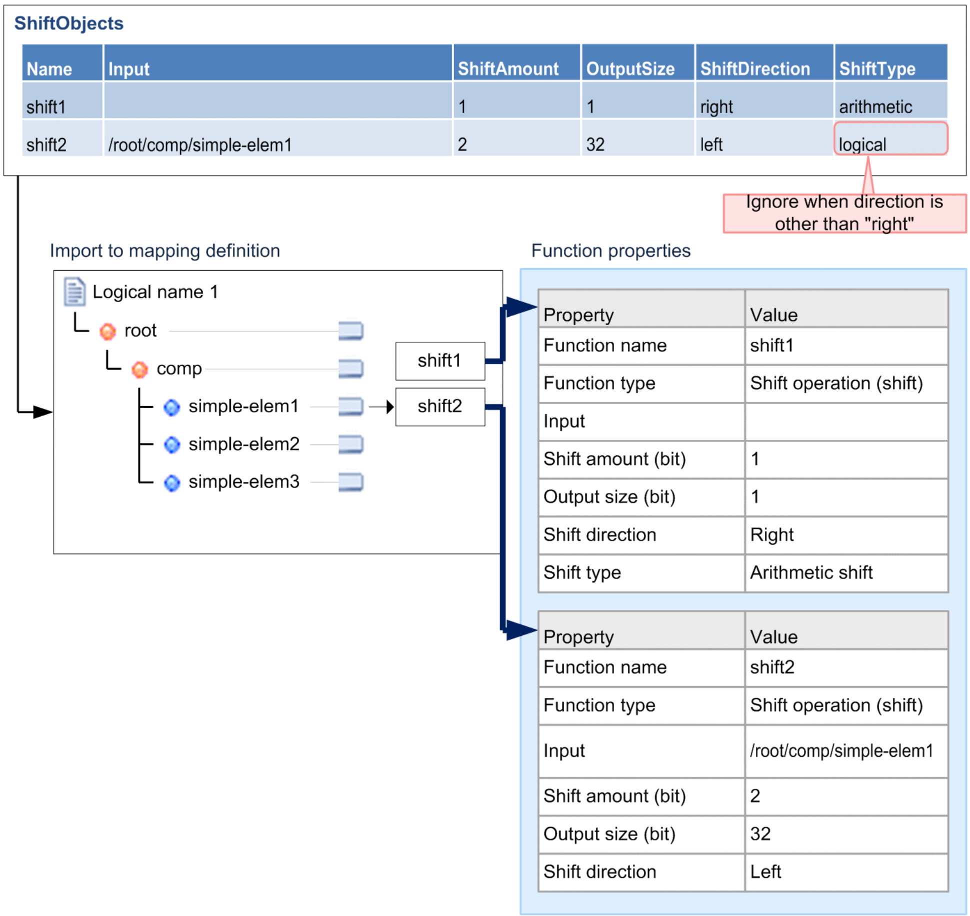
(Legend)

-: Corresponding item does not exist.

(2) Definition example

Following figure shows the definition example:

Figure 6–78: FigureShiftObjectsDefinition example



ShiftObjects

Name	Input	ShiftAmount	OutputSize	ShiftDirection	ShiftType
shift1		1	1	right	arithmetic
shift2	/root/comp/simple-elem1	2	32	left	logical

Ignore when direction is other than "right"

Function properties

Property	Value
Function name	shift1
Function type	Shift operation (shift)
Input	
Shift amount (bit)	1
Output size (bit)	1
Shift direction	Right
Shift type	Arithmetic shift

Property	Value
Function name	shift2
Function type	Shift operation (shift)
Input	/root/comp/simple-elem1
Shift amount (bit)	2
Output size (bit)	32
Shift direction	Left

6.13.15 NameObjects(Acquire node name function)

This section defines the Acquire node name function (name).

(1) Definition item

Following table describes definition items:

Table 6–40: TableNameObjectsDefinition item

Item name	Definition contents	Value that you can define	Additional description
Name	Function name	Value of NCName type	This is mandatory item. You cannot define the Name in Objects, in duplication.
Input	Input	See "*6.12.2 Setting up the mapping definition*"	-

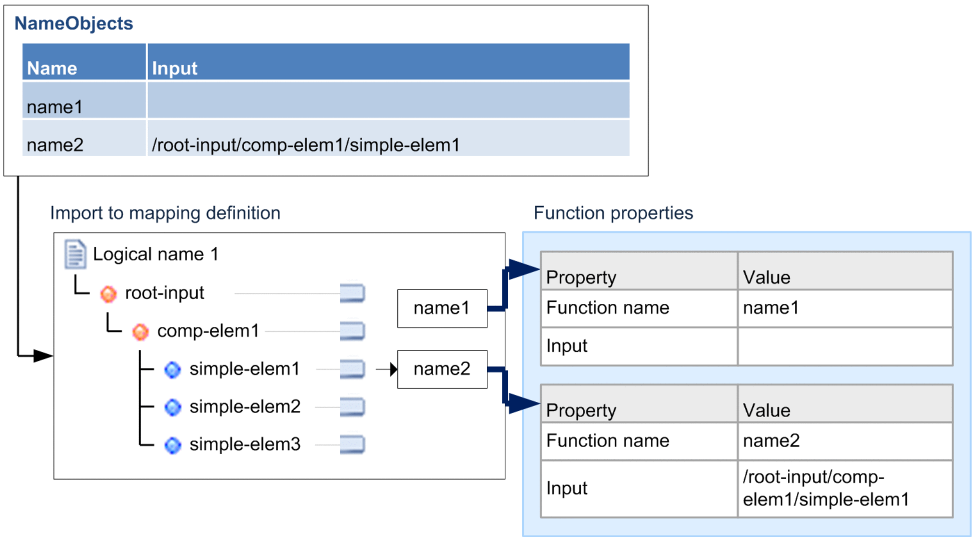
(Legend)

-: Corresponding item does not exist.

(2) Definition example

Following figure shows the definition example:

Figure 6–79: FigureNameObjectsDefinition example



NameObjects

Name	Input
name1	
name2	/root-input/comp-elem1/simple-elem1

Property	Value
Function name	name1
Input	

Property	Value
Function name	name2
Input	/root-input/comp-elem1/simple-elem1

6.13.16 CountObjects(Acquire node count function)

This section defines the Acquire node count function (count).

(1) Definition item

Following table describes definition items:

Table 6–41: TableCountObjectsDefinition item

Item name	Definition contents	Value that you can define	Additional description
Name	Function name	Value of NCName type	This is mandatory item. You cannot define the Name in Objects, in duplication.
Input	Input	See "*6.12.2 Setting up the mapping definition*"	-

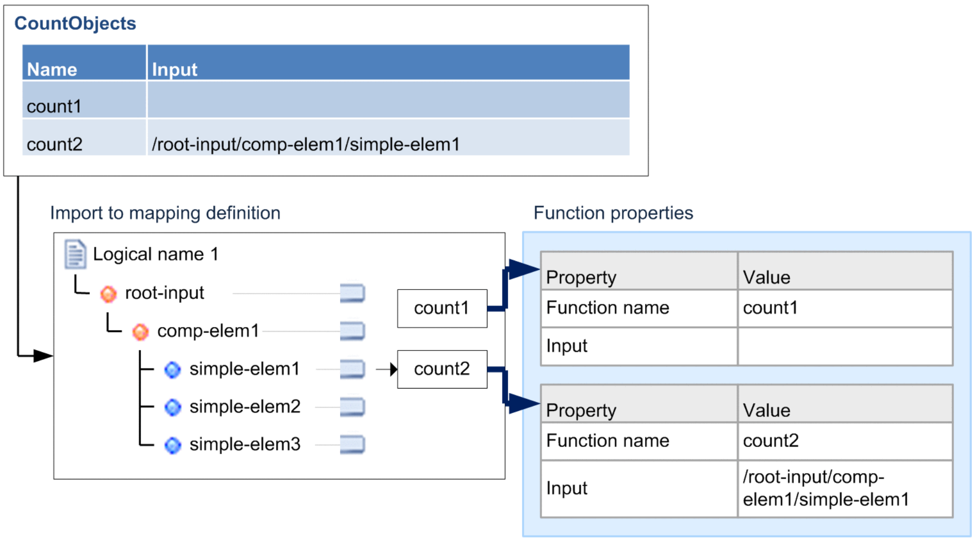
(Legend)

-: Corresponding item does not exist.

(2) Definition example

Following figure shows the definition example:

Figure 6–80: FigureCountObjectsDefinition example



CountObjects

Name	Input
count1	
count2	/root-input/comp-elem1/simple-elem1

Property	Value
Function name	count1
Input	

Property	Value
Function name	count2
Input	/root-input/comp-elem1/simple-elem1

6.13.17 ExistObjects(Check node function)

This section defines the Check node function (exist).

(1) Definition item

Following table describes definition items:

Table 6–42: TableExistObjectsDefinition item

Item name	Definition contents	Value that you can define	Additional description
Name	Function name	Value of NCName type	This is mandatory item. You cannot define the Name in Objects, in duplication.
Input	Input	See "*6.12.2 Setting up the mapping definition*"	-

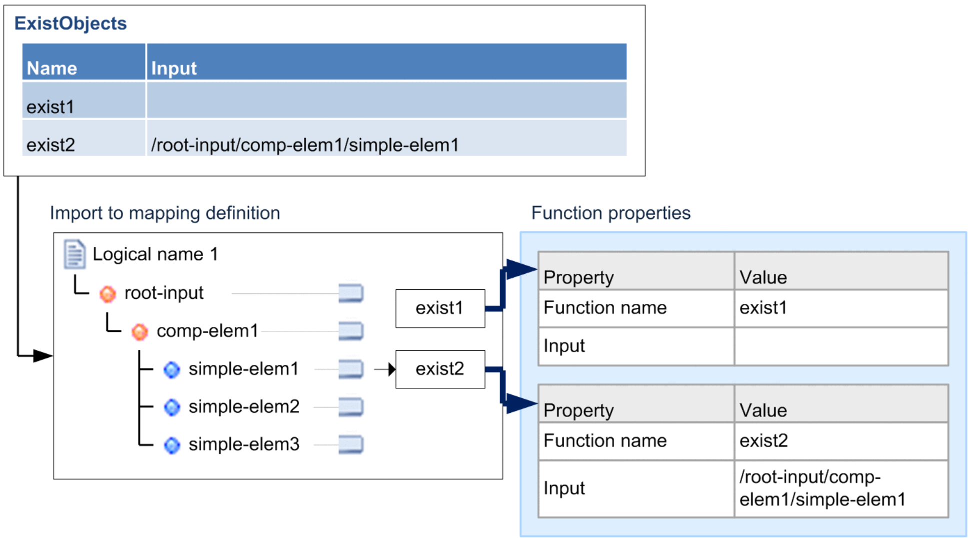
(Legend)

-: Corresponding item does not exist.

(2) Definition example

Following figure shows the definition example:

Figure 6–81: FigureExistObjectsDefinition example



6.13.18 LoopObjects(Repeat function)

This section describes Repeat function (loop).

(1) Definition item

Following table describes definition items:

Table 6–43: TableLoopObjectsDefinition item

Item name	Definition contents	Value that you can define	Additional description
Name	Function name	Value of NCName type	For details on the value that you can define, see "*6.13.24 Objects for which you can define the same Name element in multiple rows*".
Input	Input	See "*6.12.2 Setting up the mapping definition*"	You can define only one Input in the same Name.#1
RelationalPath	Linkage path	See "*6.12.2 Setting up the mapping definition*"	You can define multiple paths in the same Name. #1 Sets values in the list, in the defined order.
SortKey	Sort condition /key	See "*6.12.2 Setting up the mapping definition*"	You can specify multiple keys in the same Name.#2#3
SortOrder	Sort condition/ order	"ascending" or "descending"	You can specify multiple orders in the same Name.#2#3 In case of "ascending" For sort condition/order, set up "Ascending order". In case of "descending" For sort condition/order, set up "Descending order".
SortLanguage	Sort condition/ language	Either of "auto", "ja" or "en"	You can define multiple languages in the same Name.#2#3 In case of "auto" For sort condition/language, set "Auto".

Item name	Definition contents	Value that you can define	Additional description
SortLanguage	Sort condition/ language	Either of "auto", "ja" or "en"	In case of "ja" For sort condition/language, set "Japanese". In case of "en" For sort conditions/language, set "English".
SortDataType	Sort conditions/ data type	"text" or "numeric"	You can define multiple types in the same Name.#2#3 In case of "text" For sort condition/data type, set "Text". In case of "numeric" For sort condition/data type, set "Numeric".
SortCase	Sort condition/ priority order	"upper" or "lower"	This is mandatory item when SortDataType is "text". In other cases, setting value is not incorporated. You can specify multiple priority orders in the same Name.#2#3 In case of "upper" For sort condition/priority order, set "Upper case characters". In case of "lower" For sort condition/priority order, set "Lower case characters".

#1

For the Input and RelationPath defined in the same Name, you cannot set up nodes which are both same nodes or having relation as ancestor/grandchild.

#2

This is mandatory item when you have set either of the sort condition items (SortKey, SortOrder, SortLanguage, SortDataType, SortCase) in the same line.

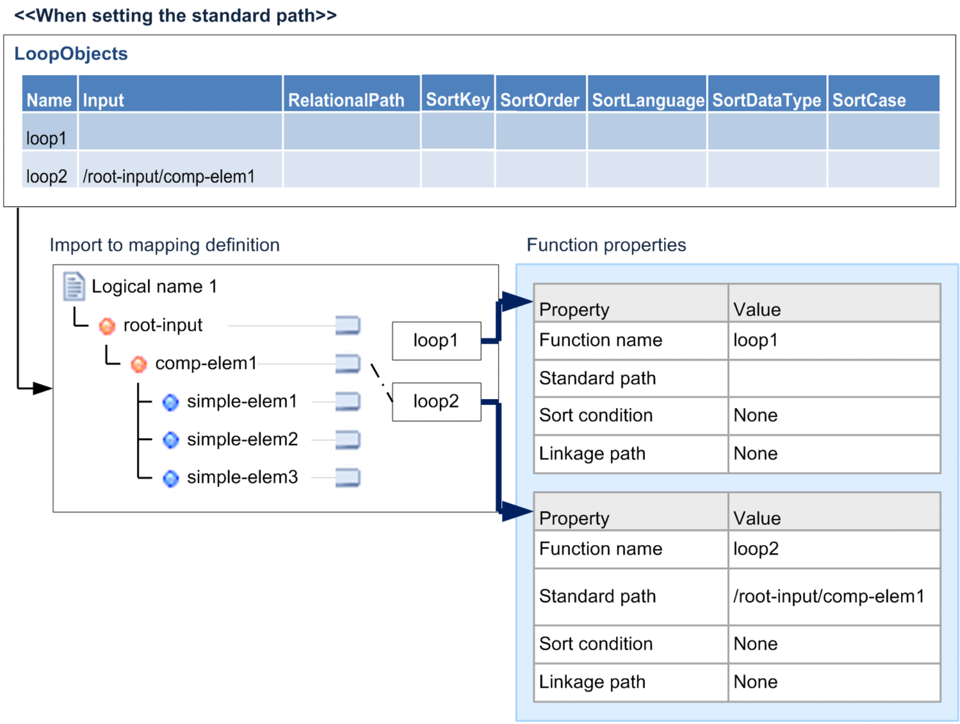
#3

Set up the sort conditions in the list, in defined order.

(2) Definition example

Following figure shows the definition example:

Figure 6–82: FigureLoopObjectsDefinition example1

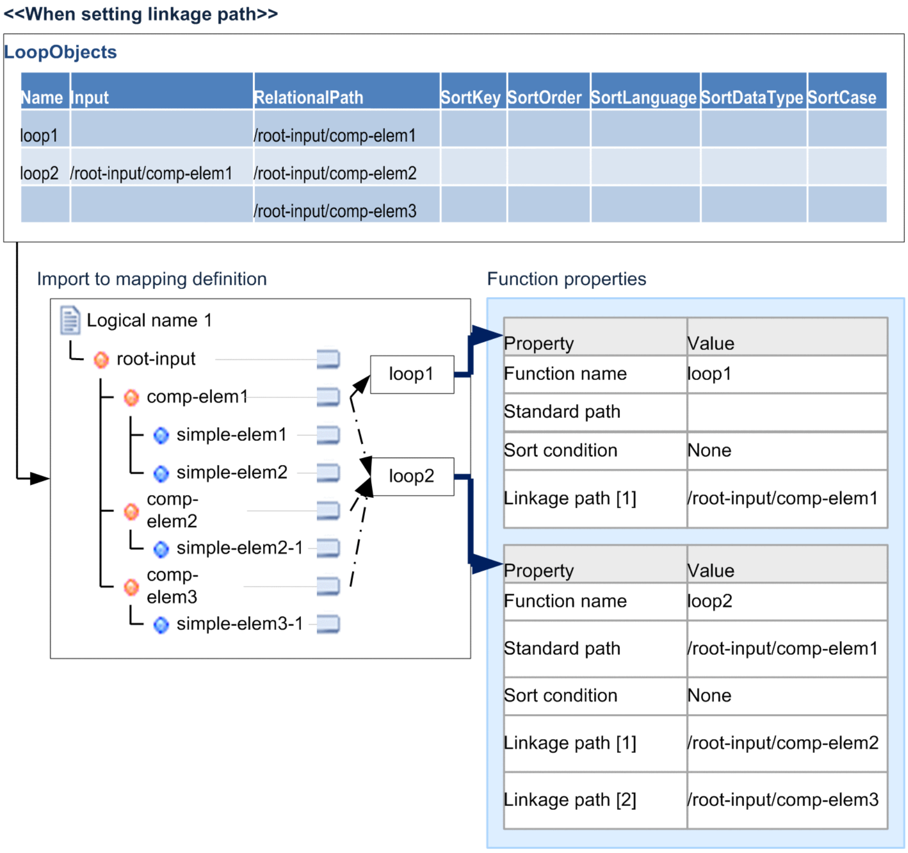


Name	Input	RelationalPath	SortKey	SortOrder	SortLanguage	SortDataType	SortCase
loop1							
loop2	/root-input/comp-elem1						

Property	Value
Function name	loop1
Standard path	
Sort condition	None
Linkage path	None

Property	Value
Function name	loop2
Standard path	/root-input/comp-elem1
Sort condition	None
Linkage path	None

Figure 6–83: FigureLoopObjectsDefinition example2

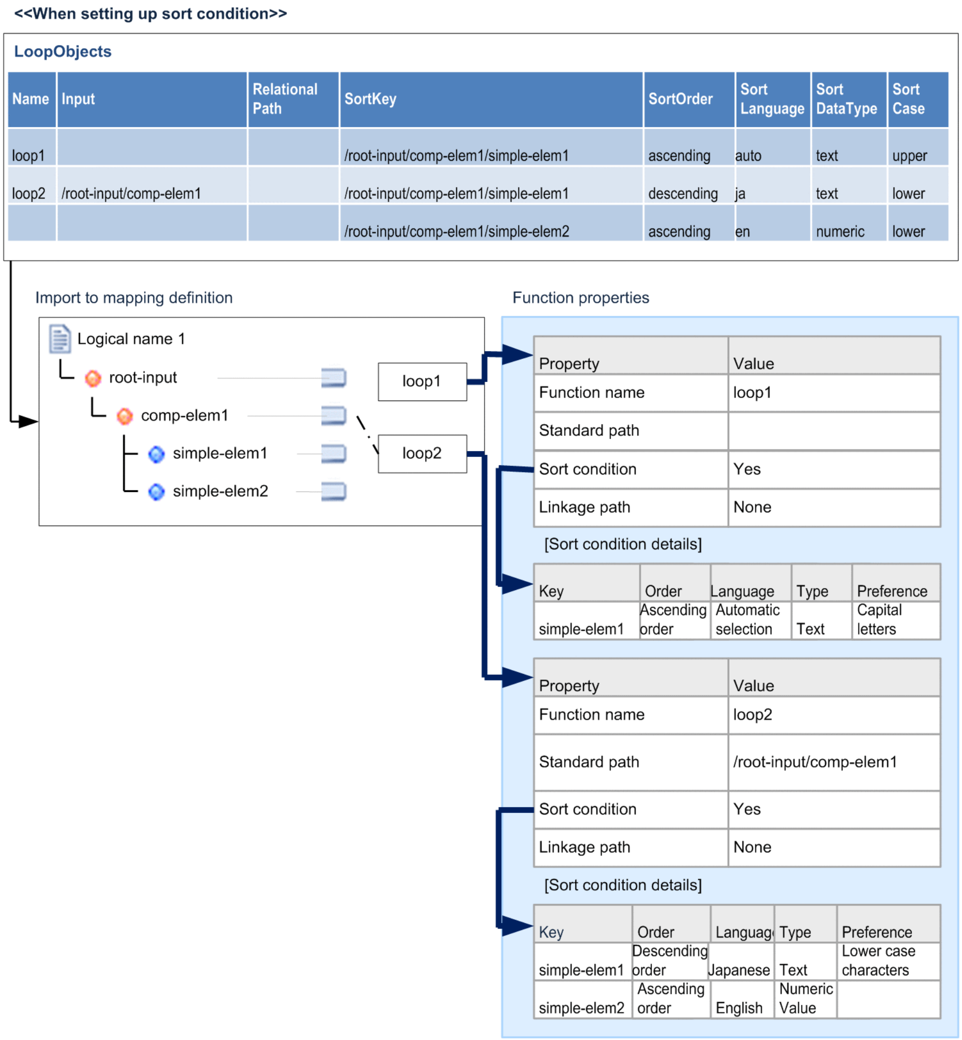


Name	Input	RelationalPath	SortKey	SortOrder	SortLanguage	SortDataType	SortCase
loop1		/root-input/comp-elem1					
loop2	/root-input/comp-elem1	/root-input/comp-elem2					
		/root-input/comp-elem3					

Property	Value
Function name	loop1
Standard path	
Sort condition	None
Linkage path [1]	/root-input/comp-elem1

Property	Value
Function name	loop2
Standard path	/root-input/comp-elem1
Sort condition	None
Linkage path [1]	/root-input/comp-elem2
Linkage path [2]	/root-input/comp-elem3

Figure 6–84: FigureLoopObjectsDefinition example3



<<When setting up sort condition>>

LoopObjects

Name	Input	Relational Path	SortKey	SortOrder	Sort Language	Sort DataType	Sort Case
loop1			/root-input/comp-elem1/simple-elem1	ascending	auto	text	upper
loop2	/root-input/comp-elem1		/root-input/comp-elem1/simple-elem1	descending	ja	text	lower
			/root-input/comp-elem1/simple-elem2	ascending	en	numeric	lower

Import to mapping definition

Function properties

Property	Value
Function name	loop1
Standard path	
Sort condition	Yes
Linkage path	None

[Sort condition details]

Key	Order	Language	Type	Preference
simple-elem1	Ascending order	Automatic selection	Text	Capital letters

Property	Value
Function name	loop2
Standard path	/root-input/comp-elem1
Sort condition	Yes
Linkage path	None

[Sort condition details]

Key	Order	Languag	Type	Preference
simple-elem1	Descending order	Japanese	Text	Lower case characters
simple-elem2	Ascending order	English	Numeric Value	

6.13.19 ChooseObjects(Select function)

This section defines the Select function (choose).

(1) Definition item

Following table describes definition items:

Table 6–44: TableChooseObjectsDefinition item

Item name	Definition contents	Value that you can define	Additional description
Name	Function name	Value of NCName type	For details on the value that you can define, see "*6.13.24 Objects for which you can define the same Name element in multiple rows*".
Condition	Conditions	See "*6.12.2 Setting up the mapping definition*"	You can define multiple conditions in the same Name.# When you want to do definition that does not match with any condition, specify "#other". In the same Name, you can specify "#other" only in one line. Also, when you do not specify even

Item name	Definition contents	Value that you can define	Additional description
Condition	Conditions	See "*6.12.2 Setting up the mapping definition*"	one "#other" in the same Name, output value of "When definition does not match with any condition" is assumed.
OutputValue	Output value	See "*6.12.2 Setting up the mapping definition*"	You can define multiple values in the same Name.# In case of "blank node" specify "#empty" and in case of "No output", specify "#notoutput".

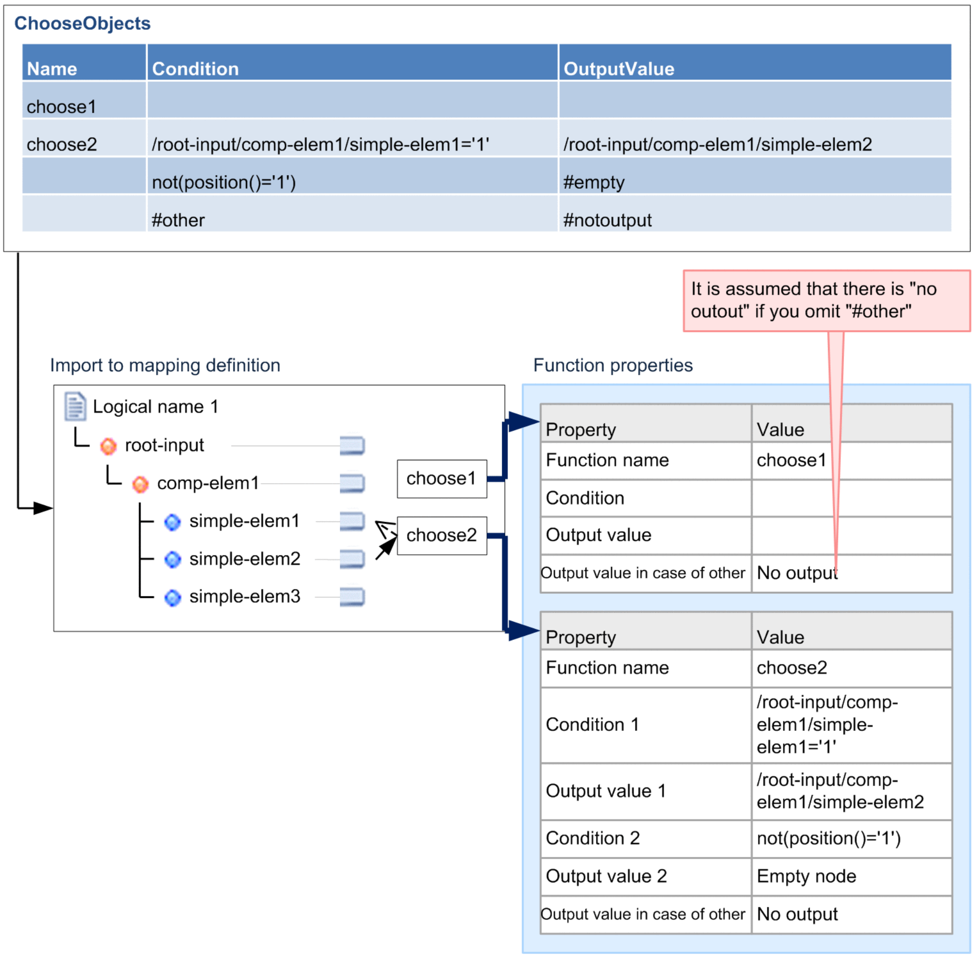
Note#

Set up the condition and output value in the list, in the defined order.

(2) Definition example

Following figure shows the definition example:

Figure 6–85: FigureChooseObjectsDefinition example



ChooseObjects

Name	Condition	OutputValue
choose1		
choose2	/root-input/comp-elem1/simple-elem1='1'	/root-input/comp-elem1/simple-elem2
	not(position()='1')	#empty
	#other	#notoutput

It is assumed that there is "no outout" if you omit "#other"

Property	Value
Function name	choose1
Condition	
Output value	
Output value in case of other	No output

Property	Value
Function name	choose2
Condition 1	/root-input/comp-elem1/simple-elem1='1'
Output value 1	/root-input/comp-elem1/simple-elem2
Condition 2	not(position()='1')
Output value 2	Empty node
Output value in case of other	No output

6.13.20 ReplaceObjects(Replace value function)

This section defines the Replace value function (replace).

(1) Definition item

Following table describes definition items:

Table 6–45: TableReplaceObjectsDefinition item

Item name	Definition contents	Value that you can define	Additional description
Name	Function name	Value of NCName type	This is mandatory item. You cannot define the Name in Objects, in duplication.
Input	Input	See "*6.12.2 Setting up the mapping definition*".	-
TableID	Transformation table ID	Character string having less than 1,024 characters (you cannot define linefeed code)	-
KeyColumn	Search key column specification	"left" or "right"	This is mandatory item. In case of "left" For search key column specification, set "Left column". In case of "right" For search key column specification, set "Right column".
FaultError	Operation in case of failure in search	"y" or "n"	This is mandatory item. In case of "y" For the operation in case of failure in search, set "Conversion error". In case of "n" For the operation in case of failure in search, set "Substitute default value".
DefaultValue	Value	Character string having less than 1,024 characters (You cannot define linefeed code)	When SortDataType is "y", the setting value is not incorporated.

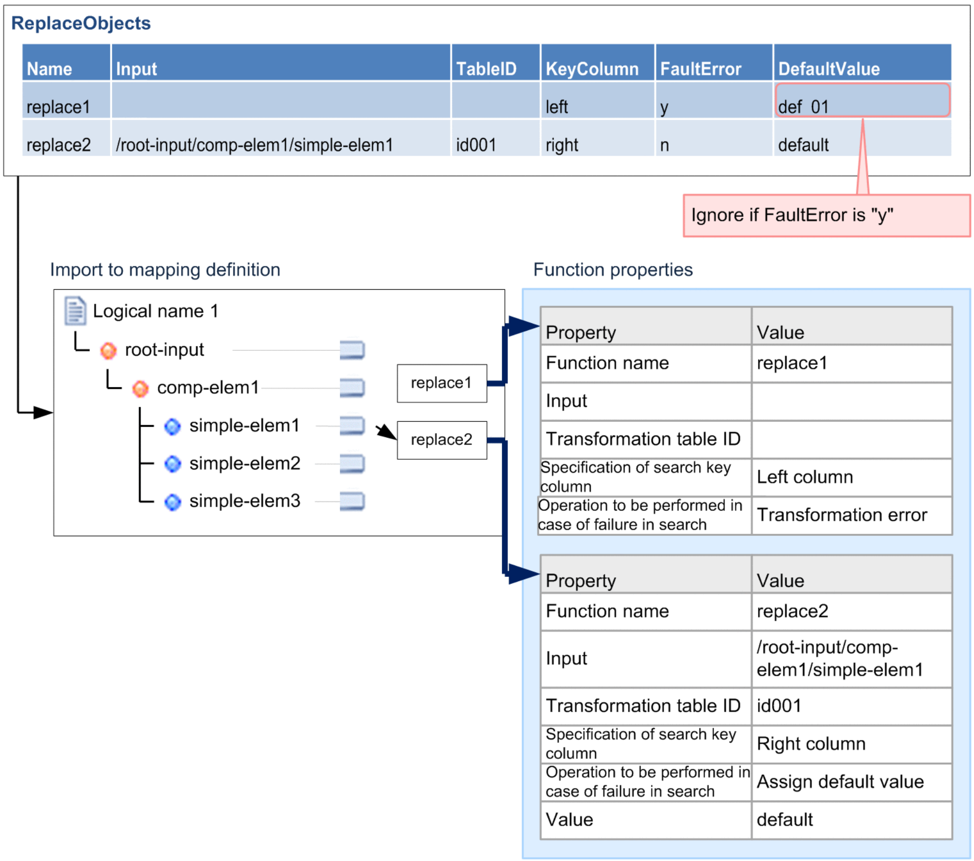
(Legend)

-: Corresponding item does not exist.

(2) Definition example

Following figure shows the definition example:

Figure 6–86: FigureReplaceObjectsDefinition example



ReplaceObjects

Name	Input	TableID	KeyColumn	FaultError	DefaultValue
replace1			left	y	def_01
replace2	/root-input/comp-elem1/simple-elem1	id001	right	n	default

Ignore if FaultError is "y"

Import to mapping definition

Function properties

Property	Value
Function name	replace1
Input	
Transformation table ID	
Specification of search key column	Left column
Operation to be performed in case of failure in search	Transformation error

Property	Value
Function name	replace2
Input	/root-input/comp-elem1/simple-elem1
Transformation table ID	id001
Specification of search key column	Right column
Operation to be performed in case of failure in search	Assign default value
Value	default

6.13.21 RadixObjects(radix conversion function)

This section defines the Radix conversion function (radix).

(1) Definition item

Following table describes definition items:

Table 6–46: TableRadixObjectsDefinition item

Item name	Definition contents	Value that you can define	Additional description
Name	Function name	Value of NCName type	This is mandatory item. You cannot define the Name in Objects, in duplication.
Input	Input	See "*6.12.2 Setting up the mapping definition*"	-
InputRadix	Input basic number	Either of "hexadecimal", "decimal" or "binary"	This is mandatory item. "hexadecimal" is hexadecimal. "decimal" is decimal. "binary" is binary number. You cannot define the same basic number in the input basic number and output basic number.
OutputRadix	output basic number	Either of "hexadecimal", "decimal" or "binary"	This is mandatory item. "hexadecimal" is hexadecimal.

Item name	Definition contents	Value that you can define	Additional description
OutputRadix	output basic number	Either of "hexadecimal", "decimal" or "binary"	"decimal" is decimal. "binary" is binary number. You cannot define the same basic number in input basic number and output basic number.

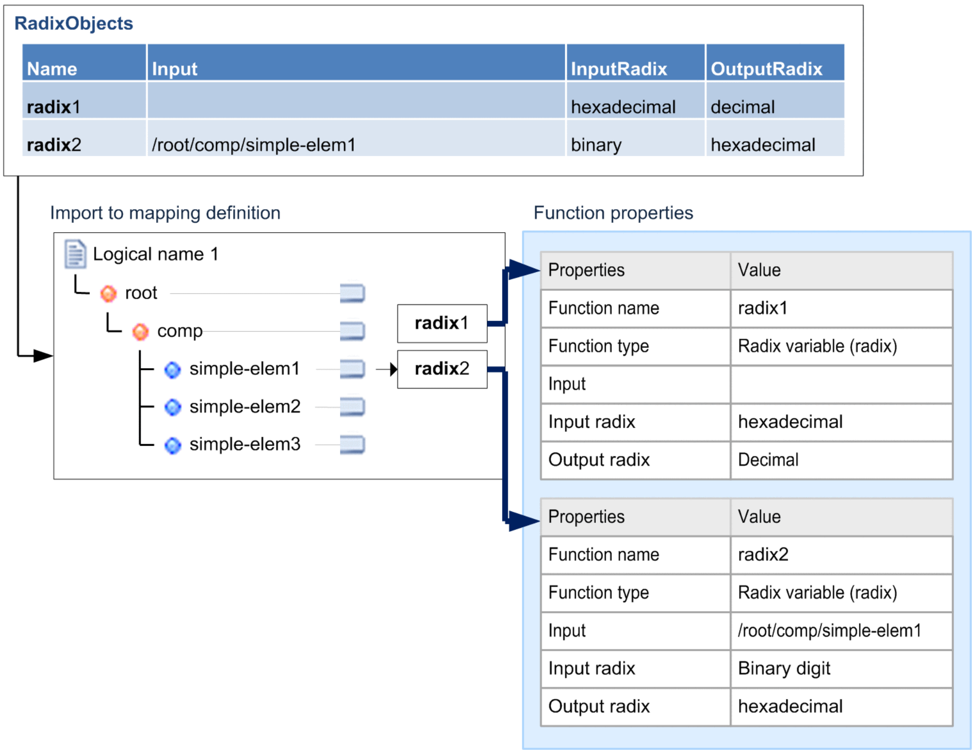
(Legend)

-: Corresponding item does not exist.

(2) Definition example

Following figure shows the definition example:

Figure 6–87: FigureRadixObjectsDefinition example



RadixObjects

Name	Input	InputRadix	OutputRadix
radix1		hexadecimal	decimal
radix2	/root/comp/simple-elem1	binary	hexadecimal

Properties	Value
Function name	radix1
Function type	Radix variable (radix)
Input	
Input radix	hexadecimal
Output radix	Decimal

Properties	Value
Function name	radix2
Function type	Radix variable (radix)
Input	/root/comp/simple-elem1
Input radix	Binary digit
Output radix	hexadecimal

6.13.22 CustomObjects(Custom function)

This section defines the Custom function (custom).

Comments exist in the class and arguments to be specified in Custom function. However, import of these items is not supported in the mapping by using Excel.

Define the package and class separately and set the value of "Package + class" defined in Excel, in the "Class" item on the screen of mapping definition editor.

(1) Definition item

Following table describes definition items:

Table 6–47: TableCustomObjectsDefinition item

Item name	Definition contents	Value that you can define	Additional description
Name	Function name	Value of NCName type	For details on the value that you can define, see "*6.13.24 Objects for which you can define the same Name element in multiple rows*".
Jar	JAR file	Value of NCName having less than 100 characters	You can define only one file in the same Name.#1
Package	Package	Value of NCName type having less than 255 characters	You can define only one package in the same Name.#1
Class	Class	Value of NCName type having less than 100 characters	You can define only one class in the same Name.#1
Method	Method	Value of NCName type having less than 100 characters	You can define only one method in the same Name.#1 Set only method name and do not add "()" at the end.
ReturnType	Return value type	Only "String" or "NodeList"	You can define only one return value type in the same Name.#1#2
Argument	Argument	Value of NCName type having less than 100 characters	You can define multiple arguments in the same Name.#3#4
ArgumentType	Argument type	Only "String" or "Object"	You can define multiple argument types in the same Name.#2#3#4
Input	Input value	See "*6.12.2 Setting up the mapping definition*"	You can specify multiple input values in the same Name. Set value in the list, in the defined order.

#1

When you specify transformation function definition item (Jar, Package, Class, Method, Return Type), always specify in first line in the same Name. Specify value same as first line, or blank in the lines other than first line in the same Name.

#2

When ReturnType is "String", you can define ArgumentType only as "String" and when the ReturnType is "NodeList", you can define ArgumentType only as "Object".

#3

This is mandatory item when you set even 1 argument item (Argument or ArgumentType) in the same line.

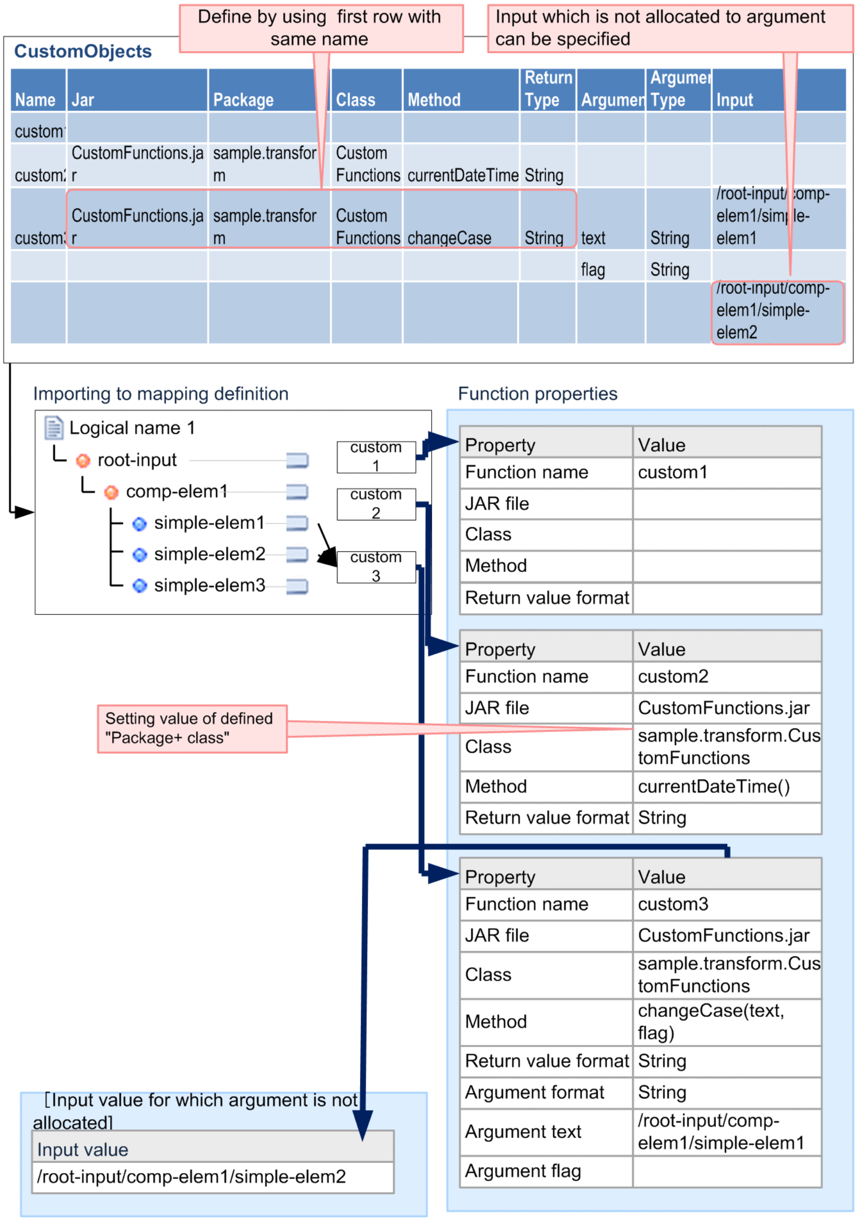
#4

Set argument in the list, in the defined order.

(2) Definition example

Following figure shows the definition example:

Figure 6–88: FigureCustomObjectsDefinition example



Define by using first row with same name

Input which is not allocated to argument can be specified

CustomObjects

Name	Jar	Package	Class	Method	Return Type	Argumen	Argumer Type	Input
custom1								
custom2	CustomFunctions.jar	sample.transform	Custom Functions	currentDateTime	String			
custom3	CustomFunctions.jar	sample.transform	Custom Functions	changeCase	String	text	String	/root-input/comp-elem1/simple-elem1
						flag	String	
								/root-input/comp-elem1/simple-elem2

Importing to mapping definition

Function properties

Property	Value
Function name	custom1
JAR file	
Class	
Method	
Return value format	

Property	Value
Function name	custom2
JAR file	CustomFunctions.jar
Class	sample.transform.CustomFunctions
Method	currentDateTime()
Return value format	String

Setting value of defined "Package+ class"

Property	Value
Function name	custom3
JAR file	CustomFunctions.jar
Class	sample.transform.CustomFunctions
Method	changeCase(text, flag)
Return value format	String
Argument format	String
Argument text	/root-input/comp-elem1/simple-elem1
Argument flag	

［Input value for which argument is not allocated］

Input value
/root-input/comp-elem1/simple-elem2

6.13.23 ConstantObjects(Set constant function)

Define the Set constant function (const).

(1) Definition item

Following table describes definition items:

Table 6–48: TableConstantObjectsDefinition item

Item name	Definition contents	Value that you can define	Additional description
Name	Function name	Value of NCName type	This is mandatory item. You cannot define the Name in Objects, in duplication.

Item name	Definition contents	Value that you can define	Additional description
Type	Type	Either of "string", "number", "boolean", "notoutput" or "empty"	This is mandatory item. In case of "string" For type, set "Character string". In case of "number" For type, set "Numeric value". In case of "boolean" For type, set "Logical value". In case of "notoutput" or "empty" For type, set "Characteristic node".
Value	Value	• When Type is "string", character string having less than 1,024 characters (you cannot define linefeed code) • When Type is "number", numeric value having less than 1,024 digits • When Type is "boolean", "true" or "false"#	This is mandatory item when Type is "boolean" or "number". When Type is "notoutput" or "empty", setting value is not incorporated. When Type is "notoutput", For value, set "Node is not output". When Type is "empty" For value, set "Empty node".

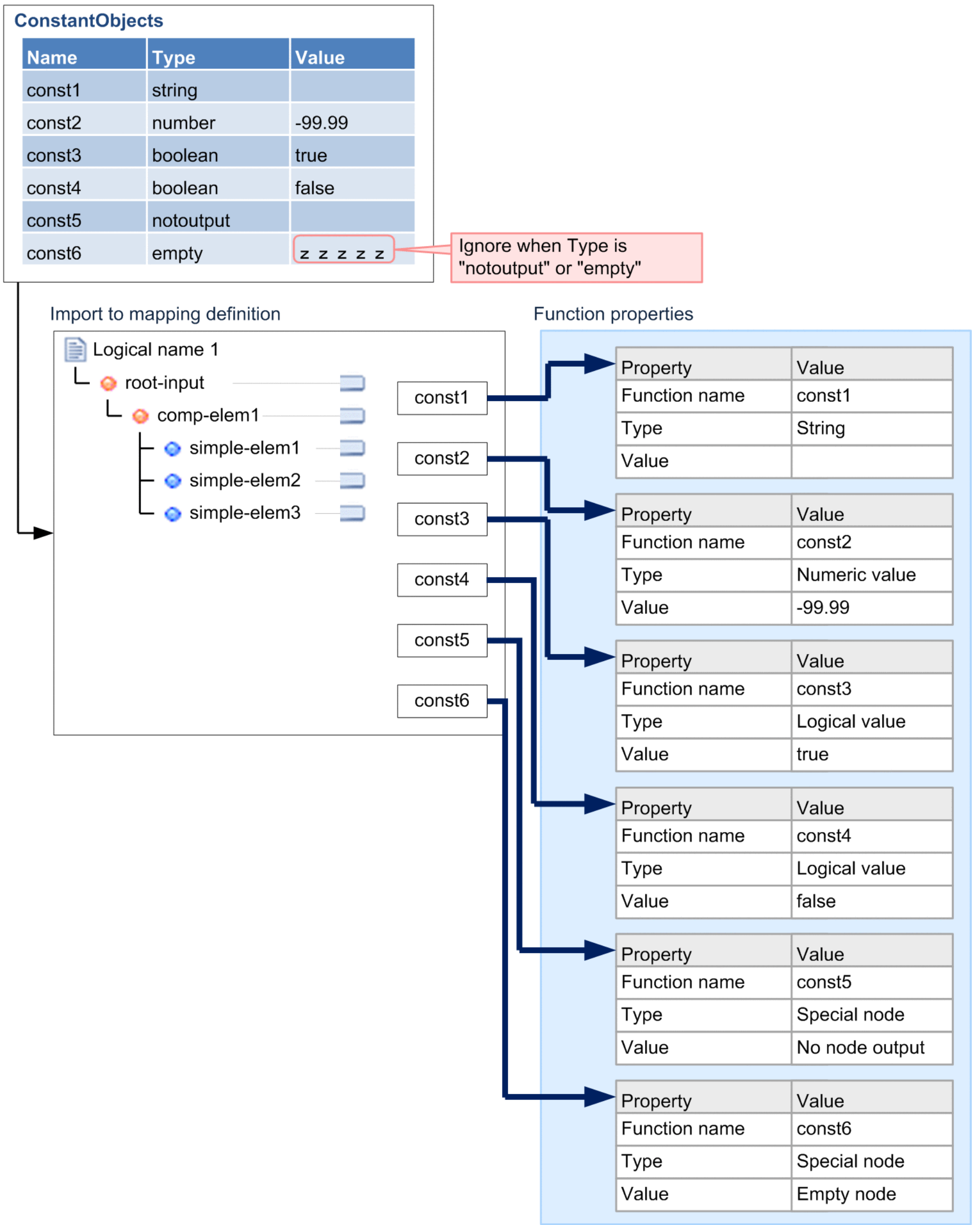
Note#

String is not case sensitive.

(2) Definition example

Following figure shows the definition example:

Figure 6–89: FigureConstantObjectsDefinition example



ConstantObjects

Name	Type	Value
const1	string	
const2	number	-99.99
const3	boolean	true
const4	boolean	false
const5	notoutput	
const6	empty	z z z z z

Ignore when Type is "notoutput" or "empty"

Property	Value
Function name	const1
Type	String
Value	

Property	Value
Function name	const2
Type	Numeric value
Value	-99.99

Property	Value
Function name	const3
Type	Logical value
Value	true

Property	Value
Function name	const4
Type	Logical value
Value	false

Property	Value
Function name	const5
Type	Special node
Value	No node output

Property	Value
Function name	const6
Type	Special node
Value	Empty node

6.13.24 Objects for which you can define the same Name element in multiple rows

For some Objects, you can define same Name element in multiple rows. Row of the same Name element is considered as definition for same one object and you can use the same in definitions of multiple values. In this case, values are set in list, in the defined order.

Following table describes objects for which you can define the same Name element in multiple rows and the items for which you can specify multiple values in the concerned object.

Table 6–49: TableObjects for which you can define the same Name element in multiple rows

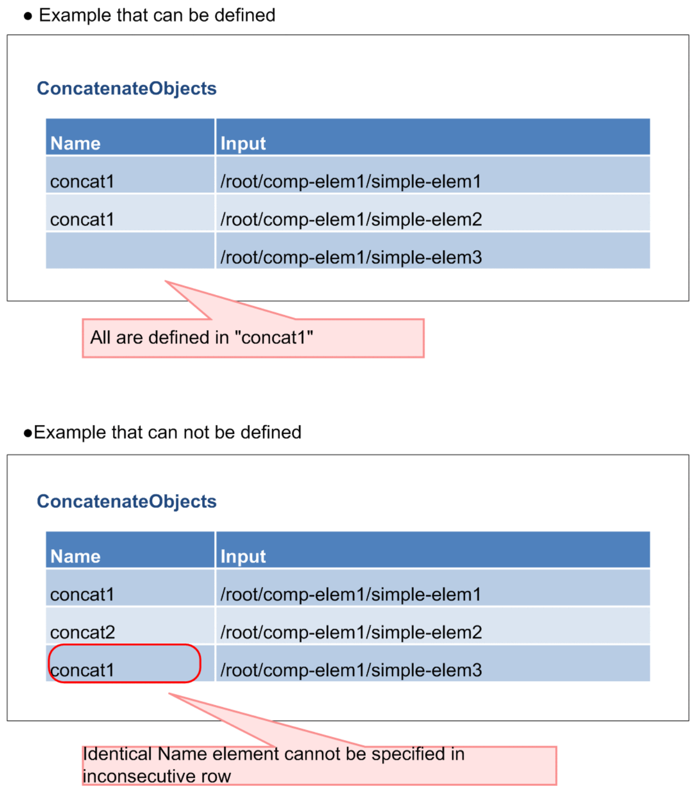
#	Objects	Items for which you can define multiple values
1	ConcatenateObjects	Input

#	Objects	Items for which you can define multiple values
2	SumObjects	Input
3	LoopObjects	RelationalPath, SortKey, SortOrder, SortLanguage, SortDataType, SortCase
4	ChooseObjects	Condition, OutputValue
5	CustomObjects	Argument, ArgumentType, Input

You can define the same Name element only in consecutive rows. If you define same Name element in non-consecutive rows, validation error occurs. Also, if the Name element has been omitted from second row onwards, value in the upper row is set.

Following figure shows the example of defining the same Name element in multiple rows.

Figure 6–90: FigureExample when setting the same Name element



● Example that can be defined

ConcatenateObjects

Name	Input
concat1	/root/comp-elem1/simple-elem1
concat1	/root/comp-elem1/simple-elem2
	/root/comp-elem1/simple-elem3

All are defined in "concat1"

●Example that can not be defined

ConcatenateObjects

Name	Input
concat1	/root/comp-elem1/simple-elem1
concat2	/root/comp-elem1/simple-elem2
concat1	/root/comp-elem1/simple-elem3

Identical Name element cannot be specified in inconsecutive row

6.14 Namespace prefix option

You can set whether to add namespace prefix, depending on the namespace prefix option. Set from the Transformation source viewer, Transformation destination Viewer and Mapping viewer of the Mapping definition editor. Display Context menu and activate or inactivate the "Output the namespace prefix to XML" option.

When following points are applicable, namespace prefix is added according to XSLT specifications, even in case of settings of not to output the namespace prefix in XML:

- When input XML has any prefix in the mapping of element or anyAttribute attribute
- When prefix is explicitly or implicitly added with Custom function
- When attribute has namespace

When you open the mapping definition file of old version, "Output namespace prefix to XML" has been set to active status.

6.14.1 Setting up default value of namespace prefix option

You can set up the default value of namespace prefix option, by setting up a property in eclipse.ini. Active/inactive status of "Output namespace prefix to XML" option at the time of defining a new mapping definition is determined by this default value. clipse.ini is stored in the following directory:

```
<Eclipse installation directory>\eclipse
```

Following table describes the setting contents of properties:

Table 6–50: TableDefault value settings of namespace prefix option

Property name	Property value#	Description
cscte.dt.default.output.namespace.prefix	true	Default value is the active status of "Output namespace prefix to XML" settings
	false	Default value is the inactive status of "Output namespace prefix to XML" settings

#

Do not specify property values by applying case sensitivity.

When you do not specify property values and invalid value is set, process is performed by considering the value as true.

Example of setting the property value of eclipse.ini is as follows:

```
-Dcscte.dt.default.output.namespace.prefix=false
```

7 Packaging HCSC Components and Defining Deployment

This chapter explains packaging of HCSC components and deployment definition.

7.1 Packaging and Defining Deployment

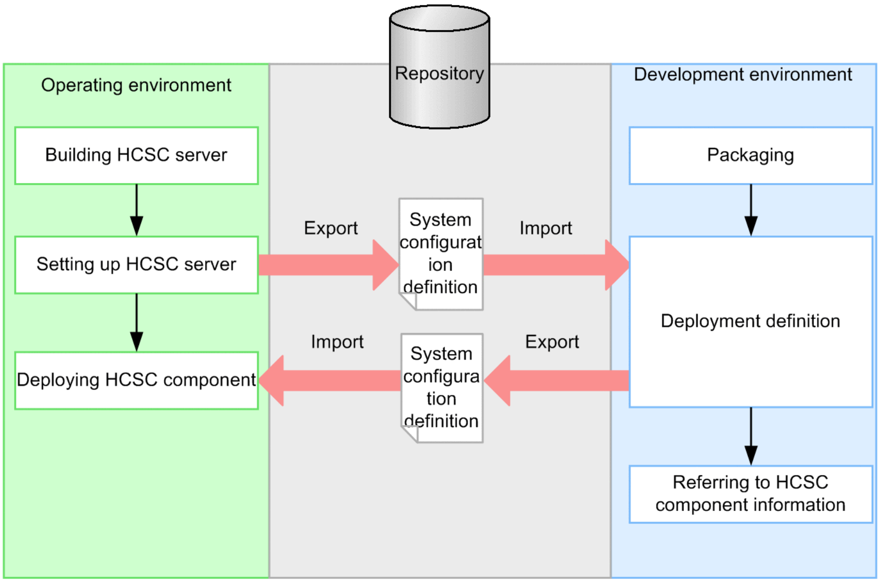
After you have defined a HCSC component, you package it or determine the cluster (or single HCSC server) in which to deploy it (deployment definition).

Before defining deployment, you must first import a repository. After deployment is defined, you export the repository to provide the system configuration definition containing deployment definition to the operating environment. The HCSC component is deployed from the operating environment to the cluster (or single HCSC server) in the execution environment according to the contents of the system configuration definition.

In the development environment, after the deployment is defined, you can reference the information about the HCSC component whose deployment was defined.

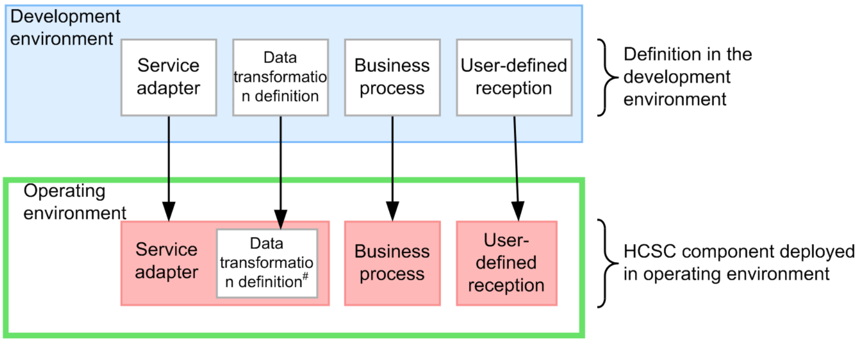
The following figure shows the flow from packaging to deployment definition.

Figure 7–1: Procedure from packaging to deployment definition



The contents defined in the development environment are deployed as HCSC components in the operating environment. The following figure shows the relationship between the definitions in the development environment and the HCSC components deployed in the operating environment:

Figure 7–2: Relationship between the definitions in the development environment and the HCSC components deployed in the operating environment


Development environment
Service adapter
Data transformation definition
Business process
User-defined reception
Definition in the development environment
Operating environment
Service adapter
Data transformation definition#
Business process
User-defined reception
HCSC component deployed in operating environment

Legend:

🟥 : HCSC component

#
Data transformation definition is not an HCSC component.
In operating environment, since data transformation definition is included in service adapter, the data transformation definition cannot be seen from operating environment.

Note that the packaging and deployment definition of the HCSC components is performed in the development environment, and the HCSC components are deployed and started in the operating environment. In the development environment, you can also execute this series of processes in a batch. In the development environment, the tasks executed individually in the development environment or the operating environment are executed in a batch, and therefore the operation load can be reduced. However, you can perform the batch execution when developing a system or during the unit testing and the integration testing. For details, see *7.5 Batch execution of processes for deploying HCSC components on the HCSC Server and then starting* and *7.6 Batch execution of processes for stopping HCSC components and deleting them from the HCSC server*.

! Important note

- When you import only partial information from a repository, you always need to package the HCSC components included in the imported repository information. For details about importing only partial information from a repository, see *3.2.3 Importing a Repository*.
- Do not implement operations such as packaging on HCSCTE, while the workspace is being built or cleaned.
- When you change the operation name of the service, perform re-packaging of the business process that calls the concerned service.

7.2 Packaging

An EAR file is created by assembling the files related to HCSC components that need to be deployed in the execution environment. This process is called *packaging*. Here, service adapters and business processes are collectively referred to as *HCSC components*.

Packaging HCSC components also packages the data transformation definitions and user-defined reception interfaces related to these HCSC components.

When packaging is performed, the HCSC components are validated. HCSC components can be packaged if they have been defined correctly, but if they have not been defined correctly, the validation process generates an error during packaging. Therefore, you can ensure smooth packaging if you validate the HCSC components before performing the packaging.

To package HCSC components:

1. In the service definition list in the tree view, right-click the HCSC component to be packaged, and then select **Packaging** or **Package multiple services**.
    - If **Packaging** is selected:
      A message dialog box appears.
    - If **Package multiple services** is selected:
      The Package multiple services dialog box appears. Select the check boxes for the services to be packaged, and then click **OK**.
2. Perform one of the following operations:
    - If packaging is successful
      Click **OK**.
      If an EAR file already exists, an overwrite confirmation dialog box appears. To overwrite the file, click **Yes**.
    - If packaging fails
      Take action according to the message displayed in the dialog box, and perform the packaging again.

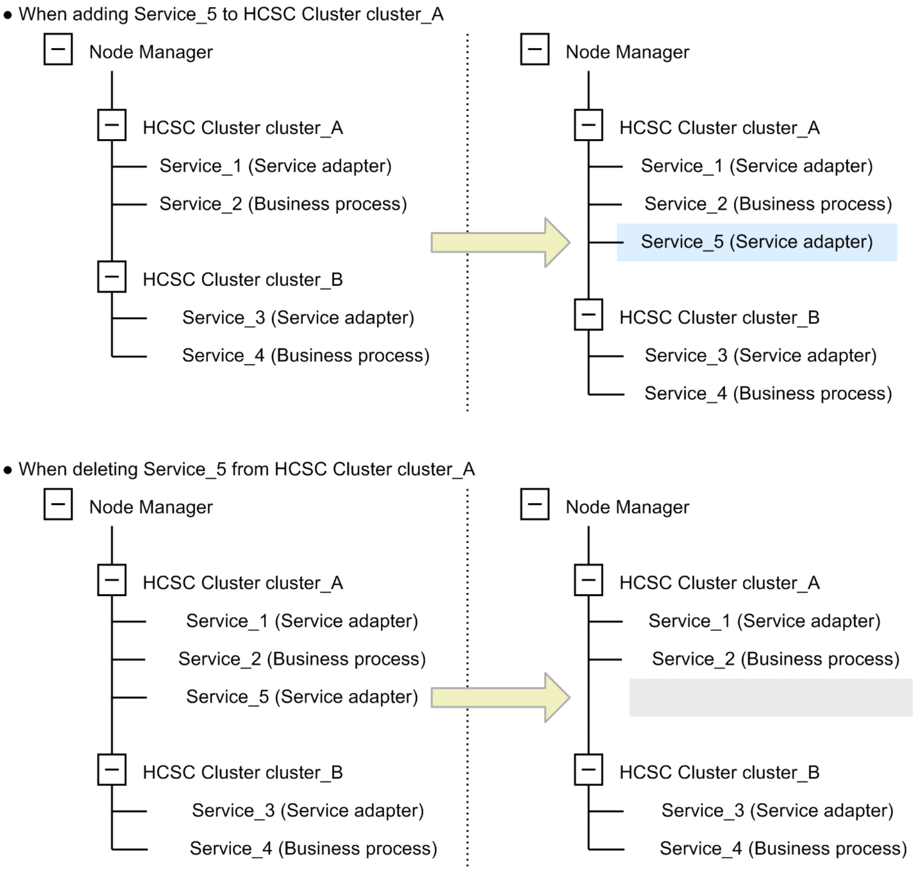
If the service adapter, business process, and user-defined reception interface have not been saved before packaging, the Save Resource dialog box appears so that you can save these definitions. If multiple service adapters, business processes, and user-defined reception interfaces are being edited, this dialog box appears more than once. To save a definition, click **OK**. To save all the definitions without displaying the subsequent confirmation dialog boxes, click **Yes to all**.

7.3 Defining Deployment of HCSC Components

Deployment definition means determining the cluster (or single HCSC server) in which to deploy the defined HCSC component.

In the system configuration definition list displayed in the tree view, a HCSC component displayed at a lower order than a cluster (or single HCSC server) is defined to be deployed in that cluster (or single HCSC server). You define deployment by adding (or deleting) the HCSC components to be deployed to (or from) a lower order than the cluster (or single HCSC server) in the system configuration definition list in the tree view. The following figure shows an example of adding and deleting HCSC components. The following figure shows an example of adding and deleting HCSC components:

Figure 7–3: Adding and deleting HCSC components


● When adding Service_5 to HCSC Cluster cluster_A
Node Manager
HCSC Cluster cluster_A
Service_1 (Service adapter)
Service_2 (Business process)
HCSC Cluster cluster_B
Service_3 (Service adapter)
Service_4 (Business process)
Node Manager
HCSC Cluster cluster_A
Service_1 (Service adapter)
Service_2 (Business process)
Service_5 (Service adapter)
HCSC Cluster cluster_B
Service_3 (Service adapter)
Service_4 (Business process)
● When deleting Service_5 from HCSC Cluster cluster_A
Node Manager
HCSC Cluster cluster_A
Service_1 (Service adapter)
Service_2 (Business process)
Service_5 (Service adapter)
HCSC Cluster cluster_B
Service_3 (Service adapter)
Service_4 (Business process)
Node Manager
HCSC Cluster cluster_A
Service_1 (Service adapter)
Service_2 (Business process)
HCSC Cluster cluster_B
Service_3 (Service adapter)
Service_4 (Business process)

With their deployment defined in the development environment, HCSC components are deployed to the cluster (or single HCSC server) in the operating environment according to the contents of the updated system configuration definition. For details about how to deploy HCSC components to a cluster (or single HCSC server), see the sections related to the deployment of service adapters and business processes in the manual *Cosminexus Service Platform System Setup and Operation Guide*.

The following subsections explain how to define deployment by adding (or deleting) HCSC components to (or from) a cluster (or single HCSC server).

Note that in some cases, you might not be able to define deployment depending on the combination of the server configuration that is defined in the system configuration definition and services. The following table describes whether the deployment can be defined based on the combination of the server configuration that is defined in the system configuration definition and services:

Table 7–1: Deployment can be defined or not

Item No.	Server configuration defined in the system configuration definition		Service type	Deployment can be defined or not
	Database usage	Cosminexus RM usage		
1	Used	Used	All	Y
2	Not used	Not used	Business processes (to be made persistent)	--
3			Business processes (not to be made persistent)	Y
4			SOAP adapter	Y
5			SessionBean adapter	Y
6			MDB (WS-R) adapter	--
7			MDB (database queue) adapter	--
8			DB adapter	Y
9			TP1 adapter	Y
10			File adapter	Y
11			Object Access adapter	Y
12			Message Queue adapter	Y
13			FTP adapter	Y
14			File operations adapter	Y
15			Mail adapter	Y
16			HTTP adapter	Y
17			Custom adapter	Y
18	Used	Not used	Business processes (to be made persistent)	--
19			Business processes (not to be made persistent)	Y
20			SOAP adapter	Y
21			SessionBean adapter	Y
22			MDB (WS-R) adapter	--
23			MDB (database queue) adapter	--
24			DB adapter	Y
25			TP1 adapter	Y
26			File adapter	Y
27			Object Access adapter	Y
28			Message Queue adapter	Y
29			FTP adapter	Y
30			File operations adapter	Y
31			Mail adapter	Y
32			HTTP adapter	Y

Item No.	Server configuration defined in the system configuration definition		Service type	Deployment can be defined or not
	Database usage	Cosminexus RM usage		
33	Used	Not used	Custom adapter	Y

Legend:

Y: Deployment can be defined.

--: Deployment cannot be defined..

7.3.1 Clusters (or Single HCSC Servers) to Which HCSC Components Can Be Deployed

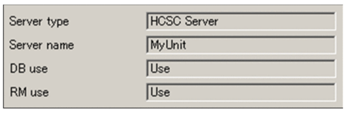
The clusters (or single HCSC servers) to which HCSC components can be deployed are displayed in the system configuration definition list in the tree view. The displayed information is the configuration information of the clusters (or single HCSC servers) that were set up in the operating environment. You acquire this information by importing into the development environment the repository information that was exported from the operating environment.

For details on importing the repository, see *3.2.3 Importing a Repository*.

7.3.2 Adding HCSC Components to a Cluster

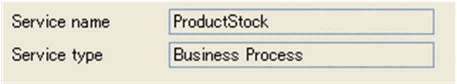
This subsection explains how to add HCSC components to a cluster or single HCSC server. Note that if you add an HCSC component to a cluster or single HCSC server, the data transformation definition and user-defined reception interface related to the added HCSC component are also added.

1. From the Eclipse menu bar, select **HCSC-Definer** and then **System Configuration List**.
   A list of clusters or single HCSC servers appears in the system configuration definition list in the tree view.
2. From the list of clusters or single HCSC servers, select and double-click the cluster or single HCSC server to which an HCSC component is to be added.
   Cluster information is displayed.



Server type: HCSC Server
Server name: MyUnit
DB use: Use
RM use: Use

3. Right-click the cluster or single HCSC server selected in step 2, and then select **Add Service** or **Add multiple services**.
   - If **Add Service** is selected
     The Add Service dialog box appears.
     If you edit a predefined HCSC component after it has been packaged, a dialog box is displayed to confirm whether to add the HCSC component to the cluster. To add the HCSC component, click **Yes**.
   - If **Add multiple services** is selected
     The Add multiple services dialog box appears.
4. Select the services to be added.
   - If **Add Service** is selected
     In the **Service to add** drop-down list box, select the HCSC component to be added.
   - If **Add multiple services** is selected
     Select the check boxes for the services to be added.
5. Click **OK**.
   Information about the added HCSC components is displayed.



Service name	ProductStock
Service type	Business Process

7.3.3 Deleting HCSC Components from a Cluster

This subsection explains how to delete HCSC components from a cluster or single HCSC server. Note that if you delete an HCSC component from a cluster or single HCSC server, the data transformation definition and user-defined reception interface related to the deleted HCSC component are also deleted.

1. From the system configuration definition list in the tree view, select the HCSC component to be deleted.
2. Use one of the following deletion methods:
   - Right-click the HCSC component to be deleted, and then select **Delete Service** or **Delete multiple services**.
   - Press the **Delete** key.

   If **Delete multiple services** is selected, the Delete multiple services dialog box appears. Select the check boxes for the services to be deleted, and then click **OK**.
   A dialog box is displayed to confirm whether to delete the selected services. To delete the services, click **Yes**.

7.4 Referencing HCSC Component Information

After you have defined deployment, you can reference the information about the HCSC component whose deployment has been defined. The information about the HCSC component is displayed in the HCSC Component Information Display screen. For details about this screen, see the manual *Cosminexus Service Platform Overview*.

Furthermore, you can also reference the user-defined reception information included in the HCSC components. The user-defined reception information is displayed in the User-defined Reception Information Display screen. For details about this screen, see the manual *Cosminexus Service Platform Overview*.

7.4.1 HCSC Component Information That Can Be Referenced

The following types of HCSC component information can be referenced:

(1) Service component information

The following information can be referenced as the service component information:

- Interface information
  Operation information, request message, response message, and fault message that correspond to the combination of a service name (HCSC component name) and an operation can be referenced.
- Server information
  Destination addresses by service component type of a cluster (or single HCSC server) that can invoke services can be referenced.
- Operation information
  The operation name and communication model specified in the interface information can be referenced.
- Request message
  Message format and message format ID can be referenced.
- Response message
  Message format and message format ID can be referenced.
- Fault message
  Fault name and message format can be referenced.

(2) Information of a user-defined reception

The following information can be referenced as the information of a user-defined reception:

- Information of a user-defined reception
  The reception name, port name, and URL information of a user-defined reception can be referenced.

7.4.2 Displaying HCSC Component Information

To display HCSC component information:

1. From the Eclipse menu, choose HCSC-Definer, and then Published Services List.
   HCSC components are listed in Published Services List in the tree view.
2. From the HCSC component list, select and double-click the HCSC component whose information is to be displayed.
   The information about the HCSC component is displayed in the HCSC Component Information Display screen.
3. Perform the following operations:
   To reference the information in Operation information, Request message, Response message, or Fault message
   In the Operation drop-down list in Interface information, select the operation whose information is to be referenced.

To reference the content of a message, click the **Display** button for the message (request message, response message, or fault message) to be referenced.

To reference the destination addresses by service type of a cluster (or single HCSC server)

The cluster name whose address is to be referenced is displayed in Server name in Server information.

To acquire a WSDL file to be used by Web Services, click **WSDL acquisition**.

To acquire a stub file to be used by SessionBean, click **Stub acquisition**.

To reference the information of a user-defined reception

Click the **User-defined Reception Information** tab at the bottom of the HCSC component Information Display screen, for displaying the User-defined Reception Information Display screen.

7.4.3 Updating the HCSC Component List

When a HCSC component is modified, the information inside the repository may not match the information displayed in the system configuration definition list in the tree view. In such a case, you can update the content displayed in the system configuration definition list in the tree view.

To update the HCSC component list:

1. Right-click the system configuration definition list in the tree view without selecting the HCSC component displayed, and choose Update.

   The HCSC component list displayed in the system configuration definition list in the tree view is updated.

! Important note

The configuration information of the clusters (or single HCSC servers) displayed in the system configuration definition list in the tree view is the configuration information of the cluster (or single HCSC server) that was set up in the operating environment.

To display the latest server configuration list in the system configuration definition list in the tree view, import the repository information exported in the operating environment into the development environment as needed. When the repository is imported, the display is automatically updated.

For details on importing the repository, see *3.2.3 Importing a Repository*.

7.5 Batch execution of processes for deploying HCSC components on the HCSC Server and then starting

In the development environment, you can execute the series of processes for deploying the HCSC components on the HCSC server and then starting them, in a batch. In the development environment, the tasks executed individually in the development environment or the operating environment are executed in a batch, and therefore the operation load can be reduced. Note that you can perform the batch execution when developing a system or during the unit testing and the integration testing.

The batch execution of the processes for deploying HCSC components on the HCSC server and then starting HCSC components is performed with the following two methods:

- Deploying the selected HCSC components on the HCSC server and then starting them.
- Deploying all the HCSC components (defined in the development environment) on the HCSC server and then starting them.

This section describes the flow of processes from deploying to starting the HCSC components, and also describes how to perform each operation.

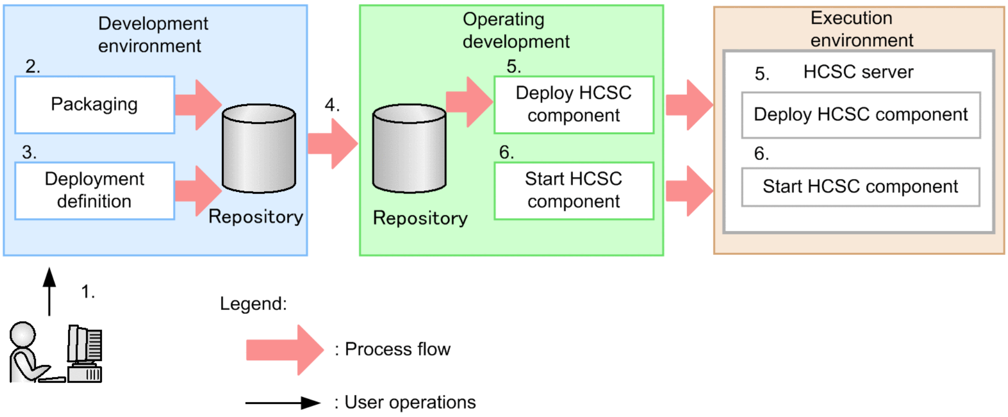
7.5.1 Flow of processes from deploying to starting HCSC components

The following are the flow of the series of processes from deploying HCSC components on the HCSC server until starting them, the range of the HCSC components to be operated and the differences in processing according to the type of HCSC components:

(1) Flow of processes

The following figure shows the flow of the processes from deploying HCSC components on the HCSC server until starting them. Note that if you perform operations in the development environment, steps 2 to 6 of the figure will execute automatically.

Figure 7–4: Flow of processes from deploying to starting HCSC components



1. Perform batch execution of operations in the development environment.
2. In the development environment, the HCSC components are packaged and saved in a repository.
3. Deployment is defined, and the system configuration definition within the repository is updated.
4. The repository of the development environment is transferred to the operating environment.
5. Based on the system configuration definition updated in the development environment, HCSC components are deployed from the operating environment onto the HCSC server of the execution environment.
6. The HCSC components deployed on the HCSC server of the execution environment are started.

(2) HCSC components to be operated

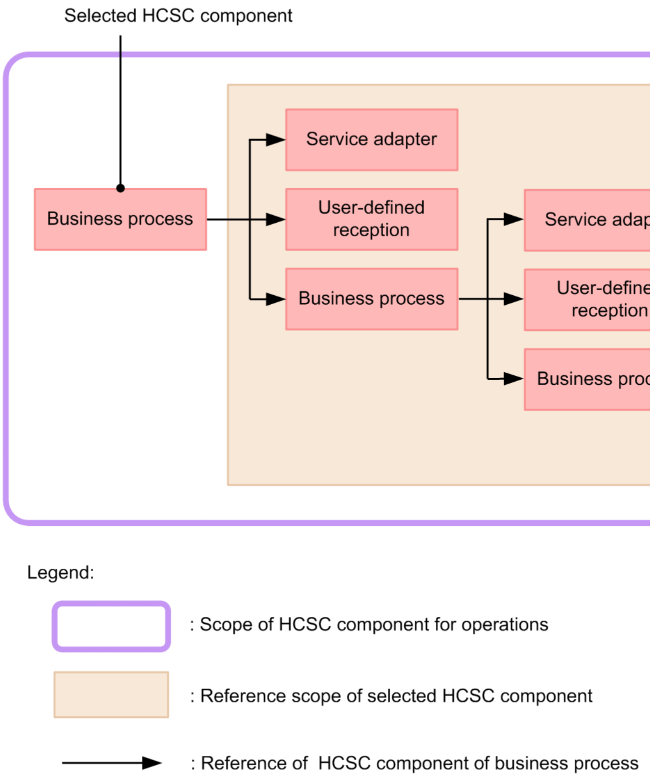
The HCSC components to be operated when only the selected HCSC components are deployed on the HCSC server and started are different from the HCSC components to be operated when all the HCSC components are deployed on the HCSC server and started. HCSC components to be operated imply the HCSC components for which the series of processes from deployment to startup is to be executed.

Deploying the selected HCSC components on the HCSC server and starting them

The HCSC components selected in the development environment are to be operated. If the selected HCSC component is a business process, the HCSC components (business processes, service adapters, and user-defined receptions) defined in that business process will also be operated.

The following figure shows the range of HCSC components to be operated when a business process is selected:

Figure 7–5: Range of HCSC components to be operated (When a business process is selected)



Here, *Referencing HCSC Components* indicates the use of other HCSC components depending on the addition of a user-defined reception and specification in the invoke service activity.

! Important note

Because a user-defined reception is processed simultaneously with the business process in the development environment, a user-defined reception cannot be operated alone.

Deploying all the HCSC components in the HCSC server and starting them

All the HCSC components defined in the development environment are to be operated.

(3) Differences in processing according to the type of HCSC components

The processing, when the HCSC component to be operated is a service adapter, is different from the processing when the HCSC component to be operated is a business process.

For a service adapter

Packaging and deployment definition are not performed again for a service adapter if the service adapter is already packaged and the deployment is defined (`public`).

For a business process

Because the Java classes might have been changed, packaging and deployment definition are performed again.

7.5.2 How to deploy HCSC components in the HCSC server and start them

This subsection describes how to deploy HCSC components in the HCSC server, and then start them.

(1) Prerequisites

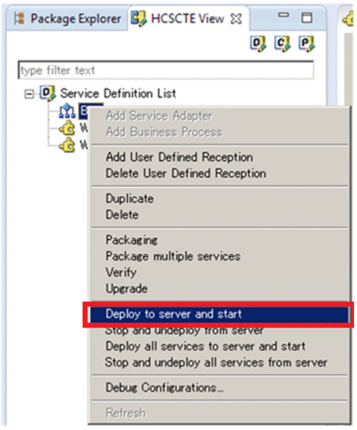
To deploy the HCSC components in the HCSC server and then start them, you must fulfill the following conditions:

- Perform the operation only during the system development or from the time of performing unit testing to the integration testing.
- Build the development environment, operating environment, and execution environment on the same machine. If these environments are set up on multiple machines, the operation cannot be guaranteed.
- The setup of the HCSC server must be complete. Set up only a single HCSC server. Even when you use the HCSC Easy Setup functionality, set up only a single HCSC server.
- The HCSC server and database must be running.

(2) Operation procedure

(a) Deploying the selected HCSC components on the HCSC server and starting them

1. From the service definition list in the tree view, select the HCSC components to be deployed on the HCSC server.
2. Right-click the selected HCSC components, and choose **Deploy on the server and start**.


Package Explorer
HCSCTE View
type filter text
Service Definition List
Add Service Adapter
Add Business Process
Add User Defined Reception
Delete User Defined Reception
Duplicate
Delete
Packaging
Package multiple services
Verify
Upgrade
Deploy to server and start
Stop and undeploy from server
Deploy all services to server and start
Stop and undeploy all services from server
Debug Configurations...
Refresh

The displayed dialog box and the procedure thereafter will differ depending on whether the HCSC components to be operated are being edited.

When the HCSC components to be operated are being edited

A dialog box confirming whether to save the HCSC components being edited (private) will be displayed. If multiple services and user-defined receptions are being edited, multiple dialog boxes will be displayed. Click the **Yes** button to save. Click the **Yes to all** button to save thereafter without displaying the confirmation dialog box. If you do not want to save HCSC components, click the **Cancel** button and end the process.

Proceed to step 3.

When HCSC components to be operated are not being edited

When batch execution is first performed after starting Eclipse, the **Account Authentication** dialog box is displayed. When the user ID and password are entered, and then the **OK** button is clicked, the processing will start. When batch execution is performed for the second time and thereafter after starting Eclipse, the processing will start immediately.

Proceed to step 4.

3. Click the **Yes** button or click the **Yes to all** button.

When batch execution is first performed after starting Eclipse, the **Account Authentication** dialog box is displayed. When the user ID and password are entered and then the **OK** button is clicked, the processing will start. When batch execution is performed for the second time and thereafter after starting Eclipse, the processing will start immediately.

The processing will differ as follows depending on whether deployment is defined (`public`) for the HCSC components being edited or deployment is not defined (`private`):

When deployment is defined for the HCSC components

Set HCSC components for which deployment is defined to a state in which deployment is not defined. After that, save the edited HCSC components and start the processing.

When deployment is not defined for HCSC components

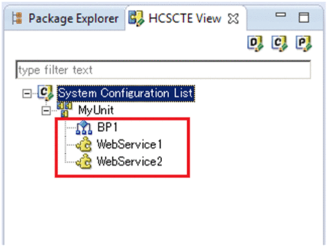
Save the HCSC components being edited and start the processing.

4. When a dialog box notifying the completion of processing appears, click the OK button.

5. Check the following as needed:

Check if deployment is defined

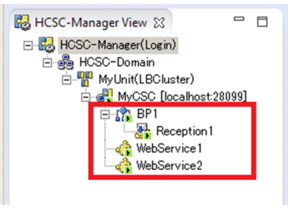
In the system configuration definition list of the tree view, make sure that the selected HCSC components are added.



In the above example, you can confirm if the selected business process (BP1) and the service adapter (WebService1) defined in the selected business process are added.

Check if the HCSC components are deployed on the HCSC server and started

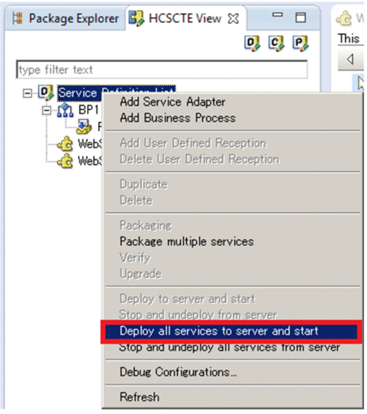
Log into the HCSC-Manager, and select the HCSC-Manager view. Make sure that the selected HCSC components are deployed on the HCSC server and started. For details about how to log in to the HCSC-Manager, see the manual *Cosminexus Service Platform System Setup and Operation Guide*.



In the above example, you can confirm if the selected business process (BP1) and the service adapter (WebService1) and user-defined reception (Reception1) defined in the selected business process are added.

(b) Deploying all HCSC components on the HCSC server and starting them

1. Right-click the service definition list in the tree view, and choose **Deploy all the services on the server and start**.


Package Explorer
HCSCTE View
type filter text
Service
BP1
Add Service Adapter
Add Business Process
Add User Defined Reception
Delete User Defined Reception
Duplicate
Delete
Packaging
Package multiple services
Verify
Upgrade
Deploy to server and start
Stop and undeploy from server
Deploy all services to server and start
Stop and undeploy all services from server
Debug Configurations...
Refresh

When the HCSC components to be operated are being edited

A dialog box, confirming whether to save the HCSC components being edited (private), will be displayed. If multiple services and user-defined receptions are being edited, multiple dialog boxes will be displayed. Click the **Yes** button to save. Click the **Yes to all** button to save thereafter without displaying the confirmation dialog box. If you do not want to save HCSC components, click the **Cancel** button and end the process.

Proceed to step 3.

When the HCSC components to be operated are not being edited

When batch execution is first performed after starting Eclipse, the **Account Authentication** dialog box is displayed. When the user ID and password are entered and then the **OK** button is clicked, the processing will start. After starting Eclipse, when batch execution is performed for the second time and thereafter, the processing will start immediately.

Proceed to step 4.

2. Click the **Yes** button or click the **Yes to all** button.

When batch execution is first performed after starting Eclipse, the **Account Authentication** dialog box is displayed. When the user ID and password are entered and then the **OK** button is clicked, the processing will start. After starting Eclipse, when batch execution is performed for the second time and thereafter, the processing will start immediately.

The processing will differ as follows depending on whether deployment is defined (`public`) for the HCSC components being edited or deployment is not defined (`private`):

When deployment is defined for the HCSC components

Set the HCSC components for which deployment is defined to a state in which deployment is not defined. After that, save the edited HCSC components and start the processing.

When deployment is not defined for the HCSC components

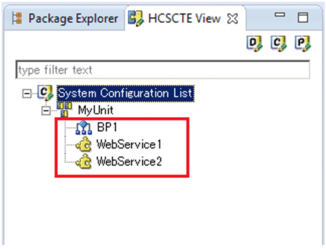
Save the HCSC components being edited and start the processing.

3. When a dialog box notifying the completion of processing appears, click the OK button.

4. Check the following as needed:

Check if deployment is defined

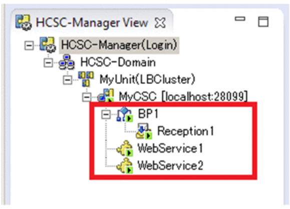
In the system configuration definition list of the tree view, make sure that all the HCSC components defined in the development environment are added.



In the above example, you can confirm that all the HCSC components (BP1, WebService1, and WebService2) defined in the development environment are added.

Check if the HCSC components are deployed on the HCSC server and started

Log into the HCSC-Manager, and select the HCSC-Manager view. Make sure that all the HCSC components defined in the development environment are deployed on the HCSC server and started. For details about how to log in to the HCSC-Manager, see the manual *Cosminexus Service Platform System Setup and Operation Guide*.



In the above example, you can confirm if all the HCSC components (BP1, WebService1, WebService2, and Reception1) defined in the development environment are added.

(3) Precautions

The following are the precautions to be taken when deploying the HCSC components on the HCSC server and then starting them:

- When deploying the selected HCSC components on the HCSC server and then starting them, if the configuration information of the HCSC server specified in the repository information of the operating environment and the development environment is wrong, this will result in an error in the following cases. In such case, deploy all the defined HCSC components on the HCSC server and then start them.

  A configuration format of the repository (combination of database and Cosminexus RM) do not match in the development and operating environment.

  The service deployed in the repository of the operating environment is updated or deleted in the development environment.

  A SOAP mode in the repository of the development and operating environment do not match.

  If an error occurs, deploy all the defined HCSC components on the HCSC server and start them.
- If an error occurs during the processing, interrupt the processing and terminate the batch execution. However, the processes, for which the batch execution had finished before the occurrence of the error, will not return to the status prior to the execution. In such a case, remove the cause of the error, and re-execute the processing.
- When batch execution is to be performed for multiple HCSC components after revising them, deploy all the HCSC components on the HCSC server and start them. However, if the revised HCSC components are included in the reference range of *7.5.1(2) HCSC components to be operated*, the selected HCSC components can be deployed on the HCSC server and started.
- If a batch execution of the process for deploying and starting the HCSC components on the HCSC server fails, you might not be able to start the service and re-execute until the HCSC component stops. If the process fails, stop the HCSC components by executing in the order of the `csccompostop -all` and `csccompoundeploy -all` command and undeploy them. After this, redeploy and restart the HCSC components on the HCSC server. Note that you cannot stop HCSC components by the method described in *7.6 Batch execution of processes for stopping HCSC components and deleting them from the HCSC server*.

  For details about the `csccompostop` and `csccompoundeploy` command, see the manual *Cosminexus Service Platform Reference*.

7.6 Batch execution of processes for stopping HCSC components and deleting them from the HCSC server

In the development environment, you can execute the series of processes for stopping HCSC components, and then delete them from the HCSC server, in a batch. In the development environment, the tasks executed individually in the development environment or the operating environment are executed in a batch, and therefore the operation load can be reduced. Note you can perform the batch execution when developing a system or during the unit testing and the integration testing.

The batch execution of the processes, for stopping HCSC components and deleting them from the HCSC server, is performed with the following two methods:

- Stopping the selected HCSC components and deleting them from the HCSC server. The deployment definition of the HCSC components is deleted in the development environment.
- Stopping all the HCSC components deployed on the HCSC server, and then deleting them from the HCSC server. The deployment definition of all HCSC components is deleted in the development environment.

This section describes the flow of processes from stopping to deleting the HCSC components, and also describes how to perform each operation.

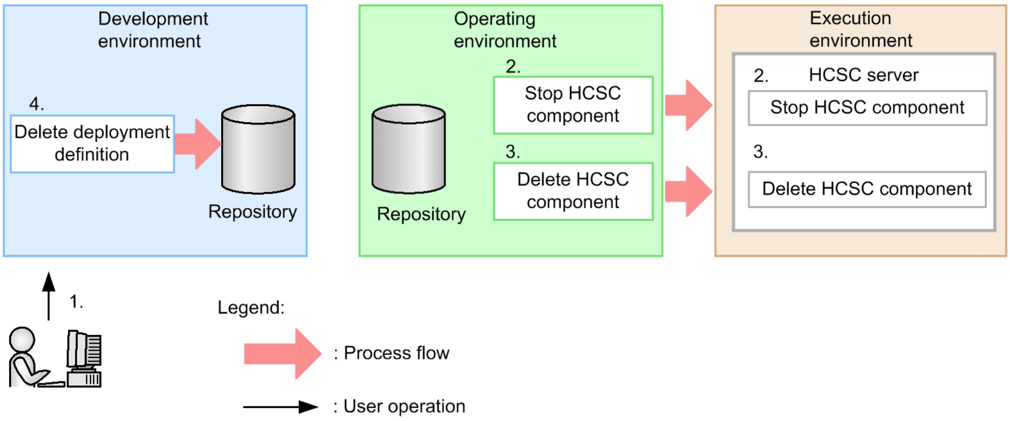
7.6.1 Flow of processes from stopping to deleting HCSC Components

The flow of the series of processes from stopping HCSC components to deleting them from the HCSC server and the range of the HCSC components to be operated are as follows:

(1) Flow of processes

The following figure shows the flow of the series of processes from stopping HCSC components to deleting them from the HCSC server. Note that if you perform operations in the development environment, steps 2 to 4 of the figure will execute automatically.

Figure 7–6: Flow of processes from stopping to deleting the HCSC components



1. Perform batch execution of operations in the development environment.
2. The HCSC components running on the HCSC server of the execution environment are stopped.
3. The HCSC components are deleted from the HCSC server of the execution environment.
4. The deployment definition of the HCSC components is deleted in the development environment, and the system configuration definition in the repository is updated.

(2) HCSC components to be operated

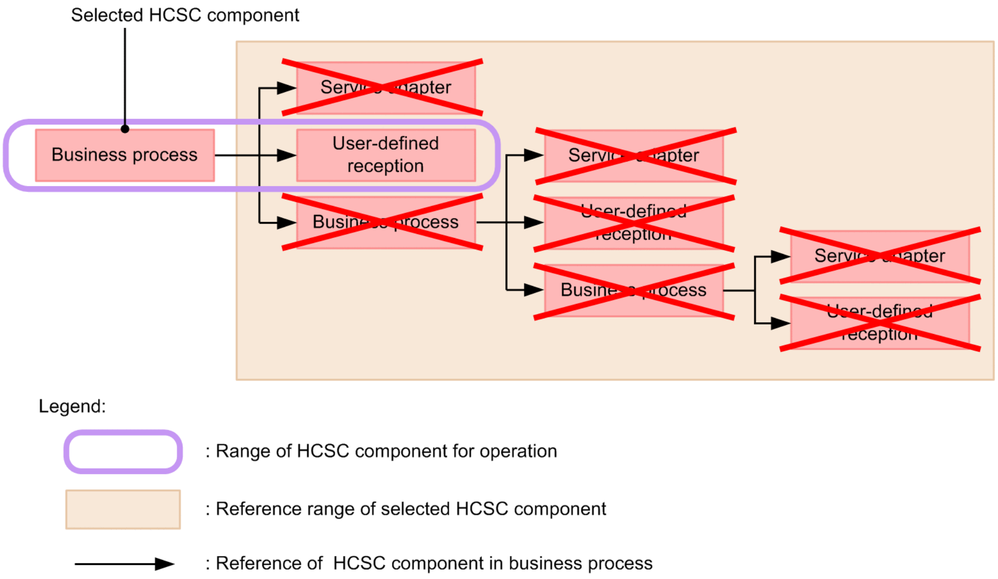
The HCSC components to be operated when only the selected HCSC components are stopped and deleted from the HCSC server are different from those to be operated when all HCSC components are stopped and deleted from the HCSC server. Here, the HCSC components to be operated imply those HCSC components for which the series of processes from stopping to deleting is to be executed.

Stopping the selected HCSC components and deleting them from the HCSC server

The HCSC components selected in the development environment are to be operated. If the selected HCSC component is a business process, the user-defined receptions defined in that business process will also be operated. The business processes and service adapters defined in the selected HCSC component will not be operated.

The following figure shows the range of HCSC components to be operated when a business process is selected:

Figure 7–7: Range of HCSC components to be operated (When a business process is selected)



Here, *Referencing HCSC Components* indicates the use of other HCSC components depending on the addition of a user-defined reception and specification in the invoke service activity.

Important note

Because a user-defined reception is processed simultaneously with the business process in the development environment, a user-defined reception cannot be operated alone.

Stopping all the HCSC components and deleting them from the HCSC server

All the HCSC components deployed on the HCSC server and all the HCSC components for which deployment is defined on the development environment are to be operated.

7.6.2 How to stop HCSC components and delete them from the HCSC server

This subsection describes how to stop HCSC components and delete them from the HCSC server.

(1) Prerequisites

To stop HCSC components and delete them from the HCSC server, the following conditions must be fulfilled:

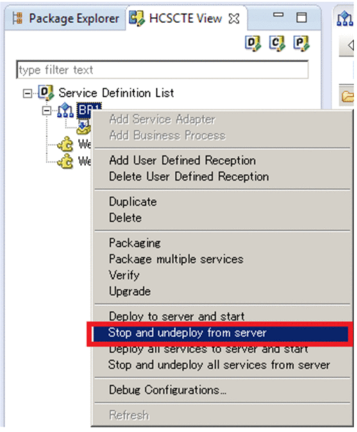
- Perform the operation only during the system development or from the time of performing unit testing to integration testing.

- Build the development environment, operating environment, and execution environment on the same machine. If these environments are set up in multiple machines, the operation cannot be guaranteed.
- The setup of the HCSC server must be complete. Set up only a single HCSC server. Even when you use the HCSC Easy Setup functionality, set up only a single HCSC server.
- The HCSC server and database must be running.

(2) Operation procedure

(a) Stopping the selected HCSC components and deleting them from the HCSC server

1. From the service definition list in the tree view, stop HCSC components and select the HCSC components to be deleted from the HCSC server.
2. Right-click the selected HCSC components, and choose **Stop and Delete from the Server**.

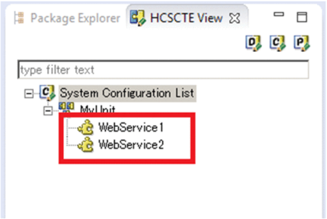

Package Explorer
HCSCTE View
type filter text
Service Definition List
BP1
We
We
Add Service Adapter
Add Business Process
Add User Defined Reception
Delete User Defined Reception
Duplicate
Delete
Packaging
Package multiple services
Verify
Upgrade
Deploy to server and start
Stop and undeploy from server
Deploy all services to server and start
Stop and undeploy all services from server
Debug Configurations...
Refresh

When the batch execution is first performed after starting Eclipse, the **Account Authentication** dialog box is displayed. When the user ID and password are entered and then the **OK** button is clicked, the processing will start. After starting Eclipse, when the batch execution is performed for the second time and thereafter, the processing will start immediately.

3. When a dialog box reporting the completion of processing appears, click the **OK** button.
4. Check the following as needed:

Check if the deployment definition has been deleted

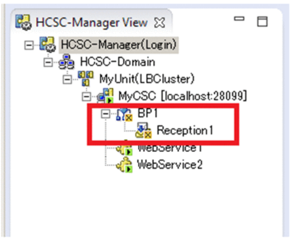
In the system configuration definition list of the tree view, make sure that the selected HCSC components have been deleted.



In the above example, you can confirm that the selected business process (BP1) has been deleted.

Check that the HCSC components are stopped and deleted from the HCSC server

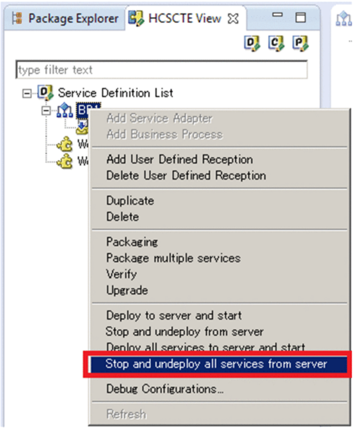
Log into the HCSC-Manager, and select the HCSC-Manager view. Make sure that the selected HCSC components are stopped, and then deleted from the HCSC server. For details about how to log in to the HCSC-Manager, see the manual *Cosminexus Service Platform System Setup and Operation Guide*.



In the above example, you can confirm that the selected business process (BP1) and the user-defined reception (Reception1) defined in the selected business process have been stopped, and then deleted.

(b) Stopping all the HCSC components and deleting them from the HCSC server

1. Right-click the service definition list in the tree view, and choose **Stop All Services and Delete from the Server**.

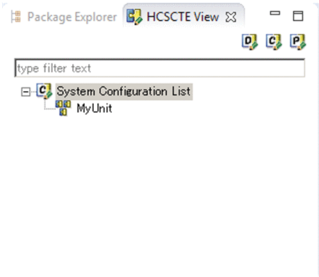

Package Explorer
HCSCTE View
type filter text
Service Definition List
BP1
Add Service Adapter
Add Business Process
Add User Defined Reception
Delete User Defined Reception
Duplicate
Delete
Packaging
Package multiple services
Verify
Upgrade
Deploy to server and start
Stop and undeploy from server
Deploy all services to server and start
Stop and undeploy all services from server
Debug Configurations...
Refresh

When the batch execution is first performed after starting Eclipse, the **Account Authentication** dialog box is displayed. When the user ID and password are entered and then the **OK** button is clicked, the processing will start. After starting Eclipse, when the batch execution is performed for the second time and thereafter, the processing will start immediately.

2. When a dialog box reporting the completion of processing appears, click the **OK** button.
3. Check the following as needed:

Check that the deployment definition has been deleted

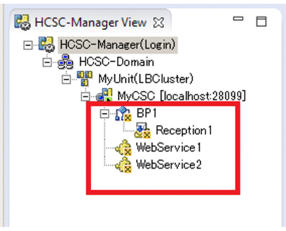
In the system configuration definition list of the tree view, make sure that all the HCSC components have been deleted.



In the above example, you can confirm that all the HCSC components (BP1, WebService1, and WebService2) have been deleted.

Check that the HCSC components are stopped and deleted from the HCSC server

Log into the HCSC-Manager, and select the HCSC-Manager view. Make sure that all the HCSC components are stopped, and then deleted from the HCSC server. For details about how to log in to the HCSC-Manager, see the manual *Cosminexus Service Platform System Setup and Operation Guide*.



In the above example, you can confirm that all the HCSC components (BP1, WebService1, WebService2, and Reception1) have been stopped and deleted.

(3) Precautions

The precautions to be taken when stopping HCSC components and deleting from the HCSC server are as follows:

- When stopping the selected HCSC components and deleting from the HCSC server, if the configuration information of the HCSC server specified in the repository information of the operating environment and the development environment is wrong, this will result in an error in the following cases. In such case, stop all the HCSC components, and delete them from the HCSC server.

  - A configuration format of the repository (combination of database and Cosminexus RM) do not match in the development and operating environment.

  - The service deployed in the repository of the operating environment is updated or deleted in the development environment.

  - A SOAP mode in the repository of the development and operating environment do not match.

  If an error occurs, stop all the HCSC components and delete them from the HCSC server.

- If an error occurs during the processing, interrupt the processing and terminate the batch execution. However, the processes, for which the batch execution had finished before the occurrence of the error, will not return to the status prior to the execution. In such a case, remove the cause of the error, and then re-execute the processing.

- If a batch execution of the process for stopping and deleting the HCSC components from the HCSC server fails, you might not be able to start the service and re-execute until the HCSC component stops. If the process fails, stop the HCSC components by executing in the order of the `csccompostop -all` and `csccompoundeploy -all` command and undeploy them. After this, delete the HCSC components from the HCSC server.

  For details about the `csccompostop` and `csccompoundeploy` command, see the manual *Cosminexus Service Platform Reference*.

8 Creating Service Requesters

This chapter explains how to create service requesters that send requests for service components to a standard reception and user-defined reception of an HCSC server.

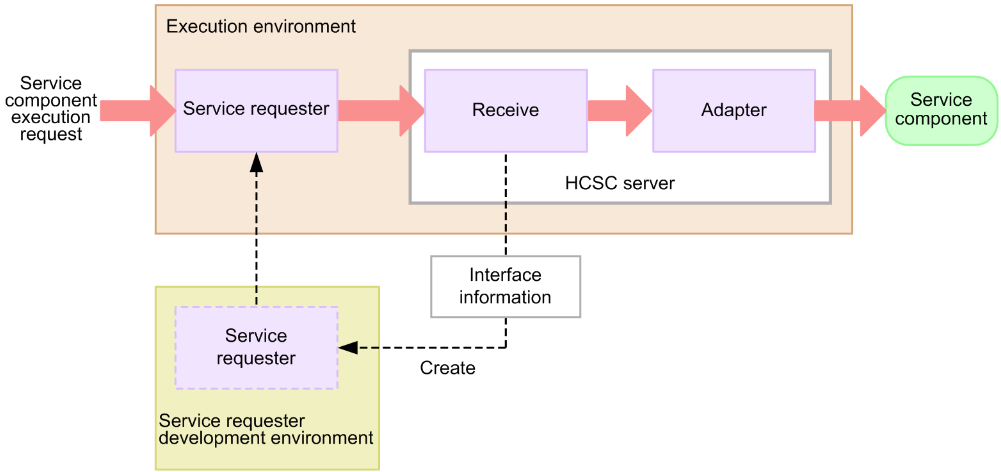
For details about how to emulate the service requester for testing and debugging, see *Appendix G. Emulating the Service Requester*.

8.1 Overview of Creating Service Requesters

A *service requester* is an application that receives a request from work in-charge and sends the request for service component execution to each HCSC component of adapters and business processes. You create a service requester in the development environment based on the interface information provided by the execution environment and deploy it in the execution environment. Development of service requesters must be carried out in an environment that enables Java program development.

The following figure provides an overview of service requesters.

Figure 8–1: Overview of service requesters



A service requester sends a service component execution request to standard reception and user-defined reception of the HCSC server. Note that the request destination differs depending on protocols used by the service requester. The following table describes the types of protocols that the service requester uses and types of standard and user-defined reception:

Table 8–1: Types of protocols used by a service requester and the types of standard and user-defined reception

Receive		Protocol
standard reception	synchronous reception (Web Services)	SOAP (HTTP)
	synchronous reception (SessionBean)	RMI-IIOP
	asynchronous reception (MDB (WS-R))	WS-R
	asynchronous reception (MDB (database queue))	• Protocol unique to Cosminexus RM • JMS
user-defined reception	synchronous reception (Web Services)	SOAP (HTTP)

! Important note

You must develop a service requester considering the procedure for processing during operations. Also, you must create the error handling with such a proper understanding so that a prompt action can be taken when an error occurs. For details, see the contents about invoking service components of an HCSC server in the manual *Cosminexus Service Platform Function Guide*, and the contents about troubleshooting in the manual *Cosminexus Service Platform System Setup and Operation Guide*.

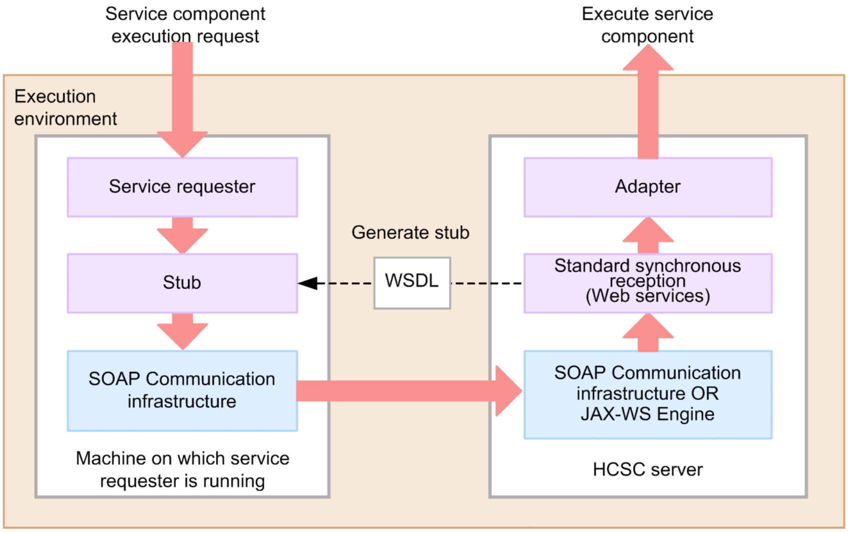
8.2 Service Requester That Sends Requests to a Standard Synchronous Reception (Web Services) (SOAP communication infrastructure)

A service requester that sends a request message to a standard synchronous reception (Web services) communicates with the standard reception using SOAP. The service requester sends a service component execution request message to a standard reception, and an HCSC server performs service component execution.

When SOAP is used, the interface information of the synchronous reception (Web Services) is acquired from the WSDL. A stub is generated from the acquired WSDL, and this stub is used for sending a request to the synchronous reception (Web Services). Therefore, the service requester must be installed such that it can utilize the generated stub.

The following figure shows the relationship between a service requester that sends requests to a standard synchronous reception (Web Services) and an HCSC server.

Figure 8–2: Relationship between a service requester that sends requests to a standard synchronous reception (Web Services) and an HCSC server (SOAP communication infrastructure)

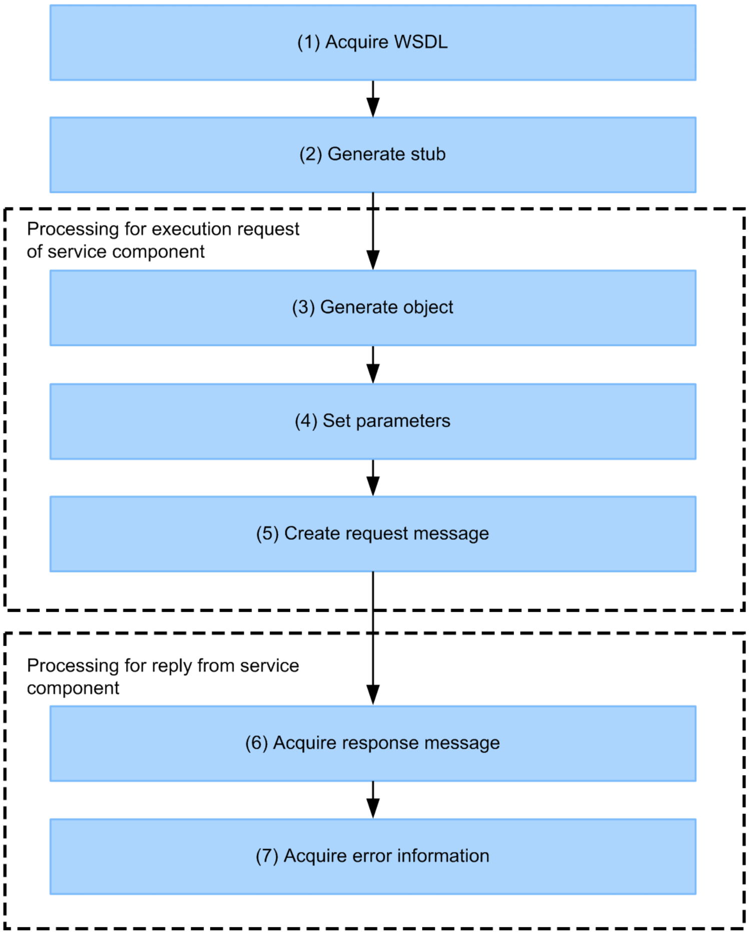

Service component execution request
Execute service component
Execution environment
Service requester
Stub
SOAP Communication infrastructure
Machine on which service requester is running
Generate stub
WSDL
Adapter
Standard synchronous reception (Web services)
SOAP Communication infrastructure OR JAX-WS Engine
HCSC server

To use the JAX-WS engine, a service class will be created instead of a stub. For details, see, *8.3 Creating a service requester using standard synchronous reception (Web Services) (JAX-WS engine)*.

8.2.1 Procedure for Creating a Service Requester (Standard Synchronous Reception (Web Services)) (SOAP communication infrastructure)

This subsection explains how to create a service requester that sends a service component execution request to a standard synchronous reception (Web Services) and invokes a service component. The creation workflow is shown in the following figure.

Figure 8–3: Workflow for creating a service requester (standard synchronous reception (Web Services)) (SOAP communication infrastructure)


(1) Acquire WSDL
(2) Generate stub
Processing for execution request of service component
(3) Generate object
(4) Set parameters
(5) Create request message
Processing for reply from service component
(6) Acquire response message
(7) Acquire error information

The tasks in the individual steps are described below.

(1) WSDL acquisition

Acquire the interface information of the synchronous reception (Web Services) of the HCSC server that invokes the service component from the WSDL. For details about WSDL acquisition, see *8.2.2 Acquiring the WSDL*.

(2) Stub creation

Create a stub from the WSDL acquired by *8.2.1(1) WSDL acquisition* (1). For details about stub creation, see *8.2.3 Creating Stubs*.

(3) Object generation

In order to invoke the method of the synchronous reception (Web Services), use the stubs created in *8.2.1(2) Stub creation* to generate objects. For details about object generation, see *8.2.4 Generating Objects*.

(4) Parameter specification

Specify the parameters that become the arguments of the method of the synchronous reception (Web Services). For details about parameter specification, see *8.2.5 Specifying Parameters*.

(5) Request message creation

Create a request message for requesting service component execution. For details about request message creation, see *8.2.6 Creating Request Messages*.

(6) Response message acquisition

Acquire a response message corresponding to the service component execution request from the synchronous reception (Web Services). For details about response message acquisition, see *8.2.7 Acquiring Response Messages*.

(7) Error information acquisition

If an error occurs at the request-destination service component, the HCSC server, or the SOAP engine, acquire the error information and take corrective action according to the information. For details about error information acquisition, see *8.2.8 Acquiring Error Information*.

8.2.2 Acquiring the WSDL

Acquire the interface information of the synchronous reception (Web Services) of the HCSC server that invokes the service component from the WSDL. You can use one of the following two WSDL acquisition methods:

Method 1

1. From the Eclipse menu, choose HCSC-Definer, and then Published Services List.

   HCSC components are listed in Published Services List in the tree view.
2. From the HCSC component list, select and double-click the service component (HCSC component) to be invoked.

   The information about the HCSC component is displayed in the HCSC Component Information Display screen. For details about this screen, see the manual *Cosminexus Service Platform Overview*.

   The information about the HCSC server that becomes the service component request destination is displayed in Server information.
3. Choose the WSDL type to be acquired by the **Binding Style** radio button and **SOAP Version** radio button as and when required.

   Choose the WSDL definition style to be acquired by the **Binding Style** radio button. Choose the corresponding SOAP specifications version by the **SOAP Version** radio button.
4. Click WSDL acquisition.

   A dialog box for saving the WSDL file opens.

   Specify the file saving destination and acquire the WSDL.

Method 2

1. From the directory in which HCSC-Messaging is installed, acquire the WSDL provided as a sample.

   Acquire the WSDL appropriate to your application from the following:

   - rpc-literal type

     `cscmsg_ws.wsdl` file
   - document-literal type

     `cscmsg_ws_doc.wsdl` file

   For details about the directory in which HCSC-Messaging is installed, see the section related to uCosminexus Service Platform installation in the manual *Cosminexus Service Platform System Setup and Operation Guide*.
2. Modify the contents of location of WSDL's <soap:address> as shown below.

```
http://host-name:port-number/#1context-root#2
            /services/CSCMsgSyncServiceDeliveryWSImpl
```

   #1

   This is the HCSC server's URL.

#2

HCSC server cluster name that is determined when the HCSC server configuration is deployed in the development environment and the operating environment.

8.2.3 Creating Stubs

Create a stub from the acquired WSDL. To create a stub, you use the `WSDL2Java` command provided by Cosminexus as a development support command.

A command input example (when the WSDL style is rpc-literal) follows.

```
WSDL2Java cscmsg_ws.wsdl
```

If the WSDL style is document-literal, replace `cscmsg_ws.wsdl` in the above example with `cscmsg_ws_doc.wsdl`.

For details about the options in the WSDL2Java command, see the manual *Cosminexus Application Server SOAP Application Development Guide*.

Executing this command creates the following directory and files:

```
/current-directory
 jp
   co
     Hitachi
       soft
         csc
           msg
             message
               reception
                 ejb
                   CSCMsgSyncServiceDeliveryWSImpl.java#1
                   CSCMsgSyncServiceDeliveryWSImplService.java#2
                   CSCMsgSyncServiceDeliveryWSImplServiceLocator.java#3
                   CSCMsgSyncServiceDeliveryWSImplSoapBindingStub.java#4
                   CSCMsgServerFaultException.java#4
```

#1

This is a user-defined data class (a service requester's interface class).

#2

This is a service component's interface class.

#3

This is a service class (a service component's interface class). Holds the connection information to the service component.

#4

This is a stub class.

8.2.4 Generating Objects

In order to invoke the method of the synchronous reception (Web Services), use the stubs created in (1) to generate objects.

(1) Stubs to be used

Use the following two classes of stubs to generate objects:

CSCMsgSyncServiceDeliveryWSImplServiceLocator.java class

This class is used to reference and set up the connection destination (endpoint) information to a service component. This class provides the methods listed in the following table.

Table 8–2: CSCMsgSyncServiceDeliveryWSImplServiceLocator.java class methods

Method name	Function explanation
`getCSCMsgSyncServiceDeliveryWSImplAddress()`	Returns the connection destination information to the service component. Return value: `java.lang.String`
`getCSCMsgSyncServiceDeliveryWSImpl()`	Returns the object pointer of the interface class to the service class. Return value: Interface class object (`CSCMsgSyncServiceDeliveryWSImpl` object)
`getCSCMsgSyncServiceDeliveryWSImpl(java.net.URL portAddress)`	Uses the specified connection destination information to the service component to return the object pointer to the service class. Return value: Interface class object (`CSCMsgSyncServiceDeliveryWSImpl` object)

`CSCMsgSyncServiceDeliveryWSImpl.java` class

Describes a list of methods that can be used as a service class. Use this class to utilize the SOAP service.

(2) Object generation procedure

To generate an object for invoking a synchronous reception (Web Services) method:

1. Create a CSCMsgSyncServiceDeliveryWSImplServiceLocator class object, which is the service component's interface class.

   Example:

```
CSCMsgSyncServiceDeliveryWSImplServiceLocator locator
             = new CSCMsgSyncServiceDeliveryWSImplServiceLocator();
```

2. Using the service component's interface class object, create a CSCMsgSyncServiceDeliveryWSImpl.java class object, which is the service requester's interface class.

   The instance of the service requester's interface class created or acquired cannot be shared by multiple threads.

   Example:

```
CSCMsgSyncServiceDeliveryWSImpl ws = null;
try {
      ws = locator.getCSCMsgSyncServiceDeliveryWSImpl();
}catch (ServiceException e) {
              e.printStackTrace();
              return;
}
```

The connection destination to the service component is the location attribute, which is an address child element of the Service element inside the WSDL definition. To acquire the connection destination to the service component within the service requester's program, use the following coding:

Example:

```
String url = locator.getCSCMsgSyncServiceDeliveryWSImplAddress();
```

To change the connection destination to the service component within the service requester's program, use the following coding:

Example:

```
java.net.URL endpoint
 = new java.net.URL("http://hostname:80/context-root
                  /services/CSCMsgSyncServiceDeliveryWSImpl");
CSCMsgSyncServiceDeliveryWSImpl locator
 = locator.getCSCMsgSyncServiceDeliveryWSImpl(endpoint);
```

3. Using the created CSCMsgSyncServiceDeliveryWSImpl.java class object, invoke a method of the synchronous reception (Web Services).

   Example: When the request message is in XML

```
String result = ws.invokeXML(              // method invocation
                       serviceName,        // service name
                       clientID,           // client correlation ID
                       requestFormatID,    // request format ID
                       responseFormatID,   // response format ID
                       operationName,      // operation name
                       userData);          // user message
```

   Example: When the request message is binary

```
byte[] resultBinary = ws.invokeBinary(     // method invocation
                          serviceName,     // service name
                          clientID,        // client correlation ID
                          requestFormatID, // request format ID
                          responseFormatID, // response format ID
                          operationName,   // operation name
                          userDataBinary.length,  // user message length
                          userDataBinary); // user message
```

! Important note

A binary request message can be sent only when the message format used on the service component side is binary.

8.2.5 Specifying Parameters

To invoke a method of the synchronous reception (Web Services), specify parameters that become the arguments of the method. The following figure shows parameter details.

Table 8–3: Parameter details (standard synchronous reception (Web Services))

Parameter name	Data type	Parameter		Explanation
		invokeXML	invokeBinary	
Service name (serviceName)	java.lang .String	in0(type="xsd:string")		This is the service name of the request destination. This parameter is required. For the service name of the request destination, specify the adapter or business process defined in the development environment.
Client correlation ID (clientID)	java.lang .String	in1(type="xsd:string")		This is a correlation identifier for uniquely identifying the request message from the service requester. Specify alphanumeric characters, underscore (_), period (.), and hyphen (-) up to 255 characters. This parameter is used to map the request message from the service requester to the execution history, logs, and traces managed by the HCSC server. Therefore, specify a different ID for each request message sent to the HCSC server. To omit the client correlation ID, specify NULL.
Request format ID (requestFormatID)	java.lang .String	in2(type="xsd:string")		This is an ID for uniquely identifying the request message format from the service requester. Specify NULL for this parameter.

Parameter name	Data type	Parameter		Explanation
		invokeXML	invokeBinary	
Response format ID (responseFormatID)	java.lang.String	in3(type="xsd:string")		This is an ID for uniquely identifying the response message from the HCSC server. Specify NULL for this parameter.
Operation name (operationName)	java.lang.String	in4(type="xsd:string")		This is an operation name corresponding to the service name at the request destination.# This operation name specifies a service component defined in the development environment. Specify the operation name with NCName definition characters of XMLSchema within 255 bytes. This parameter is required when the service component at the request destination is a synchronous service (Web Services or SessionBean) or business process. When the service at the request destination is an asynchronous service, the operation name can be omitted. To omit it, specify NULL.
User message (userData)	java.lang.String	in5 (type= "xsd:string")	--	This is the request message from the service requester.# Specify this parameter when the request message is in XML. If there is no request message, specify NULL or an empty character (""). For details about request messages, see *8.2.6 Creating Request Messages*.
User message length (userDataBinary.length)	int	--	in5(type= "xsd:int")	This is the request message length. Specify this parameter when the request message is binary. This parameter is required when the request message is binary. If there is no request message, specify 0.
User message (userDataBinary)	byte[]	--	in6(type= "base64Binary")	This is the request message from the service requester.# Specify this parameter when the request message is binary. For details about request messages, see *8.2.6 Creating Request Messages*. If there is no request message, specify NULL or a 0-byte byte array.

Legend:

in*X* (*X*=1 to 6): Parameter for each method indicated by a stub

--: Cannot be specified.

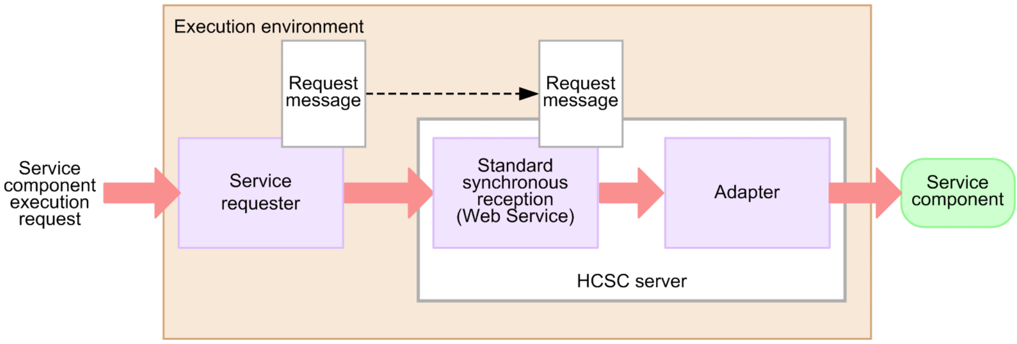
#

When the service component protocol of request destination is SOAP, decide an operation to be invoked from the name of a root element of user message (in the case of data transformation, it is the name of root element of the message after data transformation). Therefore, take note that if you specify an invalid name in the root element of user message, an unintended operation may be invoked.

8.2.6 Creating Request Messages

Create a request message for requesting a service component from the service requester to the synchronous reception (Web Services) of the HCSC server. The contents of the request message to be sent from the service requester must be created in the same message format as that used on the service component side. The following figure shows how a request message is sent.

Figure 8–4: Sending a request message (standard synchronous reception (Web Services))



For details about how to send a normal request message, see the contents related to standard reception in service invocation using a Web Service (SOAP communication), in the manual *Cosminexus Service Platform System Setup and Operation Guide*.

The request message to be sent from the service requester to the synchronous reception (Web Services) can be either an XML message or a binary message. Which message type is to be used depends on the protocol being used by the service component side. XML request messages and binary request messages are explained below.

(1) XML request message

When any one of the following is a service-module-side protocol and requests are to be sent to a business process, the XML request messages can be sent:

- SOAP (for service adapter (Web services))#1#2
- RMI-IIOP (for service adapter (SessionBean))#1#2
- WS-R (for service adapter (MDB (WS-R)))#2
- Database queue (for service adapter (MDB (DBQ)))#2

#1

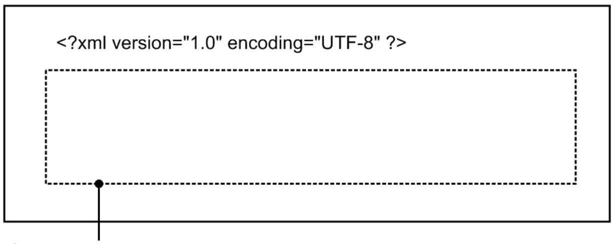
Always specify the request message.

#2

If you do not specify the request message, data transformation process will not be performed in the service adapter.

The following figure shows an XML request message.

Figure 8–5: XML request message



```
<?xml version="1.0" encoding="UTF-8" ?>
```

Set a parameter value that is passed to the service component in request message XML tag created with the message format same for the service component side.

Important note

An XML declaration must be described at the start of the XML document (request message) to be specified. If an XML declaration is not made or if an XML declaration is in a user message inside the XML document, correct operation cannot be guaranteed.

Furthermore, always specify UTF-8 in the character code specification (encoding) of an XML declaration.

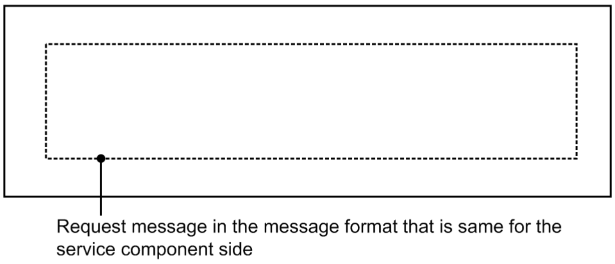
(2) Binary request message

Binary request messages can be sent only when the service component side is using the database queue protocol.

Note that if you do not specify a request message, data transformation process will not be performed in the service adapter.

The following figure shows a binary request message.

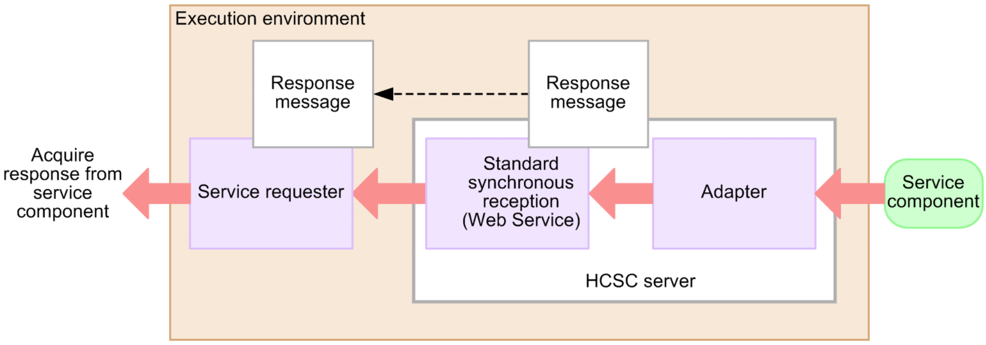
Figure 8–6: Binary request message


Request message in the message format that is same for the service component side

8.2.7 Acquiring Response Messages

Acquire a response message corresponding to the service component execution request from the synchronous reception (Web Services) of the HCSC server. The service requester acquires a response message that has the same message format as the service component side. The following figure shows how a response message is acquired.

Figure 8–7: Response message acquisition (standard synchronous reception (Web Services))

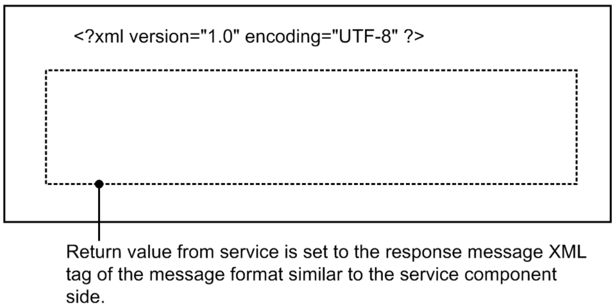


The service requester acquires a response message whose message type is XML. When there is no response message from the service component, either `NULL` or a byte array of 0 bytes is received. A case in which there is a response message and a case in which there is no response message are explained separately below.

(1) When there is a response message

The following figure shows an XML response message acquired by the service requester

Figure 8–8: XML response message

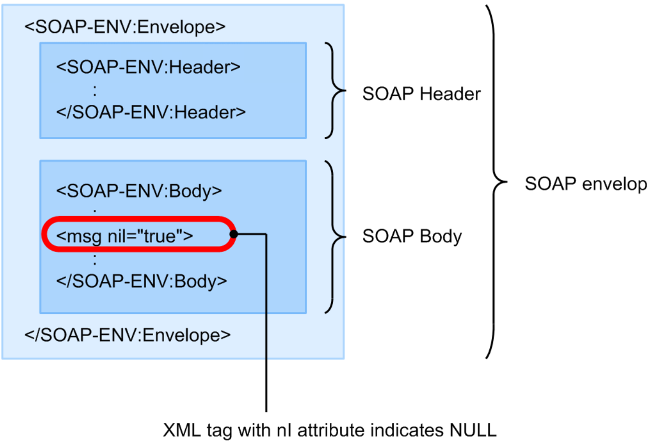

<?xml version="1.0" encoding="UTF-8" ?>
Return value from service is set to the response message XML tag of the message format similar to the service component side.

(2) When there is no response message

When there is no response message, the format received by the service requester will differ for a String-type response and byte[]-type response.

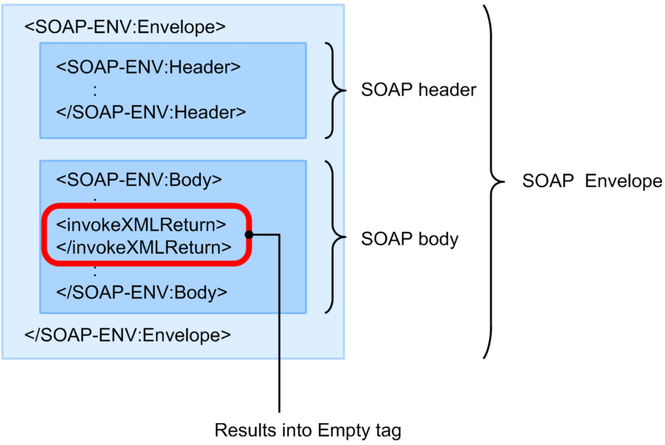
- **For a String-type response**
  The following figure shows the format in which the service requester receives `NULL`.

Figure 8–9: Format of NULL received



- **For a byte[]-type response**
  The service requester receives a byte array of 0 bytes having a format as shown in the following figure. The SOAP Envelope will become an empty tag as shown in the following figure:

Figure 8–10: Format for a byte[]-type response



8.2.8 Acquiring Error Information

If an error occurs at the request-destination service component, the HCSC server, or the SOAP engine, acquire the error information and take corrective action according to the information. For details about how to send an error, see the contents about troubleshooting during the execution of a Web Service (SOAP communication), in the manual *Cosminexus Service Platform System Setup and Operation Guide*.

(1) Service requester-side installation example

The error acquisition method differs depending on the type of SOAP Communication Infrastructure.

(a) When the SOAP Communication Infrastructure provided by Cosminexus is used

When the SOAP Communication Infrastructure provided by Cosminexus is used, catch the CSCMsgServerFaultException object and acquire the SOAP Fault error information.

To acquire the error information, the following must be installed on the service requester side:

```
        ...
 } catch (CSCMsgServerFaultException e) {
     System.err.println("Exception ErrorMessage = "
                 + e.getErrorMessage() );
     System.err.println("Exception ErrorCode = "
                 + e.getErrorCode() );
     System.err.println("Exception ProcessInstanceID = "
                 + e.getProcessInstanceID() );
     System.err.println("Exception FaultCode = "
                 + e.getCscmsgFaultCode() );
     System.err.println("Exception FaultString = "
                 + e.getCscmsgFaultString() );
     System.err.println("Exception FaultActor = "
                 + e.getCscmsgFaultActor() );
     System.err.println("Exception FaultDetails = "
                 + new String(e.getCscmsgFaultDetail(), "UTF-8"));
     System.err.println("Exception FaultName = "
                 + e.getFaultName() );
 }
        ...
```

Each method is explained below.

- **getErrorMessage**

  Explanation

  Acquires error messages.

  Use this method for acquiring the contents of the following exceptions:

  - Exceptions detected inside HCSC-Messaging
  - Faults from a service component or business process

  Format

  `public java.lang.String getErrorMessage()`

- **getErrorCode**

  Explanation

  Acquires error codes.

  Use this method for acquiring the error codes corresponding to the following exceptions:

  - Exceptions detected inside HCSC-Messaging
  - Faults from a service component or business process

  Format

  `public java.lang.String getErrorCode()`

- **getProcessInstanceID**

  Explanation

  Acquires business process instance IDs.

  Format

  `public String getProcessInstanceID()`

- **getCscmsgFaultCode**

  Explanation

  Acquires FaultCode information from a service component (Web services), business process, or custom adapter.

  Format

  `public java.lang.String getCscmsgFaultCode()`

- **getCscmsgFaultString**

  Explanation

  Acquires FaultString information from a service component (Web Services), business process, or custom adapter.

  Format

  ```
  public java.lang.String getCscmsgFaultString()
  ```

- **getCscmsgFaultActor**

  Explanation

  Acquires FaultActor information from a service component (Web Services), business process, or custom adapter.

  Format

  ```
  public java.lang.String getCscmsgFaultActor()
  ```

- **getCscmsgFaultDetail**

  Explanation

  Acquires Detail information from a service component (Web Services), business process, or custom adapter.

  Transfers the Detail information set up by the service component to the service requester in a byte array. Therefore, the acquired byte array must be converted to a character string.

  When there is no Detail information from the service component (Web Services), business process, or custom adapter, the response is sent through a byte array of 0 bytes (empty tag of a SOAP Message).

  Format

  ```
  public byte[] getCscmsgFaultDetail()
  ```

- **getFaultName**

  Explanation

  Acquires exception names from a service component (Web Services) or business process.

  Format

  ```
  public String getFaultName()
  ```

(b) When the SOAP Communication Infrastructure provided by Cosminexus is not used

The error information to be acquired depends on the SOAP engine installed on the service requester side.

(2) Error information (SOAPFault) format

The following table shows the error information (SOAPFault) format.

Table 8–4: Error information (SOAPFault) format

Element	Name	Content
`faultcode`	Fault code	Value that depends on the SOAP engine. QCName that is referenced by the message type attribute of the message part.
`faultstring`	Fault string	Value that depends on the SOAP engine. Outputs KDCCP0015-E.
`faultactor`	Fault generator	Value that depends on the SOAP engine. There is no value.
`detail`#	Fault detail	Detail that corresponds to `wsdl:fault`.

#

The element detail is for the error information detail. It is stored in the structure described in the following table.

Table 8–5: Error information (SOAPFault) detail

Field name	Explanation	
	Error (fault) from a service component, a business process, a custom adapter or for integration with HCSC server	Error (exception) detected by the HCSC server
`errorMessage`	Contents of the following errors: • Error detected inside the HCSC server. • Error from a service component, business process, or custom adapter.	
`errorCode`	Error code corresponding to the following exceptions: • Error detected inside the HCSC server. • Error from a service component, business process, or custom adapter.	
`processInstanceID`	Instance ID information of a business process A value is set when an error occurs in the business process.	
`cscmsgFaultCode`	FaultCode information from a service component (Web Services), business process, or custom adapter	--
`cscmsgFaultString`	FaultString information from a service component (Web Services), business process, or custom adapter	--
`cscmsgFaultActor`	FaultActor information from a service component (Web Services), business process, or custom adapter	--
`cscmsgFaultDetail`	Detail information from a service component (Web Services), business process, or custom adapter	--
`faultName`	Fault name (exception name) information from a service component (Web Services or SessionBean) or business process A value is set in the following cases: • In the case of SOAP Fault of a user-defined exception from a service component (Web services or SessionBean) • In the case of a fault from business process No value is set in the case of the SOAP Fault error from Web services that define URI of targetNamespace in the SOAP Fault operation definition file. For details about the SOAP Fault operation definition file, see the manual *Cosminexus Service Platform Reference*.	--

Legend:

--: Not applicable

8.2.9 Creating a service requester that sends a request for business process re-execution (Web Services and SOAP communication infrastructure)

You can create a service requester that sends a request for business process re-execution to the standard synchronous reception (Web Services).

The flow for creating a service requester that sends a request to the standard synchronous reception (Web Services) for business process re-execution is the same as that for creating an ordinary service requester that sends requests to the standard synchronous reception (Web Services). For details about the creation flow, see *8.2.1 Procedure for Creating a Service Requester (Standard Synchronous Reception (Web Services)) (SOAP communication infrastructure)*.

The tasks in the individual process are as follows:

(1) WSDL acquisition

The WSDL acquisition method is the same as that used for creating an ordinary service requester that sends requests to the standard synchronous reception (Web Services).

For details about the WSDL acquisition method, *8.2.2 Acquiring the WSDL*.

(2) Stub creation

The stub creation method is the same as the one used for creating an ordinary service requester that sends requests to the standard synchronous reception (Web Services). That is, to create a stub, you use the WSDL2Java command provided by Cosminexus as a development support command.

For details about stub creation, see *8.2.3 Creating Stubs*.

(3) Object generation

The object generation method is the same as that used for creating an ordinary service requester that sends requests to the standard synchronous reception (Web Services).

For details about object generation, see *8.2.4 Generating Objects*.

To invoke a method that sends a request for business process re-execution, use invokeBPXML().

An example of a method that sends a request for business process re-execution is shown below.

Example: Requesting business process re-execution

```
String result = ws.invokeBPXML(            // method invocation
                       serviceName,        // service name
                       bpRequestType,      // request type for business process
                       bpProcessId,        // process ID for business process
                       clientID,           // client correlation ID
                       requestFormatID,    // request format ID
                       responseFormatID,   // response format ID
                       operationName,      // operation name
                       userData);          // user message
```

(4) Parameter specification

The parameters that become the arguments of the method are different from those used in creating an ordinary service requester that sends requests to the standard synchronous reception (Web Services). The following table shows the details of the parameters that are specified for a service requester that sends a request to a synchronous reception (Web Services) for business process re-execution.

Table 8–6: Parameter details (standard synchronous reception (Web Services)/business process re-execution request)

Parameter name	Data type	Parameter (invokeBPXML)	Explanation
Service name (`serviceName`)	`java.lang.String`	`in0(type="xsd:string")`	This is the service name of the request destination. This parameter is required. For the service name of the request destination, specify the business process defined in the development environment.
Request type for business process (`bpRequestType`)	`java.lang.String`	`in1(type="xsd:string")`	Indicates a request message type. To request business process re-execution, specify a `RECOVER` string.#1
Process ID for business process (`bpProcessId`)	`java.lang.String`	`in2(type="xsd:string")`	This is a business process instance ID. Specify either a value acquired from error information or the value that is output to the message log.

Parameter name	Data type	Parameter (invokeBPXML)	Explanation
Client correlation ID (clientID)	java.lang .String	in3(type="xsd:string")	This is a correlation identifier for uniquely identifying the request message from the service requester. Specify NULL for this parameter.#1
Request format ID (requestFormatID)	java.lang .String	in4(type="xsd:string")	This is an ID for uniquely identifying the request message format from the service requester. Specify NULL or an empty character ("") for this parameter.#1
Response format ID (responseFormatID)	java.lang .String	in5(type="xsd:string")	This is an ID for uniquely identifying the response message from the HCSC server. Specify NULL or an empty character ("") for this parameter.#1
Operation name (operationName)	java.lang .String	in6(type="xsd:string")	This is an operation name corresponding to the service name at the request destination. #2 Specify NULL or an empty character ("") for this parameter.#1
User message (userData)	java.lang .String	in7(type="xsd:string")	This is the request message from the service requester.#2 Specify NULL or an empty character ("") for this parameter.#1

Legend:

in*X* (*X*=1 to 7): Parameter for each method indicated by a stub

#1

A fixed value (RECOVER, NULL, or an empty character ("")) is set as the specification value for these parameters. If you specify a value other than these fixed values, correct operation cannot be guaranteed.

#2

When the service component protocol of request destination is SOAP, decide an operation to be invoked from the name of a root element of user message (in the case of data transformation, it is the name of root element of the message after data transformation). Therefore, take note that if you specify an invalid name in the root element of user message, an unintended operation may be invoked.

(5) Response message acquisition

The response message acquisition method is the same as that used for creating an ordinary service requester that sends requests to the standard synchronous reception (Web Services).

For details about response message acquisition, see *8.2.7 Acquiring Response Messages*.

(6) Error information acquisition

The method of installing a service requester for acquiring error information is the same as that used for creating an ordinary service requester that sends requests to the standard synchronous reception (Web Services).

For details about error information acquisition, see *8.2.8 Acquiring Error Information*.

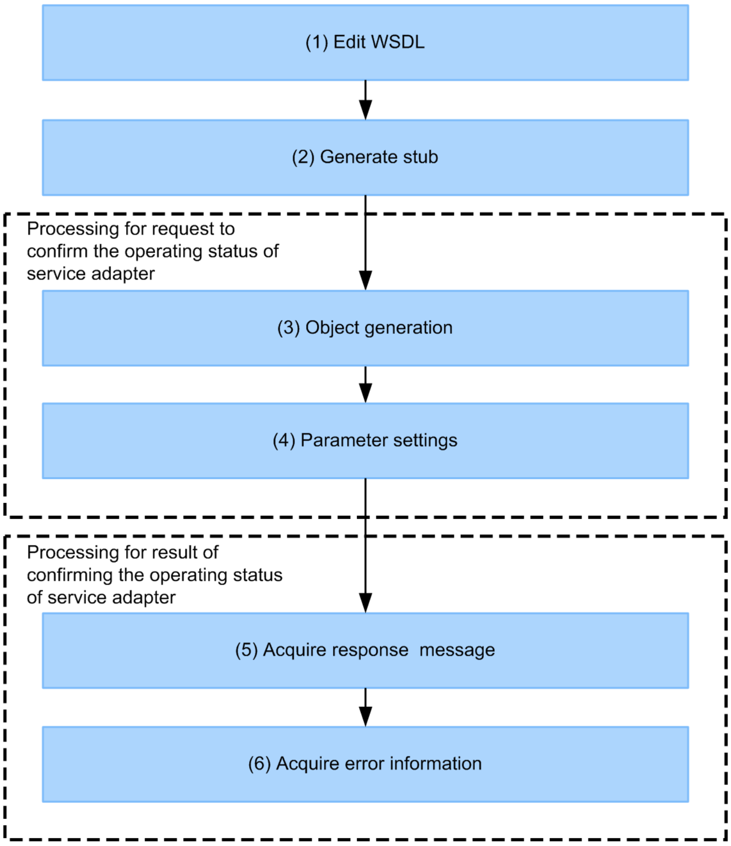
8.2.10 Creating a service requester that sends a request for the operating status of service adapter from an application (Web Services and SOAP communication infrastructure)

You can create a service requester that sends a request for the service adapter's operating status from an application to the standard synchronous reception (Web Services).

To check the operating status of a service adapter, apart from invoking the service component that sends a request to the standard synchronous reception (Web Services), you must add a process for invoking the method for checking the operating status.

The following figure shows a procedure for creating a service requester that sends a request for checking the operating status of the service adapter from an application to the standard synchronous reception (Web Services):

Figure 8–11: Procedure for creating a service requester that sends a request for checking the operating status of the service adapter (Standard synchronous reception (Web Services and SOAP communication infrastructure))


(1) Edit WSDL
(2) Generate stub
Processing for request to confirm the operating status of service adapter
(3) Object generation
(4) Parameter settings
Processing for result of confirming the operating status of service adapter
(5) Acquire response message
(6) Acquire error information

The tasks in the individual steps are described below.

(1) WSDL acquisition

The WSDL acquisition method is the same as that used for creating an ordinary service requester that sends requests to the standard synchronous reception (Web Services).

Note that you need not use different WSDLs in the process for invoking the service component and in the process for checking the operating status.

For details about the WSDL acquisition method, *8.2.2 Acquiring the WSDL*.

(2) Stub creation

The stub creation method is the same as the one used for creating an ordinary service requester that sends requests to the standard synchronous reception (Web Services). That is, to create a stub, you use the WSDL2Java command provided by Cosminexus as a development support command.

For details about stub creation, see *8.2.3 Creating Stubs*.

(3) Object generation

To generate an object for invoking a method of requesting for the service adapter's operating status (Web Services):

1. Create a CSCMsgSyncServiceDeliveryWSImplServiceLocator class object, which is the service component's interface class.

   Example:

   ```
   CSCMsgSyncServiceDeliveryWSImplServiceLocator locator
                 = new CSCMsgSyncServiceDeliveryWSImplServiceLocator();
   ```

2. Using the service component's interface class object, create a CSCMsgSyncServiceDeliveryWSImpl.java class object, which is the service requester's interface class.

   The instance of the service requester's interface class created or acquired cannot be shared by multiple threads.

   Example:

   ```
   CSCMsgSyncServiceDeliveryWSImpl ws = null;
   try {
         ws = locator.getCSCMsgSyncServiceDeliveryWSImpl();
   }catch (ServiceException e) {
                 e.printStackTrace();
                 return;
   }
   ```

   The connection destination to the service component is the location attribute, which is an address child element of the Service element inside the WSDL definition. To acquire the connection destination to the service component within the service requester's program, use the following coding:

   Example:

   ```
   String url = locator.getCSCMsgSyncServiceDeliveryWSImplAddress();
   ```

3. Using the created CSCMsgSyncServiceDeliveryWSImpl.java class object, invoke a method of requesting for the operating status of the service adapter of the standard synchronous reception (Web Services).

   Example: When `"type=status"` is specified in the option

   ```
   String result = ws.getServiceInfo(          // Method invocation
                          serviceName,         // Service name
                          clientID,            // Client correlation ID
                          "type=status");      // Option
   ```

   Example: When `"type=status,returnType=XML"` is specified in the option

   ```
   String result = ws.getServiceInfo(          // Method invocation
                          serviceName,         // Service name
                          clientID,            // Client correlation ID
                          "type=status,returnType=XML");
                                               // Option
   ```

(4) Parameter specification

Use the `getServiceInfo` method to request checking of the operating status of service adapters. The parameters that become the arguments of the method are different from those used in creating an ordinary service requester that sends requests to the standard synchronous reception (Web Services).

The following table shows the details of the parameters that are specified for a service requester that sends a request to a synchronous reception (Web Services) for the service adapter's operating status.

Table 8–7: Parameter details (Standard synchronous reception (Web Services) or service adapter operating status checking request)

Parameter name	Data type	Parameter (invokeBPXML)	Explanation
Service name (`serviceName`)	`java.lang.String`	`in0(type="xsd:string")`	This is the service name of the request destination when the operating status of service adapter is checked. This parameter is required. For the service name of the request destination, specify the service adapter deployed on the HCSC server.
Client correlation ID	`java.lang.String`	`in1(type="xsd:string")`	This is a correlation identifier for uniquely identifying the request message from the service requester.

Parameter name	Data type	Parameter (invokeBPXML)	Explanation
(`clientID`)	`java.lang.String`	`in1(type="xsd:string")`	Specify alphanumeric characters, underscore (_), period (`.`), and hyphen (`-`) up to 255 characters. This parameter is used to map the request message from the service requester to the execution history, logs, and traces managed by the HCSC server. Therefore, specify a different ID for each request message sent to the HCSC server. To omit the client correlation ID, specify `NULL`.
Option (`option`)	`java.lang.String`	`in2(type="xsd:string")`	This is an option that is selected when acquiring the checked results of the operating status. The input format of an option is `Key name=Value`. Enclose the option within double quotation marks (`"`). When specifying multiple options, demarcate with a comma (`,`).[#1] Example `"`*key-name=value*`,` *key-name=value*`"` The following options can be specified: **type={<u>all</u>/status}** This is an option that specifies the pattern of the information to be acquired. • all The entire information, corresponding to the service name specified in `serviceName`, is sent as a response. • status The status, HCSC server name, and the cluster name corresponding to the service name specified in `serviceName` is sent as a response. **returnType={<u>Properties</u>/String/XML}** This is an option that specifies the response format. • Properties All properties included in the `java.util.Properties` class are sent as a response in the XML document format[#2]. If there is no value, the property key does not exist.[#3] • String The `java.util.Properties` class is sent as a response in the `toString()` converted format. If there is no value, the property key does not exist. • XML The XML schema defined in the HCSC server is sent as a response in the XML document format. If there is no value, an empty tag is sent as a response.

Legend:

in*X* (*X*=1 to 2): Parameter for each method indicated by a stub

#1

If a space exists before or after a comma (`,`), or before or after an equal sign (=), this will result in an error. Also, make sure that a character string does not end with a comma (`,`).

#2

This is a UTF-8 encoded XML format that uses the `storeToXML()` method of the `Properties` class. This format can be restored to the `Properties` class using the `loadFromXML()` method of the `Properties` class.

#3

When the value is acquired with the `Properties` class, `null` will be sent as a response.

(5) Response message acquisition

The response message of the operating status check of service adapters is returned in the format specified in the parameter option.

The format and examples of response messages are described in the following points:

(a) Format of a response message

The information that can be acquired with a response message depends on the contents specified in the `type` option. The following information can be acquired with a response message:

Table 8–8: Information that can be acquired with a response message

Key or tag	Information contents	Response information based on the type option specification	
		all	status
`HCSCServerName`	HCSC server name	Y	Y
`ClusterName`	Cluster name	Y	Y
`ServiceName`	Service name	Y	--
`ServiceStatus`	Service adapter status	Y	Y
`ServiceKind`	Service type	Y	--
`ServiceProtocolKind`	Adapter protocol type	Y	--
`AdapterName`	Adapter name	Y	--
`AdapterLocalCall`	Local key usage status	Y#	--
`EntryTime`	Adapter definition addition time	Y	--
`ModifiedTime`	Adapter definition update time	Y	--

Legend:

Y: Can be acquired.

--: Cannot be acquired.

#

A response is sent only when `ServiceProtocolKind` is `MDB_WSR`.

The details of the acquired information are as follows:

- **HCSCServerName**

  Explanation

  Acquires the HCSC server name in which the information is acquired.

- **ClusterName**

  Explanation

  Acquires the cluster name to which the adapter (service adapter) belongs.

- **ServiceName**

  Explanation

  Acquires the deployed service name when `"type=all"` is specified in the option. If `"type=status"` is specified in the option, this information will not be acquired.

- **ServiceStatus**

  Explanation

  Acquires the status of the service adapter.

  - active: Running
  - inactive: Stopped
  - starting: Starting
  - startfailed: Failed to start
  - stopping: Stopping
  - stopfailed: Failed to stop
  - deleting: Deleting

- **ServiceKind**

  Explanation

  Acquires the type of the deployed service when `"type=all"` is specified in the option. If `"type=status"` is specified in the option, this information will not be acquired.

  - `ServiceAdapter`: Service adapter

- **ServiceProtocolKind**

  Explanation

  Acquires the type of the adapter protocol when `"type=all"` is specified in the option. If `"type=status"` is specified in the option, this information will not be acquired.

  - `WebService`: SOAP (HTTP) (Synchronous reception (Web Services))
  - `SessionBean`: RMI-IIOP (Synchronous reception (SessionBean))
  - `MDB_WSR`: WS-R (Asynchronous reception (MDB (WS-R)))
  - `MDB_DBQ`: Protocol or JMS unique to Cosminexus RM (Asynchronous reception (MDB (database queue)))
  - `Custom`: Custom adapter

- **AdapterName**

  Explanation

  Acquires the deployed adapter name when `"type=all"` is specified in the option. If `"type=status"` is specified in the option, this information will not be acquired.

- **AdapterLocalCall**

  Explanation

  Acquires the usage status of a local key of the adapter when `"type=all"` is specified in the option. If `"type=status"` is specified in the option, this information will not be acquired.

  This item is sent as a response only when `ServiceProtocolKind` is `MDB_WSR`.

  - `true`: A local key is used
  - `false`: A local key is not used

- **EntryTime**

  Explanation

  Sends (as a response) the time (time registered in the definition) at which the service adapter was deployed, when `"type=all"` is specified in the option. If `"type=status"` is specified in the option, this information will not be acquired.

  Format

  `YYYY/MM/DD hh:mm:ss.SSS`

  - `YYYY`: Year
  - `MM`: Month
  - `DD`: Date

- `hh`: Hour
- `mm`: Minutes
- `ss`: Seconds
- `SSS`: Milli seconds

- **ModifiedTime**

Explanation

Sends (as a response) the time (time registered in the definition) at which the service adapter was defined in the development environment, when `"type=all"` is specified in the option. If `"type=status"` is specified in the option, this information will not be acquired.

Format

`YYYY/MM/DD hh:mm:ss`

- `YYYY`: Year
- `MM`: Month
- `DD`: Date
- `hh`: Hour
- `mm`: Minutes
- `ss`: Seconds

(b) Examples of response messages

Examples of response messages are described below.

Example 1: When `"type=all,returnType=Properties"` is specified in the option:

```
<?xml version="1.0" encoding="UTF-8"?>
<getServiceInfoResponse>
<HCSCServerName>HCSC</HCSCServerName>
<ClusterName>Cluster</ClusterName>
<ServiceName>Service1</ServiceName>
<ServiceStatus>Active</ServiceStatus>
<ServiceKind>ServiceAdapter</ServiceKind>
<ServiceProtocolKind>WebService</ServiceProtocolKind>
<AdapterName>ad1</AdapterName><AdapterLocalCall/>
<EntryTime>YYYY/MM/DD hh:mm:ss.SSS</EntryTime>
<ModifiedTime>YYYY/MM/DD hh:mm:ss</ModifiedTime>
</getServiceInfoResponse>
```

#1

The order of the values might vary (can be in any order).

#2

When `"type=status,returnType=Properties"` is specified in the option, the information that is not to be sent as a response does not exist in the response message.

Example 2: When `"type=all,returnType=String"` is specified in the option:

```
{HCSCServerName=HCSC,ClusterName=Cluster,ServiceName=Service1,ServiceStatus=Active,
ServiceKind=ServiceAdapter,ServiceProtocolKind=WebService,AdapterName=ad1,EntryTime
=YYYY/MM/DD hh:mm:ss.SSS,ModifiedTime=YYYY/MM/DD hh:mm:ss}
```

#1

The order of the values may vary (can be in any order).

#2

When `"type=status,returnType=String"` is specified in the option, the information that is not to be sent as a response does not exist in the response message.

Example 3 When `"type=all,returnType=XML"` is specified in the option:

```
<?xml version="1.0" encoding="UTF-8"?>
<getServiceInfoResponse>
<HCSCServerName>HCSC</HCSCServerName>
<ClusterName>Cluster</ClusterName>
```

```
<ServiceName>Service1</ServiceName>
<ServiceStatus>Active</ServiceStatus>
<ServiceKind>ServiceAdapter</ServiceKind>
<ServiceProtocolKind>WebService</ServiceProtocolKind>
<AdapterName>ad1</AdapterName>
<AdapterLocalCall/>
<EntryTime>YYYY/MM/DD hh:mm:ss.SSS</EntryTime>
<ModifiedTime>YYYY/MM/DD hh:mm:ss</ModifiedTime>
</getServiceInfoResponse>
```

#1

When no value exists, an empty tag is sent as a response.

Example

`<AdapterLocalCall/>`

#2

When `"type=status,returnType=XML"` is specified in the option, <*tag*> of the information that is not to be sent as a response does not exist in the response message.

(6) Error information acquisition

When an error occurs while checking the operating status of the service adapter, catch the `CSCMsgServerFaultException` object and acquire the error information of `SOAPFault`.

For details about error information acquisition, see *8.2.8 Acquiring Error Information*.

The main causes of occurrence of an error are as follows:

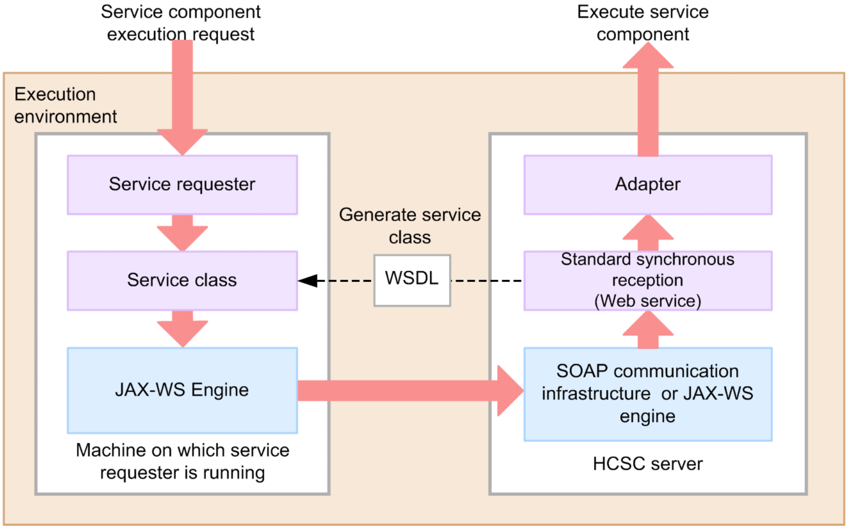
- Required arguments are not specified.
- The input value of arguments is invalid (invalid format).
- A service adapter with the corresponding service name is not deployed on the HCSC server.
- Request reception is not running (stopped, starting, failed to start, stopping, failed to stop, and deleting).

8.3 Creating a service requester using standard synchronous reception (Web Services) (JAX-WS engine)

You can create a service requester that sends a request to the standard synchronous reception (Web Services) and uses the JAX-WS engine for communication. A service adapter that uses the JAX-WS engine for communication sends a request message to standard synchronous reception (Web Services) using SOAP (document-literal type).

The following figure shows the relationship between a service requester for the JAX-WS engine and an HCSC server:

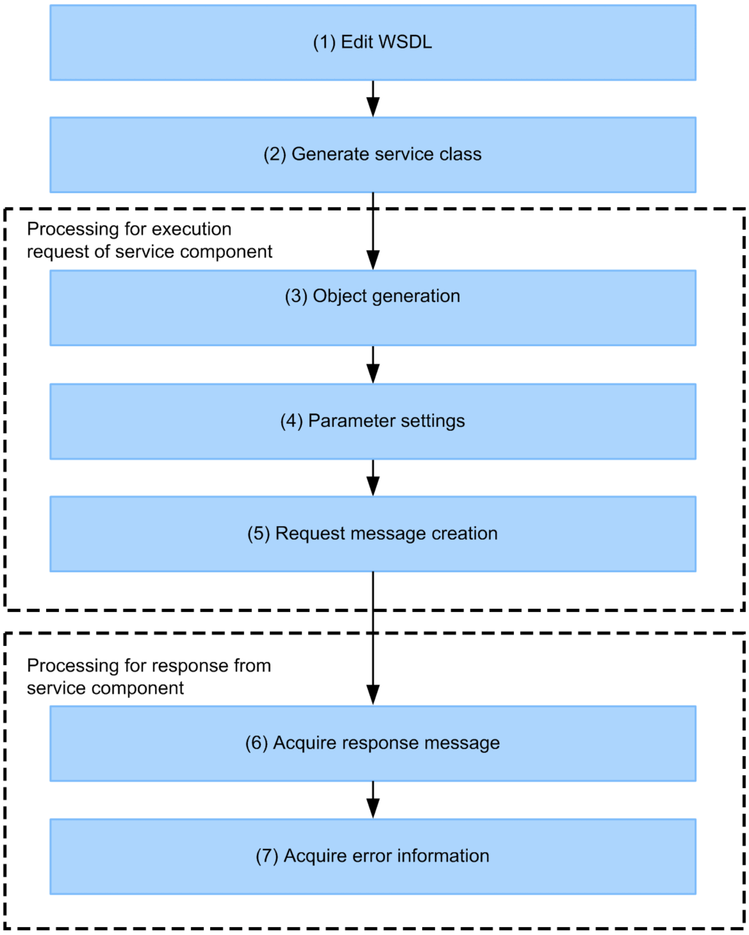
Figure 8–12: Relationship between a service requester that generates a request to Standard synchronous reception (Web Services) and an HCSC server (JAX-WS engine)


Service component execution request
Execute service component
Execution environment
Service requester
Service class
JAX-WS Engine
Machine on which service requester is running
Generate service class
WSDL
Adapter
Standard synchronous reception (Web service)
SOAP communication infrastructure or JAX-WS engine
HCSC server

8.3.1 Procedure for creating a service requester (Standard synchronous reception (Web Service)) (JAX-WS engine)

This subsection describes the procedure for sending a request for executing the service component in standard synchronous reception (Web Service) to create a service requester for invoking the service component.

Figure 8–13: Procedure for creating a service requester (standard synchronous reception (Web service) (JAX-WS engine)


(1) Edit WSDL
(2) Generate service class
Processing for execution request of service component
(3) Object generation
(4) Parameter settings
(5) Request message creation
Processing for response from service component
(6) Acquire response message
(7) Acquire error information

The operations of each process are as follows:

(1) Acquiring WSDL

Acquire interface information of synchronous reception (Web Service) of the HCSC server invoking the service component. For details about acquiring WSDL, see *8.3.2 Acquiring WSDL*.

(2) Creating service classes

Create the service class from WSDL acquired by *8.3.2 Acquiring WSDL*. For details about creating service classes, see *8.3.3 Creating service classes*.

(3) Creating objects

To invoke the synchronous reception (Web Service) method, create a proxy class object from the service class generated in *8.3.3 Creating service classes*. For details about creating the proxy class objects, see *8.3.4 Generating objects*.

(4) Specifying parameters

Specify parameters for the method argument of the synchronous reception (Web Service). For details about specifying parameters, see *8.3.5 Specifying parameters*.

(5) Creating request messages

Create the request message to request execution of the service component. For details about creating request messages, see *8.3.6 Creating request messages*.

(6) Acquiring response messages

From synchronous reception (Web Service), acquire the response message for the execution request of the service component. For details about acquiring response messages, see *8.3.7 Acquiring response messages*.

(7) Acquiring error information

If errors occur in the service component of the request destination, HCSC server, and the JAX-WS engine, acquire an error information and take action accordingly. For details about acquiring an error information, see *8.3.8 Acquiring error information*.

8.3.2 Acquiring WSDL

Acquire interface information of synchronous reception (Web Service) of the HCSC server invoking the service component. The following 2 methods can be used for acquiring WSDL:

Method 1

1. In the Eclipse menu, choose **HCSC-Definer** and **List of public services**.
   The list of HCSC components appears in the list of public services in tree view.
2. In the list of HCSC components, choose the service component to be invoked (HCSC component) and double-click.
   Information of the HCSC component appears in the HCSC Component Information Display screen. For details about the HCSC Component Information Display screen, see *Cosminexus Service Platform Reference*.
   Information of the HCSC server of the service component request destination appears in **Server information**.
3. Choose the WSDL type to be acquired by the **Binding Style** radio button and **SOAP Version** radio button, if required.
   Choose the WSDL definition style to be acquired by the **Binding Style** radio button. Choose the corresponding SOAP specifications version by the **SOAP Version** radio button.
4. Click **Acquire WSDL**.
   The **Save WSDL file** dialog box appears.
   Specify the save destination and acquire WSDL.

Method 2

1. Acquire a WSDL sample from the directory in which HCSC-Messaging is installed.
   Acquire either of the following WSDL according to use:
   - For SOAP1.1/document-literal type
     `cscmsg_ws_doc.wsdl` file
   - For SOAP1.2/document-literal type
     `cscmsg_ws_doc_1_2.wsdl` file

   For details about the directory in which HCSC-Messaging is installed, see the contents related to installation of uCosminexus Service Platform in *Cosminexus Service Platform System Setup and Operation Guide*.
2. Change the contents of `location` of WSDL `soap:address` or `soap12:address` to the following:
   ```
   http://Host name:Port number#1/Context root#2
               /services/CSCMsgSyncServiceDeliveryWSImpl
   ```
   #1
   URL of the HCSC server.

#2

Cluster name of the HCSC server decided while deploying the HCSC server configuration in the development environment and operation environment. In SOAP1.2, the cluster name is given the suffix "12".

8.3.3 Creating service classes

Create a service class from the acquired WSDL. Create a service class with the `cjwsimport` command provided by the JAX-WS functionality.

A command input example is shown below:

```
cjwsimport -s source-file-output-destination-directory -d compiled-class-file-output-
destination-directory cscmsg_ws_doc.wsdl
```

For the JAX-WS engine, set up `cscmsg_ws_doc.wsdl` in such a way so that the WSDL style becomes document-literal.

In the input example of this command, the WSDL (`cscmsg_ws_doc.wsdl`) acquired in *8.3.2 Acquiring WSDL* is saved in the current directory in which the `cjwsimport` command is executed.

For details about options of the `cjwsimport` command, see the manual *Cosminexus Application Server Web Service Development Guide*.

When you execute this command, the following directories and files will be created in the output destination directory of the specified source file:

```
/Output destination directory of the source file
 jp/co/hitachi/soft/csc/msg/message/reception/ejb/
 CSCMsgServerFaultException.java
    //Class in which fault information unique to HCSC server is saved
 CSCMsgServerFaultException_Exception.java
    //Exception class that throws CSCMsgServerFaultException
 CSCMsgSyncServiceDeliveryWSImpl.java
    //Service end point interface corresponding to the portType element of the WSDL
 CSCMsgSyncServiceDeliveryWSImplService.java
    //Service class
 GetServiceInfo.java
    //JavaBean class for the request message of the getServiceInfo operation#1
 GetServiceInfoResponse.java
    //JavaBean class for the response message of the getServiceInfo operation#1
 InvokeBinary.java
    //JavaBean class for the request message of the invokeBinary operation#1
 InvokeBinaryResponse.java
    //JavaBean class for the response message of the invokeBinary operation#1
 InvokeBPXML.java
    //JavaBean class for the request message of the invokeBPXML operation#1
 InvokeBPXMLResponse.java
    //JavaBean class for the response message of the invokeBPXML operation#1
 InvokeXML.java
    //JavaBean class for the request message of the invokeXML operation#1
 InvokeXMLResponse.java
    //JavaBean class for the response message of the invokeXML operation#1
 ObjectFactory.java
    //ObjectFactory class of the JAXB2.1 specifications
 package-info.java
      //Files used with JAXB2.1
 cscmsg_ws_doc.wsdl#2
```

#1

Used in JAXB2.1.

#2

In this output example, the `cjwsimport` command is executed in the following conditions:

- The WSDL (`cscmsg_ws_doc.wsdl`) acquired in *8.3.2 Acquiring WSDL* is saved in the directory specified in the `-s` option of the `cjwsimport` command. The WSDL specified in the `cjwsimport` command is also a WSDL that exists in the directory specified in the `-s` option.
- The `cjwsimport` command is executed by omitting the `-s` option, and the WSDL (`cscmsg_ws_doc.wsdl`) acquired in *8.3.2 Acquiring WSDL* is saved in the current directory in which the

`cjwsimport` command is executed. The WSDL specified in the `cjwsimport` command is also a WSDL for which no directory is specified and that exists in the current directory.

The WSDL (`cscmsg_ws_doc.wsdl`) specified in the `cjwsimport` command is not output in the output destination directory of the source file specified in the `-s` option after the command is executed.

! Important note

For a service requester supporting the JAX-WS engine for communication, the WSDL is read during the execution of the program. When the default constructor of the service class is used, the WSDL in the WSDL path (directory that stores the WSDL file acquired in *8.3.2 Acquiring WSDL* for generating the service class) is read. Therefore, after executing the `cjwsimport` command, do not move the WSDL file so that the configuration position of the WSDL file based on the service class does not obstruct the relative relationship. For changing the configuration position of the WSDL referenced during the execution of the service requester from the WSDL path specified with the `cjwsimport` command, use a constructor for which the URL can be specified.

8.3.4 Generating objects

In order to invoke the method of the synchronous reception (Web Services), use the created service class to generate proxy class objects. The procedures of generating proxy class objects are described below:

1. Create objects of the CSCMsgSyncServiceDeliveryWSImplService class.

   Example:

```
CSCMsgSyncServiceDeliveryWSImplService service
             = new CSCMsgSyncServiceDeliveryWSImplService();
```

2. Create the CSCMsgSyncServiceDeliveryWSImpl.java class that is the proxy class corresponding to wsdl:portType.

   Example:

```
CSCMsgSyncServiceDeliveryWSImpl ws
             = service.getCSCMsgSyncServiceDeliveryWSImpl();
```

3. Using the created CSCMsgSyncServiceDeliveryWSImpl.java class object, invoke a method of the synchronous reception (Web Services).

   Example: When the request message is in XML

```
String result = ws.invokeXML(               // Method invocation
                       serviceName,         // Service name
                       clientID,            // Client correlation ID
                       requestFormatID,     // Request format ID
                       responseFormatID,    // Response format ID
                       operationName,       // Operation name
                       userData);           // User message
```

   Example: When the request message is binary

```
byte[] resultBinary = ws.invokeBinary(      // Method invocation
                          serviceName,      // Service name
                          clientID,         // Client correlation ID
                          requestFormatID,  // Request format ID
                          responseFormatID, // Response format ID
                          operationName,    // Operation name
                          userDataBinary.length,  // User message length
                          userDataBinary);  // User message
```

! Important note

A binary request message can be sent only when the message format used on the service component side is binary.

8.3.5 Specifying parameters

The parameters that become the arguments of the method are the same as those used in creating a service requester in SOAP communication infrastructure.

For details about parameter specification in SOAP communication infrastructure, see *8.2.5 Specifying Parameters*.

8.3.6 Creating request messages

The request message creation method is the same as that used for creating a service requester in SOAP communication infrastructure.

For details about request message creation in SOAP communication infrastructure, see *8.2.6 Creating Request Messages*.

8.3.7 Acquiring response messages

The response message acquisition method is the same as that used for creating a service requester in SOAP communication infrastructure.

For details about response message acquisition in SOAP communication infrastructure, see *8.2.7 Acquiring Response Messages*.

8.3.8 Acquiring error information

If an error occurs at the request-destination service component, the HCSC server, or the JAX-WS engine, acquire the error information and take corrective action according to the information.

For a service adapter that uses the JAX-WS engine for communication, the exception (`CSCMsgServerFaultException_Exception` in SOAP1.1 and `CSCMsgServerFault1.2Exception_Exception` in SOAP1.2) in which the information is wrapped will be caught. Therefore, the save fault information class must be acquired using the `getFaultInfo()` method.

#

While executing the `cjwsimport` command, the exception name is given the suffix "_Exception" to avoid removing the underscore "_" (underscore) and the existence of duplicate names with the save class of fault information (FaultBean).

(1) Example of implementing service requester side

An example of implementation at the service requester side to acquire the save fault information class when using the `getFaultInfo()` method to acquire an exception class, is described below.

- In SOAP1.1

```
/**
 * Sample Program
 */
{
    try {
        :
      // Web service invocation
        :
    } catch (CSCMsgServerFaultException_Exception e) {
      // An exception or user-defined exception occurs in CSC

      // Get the exception object defined in CSC
      CSCMsgServerFaultException faultInfo = e.getFaultInfo();

      // Output the exception information
      System.err.println("errorCode="
                           + faultInfo.getErrorCode());
      System.err.println("errorMessage="
                           + faultInfo.getErrorMessage());
      System.err.println("processInstanceID="
                           + faultInfo.getProcessInstanceID());
      System.err.println("faultCode="
                           + faultInfo.getCscmsgFaultCode());
      System.err.println("faultActor="
                           + faultInfo.getCscmsgFaultActor());
      System.err.println("faultString="
                           + faultInfo.getCscmsgFaultString());
      System.err.println("faultName="
                           + faultInfo.getFaultName());
      byte[] faultDetail = faultInfo.getCscmsgFaultDetail();
      try {
        if(faultDetail != null) {
          System.err.println("faultDetail="
                           + new String(faultDetail, "UTF-8"));
        }
      } catch (UnsupportedEncodingException e1) {
        e1.printStackTrace();
      }
    } catch (SOAPFaultException e) {
        :
    }
}
```

- In SOAP1.2

```
/**
 * Sample Program
 */
{
    try {
        :
      // Web Service invocation
        :
    } catch (CSCMsgServerFault12Exception_Exception e) {
      // An internal exception or a user-defined exception occurred in CSC

      // Acquire fault information object defined in CSC
      CSCMsgServerFault12Exception faultInfo = e.getFaultInfo();

      // Output exception information
      System.err.println("errorCode="
                           + faultInfo.getErrorCode());
      System.err.println("errorMessage="
                           + faultInfo.getErrorMessage());
      System.err.println("processInstanceID="
                           + faultInfo.getProcessInstanceID());
      CscmsgFaultCode code = faultInfo.getCscmsgFaultCode();
      if (code == null) {
        System.err.println("Code=" + code);
      } else {
        List<String> values = code.getValues();
        for(String value : values) {
          System.err.println("Code Value=" + value);
        }
      }
      CscmsgFaultReason reason = faultInfo.getCscmsgFaultReason();
      if (reason == null) {
        System.err.println("Reason=" + reason);
      } else {
        List<CscmsgFaultReasonText> texts =
          reason.getCscmsgFaultReasonText();
        Locale locale = Locale.getDefault();
        for(CscmsgFaultReasonText text : texts) {
          if(locale.equals(text.getLocale())) {
            System.err.println("Reason=" + text.getText());
          }
        }
      }
      System.err.println("Role="
                           + faultInfo.getCscmsgFaultRole());
      System.err.println("Node="
                           + faultInfo.getCscmsgFaultNode());
      System.err.println("faultName="
                           + faultInfo.getFaultName());
      byte[] faultDetail = faultInfo.getCscmsgFaultDetail();
      try {
        if(faultDetail != null) {
          System.err.println("faultDetail="
                           + new String(faultDetail, "UTF-8"));
        }
      } catch (UnsupportedEncodingException e1) {
        e1.printStackTrace();
      }
    } catch (SOAPFaultException e) {
        :
    }
}
```

If an exception (SOAP Fault) occurs from the Web Service in the detail information of the SOAP envelope, SOAP Fault information and HCSC-Messaging returned by the service (in which SOAP Fault occurred) includes the set error message and error code.

If an exception is detected in HCSC-Messaging and in a service other than the Web Service, the error code and error information detected in HCSC-Messaging is included.

Each method is described as follows:

(a) Method of exception information maintenance class

getErrorMessage (common for SOAP1.1 and SOAP1.2)

Description

Acquires the error message.

Used for acquiring contents of the following exceptions:

- Exception detected in HCSC-Messaging
- Faults from the service component and business process

Format

```
public java.lang.String getErrorMessage()
```

getErrorCode (common for SOAP1.1 and SOAP1.2)

Description

Acquires the error code.

Used for acquiring error codes for contents of the following exceptions:

- Exception detected in HCSC-Messaging
- Faults from the service component and business process

Format

```
public java.lang.String getErrorCode()
```

getProcessInstanceID (common for SOAP1.1 and SOAP1.2)

Description

Acquires the business process instance ID.

Format

public String getProcessInstanceID()

getCscmsgFaultCode (for SOAP1.1)

Description

If communication is received from SOAP1.1, acquires FaultCode information from the service component (Web Service), business process or custom adapter.

Format

```
public java.lang.String getCscmsgFaultCode()
```

getCscmsgFaultString (for SOAP1.1)

Description

If communication is received from SOAP1.1, acquires FaultString information from the service component (Web Service), business process or custom adapter.

Format

public java.lang.String getCscmsgFaultString()

getCscmsgFaultActor (for SOAP1.1)

Description

If communication is received from SOAP1.1, acquires FaultActor information from the service component (Web Service), business process or custom adapter.

Format

```
public java.lang.String getCscmsgFaultActor()
```

getCscmsgFaultCode (for SOAP1.2)

Description

If communication is received from SOAP1.2, acquires the CscmsgFaultCode class that maintains FaultCode or Code information from the service component (Web Service), business process or custom adapter.

Format

```
public jp.co.hitachi.soft.csc.msg.message.reception.ejb.CscmsgFaultCode
getCscmsgFaultCode()
```

getCscmsgFaultReason (for SOAP1.2)

Description

If communication is received from SOAP1.2, acquires the CscmsgFaultReason class that maintains Reason information from the service component (Web Service), business process or custom adapter.

Format

```
public jp.co.hitachi.soft.csc.msg.message.reception.ejb.CscmsgFaultReason
getCscmsgFaultReason()
```

getCscmsgFaultRole (for SOAP1.2)

Description

If communication is received from SOAP1.2, acquires Role information from the service component (Web Service), business process or custom adapter.

Format

```
public java.lang.String getCscmsgFaultRole()
```

getCscmsgFaultNode (for SOAP1.2)

Description

If communication is received from SOAP1.2, acquires Node information from the service component (Web Service), business process or custom adapter.

Format

```
public java.lang.String getCscmsgFaultNode()
```

getCscmsgFaultDetail (common for SOAP1.1 and SOAP1.2)

Description

Acquires Detail information from the service component (Web service), business process or custom adapter.

Transfers the Detail information ser by the service component to the service requester in byte arrays. The acquired byte arrays must be converted to character strings.

If Detail information does not exist in the service component (Web Service), business process or custom adapter, respond with a byte array of 0 bytes (blank tab of SOAP message).

Format

```
public byte[] getCscmsgFaultDetail()
```

getFaultName (common for SOAP1.1 and SOAP1.2)

Description

Acquires exception name from the service component (Web Service) or business process.

Format

```
public String getFaultName()
```

(b) Method of the CscmsgFaultCode class

getValues (for SOAP1.2)

Description

If communication is received from SOAP1.2, acquires Value information containing FaultCode or Code information from the service component (Web Service), business process or custom adapter.

Format

```
public java.util.List<java.lang.String> getValues()
```

(c) Method of the CscmsgFaultReason class

getTexts (for SOAP1.2)

Description

If communication is received from SOAP1.2, acquires Text information containing Reason information from the service component (Web Service), business process or custom adapter.

Format

```
public java.util.List<java.lang.String> getTexts()
```

(2) Format of error information (SOAPFault)

The following table shows the format of error information (SOAPFault).

Table 8–9: Format of error information (SOAPFault)

Element		Name	Contents
SOAP1.1	SOAP1.2		
faultcode	Code	Fault code	Value depends on the JAX-WS engine. Output a violation code.
faultstring	--	Fault character string	Value depends on the JAX-WS engine. Output an error message.
--	Reason	Fault reason	Value depends on the JAX-WS engine. Output an error message.
faultactor	Role	Fault creator	Value depends on the JAX-WS engine. The value does not exist.
--	Node	Fault node	Value depends on the JAX-WS engine. The value does not exist.
detail#1	Detail#2	Fault details	Contents correspond to wsdl:fault.

Legend: Not applicable

#1

The detail element shows details of error information. Table10-10 shows the structural system in which it is saved.

#2

The detail element shows details of error information. Table10-11 shows the structural system in which it is saved.

Table 8–10: Details (SOAP1.1) of error information (SOAPFault)

Field name	Description	
	Error (fault) from service component, business process, custom adapter or integrated HCSC server	Error (exception) detected in HCSC server
errorMessage	Error contents are as follows: • Error detected in HCSC server • Error from service component, business process and custom adapter	
errorCode	Error codes correspond to the following exception contents. • Error detected in HCSC server • Error from service component, business process and custom adapter	
processInstanceID	Information of business process instance ID. Value is set if an error occurs in the business process.	
cscmsgFaultCode	FaultCode information from service component (Web Service), business process or custom adapter.	--

Field name	Description	
	Error (fault) from service component, business process, custom adapter or integrated HCSC server	Error (exception) detected in HCSC server
cscmsgFaultString	FaultString information from service component (Web Service), business process or custom adapter.	--
cscmsgFaultActor	FaultActor information from service component (Web Service), business process or custom adapter.	--
cscmsgFaultDetail	Detail information from service component (Web Service), business process or custom adapter.	--
faultName	Fault name (exception name) information from service component (Web service or SessionBean) or business process. Value is set in the following cases: • SOAP Fault of user-defined exception from service component (Web Service or SessionBean) • Fault from business process In the SOAP Fault operation definition file, value is not set in case of SOAP Fault from Web Service defining targetNamespace URI. For details about the SOAP Fault operation definition file, see *Cosminexus Service Platform Reference*.	--

Legend: Not applicable.

Table 8–11: Details (SOAP1.2) of error information (SOAPFault)

Field name	Description	
	Error (fault) from service component, business process, custom adapter or integrated HCSC server	Error (exception) detected in HCSC server
errorMessage	Error contents are as follows. • Error detected in HCSC server • Error from service component, business process and custom adapter	
errorCode	Error codes correspond to the following exception contents. • Error detected in HCSC server • Error from service component, business process and custom adapter	
processInstanceID	Information of business process instance ID. Value is set if an error occurs in the business process.	
cscmsgFaultCode	FaultCode information from service component (Web Service), business process or custom adapter.	--
cscmsgFaultReason	FaultReason information from service component (Web Service), business process or custom adapter.	--
cscmsgFaultRole	Role information from service component (Web Service), business process or custom adapter.	--
cscmsgFaultDetail	Detail information from service component (Web Service), business process or custom adapter.	--
faultName	Fault name (exception name) information from service component (Web service or SessionBean) or business process. Value is set in the following cases: • SOAP Fault of user-defined exception from service component (Web Service or SessionBean) • Fault from business process	--

Field name	Description	
	Error (fault) from service component, business process, custom adapter or integrated HCSC server	Error (exception) detected in HCSC server
faultName	In the SOAP Fault operation definition file, value is not set in case of SOAP Fault from Web Service defining targetNamespace URI. For details about the SOAP Fault operation definition file, see *Cosminexus Service Platform Reference*.	--

Legend: Not applicable.

8.3.9 Creating a service requester requesting re-execution of a business process (Web service and JAX-WS engine)

You can create a service requester requesting re-execution of the business process for the service requester that outputs a request to standard synchronous reception (Web Service).

Transmission occurs between the service requester and standard synchronous reception (Web Service) by SOAP messages in the document/literal style.

The creation flow of the service requester requesting the standard synchronous reception (Web Service) to re-execute the business process is the same as for creating an ordinary service requester by generating a request in standard synchronous reception (Web Service). For creation flow, see *8.3.1 Procedure for creating a service requester (Standard synchronous reception (Web Service)) (JAX-WS engine)*.

The following describes the contents implemented in each process.

(1) Acquiring WSDL

The method to acquire WSDL is the same as for creating an ordinary service requester by generating a request in standard synchronous reception (Web Service).

For details about how to acquire WSDL, see *8.3.2 Acquiring WSDL*.

(2) Creating service classes

The method for creating service classes is the same as for creating an ordinary service requester by generating a request in standard synchronous reception (Web Service).

For details about creating service classes, see *8.3.3 Creating service classes*.

(3) Creating objects

The method for creating objects is the same as for creating an ordinary service requester by generating a request in standard synchronous reception (Web Service).

For details about creating objects, see *8.3.4 Generating objects*.

Note that the method for requesting re-execution of the business process uses invokeBPXML() and invokes.

(4) Specifying parameters

The parameter of the method argument is the same as for creating a service requester in SOAP communication infrastructure.

For details about specifying parameters in SOAP communication infrastructure, see *8.2.5 Specifying Parameters*.

(5) Creating request messages

The method for creating request messages is the same as for creating service requester in SOAP communication infrastructure.

For details about creating request messages in SOAP communication infrastructure, see *8.2.6 Creating Request Messages*.

(6) Acquiring response messages

The method for acquiring response messages is the same as for creating a service requester in SOAP communication infrastructure.

For details about acquiring response messages in SOAP communication infrastructure, see *8.2.7 Acquiring Response Messages*.

(7) Acquiring error information

Implementation of the service requester for acquiring error information is the same as for creating an ordinary service requester by generating a request in standard synchronous reception (Web Service).

For details about acquiring error information, see *8.3.8 Acquiring error information*.

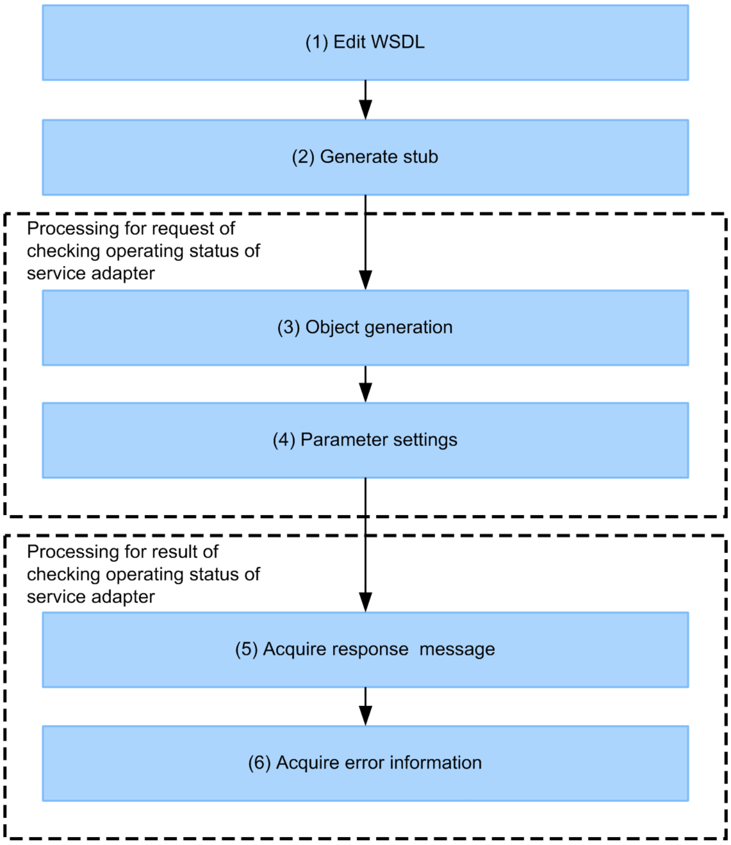
8.3.10 Creating service requester for requesting confirmation of operation status of service adapter (Web Service and JAX-WS engine)

For the service requester requesting standard synchronous reception (Web Service), you can create a service requester requesting confirmation of operation status of the service adapter from the application.

To confirm the operation status of the service adapter, the service component generating a request to standard synchronous reception (Web Service) must be invoked and the process invoking the method of operation status confirmation must be added.

The following shows the procedure for creating service requester for requesting confirmation of operation status of service adapter from the application in standard synchronous reception (Web Service).

Figure 8–14: Procedure for creating service requester for requesting confirmation of operation status of service adapter (standard synchronous reception (Web Service and JAX-WS engine))


(1) Edit WSDL
(2) Generate stub
Processing for request of checking operating status of service adapter
(3) Object generation
(4) Parameter settings
Processing for result of checking operating status of service adapter
(5) Acquire response message
(6) Acquire error information

The following describes the operation of each process.

(1) Acquiring WSDL

The method for acquiring WSDL is the same as for creating an ordinary service requester by generating a request in standard synchronous reception (Web Service).

For details about how to acquire WSDL, see *8.3.2 Acquiring WSDL*.

(2) Creating service classes

The method for creating service classes is the same as for creating an ordinary service requester by generating a request in standard synchronous reception (Web Service).

For details about creating service classes, see *8.3.3 Creating service classes*.

(3) Creating objects

The method for creating objects is the same as for creating a service requester in SOAP communication infrastructure.

For details about creating objects, see *8.3.9(3) Creating objects*.

To invoke the service method, use the class object and implement as follows:

```
String result = ws.getServiceInfo(          // Invoking method
                       serviceName,         // Service name
                       clientID,            // Client correlation ID
                       "type=all,returnType=Properties");      // Options
```

(4) Specifying parameters

The parameter of the method argument is the same as for creating a service requester in SOAP communication infrastructure.

For details about specifying SOAP communication infrastructure parameters, see *8.3.9(4) Specifying parameters*.

(5) Acquiring response messages

The method for acquiring response messages is the same as for creating a service requester in SOAP communication infrastructure.

SOAP communication infrastructure. For details about acquiring response messages, see *8.3.9(5) Creating request messages*.

(6) Acquiring error information

Implementation of the service requester for acquiring error information is the same as for creating an ordinary service requester by generating a request in standard synchronous reception (Web Service).

For details about acquiring error information, see *8.3.8 Acquiring error information*.

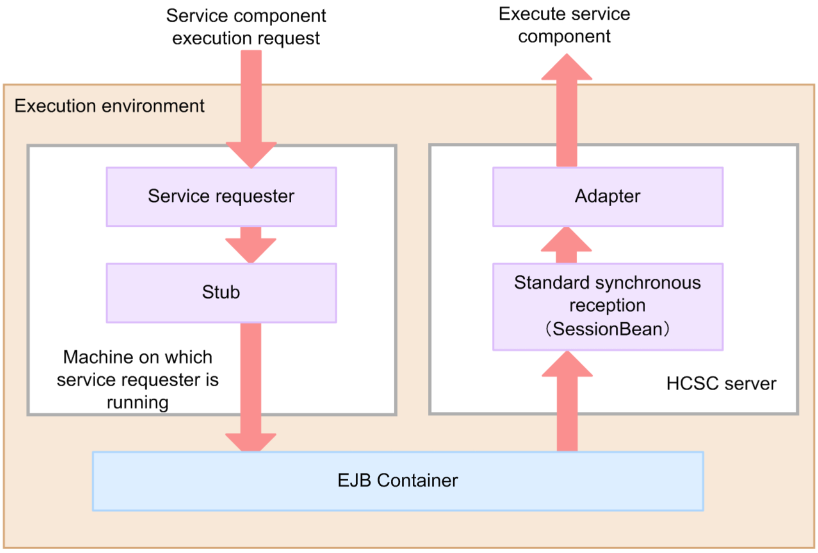
8.4 Service Requester That Sends Requests to a Standard Synchronous Reception (SessionBean) (JAX-WS engine)

A service requester that sends request messages to a standard synchronous reception (SessionBean) uses RMI-IIOP to communicate with the standard reception. The service requester sends a service component execution request message to a standard reception, and an HCSC server performs service component execution.

The user acquires a stub from the development environment and uses this stub to send a request to the synchronous reception (SessionBean). Therefore, the service requester must be installed such that it can utilize the acquired stub.

The following figure shows the relationship between a service requester for a JAX-WS engine that sends requests to a standard synchronous reception (SessionBean) and an HCSC server.

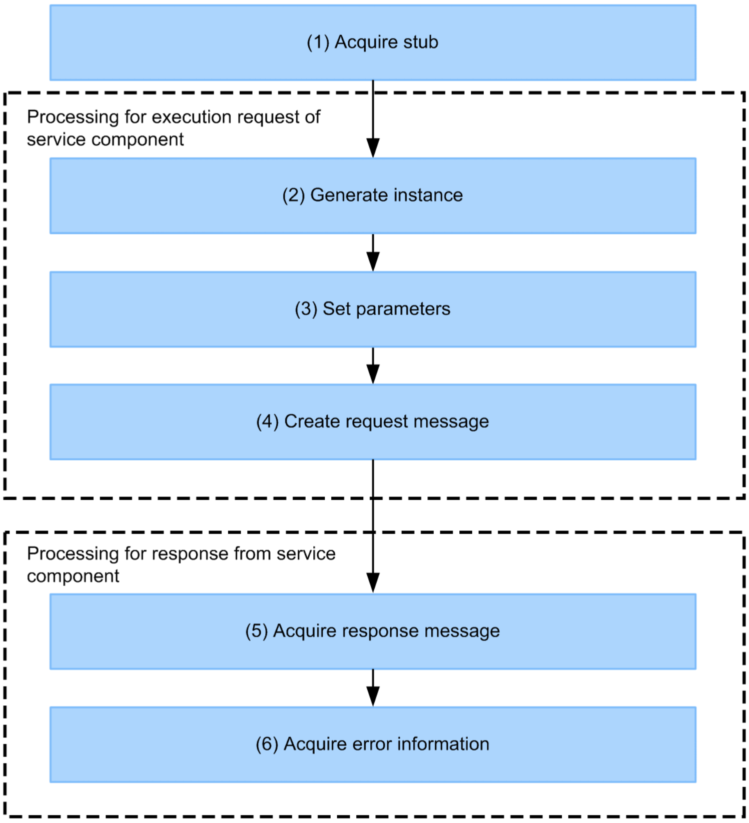
Figure 8–15: Relationship between a service requester that sends requests to a standard synchronous reception (Web Service) and an HCSC server(JAX-WS engine)


Service component execution request
Execute service component
Execution environment
Service requester
Stub
Machine on which service requester is running
Adapter
Standard synchronous reception （SessionBean）
HCSC server
EJB Container

8.4.1 Procedure for Creating a Service Requester (Standard Synchronous Reception (SessionBean))

This subsection explains how to create a service requester that sends a service component execution request to a standard synchronous reception (SessionBean) and invokes a service component. The creation workflow is shown in the following figure.

Figure 8–16: Service requester creation work flow (standard synchronous reception (SessionBean))


(1) Acquire stub
Processing for execution request of service component
(2) Generate instance
(3) Set parameters
(4) Create request message
Processing for response from service component
(5) Acquire response message
(6) Acquire error information

The tasks in the individual steps are described below.

(1) Stub acquisition

From the HCSC Component Information Display screen in the development environment, acquire the stub that corresponds to the J2EE server (JNDI name) to which the HCSC server is deployed. For details about stub acquisition, see *8.4.2 Acquiring Stubs*.

(2) Instance generation

To invoke the method of the synchronous reception (SessionBean), create an Enterprise Bean instance from the stub acquired in (1). For details about instance creation, *8.4.3 Creating Instances*.

(3) Parameter specification

Specify the parameters that become the arguments of the method of the synchronous reception (SessionBean). For details about parameter specification, see *8.4.4 Specifying Parameters*.

(4) Request message creation

Create a request message for requesting service component execution. For details about request message creation, see *8.4.5 Creating Request Messages*.

(5) Response message acquisition

Acquire a response message corresponding to the service component execution request from the synchronous reception (SessionBean). For details about response message acquisition, see *8.4.6 Acquiring Response Messages*.

(6) Error information acquisition

If an error occurs at the request-destination service component, the HCSC server, or the EJB container, acquire the error information and take corrective action according to the information. For details about error information acquisition, see *8.4.7 Acquiring Error Information*.

8.4.2 Acquiring Stubs

Acquire the stub that corresponds to the J2EE server (JNDI name) to which the HCSC server is deployed. The stub acquisition method is described blow.

1. From the Eclipse menu, choose HCSC-Definer, and then Published Services List.
   HCSC components are listed in Published Services List in the tree view.
2. From the HCSC component list, select and double-click the service component (HCSC component) to be invoked.
   The information about the HCSC component is displayed in the HCSC Component Information Display screen. For details about this screen, see the manual *Cosminexus Service Platform Overview*.
   The information about the HCSC server that becomes the service component request destination is displayed in Server information.
3. Click Stub acquisition.
   A dialog box for saving a stub file opens.
   Specify the file saving destination and acquire the stub.

The configuration of the acquired stub is shown below.

```
/current-directory
 -cscmsg_ejb_client.jar
```

8.4.3 Creating Instances

To invoke the method of the synchronous reception (SessionBean), use the acquired stub to create an Enterprise Bean instance. The procedure for creating an Enterprise Bean instance is described below.

1. Create a JNDI naming context to be used for retrieving an EJB home object reference.

   Example:

   ```
   javax.naming.Context ctx = new javax.naming.InitialContext();
   ```

2. Using the created JNDI naming context, acquire an EJB home object reference.
   To acquire an EJB home object reference, use a user-specified name space or an EJB container name space for retrieval. For details about retrieving and acquiring an EJB home object reference, see the manual *Cosminexus Application Server Function Guide - Basic Development for EJB Container*.

   Example:

   ```
   Object objref
        = initial.lookup("HITACHI_EJB/SERVERS/" + "J2EE server-name"
          + "/EJB/CSCMsgSyncServiceDelivery/CSCMsgSyncServiceDeliveryEJB");

   CSCMsgSyncServiceDeliveryHome home
        = (CSCMsgSyncServiceDeliveryHome)PortableRemoteObject
           .narrow(objref, CSCMsgSyncServiceDeliveryHome.class);
   ```

3. Use the create method of the EJB home object to create an Enterprise Bean instance. Using the created Enterprise Bean instance, invoke the method (Enterprise bean method) of the synchronous reception (SessionBean).

   Example: When the request message is in XML

   ```
   CSCMsgSyncServiceDelivery reception = home.create();
         // Enterprise Bean instance creation
   String result = reception .invokeXML(    // method invocation
                  serviceName,              // service name
                  clientID,                 // client correlation ID
                  requestFormatID,          // request format ID
   ```

```
               responseFormatID,         // response format ID
               operationName,            // operation name
               userData);                // user message
```

Example: When the request message is binary

```
CSCMsgSyncServiceDelivery reception = home.create();
      // Enterprise Bean instance creation
byte[] resultBinary = reception .invokeBinary(  // method invocation
               serviceName,                 // service name
               clientID,                    // client correlation ID
               requestFormatID,             // request format ID
               responseFormatID,            // response format ID
               operationName,               // operation name
               userDataBinary.length,       // message length
               userDataBinary);             // user message
```

! Important note

A binary request message can be sent only when the message format used on the service component side is binary.

8.4.4 Specifying Parameters

To invoke a method of the synchronous reception (SessionBean), specify the parameters that become the arguments of the method. The following figure shows parameter details.

Table 8–12: Parameter details (standard synchronous reception (SessionBean))

Parameter name	Data type	Parameter		Explanation
		invokeXML	invokeBinary	
Service name	`java.lang.String`	`serviceName`		This is the service name of the request destination. This parameter is required. For the service name of the request destination, specify the adapter or business process defined in the development environment.
Client correlation ID	`java.lang.String`	`clientID`		This is a correlation identifier for uniquely identifying the request message from the service requester. Specify alphanumeric characters, underscore (_), period (.), and hyphen (-) up to 255 characters. This parameter is used to map the request message from the service requester to the execution history, logs, and traces managed by the HCSC server. Therefore, specify a different ID for each request message sent to the HCSC server. To omit the client correlation ID, specify `NULL`.
Request format ID	`java.lang.String`	`cscRequestFormatID`		This is an ID for uniquely identifying the request message format from the service requester. Specify `NULL` for this parameter.
Response format ID	`java.lang.String`	`cscResponseFormatID`		This is an ID for uniquely identifying the response message from the HCSC server. Specify `NULL` for this parameter.
Operation name	`java.lang.String`	`serviceOperationName`		This is an operation name corresponding to the service name at the request destination.# This operation name specifies a service component defined in the development environment. Specify the operation name with

Parameter name	Data type	Parameter		Explanation
		invokeXML	invokeBinary	
Operation name	`java.lang.String`	`serviceOperationName`		NCName definition characters of XMLSchema within 255 bytes. This parameter is required when the service component at the request destination is a synchronous service (Web Services or SessionBean) or business process. When the service at the request destination is an asynchronous service, the operation name can be omitted. To omit it, specify `NULL`.
User message	`java.lang.String`	`msg`	--	This is the request message from the service requester.# Specify this parameter when the request message is in XML. If there is no request message, specify `NULL` or an empty character (`""`). For details about request messages, see *8.4.5 Creating Request Messages*.
User message length	`int`	--	`requestMessageLength`	This is the request message length. Specify this parameter when the request message is binary. This parameter is required when the request message is binary. If there is no request message, specify 0.
User message	`byte[]`	--	`msg`	This is the request message from the service requester.# Specify this parameter when the request message is binary. For details about request messages, see *8.4.5 Creating Request Messages*. If there is no request message, specify `NULL` or a 0-byte byte array.

Legend:

--: Cannot be specified.

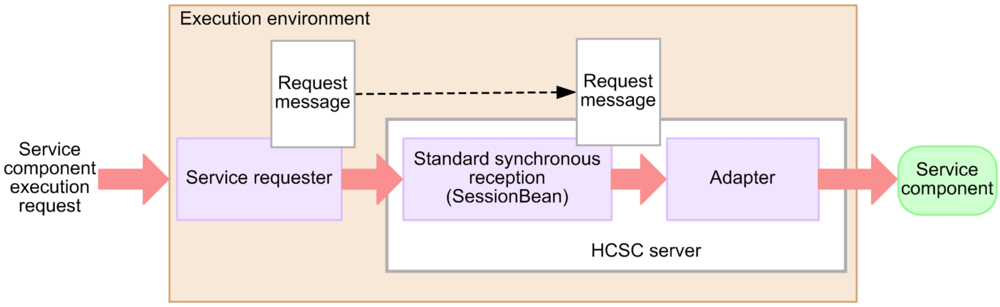
#

When the service component protocol of request destination is SOAP, decide an operation to be invoked from the name of a root element of user message (in the case of data transformation, it is the name of root element of the message after data transformation). Therefore, take note that if you specify an invalid name in the root element of user message, an unintended operation may be invoked.

8.4.5 Creating Request Messages

Create a request message for requesting a service component from the service requester to the synchronous reception (SessionBean) of the HCSC server. The contents of the request message to be sent from the service requester must be created in the same message format as that used on the service component side. The following figure shows how a request message is sent.

Figure 8–17: Sending a request message (standard synchronous reception (SessionBean))



The request message to be sent from the service requester to the synchronous reception (SessionBean) can be either an XML message or a binary message. Which message type is to be used depends on the protocol being used by the service component side.

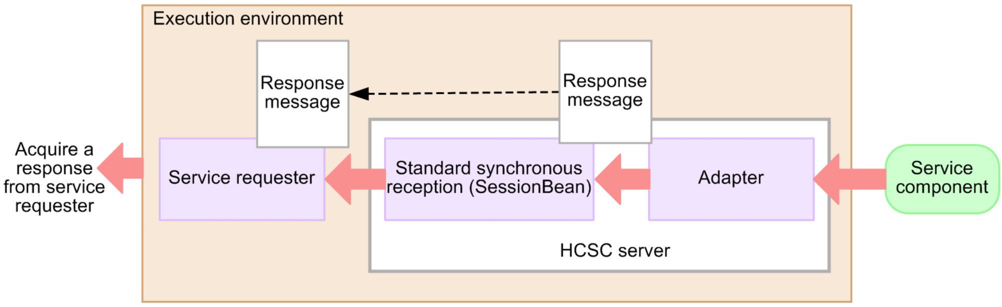
For details about XML and binary request messages, see *8.2.6 Creating Request Messages*.

For details about how to send a normal request message, see the contents about service invocation using a SessionBean, in the manual *Cosminexus Service Platform System Setup and Operation Guide*.

8.4.6 Acquiring Response Messages

Acquire a response message corresponding to the service component execution request from the synchronous reception (SessionBean) of the HCSC server. The service requester acquires a response message that has the same message format as the service component side. The following figure shows how a response message is acquired.

Figure 8–18: Response message acquisition (standard synchronous reception (SessionBean))



The service requester acquires a response message whose message type is XML. If there is no response message from the service component, `NULL` is received.

For details about receiving XML response messages and `NULL`, see *8.2.7 Acquiring Response Messages*.

8.4.7 Acquiring Error Information

If an error occurs at the request-destination service component, the HCSC server, or the EJB container, acquire the error information and take corrective action according to the information.

You can acquire error information by acquiring the CSCMsgServerException class on the service requester side. For details about the CSCMsgServerException class, see *8.4.7(2) CSCMsgServerException class*. For details on how errors are informed, see the contents related to the troubleshooting when SessionBean is executed in the manual *Cosminexus Service Platform System Setup and Operation Guide*.

To acquire error details, you use the getErrorMessage and getErrorCode methods. To acquire error information (SOAPFault) from Web Services, you use the checkSoapFault method.

(1) Service requester-side installation example

A service requester-side installation example for acquiring the CSCMsgServerException class is shown below.

```
          ...
} catch (CSCMsgServerException e) {
    System.err.println("Exception ErrorMessage = "
                + e.getErrorMessage() );
    System.err.println("Exception ErrorCode = "
                + e.getErrorCode() );
    if (e.checkFaultMessage() == true) {
        System.err.println("Exception ProcessInstanceID = "
                + e.getProcessInstanceID() );
        System.err.println("Exception FaultCode = "
                + e.getFaultCode() );
        System.err.println("Exception FaultString = "
                + e.getFaultString() );
        System.err.println("Exception FaultActor = "
                + e.getFaultActor() );
        System.err.println("Exception FaultDetails = "
                + new String(e.getFaultDetail(), "UTF-8"));
    System.err.println("Exception FaultName = "
                + e.getFaultName() );
    }
}
          ...
```

(2) CSCMsgServerException class

This is an exception class acquired by the service requester.

(a) Class definition

Package

`jp.co.Hitachi.soft.csc.msg.message.reception`

Class

```
public class CSCMsgServerException
extends java.lang.Exception
```

(b) Field list

Field name	Data type	Explanation
`errorMessage`	`java.lang.String`	Contents of the following exceptions: • Exceptions detected inside HCSC-Messaging • Fault from a service component or business process
`errorCode`	`java.lang.String`	Error code corresponding to the following exceptions: • Exceptions detected inside HCSC-Messaging • Fault from a service component or business process
`processInstanceID`	`java.lang.String`	Instance ID information of a business process A value is set when an error occurs in the business process.
`cscmsgFaultCode`	`java.lang.String`	FaultCode information from a service component (Web Services), business process, or custom adapter
`cscmsgFaultString`	`java.lang.String`	FaultString information from a service component (Web Services), business process, or custom adapter
`cscmsgFaultActor`	`java.lang.String`	FaultActor information from a service component (Web Services), business process, or custom adapter
`cscmsgFaultDetail`	`byte[]`	Detail information from a service component (Web Services), business process, or custom adapter
`faultName`	`java.lang.String`	Fault name (exception name) information from a service (Web Services or SessionBean) or business process A value is set in the following cases:

Field name	Data type	Explanation
`faultName`	`java.lang.String`	• In the case of SOAP Fault of a user-defined exception from a service component (Web services or SessionBean) • In the case of a fault from business process No value is set in the case of the SOAP Fault error from Web services that define URI of targetNamespace in the SOAP Fault operation definition file. For details about the SOAP Fault operation definition file, see the manual *Cosminexus Service Platform Reference*.

(c) Method list

Method name	Data type	Explanation
`getErrorMessage`	`java.lang.String`	Acquires error messages.
`getErrorCode`	`java.lang.String`	Acquires error codes.
`checkFaultMessage`	`Boolean`	Determines whether there is fault information from a service component, business process, or custom adapter.
`getFaultCode`	`java.lang.String`	Acquires FaultCode information.
`getFaultString`	`java.lang.String`	Acquires FaultString information.
`getFaultActor`	`java.lang.String`	Acquires FaultActor information.
`getFaultDetail`	`byte[]`	Acquires Detail information.
`getProcessInstanceID`	`java.lang.String`	Acquires the instance ID of a business process.
`getFaultName`	`java.lang.String`	Acquires a fault name (exception name).

(d) Method detail

- **getErrorMessage**

Explanation

Acquires error messages.

Use this method for acquiring the contents of the following exceptions:

- Exceptions detected inside HCSC-Messaging
- Fault from a service component or business process

Format

```
public java.lang.String getErrorMessage()
```

Parameter

None

Return value

Error message

Exception

None

- **getErrorCode**

Explanation

Acquires error codes.

Use this method for acquiring the error codes corresponding to the following exceptions:

- Exceptions detected inside HCSC-Messaging
- Fault from a service component or business process

Format

```
public java.lang.String getErrorCode()
```

Parameter

None

Return value

Error code

Exception

None

- **checkFaultMessage**

Explanation

Determines whether there is fault information from a service component, business process, or custom adapter.

If the service component or business process is Web Services, determines whether the contents of the error returned as SOAPFault is included.

Also determines whether an individual error from a custom adapter is included.

Format

```
public boolean checkFaultMessage()
```

Parameter

None

Return value

`true`: Fault message information exists.

`false`: Fault message information does not exist.

Exception

None

- **getFaultCode**

Explanation

Acquires FaultCode information from a service component (Web services), business process, or custom adapter.

Format

```
public java.lang.String getFaultCode()
```

Parameter

None

Return value

faultcode

Exception

None

- **getFaultString**

Explanation

Acquires FaultString information from a service component (Web Services), business process, or custom adapter.

Format

```
public java.lang.String getFaultString()
```

Parameter

None

Return value

faultstring

Exception

None

- **getFaultActor**

Explanation

Acquires FaultActor information from a service component (Web Services), business process, or custom adapter.

Format

```
public java.lang.String getFaultActor()
```

Parameter

None

Return value

faultactor

Exception

None

- **getFaultDetail**

Explanation

Acquires Detail information from a service component (Web Services), business process, or custom adapter.

Transfers the Detail information set up by the service component to the service requester in a byte array. Therefore, the acquired byte array must be converted to a character string.

Format

```
public byte[] getFaultDetail()
```

Parameter

None

Return value

detail

Exception

None

- **getProcessInstanceID**

Explanation

Acquires the instance ID of a business process.

Format

```
public String getProcessInstanceID()
```

Parameter

None

Return value

String

Exception

None

- **getFaultName**

Explanation

Acquires a fault name (exception name).

Format

```
public String getFaultName()
```

Parameter

None

Return value

String

Exception

None

(e) Subclass list

Class name	Subclass name and explanation	
`CSCMsgServerException`	`CSCMsgServiceException` Service component exception	`CSCMsgServiceUserException` Fault information or Exception from a service component
		`CSCMsgBusinessProcessUserException` Fault information or Exception from a business process
		`CSCMsgServiceExecuteException` Exception when an exception that is not a user-defined exception is returned from a service component (Web Services)
	`CSCMsgServiceDeliveryException` Service component request delivery error	`CSCMsgLocationSearchException` Location search failure
		`CSCMsgRoutingExecException` Routing failure
		`CSCMsgDelivererExecException` Message delivery failure
		`CSCMsgDBQServiceDeliveryException` Exception when a standard reception of an asynchronous database queue invokes a database queue service component
	`CSCMsgServerParameterException` Input parameter error	--
	`CSCMsgServerInternalException` Error other than those described above	--

Legend:

--: No subclass is available.

8.4.8 Creating a Service Requester That Sends a Request for Business Process Re-execution (SessionBean)

You can create a service requester that sends requests to a standard synchronous reception (SessionBean) for business process re-execution.

The procedure for creating a service requester that sends a request to the standard synchronous reception (SessionBean) for business process re-execution is the same as that for creating an ordinary service requester that sends requests to the standard synchronous reception (SessionBean). For details about the creation procedure, see *8.4.1 Procedure for Creating a Service Requester (Standard Synchronous Reception (SessionBean))*.

The tasks in the individual procedures are described below.

(1) Stub acquisition

The stub acquisition method is the same as that used for creating an ordinary service requester that sends requests to the standard synchronous reception (SessionBean).

For details about stub acquisition, see *8.4.2 Acquiring Stubs*.

(2) Instance generation

The instance creation method is the same as that used for creating an ordinary service requester that sends requests to the standard synchronous reception (SessionBean).

For details about instance creation, *8.4.3 Creating Instances*.

To invoke a method that sends a request for business process re-execution, use invokeBPXML().

An example of a method that sends a request for business process re-execution is shown below.

The following example shows the method for requesting re-execution of the business process.

Example: Requesting business process re-execution

```
String result = ws.invokeBPXML(          // method invocation
                       serviceName,        // service name
                       bpRequestType,      // request type for business process
                       bpProcessId,        // process ID for business process
                       clientID,           // client correlation ID
                       requestFormatID,    // request format ID
                       responseFormatID,   // response format ID
                       operationName,      // operation name
                       userData);          // user message
```

(3) Parameter specification

The parameters that become the arguments of the method are different from those used in creating an ordinary service requester that sends requests to the standard synchronous reception (SessionBean). The following table shows the details of the parameters that are specified for a service requester that sends a request to a synchronous reception (SessionBean) for business process re-execution.

Table 8–13: Parameter details (standard synchronous reception (SessionBean)/business process re-execution request)

Parameter name	Data type	Parameter (invokeBPXML)	Explanation
Service name	java.lang.String	serviceName	This is the service name of the request destination. This parameter is required. For the service name of the request destination, specify the business process defined in the development environment.
Request type for business process	java.lang.String	cscBpRequestType	Indicates a request message type. To request business process re-execution, specify a RECOVER string.#
Process ID for business process	java.lang.String	cscBpProcessID	This is a business process instance ID. Specify either a value acquired from error information or the value that is output to the message log.
Client correlation ID	java.lang.String	clientID	This is a correlation identifier for uniquely identifying the request message from the service requester. Specify NULL for this parameter.
Request format ID	java.lang.String	cscRequestFormatID	This is an ID for uniquely identifying the request message format from the service requester. Specify NULL or an empty character ("") for this parameter.#
Response format ID	java.lang.String	cscResponseFormatID	This is an ID for uniquely identifying the response message from the HCSC server.

Parameter name	Data type	Parameter (invokeBPXML)	Explanation
Response format ID	`java.lang.String`	`cscResponseFormatID`	Specify `NULL` or an empty character (`""`) for this parameter.#
Operation name	`java.lang.String`	`serviceOperationName`	This is an operation name corresponding to the service name at the request destination. Specify `NULL` or an empty character (`""`) for this parameter.#
User message	`java.lang.String`	`msg`	This is the request message from the service requester. Specify `NULL` or an empty character (`""`) for this parameter.#

#

A fixed value (`RECOVER`, `NULL`, or an empty character (`""`)) is set as the specification value for these parameters. If you specify a value other than these fixed values, correct operation cannot be guaranteed.

(4) Request message creation

The request message creation method is the same as that used for creating an ordinary service requester that sends requests to the standard synchronous reception (SessionBean).

For details about request message creation, see *8.4.5 Creating Request Messages*.

(5) Response message acquisition

The response message acquisition method is the same as that used for creating an ordinary service requester that sends requests to the standard synchronous reception (SessionBean).

For details about response message acquisition, see *8.4.6 Acquiring Response Messages*.

(6) Error information acquisition

The method of installing a service requester for acquiring error information is the same as that used for creating an ordinary service requester that sends requests to the standard synchronous reception (SessionBean).

For details about error information acquisition, see *8.4.7 Acquiring Error Information*.

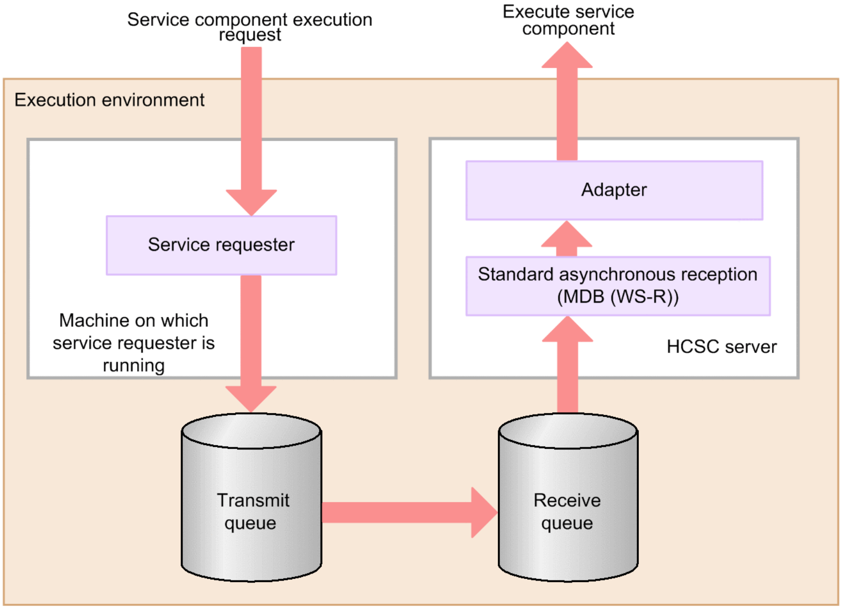
8.5 Service Requester That Sends Requests to a Standard Asynchronous Reception (MDB (WS-R))

A service requester that sends request messages to a standard asynchronous reception (MDB (WS-R)) communicates with the standard reception via a transmission queue and a receive queue.

A request message sent from the service requester to the transmission queue is sent to the asynchronous reception (MDB (WS-R)) via the receive queue.

The following figure shows the relationship between a service requester that sends requests to a standard asynchronous reception (MDB (WS-R)) and an HCSC server.

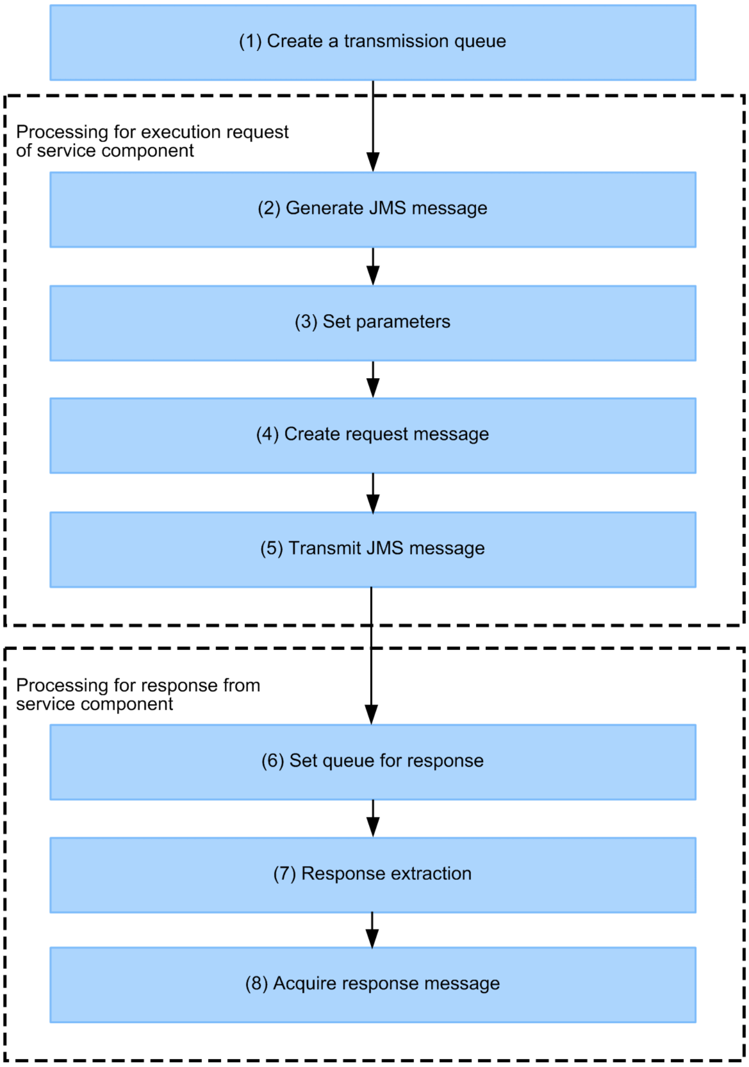
Figure 8–19: Relationship between a service requester that sends requests to a standard asynchronous reception (MDB (WS-R)) and an HCSC server


Service component execution request
Execute service component
Execution environment
Service requester
Machine on which service requester is running
Adapter
Standard asynchronous reception (MDB (WS-R))
HCSC server
Transmit queue
Receive queue

8.5.1 Procedure for Creating a Service Requester (Standard Asynchronous Reception (MDB (WS-R))

This subsection explains how to create a service requester that sends a service component execution request to a standard asynchronous reception (MDB (WS-R)) and invokes a service component. The creation workflow is shown in the following figure.

Figure 8–20: Service requester creation work flow (standard asynchronous reception (MDB (WS-R))


(1) Create a transmission queue
Processing for execution request of service component
(2) Generate JMS message
(3) Set parameters
(4) Create request message
(5) Transmit JMS message
Processing for response from service component
(6) Set queue for response
(7) Response extraction
(8) Acquire response message

The tasks in the individual steps are described below.

(1) Creating a transmission queue

Create a transmission queue needed for sending a request message from the service requester. For details about transmission queue creation, see *8.5.2 Creating a Transmission Queue*.

(2) Creating a JMS message

Create a QueueSender object and a JMS message in order to send a request to the asynchronous reception (MDB (WS-R)) for service component execution. For details about QueueSender object and JMS message creation, see *8.5.3 Creating JMS Messages*.

(3) Parameter specification

Specify parameters for the JMS message created in *8.5.1(2) Creating a JMS message*. For details about parameter specification, see *8.5.4 Specifying Parameters*.

(4) Request message creation

Create a request message for requesting service component execution. For details about request message creation, see *8.5.5 Creating Request Messages*.

(5) JMS message transmission

Send the JMS message to the transmission queue. For details about JMS message transmission, see *8.5.6 Sending JMS Messages*.

(6) Response queue setup

To receive a response from a synchronous service component, set up a response queue (transmission queue). For details about response queue setup, see *8.5.7 Setting Up a Response Queue*.

(7) Response extraction

Extract the response that was sent from the response queue (transmission queue) to the receive queue. For details about response extraction, see *8.5.8 Extracting Responses*.

(8) Response message acquisition

Acquire the response message from the extracted response. For details about response message acquisition, see *8.5.9 Acquiring Response Messages*.

8.5.2 Creating a Transmission Queue

Create a transmission queue needed for sending a request message from the service requester. The transmission queue sends the request message from the service requester to the receive queue provided by the HCSC server.

To create a transmission queue, use the hrmmkaddr and hrmmkque commands of Cosminexus RM.

This subsection explains the specification values for the options of the hrmmkaddr and hrmmkque commands. For details about the options not explained here, see the manual *Cosminexus Application Server Cosminexus Reliable Messaging*.

(1) hrmmkaddr command option specification value

The option specification value of the hrmmkaddr command is described below.

`-u` *destination-address*

Specify the destination address of the HCSC server (URL of the Web application used for transmit between queues of the after moved system).

`http://`*host-name*`:`*port-name*#1`/`*context-root*#2`/services/HRMReceiver/`

`https://`*host-name*`:`*port-number*#1`/`*context-root*#2`/services/HRMReceiver/`

#1

This is the HCSC server's URL.

#2

This is the context root of the Web application used for transmit between queues of receiving Cosminexus RM. The default value is `uCosminexusRM`.

(2) hrmmkque command option specification values

`-t` *queue-type*

Specify transmit (transmission queue).

`-m` *queue-persistence*

Specify persistent (persistent queue attribute).

`-v` *transfer-destination-queue-name*

Specify either of the following:

CSCHCSC-server-nameACPT_RCVQ

Specify this option when the HCSC server has a load-balancing cluster configuration.

CSCHCSC-cluster-nameACPT_RCVQ

Specify this option when the HCSC server has the HAcluster configuration.

HCSC-cluster-name is the cluster name of the HCSC server determined during HCSC server configuration deployment in the development environment and the operating environment.

`-i` *queue-transfer-mode*

Specify Compatible (compatible mode).

8.5.3 Creating JMS Messages

To send a service component execution request to the standard asynchronous reception (MDB (WS-R)), create a QueueSender object and a JMS message, and send the created JMS message to the transmission queue. For details about developing an application for sending JMS messages, see the manual *Cosminexus Application Server Cosminexus Reliable Messaging*.

To create a JMS message:

1. Define a startup process (lookup) for QueueConnectionFactory and Queue.

   Example:

```
Context ic = new InitialContext();
         ...
QueueConnectionFactory qcFactory
      = (QueueConnectionFactory) ic.lookup("java:comp/env/jms/qcf");
         ...
Queue queue = (Queue) ic.lookup("java:comp/env/jms/queue");
         ...
```

2. Create QueueSession.

   Example:

```
QueueSession qSession
    = qConnection.createQueueSession(true, Session.AUTO_ACKNOWLEDGE);
```

3. From QueueSession, create QueueSender.

   Example:

```
qSender = qSession.createSender(queue);
```

4. From QueueSession, create a JMS message.

   Example: When the request message is in XML

```
TextMessage textMessage = qSession.createTextMessage();
```

   Example: When the request message is binary

```
BytesMessage bytesMessage = qSession.createBytesMessage();
```

8.5.4 Specifying Parameters

To invoke a method of the asynchronous reception (MDB (WS-R)), specify parameters for the JMS message.

(1) Property specification

Specify properties for the JMS message. For details about the properties to be specified, see *8.5.4(4) Parameter details*.

Example: When the request message is in XML

```
textMessage = qSession.createTextMessage();
textMessage.setStringProperty("CSCServiceName", serviceName);
     // Service component name
textMessage.setStringProperty("CSCCorrelationID", clientID);
     // Client correlation ID
textMessage.setStringProperty("CSCRequestFormatID", requestFormatID);
     // Request format ID
textMessage.setStringProperty("CSCResponseFormatID", responseFormatID);
     // Response format ID
textMessage.setStringProperty("CSCServiceOperationName", operationName);
     // Operation name
textMessage.setStringProperty("CSCReplyToQueueName", replyToQueueName);
     // Queue name for response
textMessage.setStringProperty("CSCMessageType", "XML");
     // Message type
```

! Important note

A binary request message can be sent only when the message format used on the service component side is binary. A request message to be sent to a database queue service component must be converted to binary format. For binary conversion, use the same encoding method for the service requester side and the service component side.

(2) Headers and properties that are inherited

The headers and properties listed below are inherited from the request side to the queue on the service component side. Therefore, set up these headers and properties as needed.

- JMSReplyTo
- JMSCorrelationID#1
- JMSType
- JMSXGroupID
- JMSXGroupSeq
- User-specific property#2

#1

The maximum specifiable range is 255 bytes.

#2

Note that the following properties are not inherited by the queue on the service component side:

- JMS-defined property name that begins with JMSX
- Provider-specified property name that begins with JMS_
- Property name that begins with CSC
- Property name that begins with HCSC

(3) Payload specification

Specify a request message (user message) in the JMS message payload. For details about the payload, see *8.5.4(4) Parameter details*.

Example: When the request message is in XML

```
textMessage.setText( userData );
```

(4) Parameter details

The following figure shows parameter details.

Table 8–14: Parameter details (standard asynchronous reception (MDB (WS-R)))

Parameter name	Data type	Parameter		Property/ payload	Explanation
		TextMessage	BytesMessage		

Service name	String	`StringProperty`
		`("CSCServiceName")`
				Property	This is the service name of the request destination.
					This parameter is required.
					For the service name of the request destination, specify the adapter or business process defined in the development environment.

Client correlation ID	String	`StringProperty`
		`("CSCCorrelationID")`
				Property	This is a correlation identifier for uniquely identifying the request message from the service requester.
					Specify alphanumeric characters, underscore (_), period (.), and hyphen (-) up to 255 characters.
					This parameter is used to map the request message from the service requester to the execution history, logs, and traces managed by the HCSC server. Therefore, specify a different ID for each request message sent to the HCSC server.
					By not specifying this parameter, you can omit the client correlation ID.

Request format ID	String	`StringProperty`
		`("CSCRequestFormatID")`
				Property	This is an ID for uniquely identifying the request message format from the service requester.
					Specify alphanumeric characters, underscore (_), period (.), and hyphen (-) up to 1,024 characters.
					Specify `NULL` for this parameter. If you do not specify this parameter, `NULL` is specified.

Response format ID	String	`StringProperty`
		`("CSCResponseFormatID")`
				Property	This is an ID for uniquely identifying the response message from the HCSC server.
					Specify alphanumeric characters, underscore (_), period (.), and hyphen (-) up to 1,024 characters.
					Specify `NULL` for this parameter. If you do not specify this parameter, `NULL` is specified.

Operation name	String	`StringProperty`
		`("CSCServiceOperationName")`
				Property	This is an operation name corresponding to the service name at the request destination.#
					This operation name specifies a service component defined in the development environment. Specify the operation name with NCName definition characters of XMLSchema within 255 bytes.
					This parameter is required when the service component at the request destination is a synchronous service (Web Services or SessionBean) or business process.
					When the service at the request destination is an asynchronous service, the operation name can be omitted. By not specifying

Parameter name	Data type	Parameter		Property/ payload	Explanation
		TextMessage	BytesMessage		
Operation name	String	StringProperty ("CSCServiceOperationName")		Property	this parameter, you can omit the operation name.
Response queue name	String	StringProperty ("CSCReplyToQueueName")		Property	This is the queue name that receives a response from a service component or business process when the service component at the request destination is a synchronous service (Web Services or SessionBean) or business process. By specifying this parameter when you cannot determine whether the service component at the request destination is synchronous or asynchronous, you can receive responses. For details about the response queue, see *8.5.7 Setting Up a Response Queue*. By not specifying this parameter, you can omit the response queue name. If you omit this parameter, NULL is specified, and you cannot receive responses.
Message type	String	StringProperty ("CSCMessageType")		Property	Specify one of the following request message types: • For XML message (whose payload is TextMessage): XML • For binary message (whose payload is BytesMessage): Binary This parameter must be specified when there is a request message. When there is no request message, you can omit the message type by not specifying this parameter.
User message	--	TextMessage	N	Payload	This is the request message from the service requester.[#] Specify this parameter when the request message is in XML. If there is no request message, you do not need to specify this parameter. For details about request messages, see *8.5.5 Creating Request Messages*.
User message length	Long	N	LongProperty ("CSCMessageLength")	Property	This is the request message length. Specify this parameter when the request message is binary.
User message	--	N	BytesMessage	Payload	This is the request message from the service requester.[#] Specify this parameter when the request message is binary. If there is no request message, you do not need to specify this parameter. For details about request messages, see *8.5.5 Creating Request Messages*.

Legend:

--: Not applicable

N: Cannot be specified.

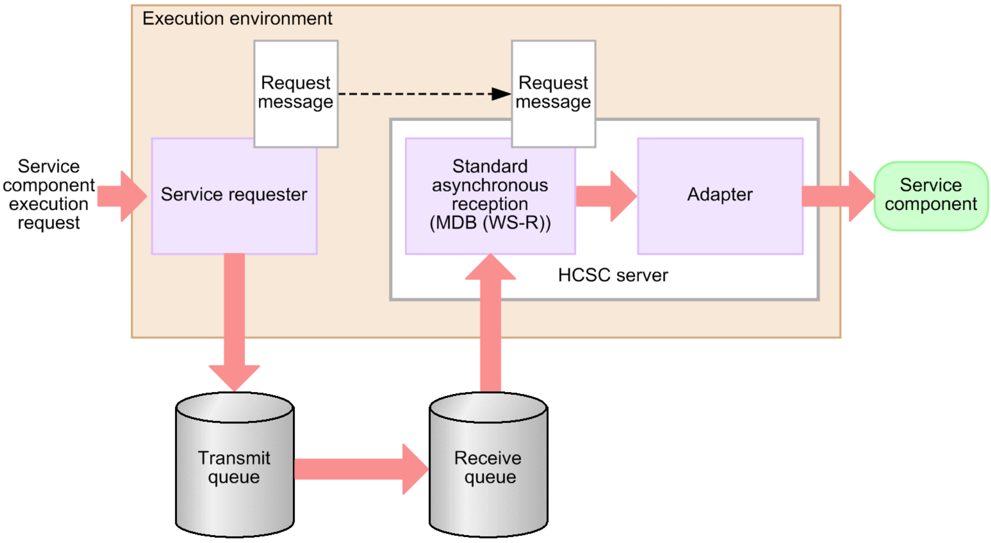
#

When the service component protocol of request destination is SOAP, decide an operation to be invoked from the name of a root element of user message (in the case of data transformation, it is the name of root element of the message after data transformation). Therefore, take note that if you specify an invalid name in the root element of user message, an unintended operation may be invoked.

8.5.5 Creating Request Messages

Create a request message for requesting a service component from the service requester to the asynchronous reception (MDB (WS-R)). The contents of the request message to be sent from the service requester must be created in the same message format as that used on the service component side. The following figure shows how a request message is sent.

Figure 8–21: Sending a request message (standard asynchronous reception (MDB (WS-R)))



The request message to be sent from the service requester to the asynchronous reception (MDB (WS-R)) can be either an XML message or a binary message. Which message type is to be used depends on the protocol being used by the service component side.

For details about XML and binary request messages, see *8.2.6 Creating Request Messages*.

For details about how to send a normal request message, see the contents about service invocation using an MDB (WS-R), in the manual *Cosminexus Service Platform System Setup and Operation Guide*.

8.5.6 Sending JMS Messages

Send the JMS message with parameters specified to the transmission queue.

To send a JMS message:

1. Start a local transaction and register the JMS message.

   Example:

```
qSender.send(message);
```

2. Define the termination process.

   Example:

```
qSession.commit();        // local transaction commit
qSender.close();          // QueueSender release
qSession.close();         // QueueSession release
qConnection.close();      // QueueConnection release
```

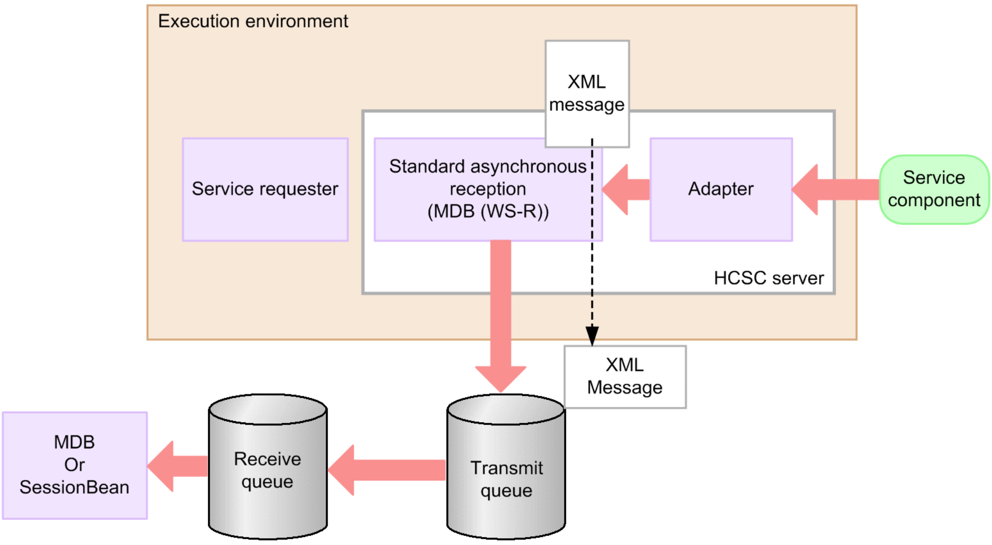
8.5.7 Setting Up a Response Queue

The standard asynchronous reception (MDB (WS-R)) is a standard reception for asynchronous request messages, and thus is used, as a rule, when there is no response from a service component. However, when the asynchronous reception (MDB (WS-R)) sends a request to a synchronous service component (Web Services or SessionBean), a response may be returned from the synchronous service component in some cases. In such a case, by specifying a response queue (transmission queue) when making a request, the service requester can receive responses from a synchronous service component. For details about specifying a response queue when making a request, see *8.5.4 Specifying Parameters*.

The standard asynchronous reception (MDB (WS-R)) sends an XML message (whose payload is TextMessage), which is a response from a synchronous service component, to the transmission queue. The response received by the transmission queue is sent to the receive queue. By extracting the response sent to the receive queue, you can acquire the response message from the service component. For details about response extraction, see *8.5.8 Extracting Responses*.

The following shows the relationship among the standard asynchronous reception (MDB (WS-R)), the transmission queue, and the receive queue.

Figure 8–22: Relationship among the standard asynchronous reception (MDB (WS-R)), the transmission queue, and the receive queue



Transmission queue (response queue)

Create a transmission queue, which functions as a response queue, inside the J2EE server (Cosminexus RM) on which the HCSC server is running.

For the transfer destination (destination address) of the transmission queue, specify the receive queue. As needed, specify transmission queues at non-transfer destinations according to the user application.

For details about how to create transmission queues, see *8.5.2 Creating a Transmission Queue* and the section related to transmission queues in the manual *Cosminexus Application Server Cosminexus Reliable Messaging*.

Receive queue (local queue)

Create a receive queue (local queue) inside the J2EE server (Cosminexus RM) on which the service requester is running.

Specify a receive queue according to the user application as needed.

For details about how to create receive queues, see the section related to local queues in the manual *Cosminexus Application Server Cosminexus Reliable Messaging*.

8.5.8 Extracting Responses

A response can be extracted from the receive queue based on either MDB or SessionBean installation. For details about how to send a normal request message, see the contents about the response queue in service invocation using an MDB (WS-R), in the manual *Cosminexus Service Platform System Setup and Operation Guide*.

(1) Response contents

A service component sends the following contents in a response:

- Message type (StringProperty property ("CSCMessageType"))
  Because the message type of the response message is XML, XML is set in the StringProperty property.
- Response message (payload (TextMessage)) from the service component
  An XML (TextMessage) response message is set.
  When a synchronous service component is executed and there is no response message, TextMessage without payload is sent.

(2) Headers and properties that are inherited

For the headers and properties listed below, the contents specified when the service requester sends a request for the service component are inherited by the receive queue via the response queue.

(a) Properties related to the HCSC server

- StringProperty("CSCServiceName")
  This is the service name of the request destination specified by the service requester.
- StringProperty("CSCCorrelationID")
  This is a correlation identifier specified by the service requester for uniquely identifying the request message from the service requester.
- StringProperty("CSCResponseFormatID")
  NULL is set.
- StringProperty("CSCServiceOperationName")
  This is an operation name corresponding to the service name of the request destination specified by the service requester.

(b) Headers and properties related to Cosminexus RM

- JMSReplyTo
- JMSCorrelationID
- JMSType
- JMSXGroupID
- JMSXGroupSeq
- User-specific property name

8.5.9 Acquiring Response Messages

Acquire the response message from the extracted response. The response message to be acquired is one of the following:

- Response message from a service component
- Error information (fault information) from a service component, business process, or custom adapter

(1) Response message from a service component

The response message to be acquired is an XML message that has the same message format as the service component side. For details about XML response messages, see *8.2.7 Acquiring Response Messages*.

When there is no response message from a synchronous service component (Web Services or SessionBean), TextMessage without payload is received.

For an asynchronous service component, there is no response.

(2) Error information (fault information) from a service component, business process, or custom adapter

The message (error information) to be acquired is the XML message that was sent by the asynchronous reception (MDB (WS-R)) to the response queue.

Output format of error information is different for SOAP1.2 and for other than SOAP1.2.

(a) Other than SOAP1.2

The following table shows the format of the XML message (error information) to be acquired while using other than SOAP1.2.

Table 8–15: Format of the XML message (error information) to be acquired (other than SOAP1.2)

Tag	Explanation
`errorcode`	Error code indicating an error from a service component, business process, or custom adapter
`errorstring`	Error message indicating an error from a service component, business process, or custom adapter
`Processinstanceid`	Instance ID information of a business process
`cscmsgcode`	FaultCode information from a service component (Web Services), business process, or custom adapter
`cscmsgstring`	FaultString information from a service component (Web Services), business process, or custom adapter
`cscmsgactor`	FaultActor information from a service component (Web Services), business process, or custom adapter
`cscmsgdetail`	Detail information from a service component (Web Services), business process, or custom adapter

The schema of the XML message (error information) to be acquired while using other than SOAP1.2 is shown below.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="cscmsgerror">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="errorcode" type="xsd:string"/>
        <xsd:element name="errorstring" type="xsd:string"/>
        <xsd:element name="processinstanceid" type="xsd:string"/>
        <xsd:element ref="errordetail"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="errordetail">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="cscmsgcode" type="xsd:string"/>
        <xsd:element name="cscmsgstring" type="xsd:string"/>
        <xsd:element name="cscmsgactor" type="xsd:string"/>
        <xsd:element name="cscmsgdetail" type="xsd:string"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

(b) In SOAP1.2

The following shows the format of the XML message (error information) to be acquired while using SOAP1.2:

Table 8–16: Format of XML message (error information) to be acquired (in SOAP1.2)

Tag	Description
errorcode	Error code indicating existence of error from service component, business process and custom adapter.

Tag	Description
errorstring	Error message indicating existence of error from service component, business process and custom adapter.
processinstanceid	Information of business process instance ID.
cscmsgcode	Code information from service component (Web Service), business process or custom adapter.
cscmsgvalue	Value information containing Code from service component (Web Service), business process or custom adapter.
cscmsgreason	Reason information from service component (Web service), business process or custom adapter.
cscmsgtext	Text information containing Reason from service component (Web Service), business process or custom adapter.
cscmsgrole	Role information from service component (Web Service), business process or custom adapter.
cscmsgnode	Node information from service component (Web Service), business process or custom adapter.
cscmsgdetail	Detail information from service component (Web Service), business process or custom adapter.

The following shows the schema of the XML message (error information) to be acquired while using SOAP1.2:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="cscmsgerror">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="errorcode" type="xsd:string"/>
        <xsd:element name="errorstring" type="xsd:string"/>
        <xsd:element name="processinstanceid" type="xsd:string"/>
        <xsd:element ref="errordetail"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="errordetail">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element ref="cscmsgcode"/>
        <xsd:element ref="cscmsgreason"/>
        <xsd:element name="cscmsgrole" type="xsd:string"/>
        <xsd:element name="cscmsgnode" type="xsd:string"/>
        <xsd:element name="cscmsgdetail" type="xsd:string"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="cscmsgcode" nillable="true">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="cscmsgvalue" type="xsd:string" maxOccurs="unbounded"
nillable="true"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>

  <xsd:element name="cscmsgreason" nillable="true">
    <xsd:complexType>
      <xsd:sequence>
      <xsd:element name="cscmsgtext" type="tns:cscmsgFaultReasonText" maxOccurs="unbounded"
nillable="true" />
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
  <xs:complexType name="cscmsgFaultReasonText">
    <xs:sequence>
      <xs:element name="locale" type="xs:string" nillable="true" />
      <xs:element name="text" type="xs:string" nillable="true" />
    </xs:sequence>
  </xs:complexType>
</xsd:schema>
```

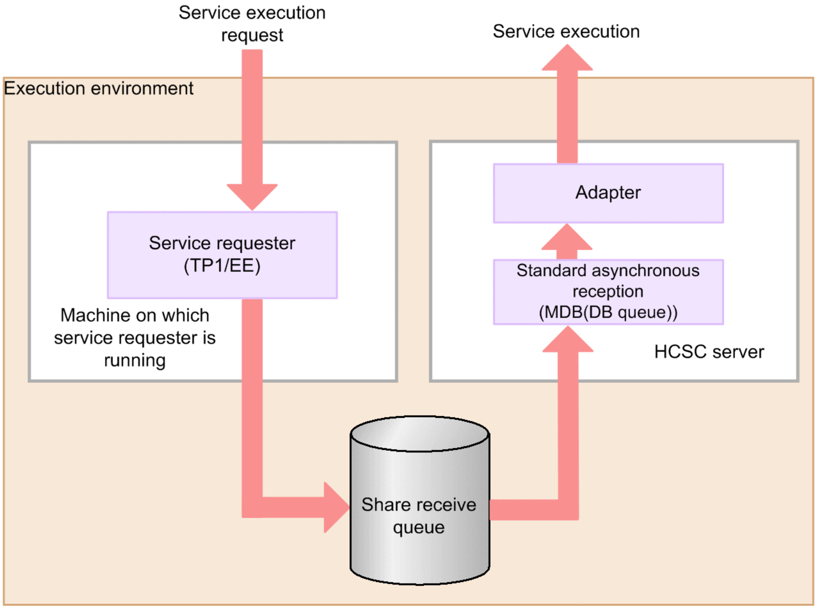
8.6 Service Requester That Sends Requests to a Standard Asynchronous Reception (MDB (database queue))

A service requester that sends a request to a standard asynchronous reception (MDB (database queue)) of an HCSC server that invokes a service component can be either TP1/Server Base Enterprise Option (TP1/EE) or JMS.

TP1/Server Base Enterprise Option (TP1/EE) service requester

The service requester (TP1/EE) communicates with the standard asynchronous reception (MDB (database queue)) of the HCSC server that invokes the service component via a shared receive queue. The following figure shows the relationship between the service requester (TP1/EE) that sends a request to the standard asynchronous reception (MDB (database queue)) and the HCSC server.

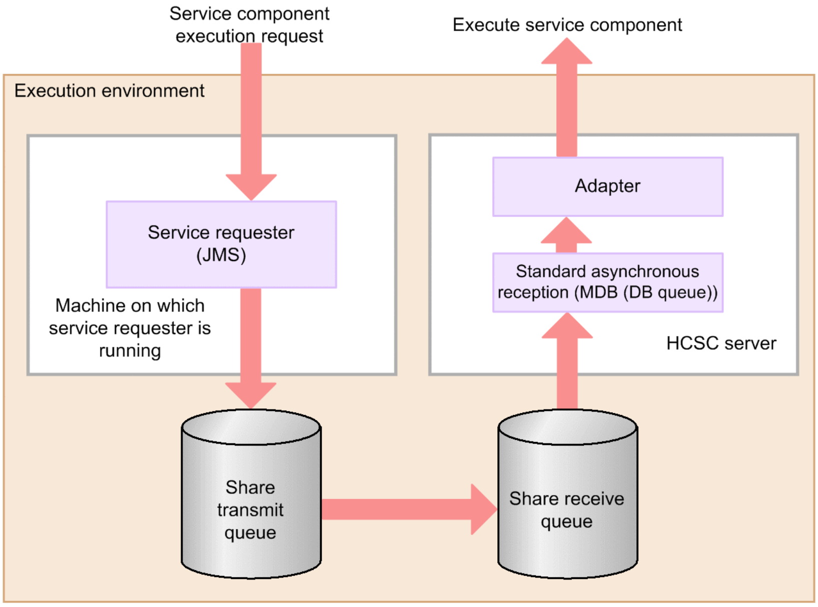
Figure 8–23: Relationship between the service requester (TP1/EE) that sends a request to standard asynchronous reception (database queue) and the HCSC server


Service execution request
Service execution
Execution environment
Service requester (TP1/EE)
Machine on which service requester is running
Adapter
Standard asynchronous reception (MDB(DB queue))
HCSC server
Share receive queue

JMS service requester

The service requester (JMS) communicates with the standard asynchronous reception (MDB (database queue)) of the HCSC server that invokes the service component via a shared transmission queue and a shared receive queue. The following figure shows the relationship between the service requester (JMS) that sends a request to the standard asynchronous reception (MDB (database queue)) and the HCSC server.

Figure 8–24: Relationship between the service requester (JMS) that sends a request to the asynchronous reception (database queue) and the HCSC server


Service component execution request
Execute service component
Execution environment
Service requester (JMS)
Machine on which service requester is running
Adapter
Standard asynchronous reception (MDB (DB queue))
HCSC server
Share transmit queue
Share receive queue

8.6.1 Service Requester (Standard Asynchronous Reception (MDB (Database Queue))) Creation Procedure

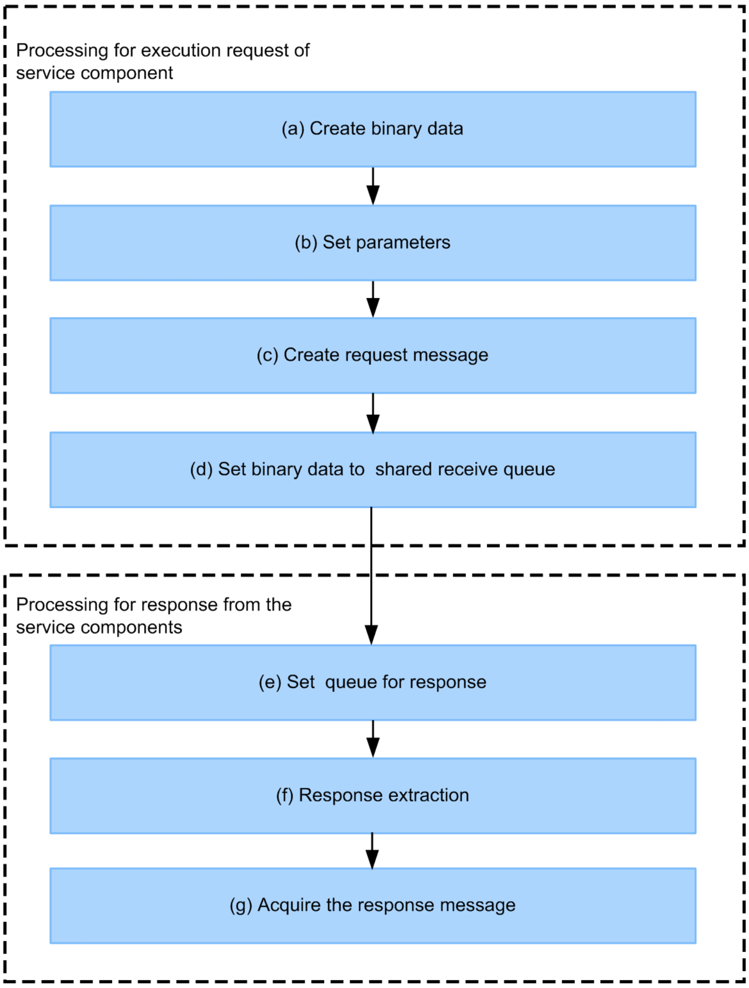
This subsection explains the workflow for creating a service requester that sends a request to the standard asynchronous reception (MDB (database queue)) to execute a service and invokes the service.

The creation workflow differs depending on whether the service requester is TP1/EE or JMS.

(1) In TP1/EE

The following figure shows the workflow for creating a service requester that sends a request to the standard asynchronous reception (MDB (database queue)) to execute a service component and invokes the service component when the service requester is TP1/EE.

Figure 8–25: Service requester (TP1/EE) creation work flow (standard asynchronous reception (MDB (database queue)))


Processing for execution request of service component
(a) Create binary data
(b) Set parameters
(c) Create request message
(d) Set binary data to shared receive queue
Processing for response from the service components
(e) Set queue for response
(f) Response extraction
(g) Acquire the response message

(a) Binary data creation

To send a request message to the standard asynchronous reception (MDB (database queue)), you create binary data. For details about binary data creation, see *8.6.4 Creating Binary Data (TP1/EE or JMS)*.

(b) Parameter specification

Assemble binary data by specifying the parameters that correspond to the individual tags in the binary data created in *8.6.1(1)(a) Binary data creation*. For details about parameter specification, see *8.6.5 Specifying Parameters (TP1/EE or JMS)*.

(c) Request message creation

Create a request message for requesting service component execution. For details about request message creation, see *8.6.6 Creating Request Messages (TP1/EE or JMS)*.

(d) Specifying binary data in the shared receive queue

Use the API (database queue control) of TP1/EE to set, in the database where the shared receive queue is located, the binary data whose parameters were specified. For details about how to specify binary data in the shared receive queue, see *8.6.7 Specifying Binary Data in the Shared Receive Queue (TP1/EE)*.

(e) Response queue setup

To receive responses from a synchronous service component (Web Services or SessionBean), set up a response queue (shared receive queue). For details about how to set up a response queue, see *8.6.9 Setting Up a Response Queue (TP1/EE or JMS)*.

(f) Response extraction

Using the API (database queue control) of TP1/EE, extract the response binary data by referencing the database in which the response queue (shared receive queue) is located. For details about response extraction, see *8.6.10 Extracting Responses (TP1/EE or JMS)*.

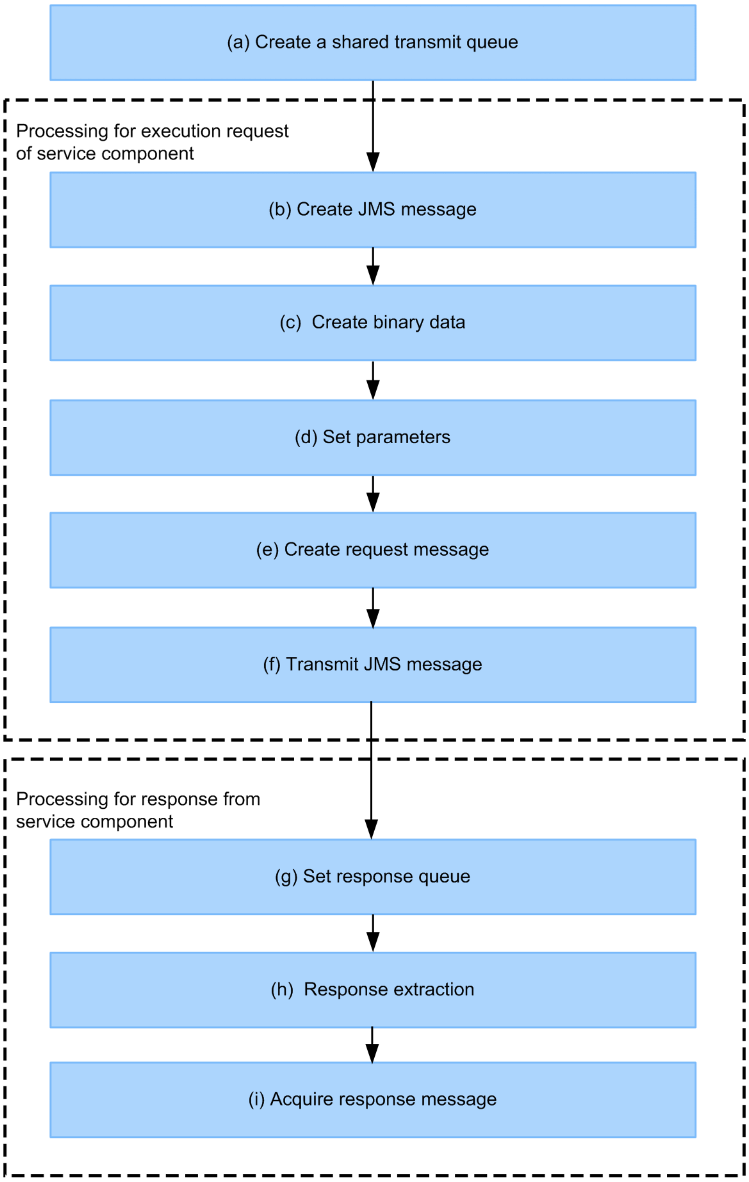
(g) Response message acquisition

Acquire the response message from the extracted response. For details about response message acquisition, see *8.6.11 Acquiring Response Messages (TP1/EE or JMS)*.

(2) In JMS

The following figure shows the workflow for creating a service requester that sends a request to the standard asynchronous reception (MDB (database queue)) to execute a service component and invokes the service component when the service requester is JMS.

Figure 8–26: Service requester (JMS) creation work flow (standard asynchronous reception (MDB (database queue)))


(a) Create a shared transmit queue
Processing for execution request of service component
(b) Create JMS message
(c) Create binary data
(d) Set parameters
(e) Create request message
(f) Transmit JMS message
Processing for response from service component
(g) Set response queue
(h) Response extraction
(i) Acquire response message

(a) Creating a shared transmission queue

Create a shared transmission queue necessary for sending a request message from the service requester (JMS). For details about creating a shared transmission queue, see *8.6.2 Creating a Shared Transmission Queue (JMS)*.

(b) Creating a JMS message

Create a QueueSender object and a JMS message in order to send a request to the asynchronous reception (MDB (database queue)) for service component execution. For details about QueueSender object and JMS message creation, see *8.6.3 Creating JMS Messages (JMS)*.

(c) Binary data creation

To send a request message to the asynchronous reception (MDB (database queue)), you create binary data. For details about binary data creation, see *8.6.4 Creating Binary Data (TP1/EE or JMS)*.

(d) Parameter specification

Assemble binary data by specifying the parameters that correspond to the individual tags in the binary data created in *8.6.1(2)(c) Binary data creation*. For details about parameter specification, see *8.6.5 Specifying Parameters (TP1/EE or JMS)*.

(e) Request message creation

Create a request message for requesting service component execution. For details about request message creation, see *8.6.6 Creating Request Messages (TP1/EE or JMS)*.

(f) JMS message transmission

Send a JMS message with binary data set as its payload to the shared transmission queue. For details about JMS message transmission, see *8.6.8 Sending JMS Messages (JMS)*.

(g) Response queue setup

To receive responses from a synchronous service component (Web Services or SessionBean), set up a response queue (shared transmission queue). For details about how to set up a response queue, see *8.6.9 Setting Up a Response Queue (TP1/EE or JMS)*.

(h) Response extraction

From the response queue (shared transmission queue), extract the response sent to the shared receive queue. For details about response extraction, see *8.6.10 Extracting Responses (TP1/EE or JMS)*.

(i) Response message acquisition

Acquire the response message from the extracted response. For details about response message acquisition, see *8.6.11 Acquiring Response Messages (TP1/EE or JMS)*.

8.6.2 Creating a Shared Transmission Queue (JMS)

Create a shared transmission queue necessary for sending a request message from the service requester. The shared transmission queue sends the request messages from the service requester to the shared receive queue provided by the HCSC server.

To create a shared transmission queue, use the hrmmkque command of Cosminexus RM. This subsection explains the specification values for the options of the hrmmkque command. For details about the options not explained here, see the manual *Cosminexus Application Server Cosminexus Reliable Messaging*.

(1) hrmmkque command option specification values

-t queue-type

Specify shr_send (shared transmission queue).

-b registration-destination-queue-name-for-using-shared-queue

Specify either of the following:

*Cosminexus-RM-system-name*_SHR_CSCHCSC-server-nameACPT_DBQ

Specify this option when the HCSC server has a load-balancing cluster configuration.

*Cosminexus-RM-system-name*_SHR_CSCHCSC-cluster-nameACPT_DBQ

Specify this option when the HCSC server has the HAcluster configuration.

HCSC-cluster-name is the cluster name of the HCSC server determined during HCSC server configuration deployment in the development environment and the operating environment.

8.6.3 Creating JMS Messages (JMS)

Create a QueueSender object and a JMS message in order to send a request to the standard asynchronous reception (MDB (database queue)) for service component execution, and send them to the shared transmission queue. For details about developing an application for sending JMS messages, see the manual *Cosminexus Application Server Cosminexus Reliable Messaging*.

For the payload of the created JMS message, specify binary data whose parameters have been specified. Create the JMS message in BytesMessage, and specify the binary data whose parameters have been specified as the payload of BytesMessage. Even if there is no request message, specify binary data as the payload of BytesMessage.

The properties of the JMS message are ignored even if specified.

For details about the JMS message creation procedure, see *8.5.3 Creating JMS Messages*.

8.6.4 Creating Binary Data (TP1/EE or JMS)

To send a request message to the standard asynchronous reception (MDB (database queue)), you create binary data in the format shown in the following table.

Table 8–17: Binary data format (request message)

Item No.	Item	Size (bytes)	Data type#1	Explanation	Required
1	Header tag	4	`char`	Header tag for delivering service component request message (eye catcher). Specify `DBQH`. Use ASCII codes.	Y
2	Byte order flag	1	`byte`	Byte order (endian) identification flag of the numeric data inside the header. Specify either of the following methods for encoding the numeric data inside the header into the binary format: • `0b00000001`: big endian • `0b00000010`: little endian	Y
3	Reserved	3	`byte`	Reserved area	Y
4	Size	4	`int`	Header size (size from Item 5 through Item 28) of the messaging of the service component request. Specify a size in bytes.	Y
5	Tag	8	`char`	Parameter identification tag (eye catcher). Specify `ServiceN`. Use ASCII codes.	P
6	Size	4	`int`	Size of the next area (Item 7). Specify a size in bytes.	
7	Service name	Optional	`String`	This is the service name of the request destination. For the service name of the request destination, specify the adapter or business process defined in the development environment. Use UTF-8 codes.	
8	Tag	8	`byte`	Parameter identification tag (eye catcher). Specify `ClientID`. Use ASCII codes.	P
9	Size	4	`int`	Size of the next area (Item 10). Specify a size in bytes.	
10	Client correlation ID	Optional	`String`	This is a correlation identifier for uniquely identifying the request message from the service requester. Specify alphanumeric characters, underscore (_), period (`.`), and hyphen (`-`) up to 255 characters.	

Item No.	Item	Size (bytes)	Data type#1	Explanation	Required
10	Client correlation ID	Optional	`String`	This parameter is used to map the request message from the service requester to the execution history, logs, and traces managed by the HCSC server. Therefore, specify a different ID for each request message sent to the HCSC server. Use UTF-8 codes. If this item is omitted, there is no need to assemble binary data having this parameter, size, and tag.	P
11	Tag	8	`byte`	Parameter identification tag (eye catcher). Specify `ReqFmtID`. Use ASCII codes.	--
12	Size	4	`int`	Size of the next area (Item 13). Specify a size in bytes.	
13	HCSC request format ID	Optional	`String`	ID for uniquely identifying the request message format. Do not assemble binary data having this parameter, size, and tag.	
14	Tag	8	`byte`	Parameter identification tag (eye catcher). Specify `ResFmtID`. Use ASCII codes.	--
15	Size	4	`int`	Size of the next area (Item 16). Specify a size in bytes.	
16	HCSC response format ID	Optional	`String`	ID for uniquely identifying the response message format. Do not assemble binary data having this parameter, size, and tag.	
17	Tag	8	`byte`	Parameter identification tag (eye catcher). Specify ReplyToQ. Use ASCII codes.	P
18	Size	4	`int`	Size of the next area (Item 19). Specify a size in bytes.	
19	Response queue name	Optional	`String`	This is the queue name that receives a response from a service component or business process when the service component at the request destination is a synchronous service (Web Services or SessionBean) or business process. By specifying this parameter when you cannot determine whether the service component at the request destination is synchronous or asynchronous, you can receive responses. For details, see *8.6.9 Setting Up a Response Queue (TP1/EE or JMS)*. Use UTF-8 codes. If you omit this parameter, `NULL` is specified, and you cannot receive responses. If this item is omitted, there is no need to assemble binary data having this parameter, size, and tag.	
20	Tag	8	`byte`	Parameter identification tag (eye catcher). Specify OperatiN. Use ASCII codes.	P
21	Size	4	`int`	Size of the next area (Item 22). Specify a size in bytes.	
22	Service operation name	Optional	`String`	This is an operation name corresponding to the service name at the request destination.#2	

Item No.	Item	Size (bytes)	Data type#1	Explanation	Required
22	Service operation name	Optional	`String`	This operation name specifies a service component defined in the development environment. Specify the operation name with NCName definition characters of XMLSchema within 255 bytes. This parameter is required when the service component at the request destination is a synchronous service (Web Services or SessionBean) or business process. Use UTF-8 codes. When the service component at the request destination is an asynchronous service component, you can omit the response queue name. If this item is omitted, there is no need to assemble binary data having this parameter, size, and tag.	P
23	Tag	8	`byte`	Parameter identification tag (eye catcher). Specify `MessageT`. Use ASCII codes.	P
24	Size	4	`int`	Size of the next area (Item 25). Specify a size in bytes.	
25	Message type	Optional	`String`	User message (request message) type. For XML message (whose payload is `TextMessage`) specify XML. For binary message (whose payload is `BytesMessage`), specify `Binary`. This parameter must be specified when there is a message (request message). Use UTF-8 codes. When there is no request message, you can omit this item. If this item is omitted, there is no need to assemble binary data having this parameter, size, and tag.	
26	Tag	8	`byte`	Parameter identification tag (eye catcher). Specify `MgLength`. Use ASCII codes.	Y
27	Size	4	`int`	Size of the next area (Item 28). Specify `4`.	
28	Message size	4	`int`	Specify the message size for the user message (request message) following binary conversion. If there is no user message (request message), specify 0.	
29	Message	Optional	`byte[]`	User message (request message).#2 Specify a binary-converted user message (request message) after the database queue headers (Items 1 through 28). If an XML message has been converted to binary format, encode it using UTF-8 codes. This item can be omitted if there is no user message (request message).	P

Legend:

Y: Required.

P: Required in some cases. Check the explanation.

--: Must not be specified.

#1

Java data type

#2

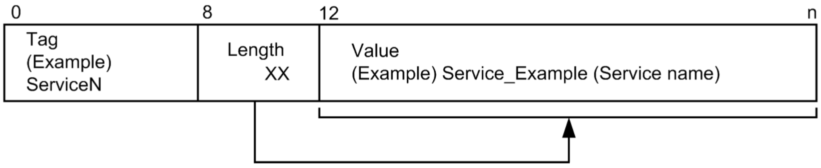
When the service component protocol of request destination is SOAP, decide an operation to be invoked from the name of a root element of user message (in the case of data transformation, it is the name of root element of the message after data transformation). Therefore, take note that if you specify an invalid name in the root element of user message, an unintended operation may be invoked.

8.6.5 Specifying Parameters (TP1/EE or JMS)

Assemble binary data by specifying the parameters that correspond to the individual tags (Items 5, 8, 11, 17, 20, 23, and 26 in *Table 8-17*) in the binary data created.

The following figure shows the binary data tag format.

Figure 8–27: Binary data tag format



The tags may occur inside the binary data in any order. However, the order in which the tag, size, and value occur is fixed as shown in Figure 8-27.

Additionally, specify a binary-converted message (request message) after the database queue headers (Items 1 through 28). If there is no request message, you can also specify MgLength for the tag of Item 26 and set its size to 0.

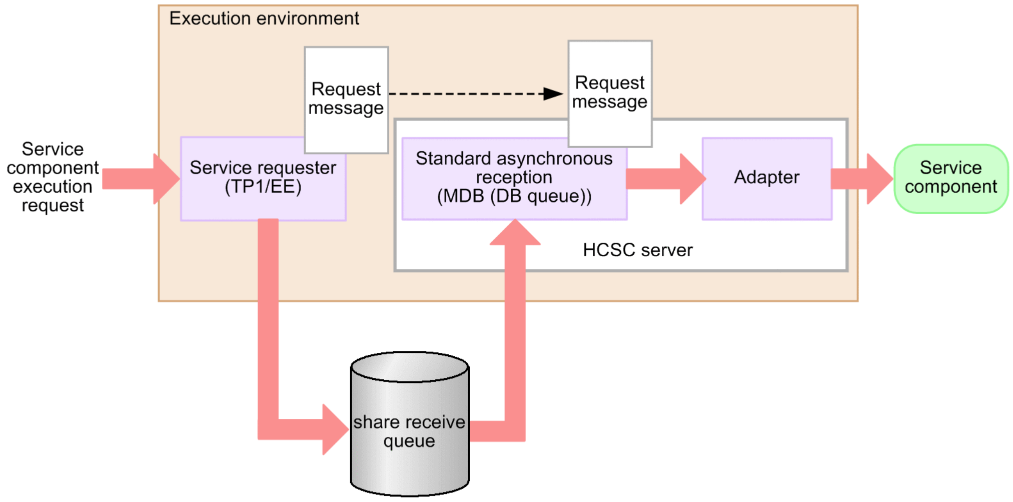
8.6.6 Creating Request Messages (TP1/EE or JMS)

Create a request message for requesting a service component from the service requester to the asynchronous reception (MDB (database queue)). The contents of the request message to be sent from the service requester must be created in the same message format as that used on the service component side.

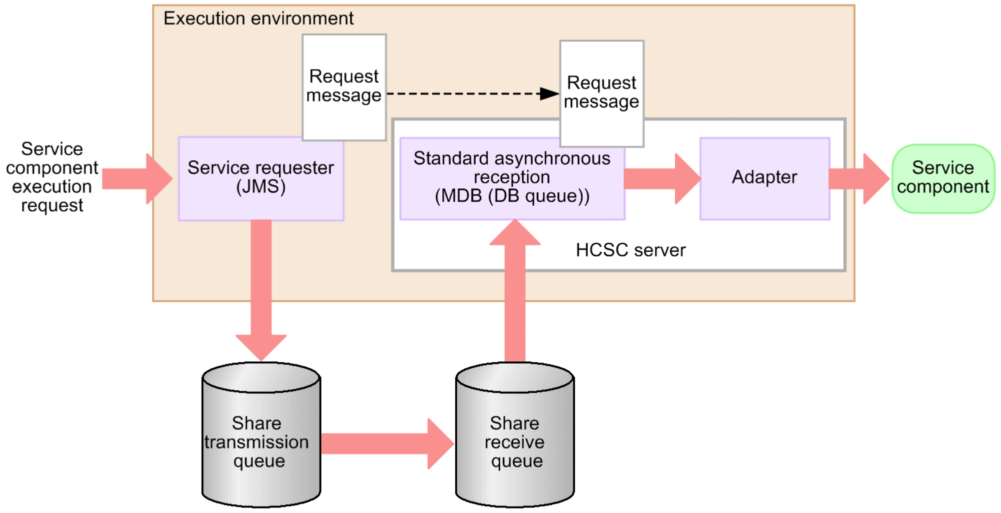
The following figure shows how a request message is sent when the service requester is TP1/EE or JMS.

Figure 8–28: Request message transmission (standard asynchronous reception (MDB (database queue)))

- For TP1/EE


Execution environment
Request message
Request message
Service component execution request
Service requester (TP1/EE)
Standard asynchronous reception (MDB (DB queue))
Adapter
Service component
HCSC server
share receive queue

- For JMS



For details about how to send a normal request message, see the contents about service invocation using an MDB (database queue), in the manual *Cosminexus Service Platform System Setup and Operation Guide*.

The service requester creates a request message whose message type is XML. Convert the created request message to binary format. For binary conversion, use the same encoding method for the service requester side and the service component side. For details about XML request messages, see *8.2.6 Creating Request Messages*.

Note that when the protocol type is a database queue service component, the asynchronous reception (MDB (database queue)) cannot send request messages. The asynchronous reception (MDB (database queue)) can send request messages only when the service component side is using one of the following protocols:

- SOAP
- RMI-IIOP
- WS-R

8.6.7 Specifying Binary Data in the Shared Receive Queue (TP1/EE)

Use the API (database queue control) of TP1/EE to set, in the database where the shared receive queue is located, the binary data whose parameters were specified.

For details, see the manual *TP1/Server Base Enterprise Option Program Creation Guide*.

8.6.8 Sending JMS Messages (JMS)

Send a JMS message with binary data set as its payload to the shared transmission queue.

For details about the JMS message transmission procedure, see *8.5.6 Sending JMS Messages*.

8.6.9 Setting Up a Response Queue (TP1/EE or JMS)

The standard asynchronous reception (MDB (database queue)) is a standard reception for asynchronous request messages, and thus is used, as a rule, when there is no response from a service. However, when the asynchronous reception (MDB (database queue)) sends a request to a synchronous service component (Web Services or SessionBean), a response may be returned from the synchronous service component in some cases. In such a case, by specifying a response queue when making a request, the service requester can receive responses from a synchronous service component. For details about specifying a response queue when making a request, see *8.5.6 Sending JMS Messages*.

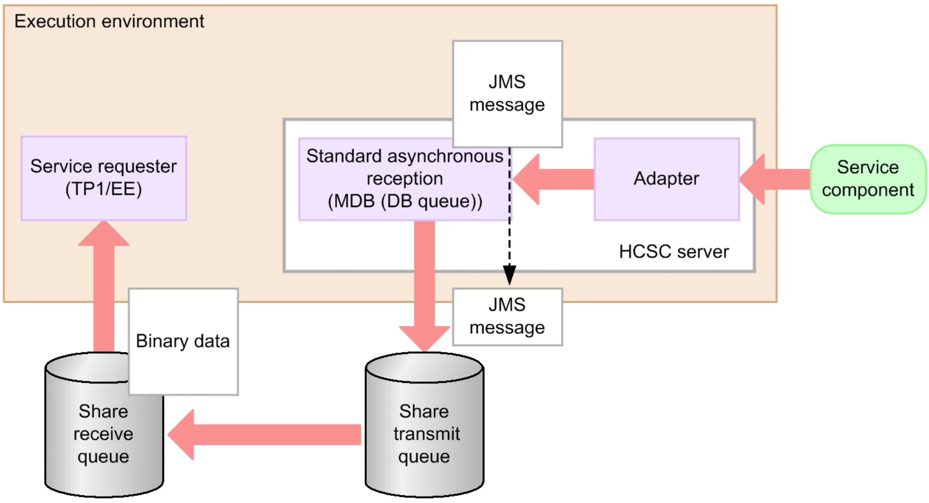
(1) In TP1/EE

The asynchronous reception (MDB (database queue)) sends the JMS message, in which binary data including a response message is set in the payload, to the response queue (shared transmission queue).

The shared transmission queue extracts the binary data from the received response and sends it to the shared receive queue. You can extract the response sent to the shared receive queue and acquire the response message from the service component. For details about response extraction, see *8.6.10 Extracting Responses (TP1/EE or JMS)*.

The following figure shows the relationship among the standard asynchronous reception (MDB (database queue)), the shared transmission queue, and the shared receive queue.

Figure 8–29: Relationship among the standard asynchronous reception (MDB (database queue)), the shared transmission queue, and the shared receive queue (TP1/EE)



Shared transmission queue (response queue)

Create a shared transmission queue, which functions as a response queue, inside the J2EE server (Cosminexus RM) on which the HCSC server is running. For the registration destination queue of the shared transmission queue, specify the shared receive queue.

For details about how to create a shared transmission queue, see the section related to shared transmission queues in the manual *Cosminexus Application Server Cosminexus Reliable Messaging*.

Shared receive queue

Create a TP1/EE database queue inside the same schema definition of the database in which the shared transmission queue is located (which is being used by Cosminexus RM). For details, see the section related to shared receive queues in the manual *Cosminexus Application Server Cosminexus Reliable Messaging*.

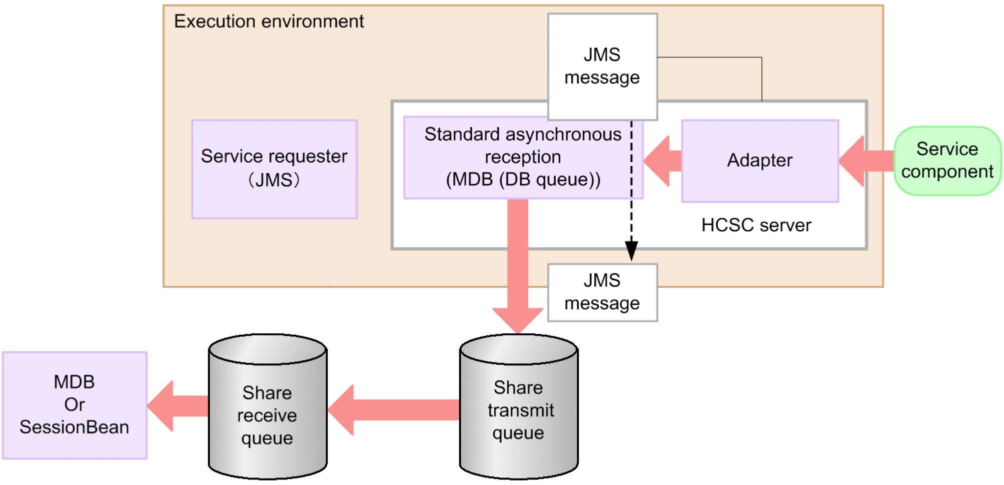
Also, for details about how to create a database queue, see the contents about the database queue in the manual *TP1/Server Base Enterprise Option Usage Guide*.

(2) In JMS

As a response, the standard asynchronous reception (MDB (database queue)) sends the JMS message, in which binary data including a response message is set in the payload, to the response queue (shared transmission queue). The response received by the shared transmission queue is sent to the shared receive queue. By extracting the response sent to the shared receive queue, you can acquire the response from the service component. For details about response extraction, see *8.6.10 Extracting Responses (TP1/EE or JMS)*.

The following figure shows the relationship among the standard asynchronous reception (MDB (database queue)), the shared transmission queue, and the shared receive queue.

Figure 8–30: Relationship among the standard asynchronous reception (MDB (database queue)), the shared transmission queue, and the shared receive queue (JMS)



Shared transmission queue (response queue)

Create a shared transmission queue, which functions as a response queue, inside the J2EE server (Cosminexus RM) on which the HCSC server is running. For the registration destination queue of the shared transmission queue, specify the shared receive queue.

For details about how to create a shared transmission queue, see the section related to shared transmission queues in the manual *Cosminexus Application Server Cosminexus Reliable Messaging*.

Shared receive queue

Create a shared receive queue inside the J2EE server (Cosminexus RM) on which the service requester is running.

Specify a receive queue according to the user application as needed.

For details about how to create receive queues, see the section related to shared receive queues in the manual *Cosminexus Application Server Cosminexus Reliable Messaging*.

8.6.10 Extracting Responses (TP1/EE or JMS)

(1) In TP1/EE

Using the API (database queue control) of TP1/EE, extract the response binary data by referencing the database in which the response queue (shared receive queue) is located. For details about how to send a normal request message, see the contents about the response queue in service invocation using an MDB (database queue), in the manual *Cosminexus Service Platform System Setup and Operation Guide*.

For details, see the manual *TP1/Server Base Enterprise Option Program Creation Guide*.

The following table shows the binary data format for the response message.

Table 8–18: Binary data format (response message)

Item No.	Item	Size (bytes)	Data type#	Explanation
1	Header tag	4	`char`	Header tag for delivering HCSC messages (eye catcher). DBQH is set. ASCII-encoded binary data is set.
2	Byte order flag	1	`byte`	The byte order flag that was specified during a request is set.
3	Reserved	3	`byte`	Reserved area
4	Size	4	`int`	Header size (size from Item 5 through Item 19) for HCSC messaging is set.
5	Tag	8	`char`	Parameter identification tag (eye catcher). ServiceN is set. ASCII-encoded binary data is set.
6	Size	4	`int`	Size of the next area (Item 7). Size in bytes is set.
7	Service name	Optional	`String`	The adapter or business process specified during the request is set. UTF-8-encoded binary data is set.
8	Tag	8	`byte`	Parameter identification tag (eye catcher). ClientID is set. ASCII-encoded binary data is set.
9	Size	4	`int`	Size of the next area (Item 10). Size in bytes is set.
10	Client correlation ID	Optional	`String`	This is a correlation identifier for uniquely identifying the request message from the service requester. Specified with alphanumeric characters, underscore (_), period (.), and hyphen (-) up to 255 characters. UTF-8-encoded binary data is set.
11	Tag	8	`byte`	Parameter identification tag (eye catcher). Specify ResFmtID. Use ASCII codes.
12	Size	4	`int`	Size of the next area (Item 13). Specify a size in bytes.
13	HCSC response format ID	Optional	`String`	ID for uniquely identifying the response message format. This parameter, size, and tag are not specified.
14	Tag	8	`byte`	Parameter identification tag (eye catcher). OperatiN is set. ASCII-encoded binary data is set.
15	Size	4	`int`	Size of the next area (Item 16). Size in bytes is set.

Item No.	Item	Size (bytes)	Data type#	Explanation
16	Service operation name	Optional	`String`	This is an operation name corresponding to the service name at the request destination. The operation name is set with the 255-byte NCName definition characters of XMLSchema. UTF-8-encoded binary data is set.
17	Tag	8	`byte`	Parameter identification tag (eye catcher). MessageT is set. ASCII-encoded binary data is set.
18	Size	4	`int`	Size of the next area (Item 19). Size in bytes is set.
19	Response message type	Optional	`String`	Message (response message) type. If the message type is XML, XML is set. If there is no message (response message), this parameter is omitted. UTF-8-encoded binary data is set.
20	Tag	8	`byte`	Parameter identification tag (eye catcher). MgLength is set. ASCII-encoded binary data is set.
21	Size	4	`int`	Size of the next area (Item 22). 4 is set.
22	Message size	4	`int`	Message (response message) size is set. If there is no message (response message), 0 bytes is set.
23	Message	Optional	`byte[]`	Either of the following types of information is set: • Response message from a service component • Error information (fault information) from a service component, business process, or custom adapter UTF-8-encoded binary data is set.

#

Java data type

(2) In JMS

A response (JMS message) can be extracted from the shared receive queue based on MDB or SessionBean installation. For details about MDB and SessionBean installation, see the manual *Cosminexus Application Server Cosminexus Reliable Messaging*.

For details about the format of the binary data that is set in the payload of the extracted response (JMS message), see *Table 8-18*.

8.6.11 Acquiring Response Messages (TP1/EE or JMS)

Acquire the response message from the extracted response. The response message to be acquired is one of the following:

- Response message from a service component
- Error information (fault information) from a service component, business process, or custom adapter

(1) Response message from a service component

The response message to be acquired is an XML message that has the same message format as the service component side. For details about XML response messages, see *8.2.7 Acquiring Response Messages*.

(2) Error information (fault information) from a service component, business process, or custom adapter

The message (error information) to be acquired is the XML message that was sent by the asynchronous reception (MDB (database queue)) to the response queue. For details about the format of the XML message (error information) to be acquired, see *8.5.9 Acquiring Response Messages*.

8.7 Service Requester That Sends Requests to a User-defined Reception (Web Services)

A service requester that sends request messages to a user-defined reception (Web services) communicates with the user-defined reception using SOAP. The service requester sends a service component execution request message to a user-defined reception, and an HCSC server performs service component execution.

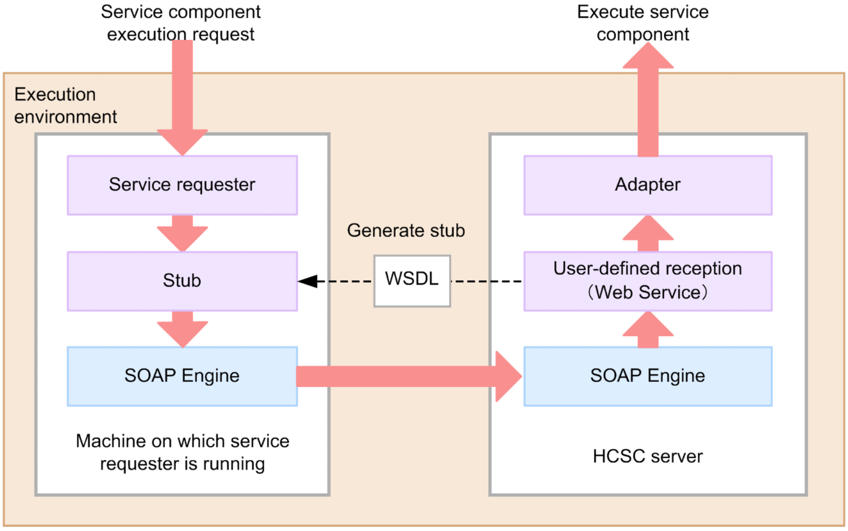
The interface information of the user-defined reception (Web services) becomes the WSDL specified when you define the user-defined reception in the development environment. You generate a stub from the WSDL and use the stub to send a request to the user-defined reception (Web services).

Tip

When creating a service requester that sends requests to a user-defined reception (Web services), unlike a standard reception, you need not create a request message. A request message (SOAP message) of a format that matches WSDL is automatically generated in the service requester stub and a service component execution request is sent.

The following figure shows the relationship between a service requester that sends requests to a user-defined reception (Web Services) and an HCSC server:

Figure 8–31: Relationship between a service requester that sends requests to a user-defined reception (Web services) and an HCSC server

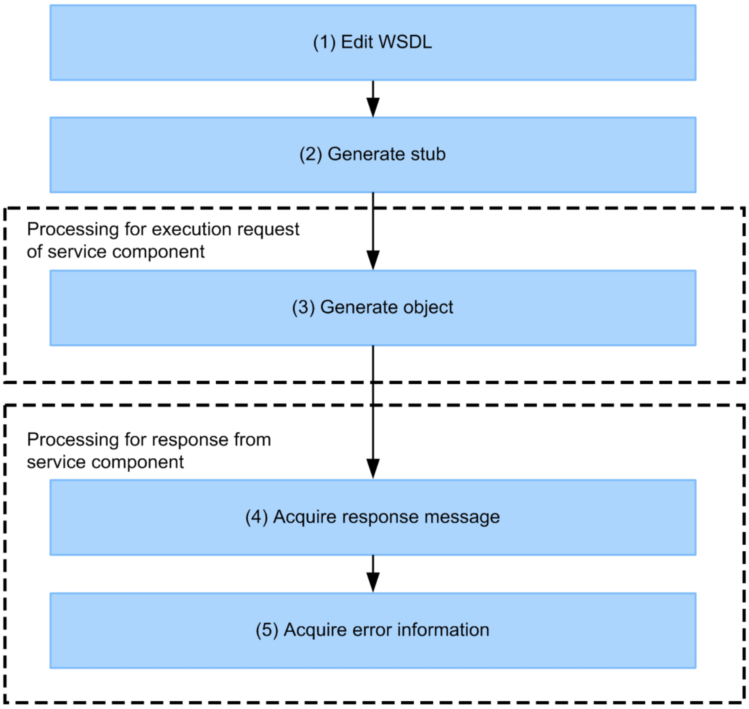

Service component execution request
Execute service component
Execution environment
Service requester
Stub
SOAP Engine
Machine on which service requester is running
Generate stub
WSDL
Adapter
User-defined reception（Web Service）
SOAP Engine
HCSC server

For a service requester that uses JAX-WS engine for communication, a service class is created instead of a stub. For details about how to create a service requester for a JAX-WS engine, see *8.7.7 Procedure for creating a service requester (User-defined Reception (Web Service)) (JAX-WS engine)*.

8.7.1 Procedure (SOAP communication infrastructure)for creating a service requester (User-defined reception (Web Service))

This subsection explains how to create a service requester that sends a service component execution request to a user-defined reception (Web services) and invokes a service component.

Figure 8–32: Workflow for creating a service requester (user-defined reception (Web services)) (SOAP communication infrastructure)


(1) Edit WSDL
(2) Generate stub
Processing for execution request of service component
(3) Generate object
Processing for response from service component
(4) Acquire response message
(5) Acquire error information

The tasks in the individual steps are described below.

(1) WSDL editing

Check the definition contents of the user-defined reception and edit the WSDL contents required for stub creation. For details about WSDL editing, see *8.7.2 Editing a WSDL*.

(2) Stub creation

Create stubs from the WSDL edited in *8.7.1(1) WSDL editing*. For details about stub creation, see *8.7.3 Creating Stubs*.

(3) Object generation

To invoke a method of user-defined reception (Web services), use stubs created in *8.7.1(2) Stub creation* to generate objects. For details about object generation, see *8.7.4 Generating Objects*.

(4) Response message acquisition

Acquire a response message corresponding to the service component execution request from the user-defined reception (Web services). For details about response message acquisition, see *8.7.5 Acquiring Response Messages*.

(5) Error information acquisition

If an error occurs at the request-destination service component, the HCSC server, or the SOAP engine, acquire the error information and take corrective action according to the information. For details about error information acquisition, see *8.7.6 Acquiring Error Information*.

8.7.2 Editing a WSDL

To create a service requester that sends execution requests to user-defined reception (Web Services), use a WSDL that is specified when the user-defined reception is defined in the development environment. For details about creating a WSDL, see *8.3 Creating WSDL*.

In a WSDL that is specified when the user-defined reception is defined in the development environment, a temporary value is set in the value for the service location (value set in the location attribute of the soap-address element within the wsdl:port element). As a result, you need to edit the service location value of the WSDL to the URL information of the user-defined reception that you want to use.

When setting up an HCSC server, if there is a specification (request-userdef-soap=ON is specified) for using a user-defined reception (Web Services) in the HCSC server setup definition file, a URL will be displayed for Web Services in the User-defined Reception Information Display screen of the development environment. The displayed URL is the URL information of the user-defined reception. Set this URL in the service location value of the WSDL.

If an HCSC server is not set up or an HCSC server is set up but there are specifications for not using the user-defined reception (Web services) in the HCSC server setup definition file, set a service location value based on the following rule:

```
http://host-name:port-number#1/context-root #2
            /services/CSCMsgUserDefinedReception
```

#1

This is the URL (host name and port number) of the HCSC server.

#2

For the context root name, the same value is set by default as the reception ID that is allocated when a user-defined reception is defined in the development environment. It is displayed in **Context root** in the User-defined Reception Definition screen.

For details about how to display the User-defined Reception Information Display screen, see *7.4.2 Displaying HCSC Component Information*.

8.7.3 Creating Stubs

Create stubs from the edited WSDL. You can create a stub with the WSDL2Java command provided by Cosminexus as a development support command.

A command input example is shown below:

```
WSDL2Java xxxxx.wsdl
```

If you execute this command, directories and files will be created based on the contents coded in the specified WSDL.

For details about options of the WSDL2Java command and contents of the generated stubs, see the manual *Cosminexus Application Server SOAP Application Development Guide*.

8.7.4 Generating Objects

To invoke a method of user-defined reception (Web Services), use the created stubs and generate objects.

To generate an object for invoking a method of user-defined reception (Web Services):

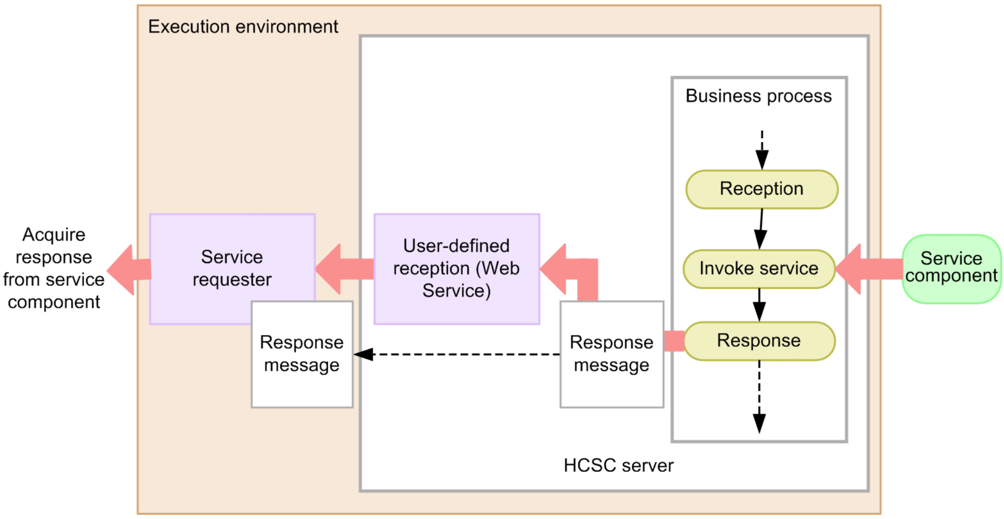
1. Generate an object of the Locator class that is an interface class.
2. Use the Locator class, and generate an object of the interface class of the user-defined reception.
   The instance of the service requester's interface class created or acquired cannot be shared by multiple threads.

By invoking a method of the generated object, a request for service component execution is sent to the user-defined reception. Within the objects of a stub, request messages (SOAP messages) are automatically generated in the format defined in the WSDL.

8.7.5 Acquiring Response Messages

Acquire a response message corresponding to the service component execution request from the user-defined reception (Web services). The objects of a stub receive a response message (SOAP message) from the user-defined reception in a format defined in WSDL, and return the response to a service requester. The following figure shows how a response message is acquired.

Figure 8–33: Response message acquisition (User-defined reception (Web services))



If there is no response message from the service component, `NULL` is received. A case in which there is a response message and a case in which there is no response message are explained separately below.

(1) When there is a response message

The service requester acquires a response message from a business process as a return value for invoking a method of the generated object.

(2) When there is no response message

The service requester acquires `NULL` as a return value for invoking a method of the generated object.

8.7.6 Acquiring Error Information

If an error occurs at the request-destination service component, the HCSC server, or the SOAP engine, acquire the error information and take corrective action according to the information.

For details about how to send an error, see the contents of a user-defined reception in the troubleshooting during the execution of a Web Service (SOAP Communication), in the manual *Cosminexus Service Platform System Setup and Operation Guide*.

(1) Service requester-side installation example

The error acquisition method differs depending on the type of SOAP Communication Infrastructure.

(a) When the SOAP Communication Infrastructure provided by Cosminexus is used

How to acquire SOAPFault from a service component

Catch an exception object used for the error information defined in the WSDL and acquire the SOAPFault error information. The format of SOAPFault is the Fault format (Fault format on the service component machine) defined in WSDL.

To acquire SOAPFault from a service component, you need to implement the following at the service requester-side:

```
{
    try {
        ...
      // Invoke Web Services
        ...
    } catch (xxxxxxxxxxException e) {
        ...
    } catch (C4Fault e) {
        ...
    }
}
```

How to acquire an exception that occurred in the HCSC server

Catch the `C4Fault` object provided in the SOAP Communication Infrastructure of Cosminexus, and acquire the error information.

The process flow at the service requester machine is as follows:

1. Catch with either `C4Fault` or `RuntimeException`.
2. Extract `Element[]` with the `getFaultDetails()` method from `C4Fault`, and acquire `NodeList` for each array with the `getChildNodes()` method.
3. Extract `Node` from `NodeList`, and then use the `getTextContent()` method to extract the error information for each Node.

An example of implementation at the service requester side when acquiring an exception that occurred in the HCSC server, is described below:

```
{
    try {
        ...
    } catch (xxxxxxxxxxException e) {
        ...
    } catch (C4Fault e) {
      // Acquire Fault information from C4Fault
      System.out.println("C4Fault Message          = " + e.getMessage());
      System.out.println("C4Fault FaultCode        = " + e.getFaultCode());
      System.out.println("C4Fault FaultActor       = " + e.getFaultActor());
      System.out.println("C4Fault FaultString      = " + e.getFaultString());
      // Acquire Detail from C4Fault
      Element ele[] = e.getFaultDetails();
      printElement(ele);
    }
}

/**
 * Print Element
 */
  private static void printElement(Element[] ele) {
    int eleNumber = ele.length;
    System.out.println("Element count = " + eleNumber);
    for (int i=0; i<eleNumber; i++) {
      // Acquire NodeList from Element array
      if (ele[i] != null) {
        printNodeList(ele[i].getChildNodes());
      } else {
        System.out.println("Element[" + i + "] = null");
      }
    }
  }

/**
 * Print NodeList
 */
  private static void printNodeList(NodeList nodelist) {
    int nodelistcount = nodelist.getLength();
    for (int j=0; j<nodelistcount; j++) {
      Node node1 = nodelist.item(j);
      NodeList nodelist1 = node1.getChildNodes();
      int nodelist1len = nodelist1.getLength();

      if (nodelist1len == 0) {
        System.out.println("empty ChildNode");
      } else if(nodelist1len == 1) {
        Node node2 = node1.getFirstChild();
        if(node2 != null) {
          System.out.println("ChildNode[" + j + "] getTextContent = "
                              + node2.getTextContent());
        } else {
          System.out.println("ChildNode[" + j + "] = null");
        }
      } else {
        printNodeList(nodelist1);
      }
    }
  }
```

The structure of `Element[]` that is acquired with the `getFaultDetails()` method is described below:

Table 8–19: Structure of Element[] acquired with the getFaultDetails() method

Name	Explanation
`errorMessage`	This is an error message set in the exception that occurred in the HCSC server.
`errorCode`	This is an error code corresponding to the exception that occurred in the HCSC server.
`processInstanceID`	Instance ID information of a business process Since this information is not set in an exception that occurred in the HCSC server, it becomes null (`nil` attribute).

(b) When the SOAP Communication Infrastructure provided by Cosminexus is not used

How to acquire SOAPFault from the service component

Catch an exception object used for the error information defined in the WSDL and acquire the SOAPFault error information. The format of SOAPFault is the Fault format (Fault format on the service component machine) defined in WSDL.

How to acquire an exception that occurred in the HCSC server

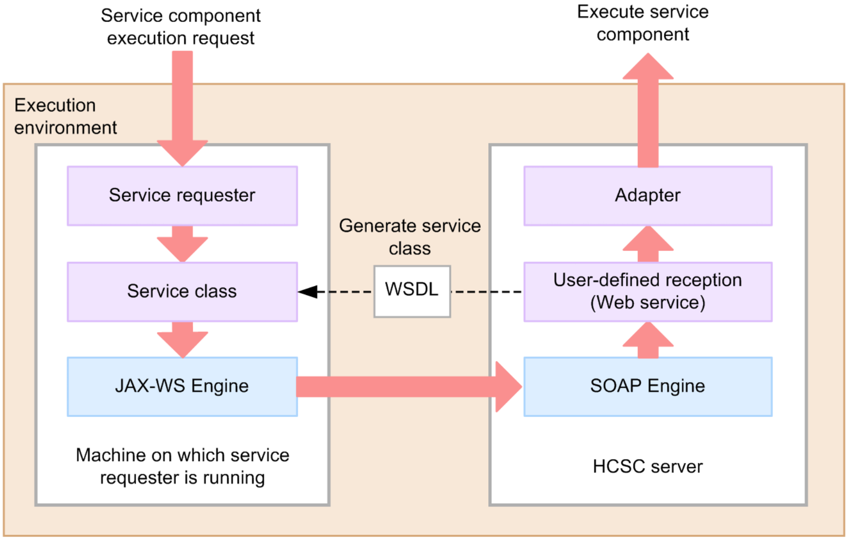
The acquisition method depends on the SOAP engine that is implemented at the service requester side.

8.7.7 Procedure for creating a service requester (User-defined Reception (Web Service)) (JAX-WS engine)

A service requester that uses JAX-WS engine for communication can be created as the service requester for sending requests to a user-defined reception (Web Services). The service requester that uses the JAX-WS engine for communication sends request messages to the user-defined reception (Web Services) using SOAP (document-literal type).

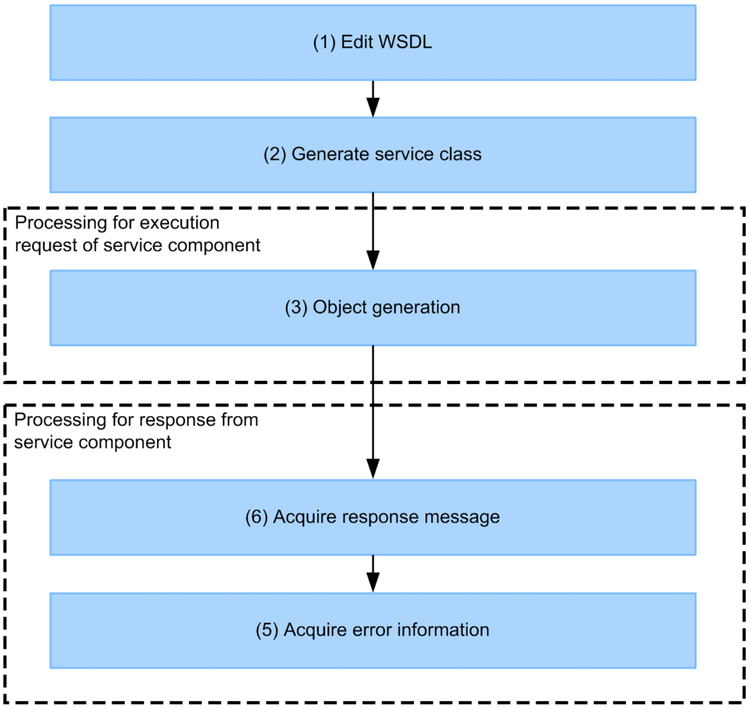
The following figure shows the relationship between a service requester that uses the JAX-WS engine for communication and an HCSC server:

Figure 8–34: Relationship between a service requester that uses the JAX-WS engine for communication and an HCSC server (User-defined reception (Web Services))


Service component execution request
Execute service component
Execution environment
Service requester
Service class
JAX-WS Engine
Machine on which service requester is running
Generate service class
WSDL
Adapter
User-defined reception (Web service)
SOAP Engine
HCSC server

The procedure for creating a service requester that uses the JAX-WS engine for communication is as follows:

Figure 8–35: Procedure for creating a service requester that uses the JAX-WS engine for communication (User-defined reception (Web Services))


(1) Edit WSDL
(2) Generate service class
Processing for execution request of service component
(3) Object generation
Processing for response from service component
(6) Acquire response message
(5) Acquire error information

The tasks in the individual steps are described below.

(1) WSDL editing

The WSDL editing method is the same as that used for creating an ordinary service requester in SOAP communication infrastructure.

For details about how to edit WSDL, see *8.7.2 Editing a WSDL*.

(2) Service class creation

Create a service class from the edited WSDL. To create a service class, you use the cjwsimport command provided by Cosminexus as a development support command.

A command input example is shown below:

```
cjwsimport -s source-file-output-destination-directory -d compiled-class-file-output-
destination-directory WSDL-file
```

For details about options of the `cjwsimport` command, see the manual *Cosminexus Application Server Web Service Development Guide*.

If the WSDL file edited in *10.6.7(1) WSDL editing* is saved at a location different from the current directory in which the `cjwsimport` command is executed, also specify the directory.

If you execute this command, the directories and files will be created in the output destination directory of the specified source file based on the contents described in the WSDL.

! Important note

For a service requester that uses JAX-WS engine for communication, the WSDL will be read during the execution of the program. When the default constructor of the service class is used, the WSDL in the WSDL path (directory that stores the WSDL file acquired in *10.6.7(1) WSDL acquisition* for creating the service class) specified with the `cjwsimport` command will be read. Therefore, after executing the `cjwsimport` command, do not move the WSDL file so that the configuration position of the WSDL file based on the service class does not destroy the relative relationship. For changing the configuration position of the WSDL referenced during the execution of the service requester from the WSDL path specified with the `cjwsimport` command, use a constructor for which the URL can be specified.

(3) Object generation

To invoke a method of user-defined reception (Web Services), use the created service class and generate proxy class objects.

The following describes the procedure for creating an object of the proxy class for invoking user-defined reception (Web Services) on the basis of the created service class:

1. Create the service class.
2. Create the proxy class corresponding to `wsdl:portType`.
3. Use objects of the created proxy class and invoke service methods.

By invoking a method of the generated object, a request for service component execution is sent to the user-defined reception. Within the objects of a service class, request messages (SOAP messages) are automatically generated in the format defined in the WSDL.

(4) Response message acquisition

The response message acquisition method is the same as that used for creating an ordinary service requester in SOAP communication infrastructure.

For details about response message acquisition, see *8.7.5 Acquiring Response Messages*.

(5) Error information acquisition

If an error occurs at the request-destination service component, the HCSC server, or the SOAP engine, acquire the error information and take corrective action according to the information.

How to acquire SOAPFault from a service component

For a service requester that uses the JAX-WS engine for communication, the exception in which the information is wrapped will be caught. Therefore, to acquire a user-defined reception, the fault information save class must be acquired using the `getFaultInfo()` method.

How to acquire an exception that occurred in the HCSC server

Acquire the error information by catching the `javax.xml.ws.soap.SOAPFaultException` object.

The process flow at the service requester machine is as follows:

1. Catch with `SOAPFaultException`.
2. Extract SOAPFault from `SOAPFaultException` using the `getFault()` method.
3. From SOAPFault, extract the information about the SOAP Fault.

An example of implementation at the service requester side when acquiring an exception that occurred in the HCSC server, is described below:

```
/**
 * Sample Program
 */
{
    try {
         ...
    } catch (xxxxxxxxxxException_Exception e) {
       // When the SOAP fault defined in WSDL is returned back
      xxxxxxxxxxException faultInfo = e.getFaultInfo();
         ...
    } catch (SOAPFaultException e) {
       // When an error occurs in the HCSC server
      SOAPFault soapFault = e.getFault();
      if(soapFault != null) {
         // Output the information about SOAP fault
         System.err.println("faultCode=" + soapFault.getFaultCode());
         System.err.println("faultActor=" + soapFault.getFaultActor());
         System.err.println("faultString=" + soapFault.getFaultString());
         Detail detail = soapFault.getDetail();
         if(detail != null) {
           for(Iterator ite = detail.getDetailEntries(); ite.hasNext(); ) {
             printDetail((Element)ite.next());
          }
        }
      }
    }
}
```

The structure of `Element[]` acquired with the `getDetail()` method is as shown in the following table:

Table 8–20: Structure of Element[] acquired with the getDetail() method

Name	Content
`errorMessage`	This is the error message set up in the exception.
`errorCode`	This is the error code corresponding to the error message set up in an exception.
`processInstanceID`	This is the process instance ID of the business process. Because the process instance ID is not set up for an error detected in the messaging infrastructure, this will become `null` (`nil` attribute). This is applicable for an error occurring in the business process.

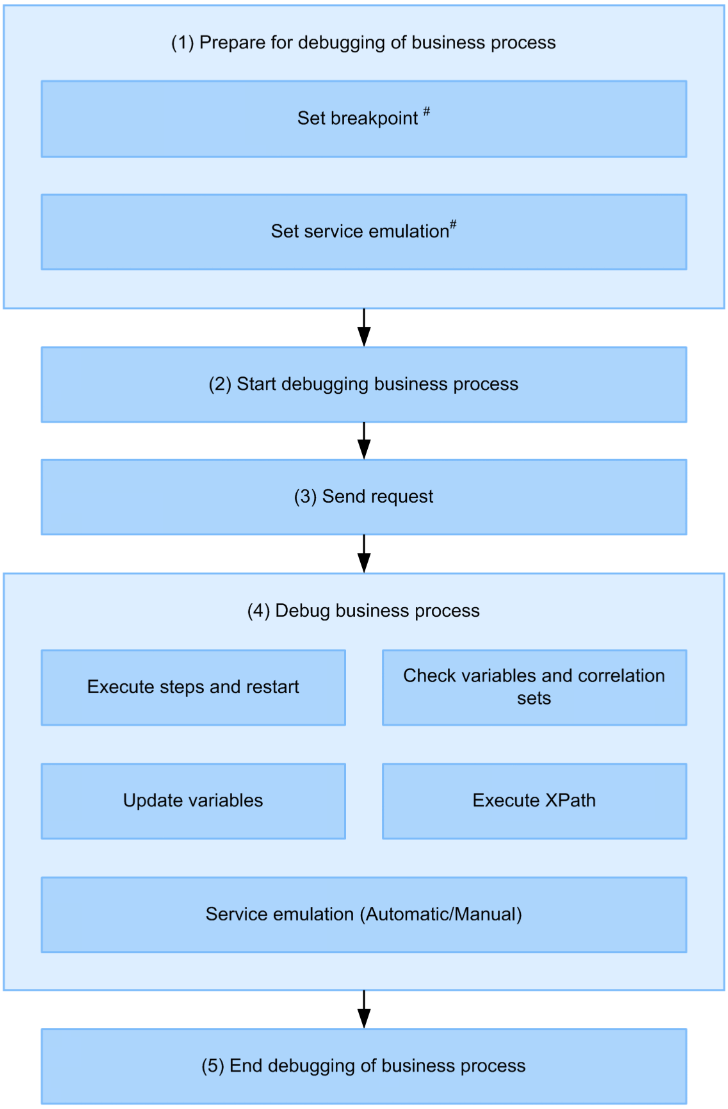
9 Debugging Business Processes

This section describes the flow of debugging and the operation procedure of the business process.

9.1 Flow of Debugging

The following figure shows the flow of debugging of the business process.

Figure 9–1: Flow of debugging of the business process


(1) Prepare for debugging of business process
Set breakpoint #
Set service emulation#
(2) Start debugging business process
(3) Send request
(4) Debug business process
Execute steps and restart
Check variables and correlation sets
Update variables
Execute XPath
Service emulation (Automatic/Manual)
(5) End debugging of business process

The breakpoint and service emulation can be set when debugging the business process.

The following describes operations related to debugging of the business process.

(1) Preparing for debugging of business processes

Implement the following to prepare for debugging of the business process.

- Set breakpoints
  To interrupt processing of the process instance in any activity, set a breakpoint in the activity.
  For setting breakpoints, see *9.2.1 Setting Breakpoints*.
- Set service emulation
  To use a created message instead of the response message for actually invoking the service, set service emulation.

For setting service emulation, see *9.2.2 Setting Service Emulation*.

(2) Starting debugging of business processes

Start debugging the business process.

For the method for starting debugging of the business process, see *9.3 Starting debugging of business processes*.

(3) Sending requests

Use the service requester and service requester emulation for sending requests to the receive activity. Once the request is sent, processing of the process instance continues till the activity for which a breakpoint is set and then processing of the process instance is automatically interrupted.

For sending requests, see *9.4 Sending requests*.

(4) Debugging business processes

Implement the following for debugging the business process.

- Executing steps and restarting

  You can continue sequential processes while processing of the process instance in activity units is interrupted. You can execute them while processing of the process instance is interrupted.

  For the method for executing steps and restarting, see *9.5.1 Step-by-Step Execution and Restarting*.

- Checking variables and correlation sets

  Check the variable and correlation set used in the business process. You can execute while processing of the process instance is interrupted.

  For the method for checking variables and correlation sets, see *9.5.2 Checking Variables and Correlation Sets*.

- Updating variables

  Change the value of the variable used in the business process and reflect it in the business process. You can execute while processing of the process instance is interrupted.

  For the method for updating variables, see *9.5.3 Updating Variables*.

- Evaluating XPath

  You can evaluate the validity of the conditional expression specified in the switch and assign activities and see a part of the variable values. You can execute while processing of the process instance is interrupted.

  For the method for evaluating XPath, see *9.5.4 Evaluating XPath*.

- Service emulation (automatic/manual)

  Use the set response message instead of the service response message.

  For the method for executing automatic service emulation, see *9.5.5 Automatic Service Emulation*. For the method for executing manual service emulation, see *9.5.6 Manual Service Emulation*.

(5) Ending debugging of business processes

Debugging of the business process ends when the business process is redefined and its operations are checked.

For the method for ending debugging of the business process, see *9.6 Ending Debugging of Business Processes*.

9.2 Preparing for Debugging of Business Processes

Set breakpoints and service emulation to prepare for debugging of business processes.

You can also set the breakpoint and service emulation after debugging of the business process starts. In such cases, implement when the processing of the process instance is interrupted. You can check the processing status of the process instance in debug view. For details about debug view, see *Cosminexus Service Platform Reference*.

9.2.1 Setting Breakpoints

To interrupt processing of the process instance in an activity, add the breakpoint in the activity to be interrupted. The added breakpoint changes, exports and imports criteria setting if required.

For the types of activities for which breakpoints can be set, see *9.5.1(1) Activities that can interrupt the processing of process instance*.

(1) Adding breakpoints

Two types of breakpoints exist-ordinary breakpoints and conditional breakpoints. If an ordinary breakpoint is added, the processing of the process instance is always interrupted when the process continues in the activity. If a conditional breakpoint is added, when the process continues in the activity and the conditional expression entered is fulfilled, (evaluation result is true), processing of the process instance is interrupted.

The following describes the procedure for adding ordinary breakpoints and conditional breakpoints in an activity.

(a) Adding ordinary breakpoints

1. In the Business Process Definition screen, right click the activity in which the breakpoint is to be added and choose **Add breakpoint**.

   The breakpoint is added in the activity and a check expressing the breakpoint is added next to the activity. For details about the Business Process Definition screen, see *Cosminexus Service Platform Reference*.

(b) Adding conditional breakpoints

1. In the Business Process Definition screen, right click the activity in which the breakpoint is to be added and choose **Add conditional breakpoint**.

   The **Set criteria** dialog box appears.
2. Enter the conditional expression in XPath.#
3. Click OK.

   The breakpoint is added in the activity and a check expressing the breakpoint is added next to the activity.

   When the process continues in the activity in which a breakpoint is added, if the conditional expression of the entered XPath is fulfilled (evaluation result is true), processing of the process instance is interrupted.

# If the conditional expression is not entered, the breakpoint is treated as an ordinary breakpoint.

(2) Changing criteria setting of breakpoints

You can change the interruption criteria if required for the ordinary and conditional breakpoints added.

The following describes the procedure for changing criteria setting of breakpoints.

1. In the Business Process Definition screen, right click the activity in which the breakpoint is to be added and choose **Change criteria**.

   The Set criteria dialog box appears.
2. Enter the conditional expression in XPath.#
3. Click OK.

   Criteria setting of the breakpoint changes.

When the process continues in the activity in which criteria setting of the breakpoint has changed, if the conditional expression of the entered XPath is fulfilled (evaluation result is true), processing of the process instance is interrupted.

# If the conditional expression is not entered, the breakpoint is treated as an ordinary breakpoint.

(3) Deleting breakpoints

In the Business Process Definition screen, right click the activity in which the breakpoint is to be deleted and choose **Delete breakpoint**. The target activity breakpoint is deleted.

(4) Disabling breakpoints

You can use standard functionalities of Eclipse for temporarily disabling from breakpoint view the breakpoint added in the activity.

For details about breakpoint view, see *Cosminexus Service Platform Reference*.

(5) Exporting and importing breakpoints

You can use standard functionalities of Eclipse for exporting the breakpoint added in the activity. You can also import the exported breakpoint.

For details about how to export and import see Eclipse documents.

! Important note

Before importing a breakpoint, breakpoints set in all activities must be deleted. If a breakpoint setting remains in an activity, breakpoints might not be imported normally.

9.2.2 Setting Service Emulation

To use a created response message instead of invoking the actual service, set service emulation.

You can use service emulation to continue processing process instances even if the service is not invoked from the business process.

Two types of service emulation functionalities exist: automatic emulation and manual emulation.

Automatic service emulation is the functionality that executes service emulation automatically when processing of the process instance continues to the service invocation activity. Before invoking the service from the service invocation activity, the operation of the service for emulation and the operation response messages must be set..

Manual service emulation is the functionality that sets response messages of services for emulation when processing of the process instance is interrupted by the service invocation activity. Service emulation is executed manually in this functionality.

(1) Setting automatic service emulation

In automatic service emulation, service operations, response types and response messages for emulation are set.

For the procedure for setting automatic service emulation, see *9.5.5 Automatic Service Emulation*.

(2) Setting manual service emulation

In manual service emulation, response types and response messages are set to be used when processing of process instances is interrupted by service invocation activities.

For the procedure for setting manual service emulation, see *9.5.6 Manual Service Emulation*.

9.3 Starting debugging of business processes

The following describes the procedure for starting the business process.

1. Start the test environment.

   For details about how to start the test environment, see *2.4.3(1) Starting and stopping the test environment*.

2. Start standard reception.

   This step is not required if standard reception is not used in the business process.

   For details about how to start standard reception, see *Cosminexus Service Platform System Setup and Operation Guide*.

3. Any of the following methods displays the Debug configuration dialog box.

   - In the Eclipse toolbar, choose Debug ( ), **Configure debug**.
   - In the List of service definitions of tree view, right click the business process for debugging and choose **Configure debug**.
   - Right click the target project in the package explorer and choose **Debug** and **Configure debug**.
   - In the Eclipse menu, choose **Execute** and **Configure debug**.

4. In the menu of the Debug configuration dialog box, right click **HCSC-BP** and choose **New**.

5. Enter the debug name in **Name**.

6. In the **Debug target** drop down list of the **Business process setting** tab, choose the business process for debugging.

7. Choose the **Debug setting** tab and specify the required information.

   For contents displayed or specified in the **Debug setting** tab, see *Cosminexus Service Platform Reference*.

8. Click **Debug**.

   Console view displays the expiry of the debug start process. If the business process, user-defined reception and service adapter are being edited, a dialog box appears for confirming whether to save the definitions being edited.

   The business process editor appears. For details about the business process editor, see *Cosminexus Service Platform Reference*.

   Once Eclipse starts, the **Authenticate account** dialog box appears when debugging of the business process first starts. Enter the user ID and password to start the start process of debugging the business process.

9. Click OK.

   Debugging of the business process starts.

   Once debugging of the business process starts, a shortcut is created for the debug configuration in the Debug menu of Eclipse. Thereafter, you can choose the shortcut to skip operations of the steps from 3-8 and start debugging.

   ! Important note

   - Configure the development environment, operation environment and execution environment in the same machine. Operations cannot be guaranteed for multiple machine configurations.
   - The HCSC server must be set up. Note that setup must be executed by a single HCSC server configuration. Even if HCSC easy setup is used, single HCSC server configuration must be set up.
   - Start the HCSC server and database in advance.
   - The following errors might occur when debugging of the business process starts.

     The repository configuration forms in the development environment and operation environment (combination of database and Cosminexus RM) do not match.

     The service deployed in the repository of the operation environment has changed or is deleted from the development environment.

     SOAP modes in the repository of the development environment and operation environment do not match.

     If an error occurs, stop all defined HCSC components, delete them from the HCSC server and restart debugging of the business process.
   - If an error occurs while debugging of the business process is starting, interrupt the process and exit. Processes that exited before the error occurred do not return to status before starting. In such cases, remove the cause of the error and restart.

- If the business process and service adapter change after debugging of the business process starts, the functions of debug might not operate normally. If the business process and service adapter change while debugging, end the debugging of the business process once and then restart debugging.
- Start debugging the business process in a single machine. Operations cannot be guaranteed for debugging started between multiple machines.
- Debugging of the business process can be started or ended only once. If multiple debug configurations are selected in Debug view, do not start or end debug.
- Debug starts only for the business process selected in **Debug target** of the **Debug configuration** dialog box. To debug simultaneously multiple business processes such as other business processes invoked from the business process, start debugging each business process.
- If debugging fails to start, the service cannot be started. Debugging of the business process might not start till the HCSC component stops. If debugging fails to start, execute in the order of the `csccompostop -all` command and the `csccompoundeploy -all` command, stop the HCSC component and undeploy. Then restart debugging the business process. You cannot stop the HCSC component by the methods in *7.6 Batch execution of processes for stopping HCSC components and deleting them from the HCSC server*.

  For details about the `csccompostop` command and `csccompoundeploy` command, see *Cosminexus Service Platform Reference*.
- The same business process cannot be duplicated and started.
- You can start debugging of only the business process having latest version. You cannot execute debugging of a business process having any other version.
- If you execute debugging of the business process, the following message is output in the message log multiple times.

  Message output when you try to read a value of variable, which is not initialized (KDEC20052-E)

  Message output when you try to read a value of correlation set, which is not initialized (KDEC20065-E)
- Do not execute following functions after starting the debugging of business process.

  Closing HCSCTE project

  Deleting HCSCTE project

  Changing repository directory

  Initializing repository

  Importing repository

  Importing services

  Exporting services

Reference note

While starting debugging of the business process, the process to deploy and start HCSC components in the HCSC server is executed in a batch. At this stage, the business process, service adapter and user defined reception are deployed and defined as follows:

- Business process

  The business process selected in the **Debug configuration** dialog box is repackaged and then deployed and defined.

  If the business process seen from the business process selected in the **Debug configuration** dialog box is already deployed and defined (public), it is not redeployed and defined when the debugging of the business process starts. In other cases, it is repackaged and then deployed and defined.
- Service adapter

  If it is already packaged and if it is already deployed and defined (public), it is not repackaged and then deployed and defined when the debugging of the business process starts.
- User-defined reception

  The user-defined reception of the business process selected in the **Debug configuration** dialog box is redeployed and defined.

  If the user-defined reception of the business process seen from the business process selected in the **Debug configuration** dialog box is already deployed and defined (public), it is not redeployed and defined when the debugging of the business process starts. In other cases, it is redeployed and defined.

9.4 Sending requests

Use the service requester and service requester emulation for sending requests to the receive activity. For details about service requester emulation, see *Appendix G. Emulating the Service Requester*.

If a request is sent, a dialog box appears for confirming whether to switch perspectives. Click **Yes** to switch from the **HCSCTE** perspective to the **Debug** perspective. Debug view, variable view, etc appear.

Once the request is sent, processing of the process instance starts and processing of the process instance is interrupted in the activity in which a breakpoint is set.

9.5 Debugging Business Processes

This section describes the operations that can be executed by debugging business processes.

Execute each operation for debugging the business process while processing of the process instance is interrupted. You can check the processing status of the process instance in debug view.

9.5.1 Step-by-Step Execution and Restarting

This subsection describes the types of activities that can interrupt the processing of process instances, and step-by-step execution and restart operations.

(1) Activities that can interrupt the processing of process instances

The following table lists the types of activities that can interrupt the processing of process instances.

Table 9–1: Types of activities that can interrupt the processing of process instances

Activity type	Activity	Whether the activity can be Interrupted
Basic activity	Start activity	No
	Receive activity	Yes
	Reply activity	Yes
	Invoke service activity	Yes
	Invoke java activity	Yes
	Data transformation activity	Yes
	Assign activity	Yes
	Empty activity	Yes
	Throw activity	Yes
	Standby activity	Yes
	Validate activity	Yes
	End activity	No
Structure activity	Scope activity	Yes
	While activity	Yes
	Switch start activity	Yes
	Switch end activity	No
	Flow start activity	Yes
	Flow end activity	No

Legend:

Yes: Can be interrupted

No: Cannot be interrupted

(2) Step-by-step execution and restart operations

To execute a process instance step-by-step or to restart an operation, select the interrupted activity in the debug view, and then click the icon of the operation to be executed.

The following table describes step-by-step execution and restart operations, and debugging operations.

Table 9–2: Step-by-step execution and restart operations, and debugging operations

Step-by-step execution	Icon	Debugging operations
Step in	⤵	Step in is executed. For details about the activities interrupted when a step in is executed for each activity, see *(3) Activities interrupted when a step in is executed*.
Step over	↷	Step over is executed. For details about the activities interrupted when a step over is executed for each activity, see *(4) Activities interrupted when a step over is executed*.
Step return	⤴	Step return is executed. For details about the activities interrupted when a step return is executed for each activity, see *(5) Activities interrupted when a step return is executed*.
Restart	▶	Restart processing of the process instance continues until the next activity for which a breakpoint is set.

To make fault handling interrupt the processing of a process instance when an invoke service activity switches to the fault handling, you need to set a breakpoint in the activity performing the fault handling or execute a step in.

(3) Activities interrupted when a step in is executed

The following table and figure show activities that are interrupted when a step in is executed for each activity.

Table 9–3: Activities interrupted when a step in is executed for each activity

Activity type	Activity	Explanation
Basic activity		Proceeds to the next activity.
Structure activities	Scope activity	Proceeds to the first activity to be processed within the scope activity.
	While activity	Proceeds to the first activity to be processed within the while activity.
	Switch start activity	Proceeds to the first activity to be processed between the switch start activity and the switch end activity.
	Flow start activity	Proceeds to the first activity to be processed between the flow start activity and the flow end activity.

Figure 9–2: Activities interrupted when a step in is executed in a basic activity

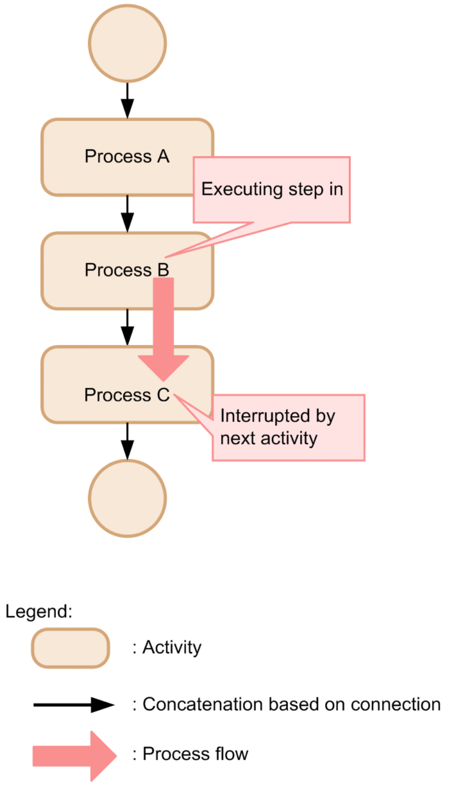


Figure 9–3: Activities interrupted when a step in is executed in a structure activity such as a scope or while activity

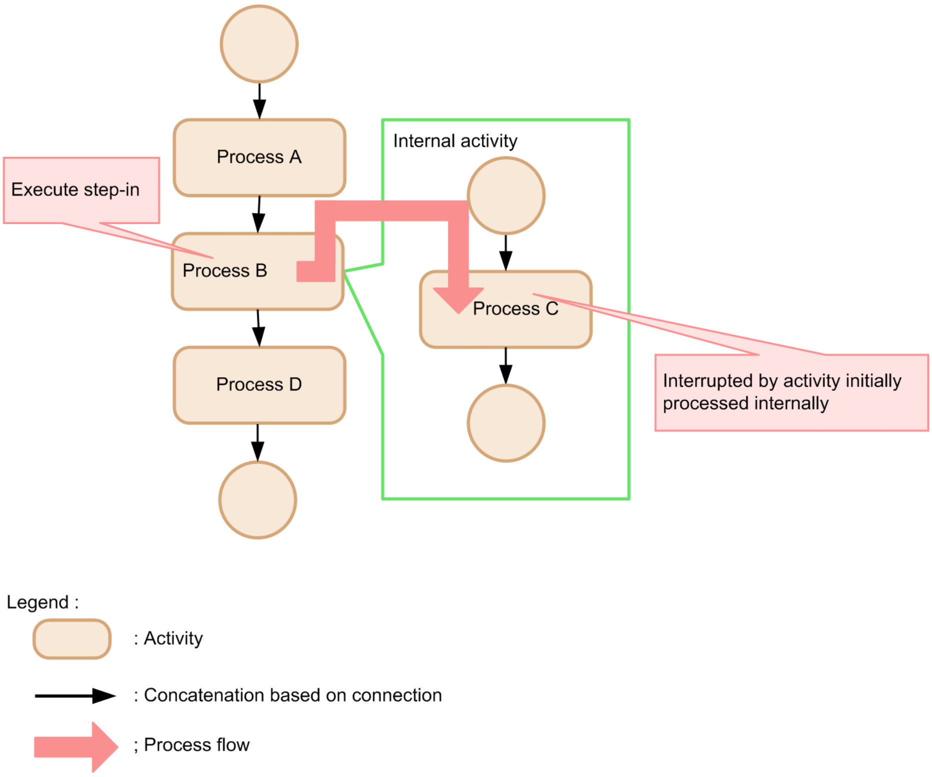
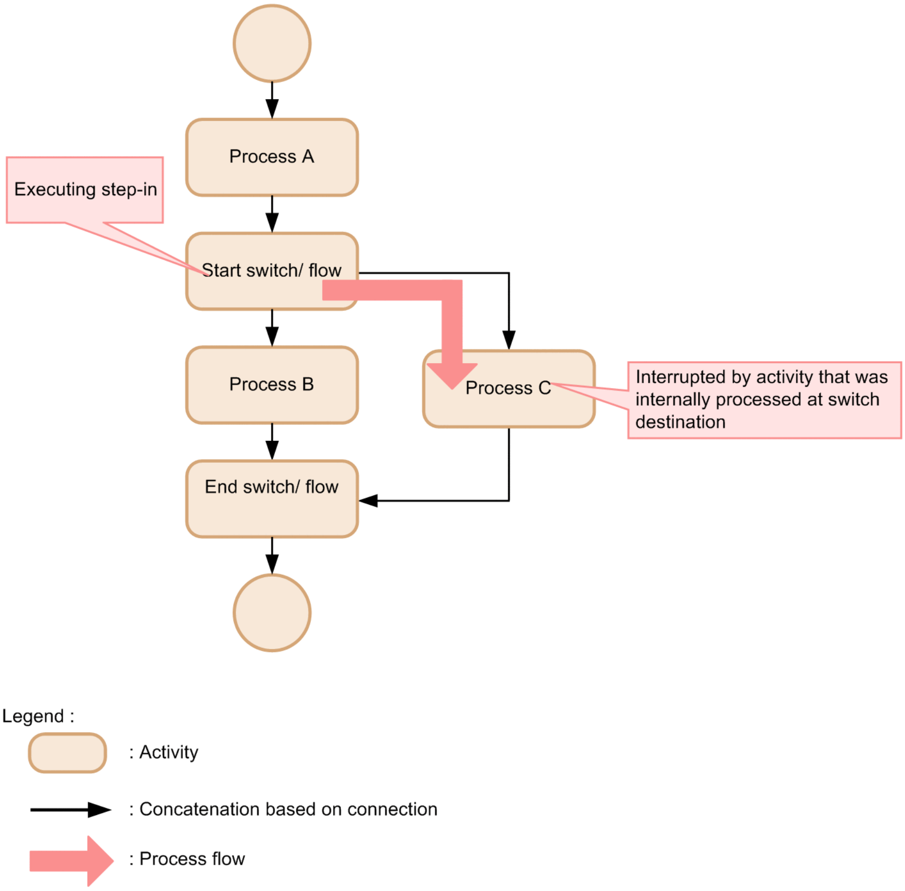


Figure 9–4: Activities interrupted when a step in is executed in a structure activity such as a switch or flow activity


Process A
Executing step-in
Start switch/ flow
Process B
Process C
Interrupted by activity that was internally processed at switch destination
End switch/ flow
Legend :
: Activity
: Concatenation based on connection
: Process flow

(4) Activities interrupted when a step over is executed

The following table and figure show activities that are interrupted when a step over is executed for each activity.

Table 9–4: Activities interrupted when a step over is executed for each activity

Activity type	Activity	Explanation
Basic activity		Proceeds to the next activity.
Structure activities	Scope activity	Processes the activities within the scope activity, and then proceeds to the next activity. If a breakpoint is set in an activity, processing is interrupted at that breakpoint.
	While activity	Processes the activities within the while activity, and then proceeds to the next activity. If a breakpoint is set in an activity, processing is interrupted at that breakpoint.
	Switch start activity	Processes until the activity following the switch end activity. If a breakpoint is set in an activity, processing is interrupted at that breakpoint.
	Flow start activity	Processes until the activity following the flow end activity. If a breakpoint is set in an activity, processing is interrupted at that breakpoint.

Figure 9–5: Activities interrupted when a step over is executed in a basic activity

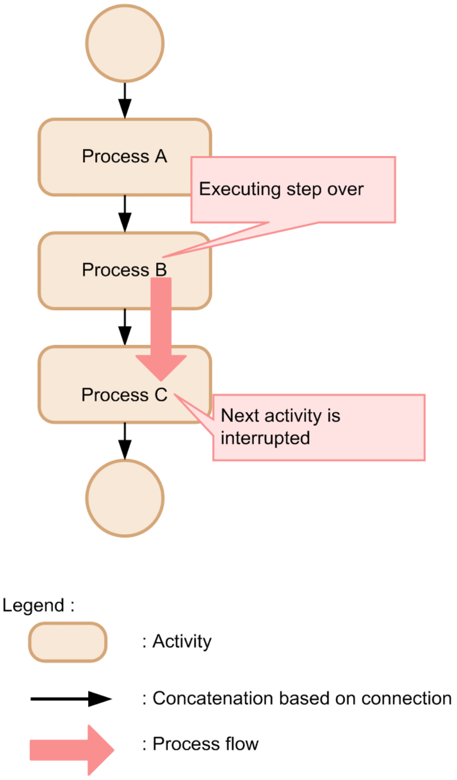


Figure 9–6: Activities interrupted when a step over is executed in a structure activity such as a scope or while activity

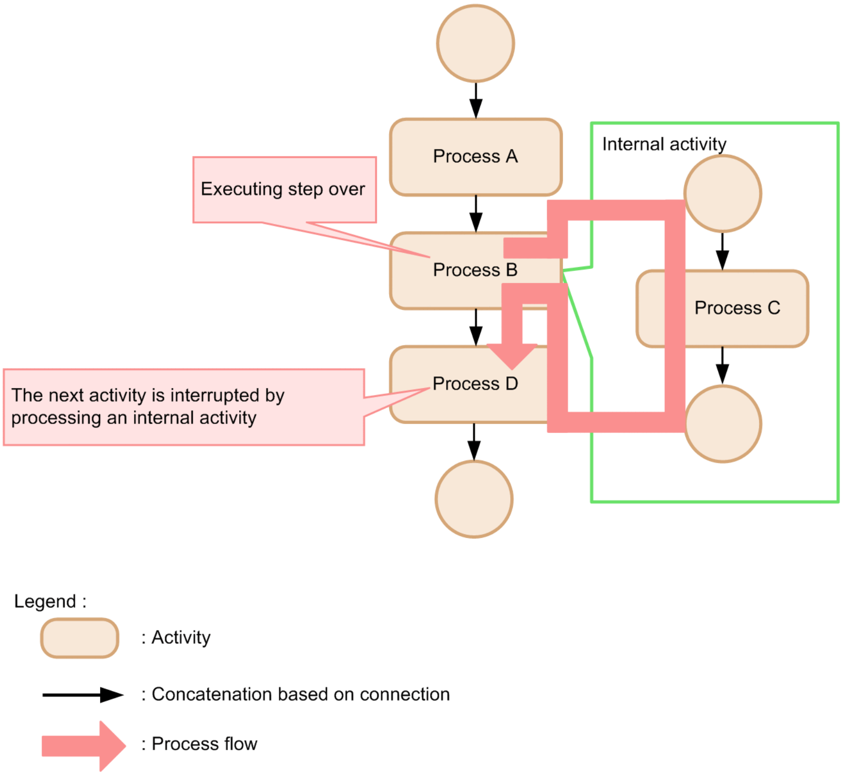
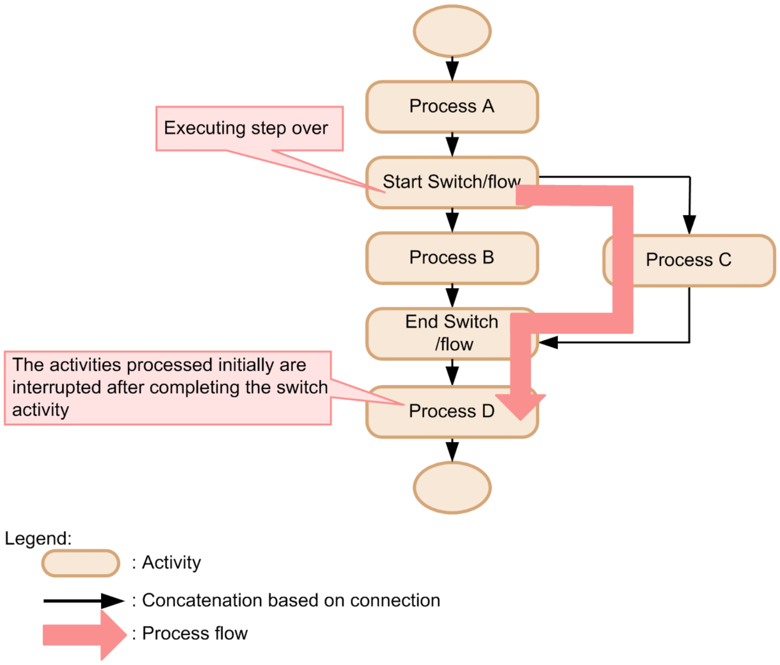


Figure 9–7: Activities interrupted when a step over is executed in a structure activity such as a switch or flow activity


Process A
Executing step over
Start Switch/flow
Process B
Process C
End Switch /flow
The activities processed initially are interrupted after completing the switch activity
Process D
Legend:
: Activity
: Concatenation based on connection
: Process flow

(5) Activities interrupted when a step return is executed

This subsection describes activities that are interrupted when a step return is executed for each activity.

If a step return is executed in a basic activity within a structure activity, processing continues until the activity following the structure activity to which the basic activity belongs. If the structure activity containing the activity where the step return is executed is a scope activity or while activity, it processes all its internal activities, and then proceeds to the next activity. If the structure activity containing the activity where the step return is executed is a switch start activity or flow start activity, processing continues until the activity following the corresponding switch end activity or flow end activity.

If the interrupted activity does not exist in the structure activity, the same operation as for a restart operation is performed.

If a breakpoint is set in an activity, processing is interrupted at that breakpoint.

The following figure shows the activities that are interrupted when a step return is executed in each activity within a structure activity.

Figure 9–8: Activities interrupted when a step return is executed in a basic activity within a structure activity

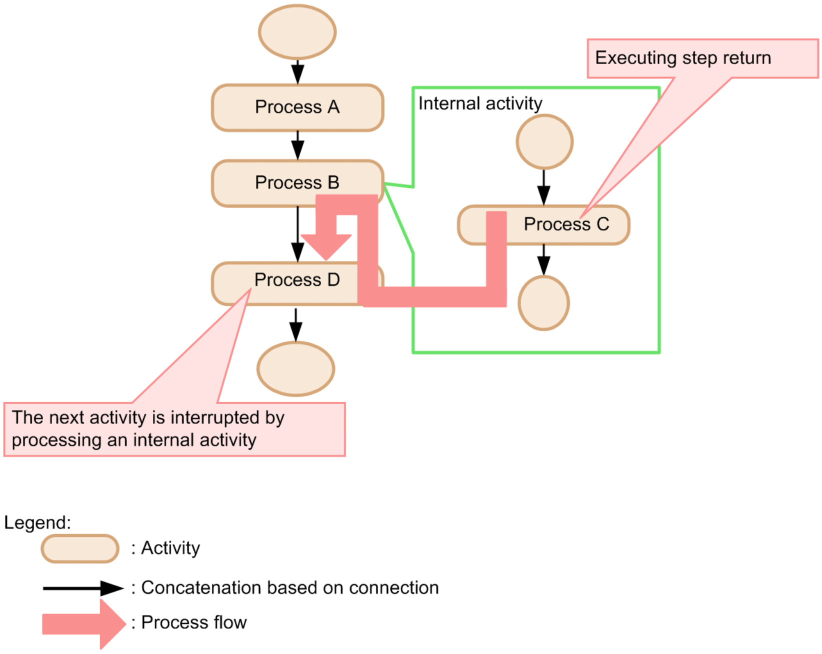


Figure 9–9: Activities interrupted when a step return is executed in a structure activity such as a scope or while activity within a structure activity

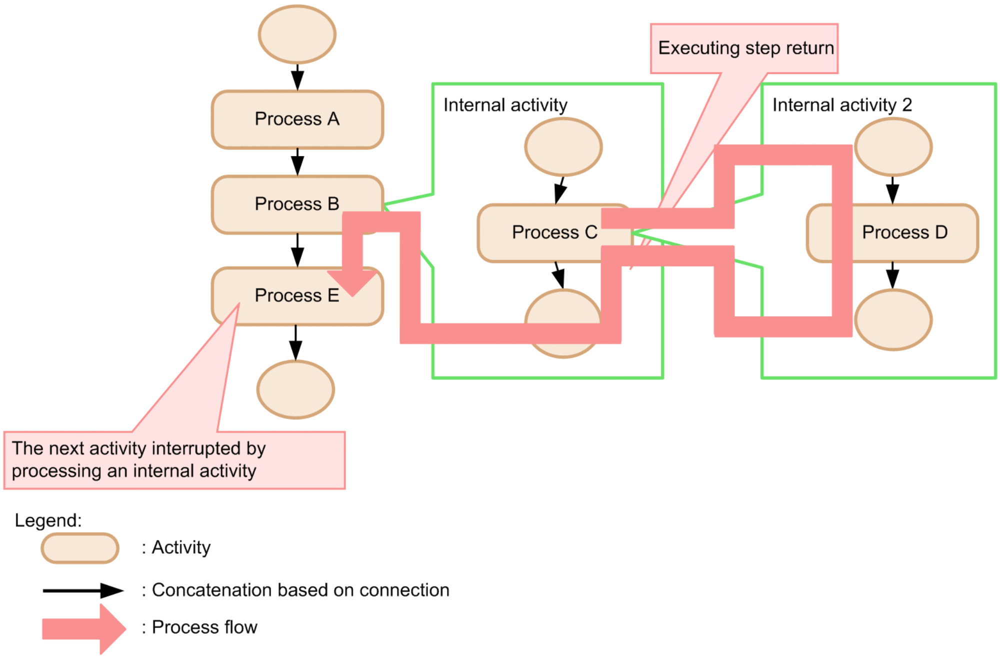
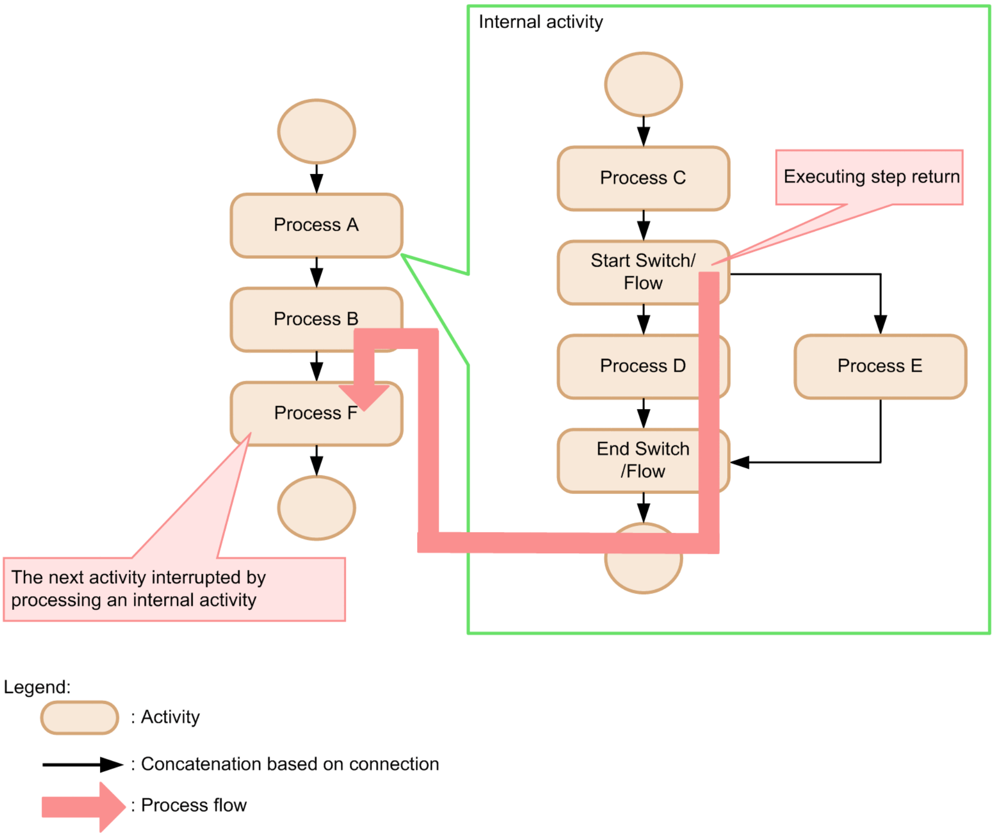


Figure 9–10: Activities interrupted when a step return is executed in a structure activity such as a switch or flow activity within a structure activity



9.5.2 Checking Variables and Correlation Sets

When processing of a process instance is interrupted, if you choose the interrupted activity in debug view, the variable name, variable value, correlation set name and correlation ID currently used in the business process appear in variable view.

For details about variable view, see *Cosminexus Service Platform Reference*.

9.5.3 Updating Variables

Change the value of the variable appearing in variable view and reflect the change in the business process. If the value is not set for the variable, it cannot be updated.

The following table shows the scope of values that can be entered for each variable type.

Table 9–5: Scope of values that can be entered for each variable type

Variable type	Scope of values
numeric type	Value that can be interpreted by the valueOf(java.lang.String) method of java.lang.Double (Example:+1, 3.14, 1e-2d)
boolean type	True or false
string type	Any character string (Example:15-inch LCD display)
message type#1	• In case of XML type Any XML(example: <?xml version="1.0" encoding="UTF-8"?><message>Hello</message>) • In case of non-XML type#2

Variable type	Scope of values
message type[#1]	Any binary data (example: Hello) - In case of any type[#2] Any format

#1

If the message type variable changes, the XML character always changes to UTF-8 (even if a character code other than UTF-8 is specified, it converts to UTF-8).

#2

When the variable type is non-XML type and any type, enter the value in hexadecimal expression.

Variables can be updated by the method for changing in the **Set value** dialog box, the method for changing from the input field of Variable view or the method for changing from the **Value** cell of Variable view.

The following describes the procedure for updating variables.

(1) Updating in the Set value dialog box

1. Right click the variable to be updated in Variable view and choose **Change value**.

   The **Set value** dialog box appears. The contents displayed in the **Set value** dialog box differ for message type and other than message type. For details about the **Set value** dialog box in each case, see *Cosminexus Service Platform Reference*.

2. Enter a value in the input field of the **Set value** dialog box.

   If the variable type is message type, click Save XML to save the entered variable value in the XML file. Click **Read XML** to read in the input field the variable value saved in the XML file.

3. Click **OK** button.

   The entered variable value is reflected in the business process. The cell of the updated variable appears in yellow.

(2) Updating from the input field of Variable view

1. In Variable view, choose the variable to be updated.

   The selected variable value appears in the input field in the lower part of Variable view.

2. Enter the value in the input field.

3. Update the variable by either of the following methods:

   - Click the **Ctrl** key + **S** key
   - Right click and choose **Assigned value**

   The entered variable value is reflected in the business process. The cell of the updated variable appears in yellow.

(3) Updating from the **Value** cell of Variable view

1. In Variable view, choose the **Value** cell of the variable to be updated.

2. Enter the value in the **Value** cell.

3. Click the **Enter** key.

   The entered variable value is reflected in the business process. The cell of the updated variable appears in yellow.

9.5.4 Evaluating XPath

You can evaluate the validity of the conditional expression specified in the switch and assign activities and see a part of the variable values. You can execute while processing of the process instance is interrupted.

! Important note

When debugging a business process, do not evaluate an XPath expression that contains linefeed code in the data. If you need to evaluate an XPath expression that contains linefeed code in the data, use the `normalize-space()` function in XPath to replace linefeed characters with single spaces before evaluating the XPath expression.

The following describes the procedure for evaluating XPath.

1. In the Eclipse menu, choose **Window**, **View display** and **Others**.
   The **View display** dialog box appears.
2. Choose **Debug** and **Evaluate HCSC XPath** and then click the **OK** button.
   HCSC XPath evaluation view appears. For details about HCSC XPath evaluation view, see *Cosminexus Service Platform Reference*.
3. In Debug view, choose the business process activity for evaluating the conditional expression.
4. Enter the XPath expression in the input field of HCSC XPath evaluation view and click **Evaluate**.
   The evaluation result for the XPath expression appears below the input field.

9.5.5 Automatic Service Emulation

This subsection explains how to set up and execute automatic service emulations.

(1) Setting up an automatic service emulation

1. From the Eclipse menu, select **Window**, **Show View**, and then **Others**.
   The Show View dialog box appears.
2. Select **Debug** and then **HCSC Auto Emulate**, and then click **OK**.
   The HCSC Auto Emulate view appears.
   For details about the HCSC Auto Emulate view, see *1.2.4 Debug Business Process screen* in the manual *uCosminexus Service Platform Reference Guide*.
3. Click **Add** in the HCSC Auto Emulate view.
   A row is added to the table in the HCSC Auto Emulate view.
4. Click the **Service name** cell, and then select the service to be emulated from the drop-down list.
5. Click the **Operation name** cell, and then select the operation of the service to be emulated from the drop-down list.
6. Click the **Response Type** cell, and then select **Normal Response** or **Fault Response** from the drop-down list.
7. Click the **Response Message** cell, and then click the **...** button.
   The Select file dialog box appears.
8. Select an applicable response message file, and then click **OK**.#
   If you specify a file with the `.xsl` extension, the result of applying the specified file appears as a response message in the request message for the invoke service activity.
   If you specify a file with an extension other than `.xsl`, the contents of the file are used directly as response messages.
   #
   You can specify only response messages where UTF-8 is specified for the XML character encoding.
9. If necessary, select the **Condition** cell, and then enter conditions in the XPath expression.
   If the result of evaluating the conditional expression against the service request message is true, automatic service emulation is executed. If no condition is entered, the result is always evaluated as true.
10. Select the check box for the **Valid** cell.
    Automatic service emulation is set.

To save the settings of the HCSC Auto Emulate view to an external file, click **Save** in the HCSC Auto Emulate view.

By clicking **Load**, you can obtain the settings saved in the external file. If settings already exist in the HCSC Auto Emulate view, a dialog box is displayed to confirm whether to overwrite the existing settings of the HCSC Auto Emulate view.

If multiple combinations of the same service and operation are set in the HCSC Auto Emulate view, emulation for the service with the highest priority is executed. To change the priority order of the service, select the row to be changed, and then click **Up** or **Down**.

If you right-click the file name of a response message and then select **Show file**, you can open the target file.

If multiple business processes are debugged, automatic emulation settings (service names and operation names) are enabled for all business processes.

Note that if manual emulation is executed on an invoke service activity that is the target for automatic emulation, manual emulation takes precedence over automatic emulation.

(2) Executing automatic service emulation

As the processing of a process instance proceeds in the invoke service activity specified in the HCSC Auto Emulate view, the specified response message is used automatically instead of the response message for the service.

If you specify an XML format file for a response message, the contents of the file are emulated directly as the response message for the service. If an XSL format file is specified for a response message, the result of applying the contents of the file to the request message for the service is emulated as the response message for the service.

Note that if automatic service emulation is not set, normal service invocation is executed.

! Important note

If the HCSC Auto Emulate view is closed, automatic service emulation is not executed.

9.5.6 Manual Service Emulation

This subsection explains how to set up and execute manual service emulation.

(1) Setting up manual service emulation

1. From the Eclipse menu, select **Window**, **Show View**, and then **Others**.
   The Show View dialog box appears.
2. Select **Debug** and then **HCSC Emulate**, and then click **OK**.
   The HCSC Emulate view appears.
   For details about the HCSC Emulate view, see *1.2.4 Debug Business Process screen* in the manual *uCosminexus Service Platform Reference Guide*.
3. In the Debug view, select the activity to be manually emulated.
4. Click **Add** in the HCSC Emulate view.
   A row is added to the table in the HCSC Emulate view.
5. Click the **Response Type** cell, and then select **Normal Response** or **Fault Response** from the drop-down list.
6. Click the **Response Message** cell, and then click the **...** button.
   The Select file dialog box appears.
7. Select an XML file (for XML format)# or a file in any format (for non-XML or any format) for a response message, and then click **OK**.
   Service emulation is set.
   #
   You can specify only XML files where UTF-8 is specified for the XML character encoding.

To save the settings of the HCSC Emulate view to an external file, click **Save** in the HCSC Emulate view.

By clicking **Load**, you can obtain the settings saved in the external file. If settings already exist in the HCSC Emulate view, a dialog box is displayed to confirm whether to overwrite the existing settings of the HCSC Emulate view.

If you right-click the file name of a response message and then select **Show file**, you can open the target file.

(2) Executing manual service emulation

If you click **Resume** or **Step Over** in the HCSC Emulate view when the processing of a process instance is interrupted in an invoke service activity, the specified response message is used instead of the response message for the service.

To make fault handling interrupt the processing of a process instance when a fault response is emulated to switch from an invoke service activity to the fault handling, you need to set a breakpoint in the activity performing fault handling.

9.6 Ending Debugging of Business Processes

This section describes the procedure to end debugging and the status of business processes and activities when debugging ends.

End debugging of the business process when the process instance is interrupted. You can check the processing status of the process instance in debug view.

The following describes the procedure for ending debugging.

1. Choose the debugging business process in Debug view.
2. Click the **End** icon (🟥).

   Debugging of the business process ends.

When debugging ends in the processing of the process instance, the status of the business process and activity is the same as the status when transaction commit is executed just before debugging ends.

For the transaction commit time, see *5.6 Defining Activities*.

! Important note

Even after debugging ends for the business process and service adapter used for business process debugging, it remains in the deployed and started state in the HCSC server. If required, stop the HCSC component by the method described in *7.6 Batch execution of processes for stopping HCSC components and deleting them from the HCSC server*and delete it from the HCSC server.

Appendixes

A. Migrating from an Earlier Version

This appendix describes how to upgrade a version when the repository information that is used in the development environment of the earlier version is also to be used in the upgraded version.

For details about how to upgrade versions when the repository information used in the earlier version is to be used as it is for executing operations in the upgraded version, see the manual *Cosminexus Service Platform System Setup and Operation Guide*.

Note

> You can migrate the repository information used in the development environment in the versions prior to 08-10, only in case of SOAP 1.1 mode. From version 08-50 onwards, you can migrate the information in case of SOAP 1.1 mode and SOAP 1.1/1.2 combined mode.

A.1 Versions Wherein Migration Is to Be Performed

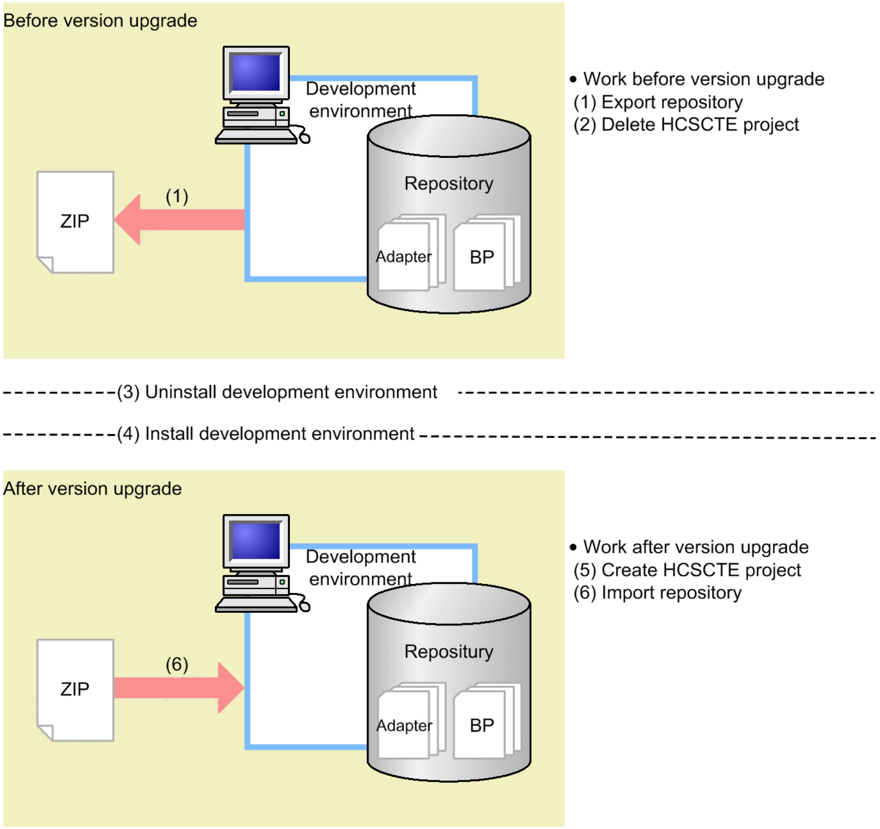
The old versions wherein the repository information is to be migrated are described below. You can migrate the repository information of these versions, and upgrade development environment to version 09-60.

- uCosminexus Service Architect 07-10
- uCosminexus Service Architect 07-20
- uCosminexus Service Architect 07-50
- uCosminexus Service Architect 07-60
- uCosminexus Service Architect 08-00
- uCosminexus Service Architect 08-10
- uCosminexus Service Architect 08-50
- uCosminexus Service Architect 08-51
- uCosminexus Service Architect 08-53
- uCosminexus Service Architect 08-70
- uCosminexus Service Architect 09-00
- uCosminexus Service Architect 09-50
- uCosminexus Service Architect 09-51

A.2 Migrating from an Earlier Version

Migrate Service Platform (development environment) from an old version to 09-60 according to steps (1) to (6) in the following figure:

Figure A–1: Procedure for migrating from an earlier version



• Work before version upgrade
(1) Export repository
(2) Delete HCSCTE project

• Work after version upgrade
(5) Create HCSCTE project
(6) Import repository

Steps (1) to (6) in Figure A-1 are explained in detail, as below.

(1) Exporting the repository

Before upgrading the version, export and save the repository of the development environment. If multiple repositories are being used, export and save all the necessary repositories.

For details about how to export a repository, see *3.2.2 Exporting a Repository*.

(2) Deleting the HCSCTE projects

Delete the HCSCTE projects.

For details about the procedure, see *3.1.4 Deleting a Project*.

(3) Uninstalling the development environment

Uninstall the old version of Service Platform from the development environment. Hitachi recommends that you back up the directories and files in the installation directory for Service Platform before uninstalling the old version.

Note

- Stop the components of the execution environment (such as the J2EE server, Management Server, and PRF) before uninstalling Service Platform.
- If you are using an embedded database, stop the database before uninstalling Service Platform.

(4) Installing the development environment

In the development environment, install version 09-60 of Service Platform to upgrade the existing version.

(5) Creating an HCSCTE project

Create a new HCSCTE project in the upgraded development environment. Specify the directory to be used as a repository. For the directory to be used as a repository, do not specify the repository of the production environment.

For details about the procedure, see *3.1.1 Creating a Project*.

If an external binding file has been used in the old version, you need to specify the file again in Eclipse. For details about the procedure, see *Appendix L. Customizing WSDL using the external binding file*.

(6) Importing the repository

Import the repository information (exported in *(1) Exporting the repository*) into the development environment. Before importing the repository, select the **Project** menu of Eclipse, and then clear the **Build Automatically** option. After importing the repository, select the **Build Automatically** option again.

When you import the repository of the old version, packaging and deployment definition are executed automatically, and the repository information is inherited into the current version. Note that packaging and deployment definition are automatically executed only for service adapters, business processes, and user-defined reception interfaces for which packaging and deployment definition have been executed in the old version. Add a new database adapter rather than using the old version of the database adapter. For details about how to add new database adapters, see *3.2.5 Adding New Database Adapters* in the *Service Platform Reception and Adapter Definition Guide*.

For details about how to import a repository, see *3.2.3 Importing a Repository*.

Note

Notes on migrating from a version earlier than 07-50

- If the following file names are specified in the user-defined class of the SessionBean adapter, a warning message appears:
  - `csmsvcadpdef.jar`
  - `cscmsg_adpejb.jar`

  If the warning message appears, delete the above user-defined class in the Service Adapter Settings window, change the file name, and then specify the file name again.
- If a message format schema file where the default namespace is not specified is set as the variable for a business process defined in a version earlier than 07-50, register the message format file again after the version is upgraded.

Notes on migrating from version 07-60 or later

When migrating from version 07-60 or later, import the service adapter and user-defined reception interface to be deployed in the execution environment into the development environment, and then repackage them. If you do not repackage the service adapter, the `KDEC03007-E` message might not be output. Similarly, if you do not repackage the user-defined reception interface, invalid padding characters might be output in the `KDEC00001-E` message. For details about packaging methods, see *7.2 Packaging*.

Notes on migrating from a version earlier than 08-10

If the following elements (compositors) with the occurrence count fixed at "once" are defined below the `sequence` or `choice` element in the data transformation definition, the node display is changed. Therefore, an error message is displayed indicating that the XML Schema used in the mapping definition file has changed.

- `sequence`
- `choice`

If an error message is displayed, start the mapping definition again to apply the changes.

Notes on migrating from a version earlier than 09-50

If a schema that meets the conditions below is defined in the data transformation definition, the namespace is changed. Therefore, an error message is displayed indicating that the XML Schema used in the mapping definition file has changed.

1. A schema that has its target namespace (`targetNamespace`) defined at the `import` destination is specified for the transformation source node.
2. The namespace in 1 above is not defined in any namespace declarations (`xmlns[:prefix]`) including `import` and `include` destinations.

If an error message is displayed, start the mapping definition again to apply the changes.

Notes on memory size

When importing an earlier-version repository, you need a large amount of memory. Therefore, use the following procedure to check whether the memory size is sufficient. If the memory is insufficient, increase the memory size before importing the repository.

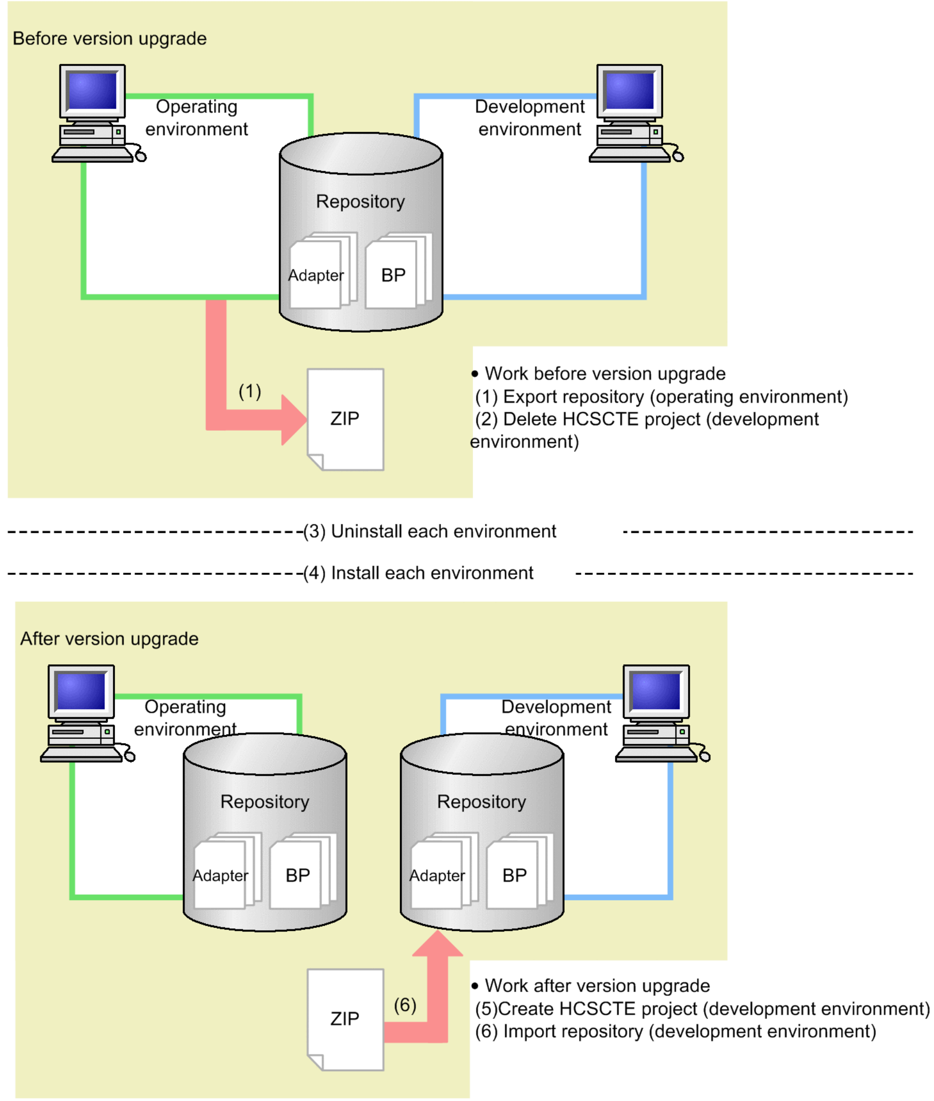
1. From the Eclipse menu, select **Window** and then **Preferences**.
2. Select **General** in the tree view on the left side of the dialog box.
3. Select the **Show heap status** check box on the right side of the dialog box.
4. Select the recycle bin icon on the bottom right of the Eclipse window, and then run the garbage collector.
5. Check the heap size shown to the left of the recycle bin icon.
6. If the amount of unused heap memory is insufficient, edit the `eclipse.ini` file to increase the memory size specified in `-Xmx`. For details about how to edit `eclipse.ini`, see *Appendixes B.1 Installation* in the *Application Server Application Development Guide*.

To perform actual operations by using the repository information imported into the development environment, export this repository information from the development environment to the operation environment, and deploy the HCSC components from the operation environment.

A.3 Migrating procedure when a repository is shared between development environment and operating environment in earlier version

In 07-60 and higher versions, you cannot share the same repository between a development environment and an operating environment. If the same repository is shared between a development environment and an operating environment, migrate the Cosminexus Service Platform from the earlier version to the version 09-60 according to steps (1) to (6) shown in the following figure:

Figure A–2: Procedure for migrating from an earlier version (When the repository is shared between the development environment and operating environment)


Before version upgrade
Operating environment
Development environment
Repository
Adapter
BP
(1)
ZIP
• Work before version upgrade
(1) Export repository (operating environment)
(2) Delete HCSCTE project (development environment)
(3) Uninstall each environment
(4) Install each environment
After version upgrade
Operating environment
Development environment
Repository
Adapter
BP
Repository
Adapter
BP
ZIP
(6)
• Work after version upgrade
(5)Create HCSCTE project (development environment)
(6) Import repository (development environment)

The details of steps (1) to (6) of Figure A-2 are described below.

(1) Repository export (operating environment)

Execute the `cscrepctl` command (the `-export` option) in the operating environment and export the repository prior to the version upgrade. If multiple repositories are being used, export all the required repositories and save them.

(2) Deleting the HCSCTE project (Development environment)

Delete the HCSCTE project once.

For details about the procedure, see *3.1.4 Deleting a Project*.

(3) Uninstalling the environment

Uninstall the Cosminexus Service Platform of the earlier version in the development environment and the operating environment. Hitachi recommends that you save the directory information under Cosminexus, and then uninstall the Cosminexus Service Platform of the earlier version.

#

- Uninstall components of the execution environment, such as the J2EE server, the Management Server, and PRF after stopping them.
- If you are using an embedded database, stop the embedded database before uninstalling.
- After unsetting up the HCSC server, delete all the entries under the repository root.

(4) Installing each environment

In the development environment and the operating environment, install the Cosminexus Service Platform of version 09-60 to upgrade the version.

(5) Creating an HCSCTE project (development environment)

Create a new HCSCTE project in the upgraded development environment. Specify the directory to be used as a repository. Do not specify the repository of the operating environment in the directory to be used as repository.

For details about the procedure, see *3.1.1 Creating a Project*.

(6) Importing the repository (Development environment)

Import the repository information that was exported in *Appendix A.3(1) Repository export (operating environment)* to the development environment. When importing, in the Eclipse menu, turn OFF **Project** - **Build automatically**, and then import the repository. After you have imported the repository, turn it back to ON.

When you import the repository of an earlier version, packaging and deployment definition are executed automatically, and the repository information is inherited into the current version. Note that automatic execution of packaging and deployment definition is performed only for the HCSC components for which packaging and deployment definition were executed in the earlier version.

For details about how to import a repository, see *3.2.3 Importing a Repository*.

#

Notes when migrating from a version earlier than 07-50

If the following file names are set up in the user-defined class of the service adapter (SessionBean), a warning message will appear.

- `csmsvcadpdef.jar`
- `cscmsg_adpejb.jar`

When the warning message appears, delete the above user-defined class in the Service Adapter Definition screen, change the file name, and then set up the name again.

Notes when migrating from versions prior to 08-10

If the following elements (compositor) with the occurrence count fixed at "once" are defined below the `sequence` or `choice` elements in the data transformation definition, the node display is changed; therefore, an error message is displayed indicating that the XML Schema used in the mapping definition file is changed:

- `sequence`
- `choice`

If the error message is displayed, start the mapping definition again to apply the changes.

Notes on migrating from a version earlier than 09-50

If a schema that meets the following conditions is defined in the data transformation definition, the namespace is changed. Therefore, an error message is displayed indicating that the XML Schema used in the mapping definition file has changed.

1. A schema that has its target namespace (`targetNamespace`) defined at the `import` destination is specified for the transformation source node.
2. The namespace in 1 above is not defined in any namespace declarations (`xmlns[:prefix]`) including `import` and `include` destinations.

If an error message is displayed, start the mapping definition again to apply the changes.

Notes regarding the memory

When importing an earlier-version repository, you need a large amount of memory. Therefore, use the procedure below to check whether the memory size is sufficient. If the memory is insufficient, increase the memory size before importing the repository.

1. From the Eclipse menu, select **Window** and then **Preferences**.
2. Select **General** in the tree view on the left side of the dialog box.
3. Select the **Show heap status** check box on the right side of the dialog box.
4. Select the recycle bin icon on the bottom right of the Eclipse window, and then run the garbage collector.
5. Check the heap size shown to the left of the recycle bin icon.
6. If the amount of unused heap memory is insufficient, edit the `eclipse.ini` file to increase the memory size specified in `-Xmx`. For details about how to edit `eclipse.ini`, see *Appendixes B.1 Installation* in the *Application Server Application Development Guide*.

To perform actual operations by using the repository information imported into the development environment, export this repository information from the development environment to the operating environment, and then deploy the HCSC components from the operating environment.

B. Migrating from the Evaluation Version

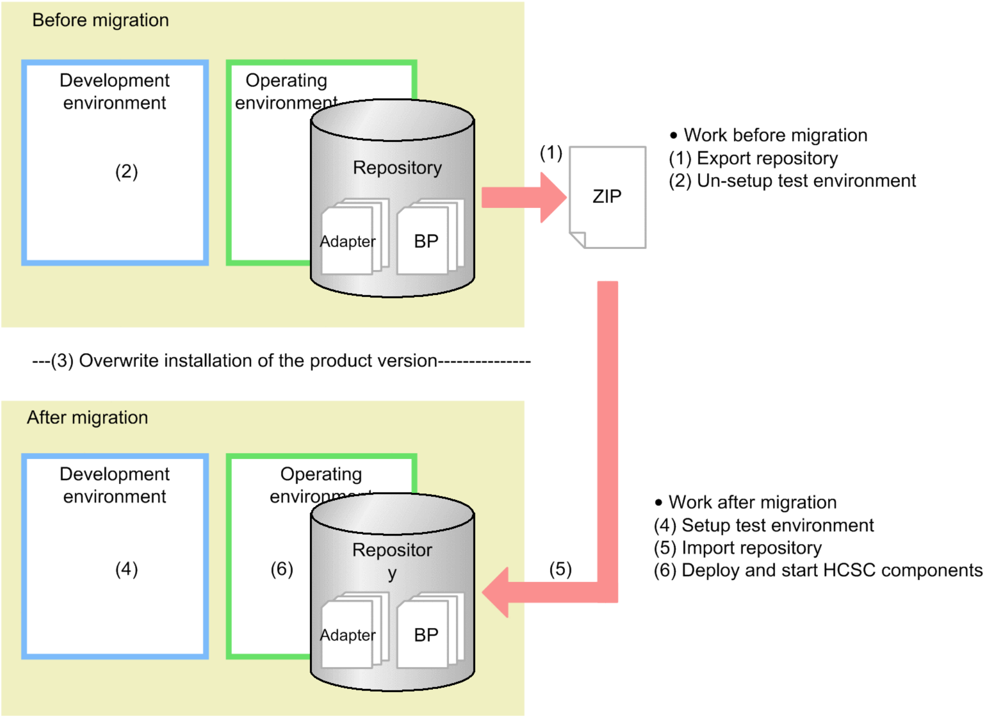
If you have been using the evaluation version of uCosminexus Service Architect, you can migrate the repository information of the operating environment of the evaluation version to the operating environment of the product version.

! Important note

The information about the process instance execution log and the message execution log cannot be migrated.

The following figure shows the procedure of migration from the evaluation version to the product version:

Figure B–1: Procedure of migration from the evaluation version



(1) Exporting the repository

Export the repository of the operating environment of the evaluation version, and save once. If multiple repositories are being used, export all the required repositories and save them.

1. Enter the following command:

```
cscrepctl -user admin -pass admin -export file-name-of-the-repository-to-be-exported-
(extension:.zip)
```

For details about the `cscrepctl` command, see the manual *Cosminexus Service Platform Reference*.

(2) Unsetup of the test environment

Unsetup the test environment set up in the development environment of the evaluation version.

For details about how to unsetup the test environment, see *2.4.2(2) Unsetting up the test environment*.

(3) Overwrite installation of the product version

Overwrite install the product version of uCosminexus Service Architect.

Note

- Stop the execution environment components such as the J2EE server, Management Server, and PRF and then perform overwrite installation.
- If an embedded database is being used, stop the embedded database and then perform overwrite installation.

(4) Setup of the test environment

Set up the test environment in the development environment.

For details about the procedure, see *2.4.2(1) Setting up the test environment*.

On the HCSC Easy Setup screen (**Main** tab), make sure that you select the same SOAP mode as the SOAP mode (SOAP1.1 mode or SOAP1.1/1.2 combined mode) specified when you set up the test environment of the evaluation version. If a different SOAP mode is selected, the procedure might not operate normally.

(5) Importing the repository

Import the exported repository information into the operating environment.

1. Enter the following command:

```
cscrepctl -user admin -pass admin -import file-name-of-the-repository-to-be-imported-
(extension:.zip)
```

To use the imported repository information for development, import this repository information into the development environment.

For details about how to import the repository into the development environment, see *3.2.3 Importing a Repository*.

(6) Deploying or starting an HCSC component

Start the HCSC server and then deploy or start the imported HCSC component.

1. Enter the following command to start the HCSC server:

```
cscsvstart -user admin -pass admin -system
```

2. Enter the following command to deploy the HCSC component on the HCSC server:

```
csccompodeploy -user admin -pass admin -csc MyCSC -all
```

3. Enter the following command to start the deployed HCSC component:

```
csccompostart -user admin -pass admin -csc MyCSC -all
```

4. Enter the following command to start standard reception:

```
cscrcptnstart -user admin -pass admin -csc MyCSC
```

C. System development using High Level Design Tools

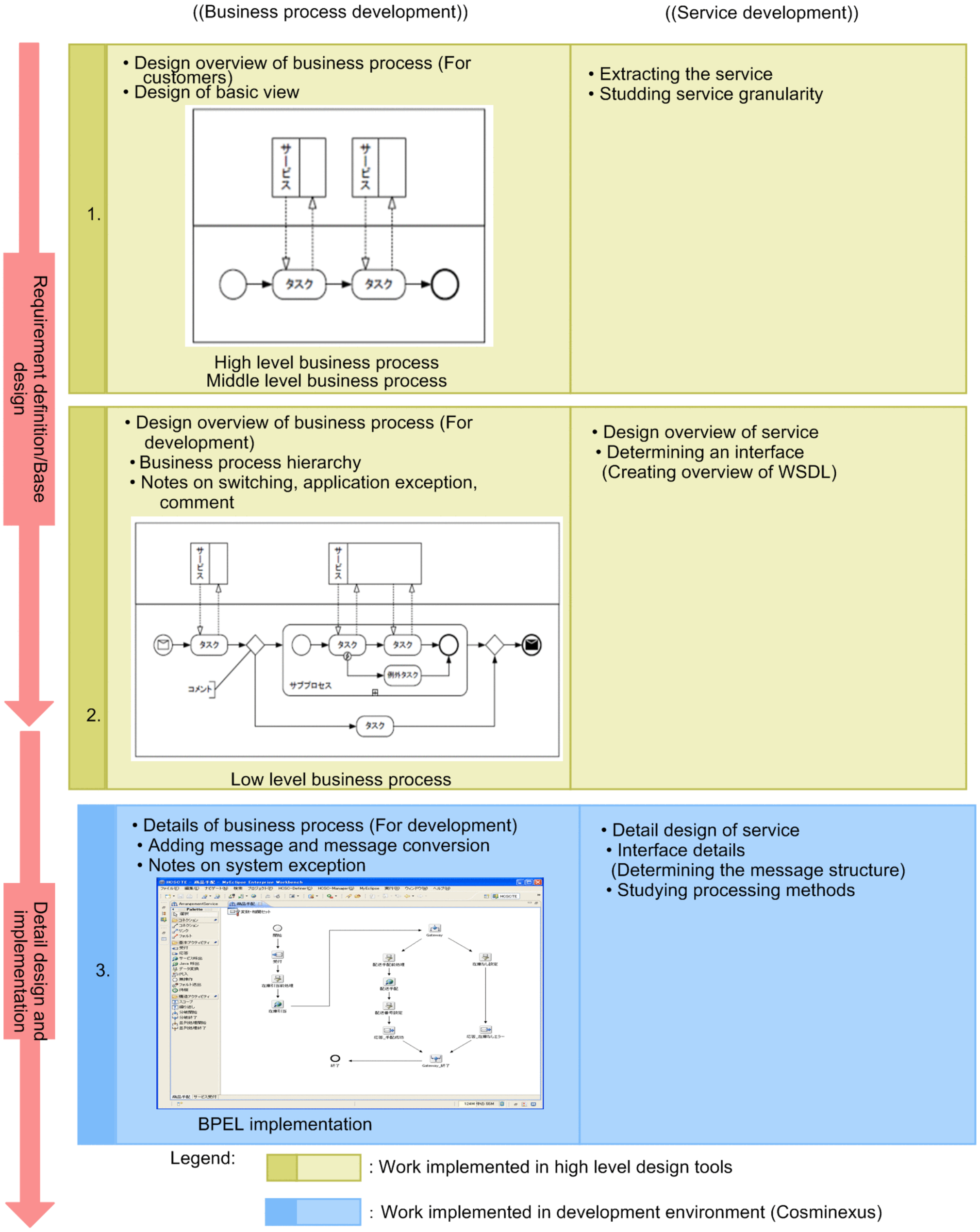
In the development environment of the Cosminexus Service Platform, you can develop business processes integrated with the high level design tools that support BPEL. Many of the high level design tools support the easy-to-understand flow notations such as BPMN; and therefore, possess the feature whereby the business process can be reviewed by involving the customer in the upper processes of system development.

This appendix describes how to develop the system by combining the high level design tools and the Cosminexus Service Platform.

C.1 Overview of system development using high level design tools

The following figure shows the procedure of system development combining the high level design tools and the Cosminexus Service Platform and the image of the deliverables:

Figure C‒1: System development combining the high level design tools and the Cosminexus Service Platform



((Business process development))

((Service development))

Requirement definition/Base design

Detail design and implementation

1.

- Design overview of business process (For customers)
- Design of basic view

High level business process
Middle level business process

- Extracting the service
- Studding service granularity

2.

- Design overview of business process (For development)
- Business process hierarchy
- Notes on switching, application exception, comment

Low level business process

- Design overview of service
- Determining an interface
  (Creating overview of WSDL)

3.

- Details of business process (For development)
- Adding message and message conversion
- Notes on system exception

BPEL implementation

- Detail design of service
- Interface details
  (Determining the message structure)
- Studying processing methods

Legend:

🟨 : Work implemented in high level design tools

🟦 : Work implemented in development environment (Cosminexus)

System development based on SOA involves business process development and service development and the operations proceed concurrently while drilling down the respective deliverables.

The following table describes the operating procedure and the details of the developments:

Table C‒1: Development procedure and operation details of business process development and service development

Step	Operations in business process development	Operations in service development
1	**Outline design of business process (for the customer)** In order to review the details and scope of the business to be systematized with the customer, the basic flow of business is created as a high-level business process and a middle level business process.	**Extraction of service** In parallel with the business process design, the granularity of the service that processes the business is studied.

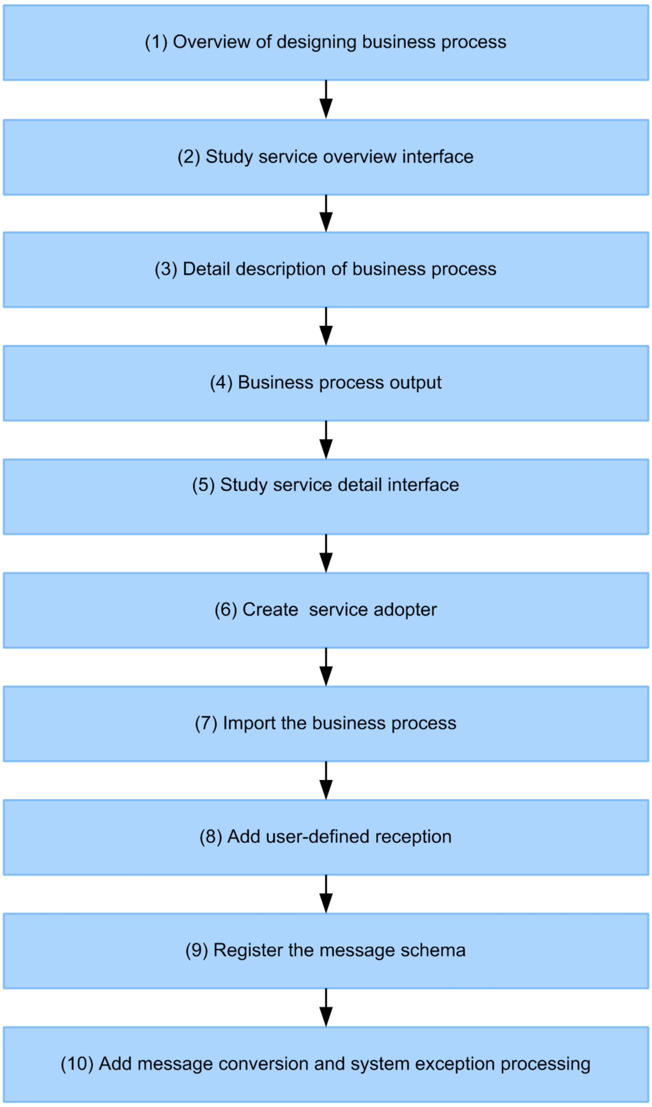
Step	Operations in business process development	Operations in service development
2	**Outline design of business process (for development)** The business process is detailed and changed into a low-level business process. In detailing, processes such as stratification of flow considering the reusability and readability, supporting the business exceptions, and adding comments are performed.	**Outline design of service** The service interface (list of operations that the service releases) is determined and the WSDL overview is created.
3	**Detailing of the business process (for development)** The results of service extraction and outline design are received and then message definition or message transformation is added to the business process. Also, support is added for the system exceptions assumed to occur actually.	**Detailed design of service** The message structure used in the service operations and the service processing method is determined and implemented.

When you develop a system combining the high level design tools and the Cosminexus Service Platform, you use the high level design tools in step 1 and step 2 and the development environment of the Cosminexus Service Platform from step 3. The design is inherited from the high level design tools to the development environment through the BPEL file.

C.2 Procedure of system development using high level design tools

The following figure shows the procedure of system development using the high level design tools. In *Appendix D. Examples of System Development Using High Level Design Tools*, you develop the sample programs according to this procedure.

Figure C–2: Procedure of system development using high level design tools


(1) Overview of designing business process
(2) Study service overview interface
(3) Detail description of business process
(4) Business process output
(5) Study service detail interface
(6) Create service adopter
(7) Import the business process
(8) Add user-defined reception
(9) Register the message schema
(10) Add message conversion and system exception processing

(1) Overview of designing business process

You create a high-level business process using the high level design tools. Reviews are repeated with the customer for the high-level business process that is then changed into a more concrete middle level business process.

(2) Overview of service interfaces

When the review of the middle level business process ends, review the outline interface of the service. Also, based on the reviewed results, create the WSDL overview.

(3) Detail description of business process

A business process is detailed and changed into a low-level business process. To design an optimum business process for the Cosminexus Service Platform, design the flow according to *Appendix C.3 Prerequisites for using high level design tools*.

(4) Business process output

A business process designed using an high level design tool is output in the BPEL file format. In this case, the WSDL of the service integrated with the business process is also compiled.

(5) Detail description of service interface

Based on the specifications reviewed until now, review the message structure for the service operations. Also, on the basis of the reviewed results, add the message structure declaration to the WSDL overview.

(6) Creating the service adapter

In the development environment of the Cosminexus Service Platform, you create the service adapter from WSDL. The service name must be the same as the port type (`portType`) attribute described in WSDL. For details about how to create a service adapter, see *5.2.1 Adding New Service Adapters*.

(7) Importing the business process

In the development environment of the Cosminexus Service Platform, you import the BPEL file that was output in *Appendix C.2(4) Business process output*, as a business process. For details about how to import the BPEL file, see *5.2.2 Using an Already Defined Business Process to Add Business Processes*.

(8) Adding user-defined reception

Add the user-defined reception to the business process. For details about how to add the user-defined reception, see *8.4.1 Adding a New User-Defined Reception*.

(9) Registering the message schema

Register the schema (structure) of the messages (variable) used in the business process.

(10) Adding message conversion and system exception processing

As and when required, add message transformation processing and processing to support the system exceptions to the business process.

C.3 Prerequisites for using high level design tools

This section describes the prerequisites (points to remember when using high level design tools) for designing a business process for the Cosminexus Service Platform.

(1) Prerequisites related to the overall business process

(a) Settings for instance correction

The Cosminexus Service Platform does not perform instance correction. When instance correction can be set up for a business process using the high level design tools, specify "No correction". When the settings cannot be specified, design the flow assuming that instance correction is not performed.

(b) Specifying the abstract process

The Cosminexus Service Platform can only handle the business processes with an executable implementation level. When a business process can be set up as an abstract process using the high level design tools, do not specify the process as an "Abstract Process".

(c) Controlling the joinFailure fault

The Cosminexus Service Platform always controls and operates the `joinFailure` fault. When the controlling of `joinFailure` fault can be set up using the high level design tools, specify "Control". When the settings cannot be specified, design the flow assuming that the `joinFailure` fault is controlled.

(2) Prerequisites related to activities

(a) Using unsupported elements and attributes

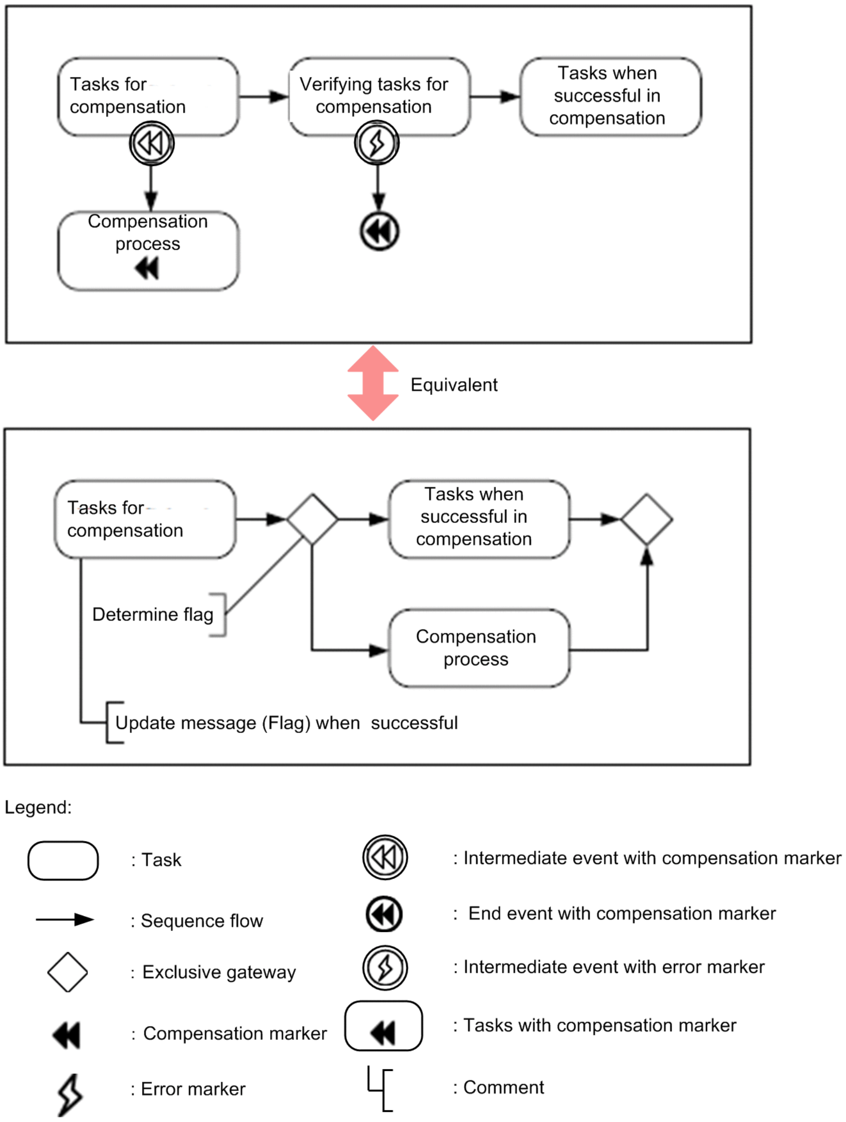
Do not use BPEL elements and attributes that are not supported by the BPEL import functionality. Unsupported elements and attributes skip the fetching operation or are replaced by other elements; therefore, the process might change into an unintended flow. For details about the supported status of BPEL import functionality, see *Appendix E. Support Range of BPEL Used by Linking with an High Level Design Tool*.

For details about the elements (BPMN elements) and settings for the high level design tools and the mapping to BPEL elements and attributes, see the documentation for the BPMN specifications or high level design tools.

(b) Using the compensation handler

The BPEL import functionality does not support the `compensationHandler` element. If a compensation marker is used in the high level design tools that use BPMN, the compensation marker is output in BPEL as the compensationHandler element; therefore, it is recommended that you do not use the compensation marker in the design. The following figure shows an example of the same design without using the compensation marker:

Figure C–3: Example of a design without using the compensation marker


Tasks for compensation
Verifying tasks for compensation
Tasks when successful in compensation
Compensation process
Equivalent
Tasks for compensation
Tasks when successful in compensation
Determine flag
Compensation process
Update message (Flag) when successful
Legend:
: Task
: Sequence flow
: Exclusive gateway
: Compensation marker
: Error marker
: Intermediate event with compensation marker
: End event with compensation marker
: Intermediate event with error marker
: Tasks with compensation marker
: Comment

(c) Character types of the service name

The Cosminexus Service Platform can only use NCName as the name of the service integrated with a business process. When specifying the service name (interface attribute in BPMN) using the high level design tool, specify the value with NCName. If characters other than NCName are used, the fetching of the information is partially skipped.

(d) Partially specifying a message

Do not partially specify messages with the high level design tools. Specify a message partially after fetching the business process definition into the development environment, since the message structure must be determined. If partially specified (part) attributes are specified in the BPEL assign activity, the specification is ignored by the BPEL import functionality.

(3) Prerequisites related to messages

(a) Specifying the message type

Do not specify the message structure in the high level design tools. Specify the message structure after fetching the business process into the development environment, since the service interface must be determined. If the message type (messageType) attribute is specified in BPEL, the attribute is fetched after being replaced by the string type variable.

(b) Specifying the type

If the message is of a basic type (without a structure), specify a type belonging to the XML schema namespace (http://www.w3.org/2001/XMLSchema). If another type is specified in the BPEL type (type) attribute, the type is fetched after being replaced by a type that is presumed to be compatible.

C.4 Troubleshooting when the high level design tools are used

This section describes the message details and the solutions for messages displayed when a BPEL file, created by the high level design tools, is imported.

The message format described in this section is as follows:

Message text

Description of the message

Output conditions

Message output conditions

Action

Action and supplementary notes for the message

Message text

Indicates the message text output on the Cosminexus Service Platform.

The part enclosed within [ ] (square brackets) is displayed by XPath and the part enclosed within { } (curly brackets) is displayed by the value.

Message description

Indicates the supplementary description for the message text.

Output conditions

Indicates the conditions in which the message is output.

Action

Indicates the action to be taken when the message is output or the supplementary notes related to the action.

The following points describe the messages output for each defined BPEL element:

(1) Process

The attribute targetNamespace of the element [process] is not supported and, hence, not applied.

On the Cosminexus Service Platform, the fixed value reserved for the system is always used as the targetNamespace attribute value. Even if the targetNamespace attribute value is specified, when the BPEL file is imported, the value is replaced by the fixed value reserved for the system.

Output conditions

When the targetNamespace attribute is defined

Action

Does not affect the execution of the business process. Specific action need not be taken.

The element [process] attribute enableInstanceCompensation is not supported and, hence, not applied.

On the Cosminexus Service Platform, the enableInstanceCompensation attribute (instance correction) is not supported. Therefore, the operation is always the same as when "no" is specified in the enableInstanceCompensation attribute.

Note that in the case of BPEL 1.1, the default value of the enableInstanceCompensation attribute is "no". In the case of BPEL 2.0, the enableInstanceCompensation attribute cannot be specified.

Output conditions

When the enableInstanceCompensation attribute is defined

Action

When this message is output, check the specification of the enableInstanceCompensation attribute in the BPEL file and take action from the following viewpoints:

- When "no" is specified in the enableInstanceCompensation attribute

  The operation is same as that on the Cosminexus Service Platform. Specific action need not be taken.

- When "yes" is specified in the enableInstanceCompensation attribute

  When the BPEL file is imported, the specification of the enableInstanceCompensation attribute is replaced by "no". Therefore, if the flow is designed assuming instance correction, the execution results of the business process might differ. Open the imported business process in a business process editor and review whether the business process is defined for the intended details.

The details of the instance correction depend on the header. If the business process is created assuming instance correction, sometimes the runtime compatibility cannot be guaranteed.

Therefore, either do not specify the enableInstanceCompensation attribute from the upper-level design stage, or design the process by specifying "no".

The element [process] attribute abstractProcess is not supported and, hence, not applied.

On the Cosminexus Service Platform, the abstractProcess attribute is not supported. The Cosminexus Service Platform always targets the business processes with executable implementation level. Therefore, the operation is always the same as when "no" is specified for the abstractProcess attribute.

Note that in the case of BPEL 1.1, the default value of the abstractProcess attribute is "no". In the case of BPEL 2.0, the abstractProcess attribute cannot be specified.

Output conditions

When the abstractProcess attribute is defined

Action

When this message is output, check the specification of the abstractProcess attribute in the BPEL file and take action from the following viewpoint:

- When "no" is specified for the abstractProcess attribute

  The operation is the same as on the Cosminexus Service Platform. Specific action need not be taken.

- When "yes" is specified for the abstractProcess attribute

The business process might be of a non-executable abstract level. Open the imported business process using a business process editor and review the business process definition.

(2) Partner link

The element [`partnerLinks`] is not supported and, hence, not applied. Also, if lower attributes and elements exist, these attributes and elements are also not applied.

On the Cosminexus Service Platform, the `partnerLinks` attribute is not supported. When the BPEL file is imported, the `partnerLinks` element indicating the relationship between the business process and the external service is not fetched.

Output conditions

When the `partnerLinks` element is defined

Action

Does not affect the execution of the business process. Specific action need not be taken.

On the Cosminexus Service Platform, the partner link information of the BPEL file corresponds to the user-defined reception and service adapter.

As and when required, create the user-defined reception and service adapter from the WSDL file that makes a set with the BPEL file.

See the actions for the following messages as well:

- Reference: *Appendix C.4(5) Activity (receive)*
  The attribute `partnerLink` of the element [`receive`] is not supported and, hence, not applied.
- Reference: *Appendix C.4(6) Activity (reply)*
  The attribute `partnerLink` of the element [`reply`] is not supported and, hence, not applied.
- Reference: *Appendix C.4(8) Activity (invoke)*
  The attribute `partnerLink` of the element [`invoke`] is not supported and, hence, not applied.

(3) Variable or correlation set (variable)

The variable {*variable-name*} was replaced by the string type. Change the variable type to `messageType` and register the message format.

When the BPEL file is imported, a structured message is not fetched as a variable.

In the high level design tools, specify only the name of the message (variable) and define the message structure when you enter the detailed design phase.

Output conditions

When the `messageType` attribute of the variable element is defined

Action

After importing the BPEL file, change the variable type into `messageType` using the business process editor and then specify the schema file.

The variable {*variable-name*} type {*old-variable-type*} must belong to the namespace "`http://www.w3.org/2001/XMLSchema`". The type is applied as the {*new-variable-type*} type, but check whether there is a problem.

The types that are not defined in the XML schema namespace are specified in the variable. The variable is fetched after the type is replaced by a type that is presumed to be compatible.

Output conditions

When a type that is not defined in the XML schema namespace is specified in the variable

Action

Check whether the replaced variable type affects the business process. As and when required, import the BPEL file and then review the variable types using the business process editor.

(4) Variable or correlation set (correlation set)

The element [`correlationSets`] is not supported and, hence, not applied. Also, if lower attributes and elements exist, these attributes and elements are also not applied.

When the BPEL file imported, the `correlationSets` element defining the correlation set is not fetched.

Do not define the correlation set in the high level design tools.

Output conditions

When the `correlationSets` element is defined

Action

After importing the BPEL file, redefine the correlation set using the business process editor.

(5) Activity (receive)

The attribute `partnerLink` of the element [`receive`] is not supported and, hence, not applied.

When the BPEL file is import, the `partnerLink` attribute is not fetched.

Output conditions

When the `partnerLink` attribute is defined

Action

Need not be a cause for concern within the usage of standard reception.

The `partnerLink` attribute of the `receive` element is equivalent to the user-defined reception on the Cosminexus Service Platform.

As and when required, import the BPEL file and then create the user-defined reception from the WSDL file of the business process.

The attribute `portType` of the element [`receive`] is not supported and, hence, not applied.

When the BPEL file is imported, the `portType` attribute is not fetched.

Output conditions

When the `portType` attribute is defined

Action

The `portType` attribute is specified for the readability of the BPEL file and WSDL file. Does not affect the execution of the business process. Specific action need not be taken.

The operation name of the activity [`receive`] is repeated and, therefore, replaced by {*operation-name*}.

The same operation name is specified for multiple `receive` elements having different `portType` attribute values. The operation name is replaced such that the name is not repeated when the BPEL file is imported and then fetched.

Output conditions

When the same operation name is specified for multiple `receive` elements having different `portType` attribute values

Action

Does not affect the execution of the business process. Specific action need not be taken. However, the operation name might not match the business process WSDL; therefore, as and when required, change the operation name using the business process editor.

The `receive` activity [receive] was replaced with an empty activity since an activity with the same `portType` and `operation` already exists.

The same operation name was specified for multiple `receive` elements having the same `portType` attribute values. When the BPEL file is imported, the second and subsequent `receive` elements are replaced by empty activities and then fetched.

Output conditions

When the same operation name was specified for multiple `receive` elements having the same `portType` attribute values

Action

Import the BPEL file and then redefine the empty activity as an appropriate receive activity.

The value "no" specified in the attribute [`createInstance`] of the element [`receive`] is replaced with "yes".

When the BPEL file is imported, the `createInstance` attribute of the `receive` element is always fetched as "yes".

Output conditions

- When the `createInstance` attribute is not specified
- When "no" is specified for the `createInstance` attribute

Action

Import the BPEL file and then re-specify the instance generation for the receive activity.

The element [`correlations`] is not supported and, hence, not applied. Also, if lower attributes and elements exist, these attributes and elements are also not applied.

When the BPEL file is imported, the `correlations` element specifying the correlation set is not fetched. Do not specify the correlation set in the high level design tools.

Output conditions

When the `correlations` element is defined

Action

Import the BPEL file and then specify the correlation set using the business process editor.

(6) Activity (reply)

The attribute `partnerLink` of the element [`reply`] is not supported and, hence, not applied.

When the BPEL file is imported, the `partnerLink` attribute is not fetched.

Output conditions

When the `partnerLink` attribute is defined

Action

On the Cosminexus Service Platform, the `partnerLink` attribute of the `reply` element is equivalent to the user-defined reception.

If necessary, import the BPEL file and then create the user-defined reception from the WSDL file of the business process.

The attribute `portType` of the element [`reply`] is not supported and, hence, not applied.

When the BPEL file is imported, the `portType` attribute is not fetched.

Output conditions

When the `portType` attribute is defined

Action

The `portType` attribute is specified for the readability of the BPEL file and WSDL file. Does not affect the execution of the business process. Specific action need not be taken.

The value {*namespace*}:{*local-name*} specified in the attribute `faultName` of the element [`reply`] was replaced by {*local-name*}.

On the Cosminexus Service Platform, the `fault` namespace is not supported by default.

If the fault namespace is specified in the BPEL file, only the local name is fetched.

Output conditions

When the namespace is specified in the faultName attribute value

Action

When the standard reception is used, the fault message only stores the local name.

When the user-defined reception is used, the fault message stores both, the namespace and the local name.

As and when required, create the user-defined reception from the WSDL file of the business process such that the fault message conforms to the business process interface.

The operation name of the activity [reply] is repeated and, therefore, replaced by {*operation-name*}.

The same operation name is specified for multiple reply elements having different portType attribute values. The operation name is replaced such that the name is not repeated when the BPEL file is imported and then fetched.

Output conditions

When the same operation name is specified for multiple reply elements having different portType attribute values

Action

Does not affect the execution of the business process. Specific action need not be taken. However, the operation name might not match the business process WSDL; therefore, as and when required, change the operation name using the business process editor.

The element [correlations] is not supported and, hence, not applied. Also, if lower attributes and elements exist, these attributes and elements are also not applied.

When the BPEL file is imported, the correlations element specifying the correlation set is not fetched. Do not specify the correlation set in the high level design tools.

Output conditions

When the correlations element is defined

Action

Import the BPEL file and then specify the correlation set using the business process editor.

(7) Activity (assign)

The attribute part of the element [assign/copy/from] is not supported and, hence, not applied.

When the BPEL file is imported, the part attribute is not fetched.

In the high level design tools, specify only the name of the message (variable) and define the message structure when you enter the detailed design phase.

Output conditions

When the part attribute is defined

Action

Import the BPEL file and then define the message structure and specify the parts using the business process editor.

The attribute part of the element [assign/copy/to] is not supported and, hence, not applied.

When the BPEL file is imported, the part attribute is not fetched.

In the high level design tools, specify only the name of the message (variable) and define the message structure when you enter the detailed design phase.

Output conditions

When the part attribute is defined

Action

Import the BPEL file and then define the message structure and specify the parts using the business process editor.

The `copy` element of the assign activity [`assign`] is not applied because the format of the attributes defined in `copy/from` is not supported.

To fetch the `copy` element of the `assign` activity, the attributes of the `copy/from` element must be defined in a fetchable combination.

For details about the fetchable combinations, see *Appendix E. Support Range of BPEL Used by Linking with an High Level Design Tool*.

Output conditions

When the attributes of the `copy/from` element are defined using an unfetchable combination

Action

Import the BPEL file and then redefine the assign settings for the `assign` activity.

The `to` element of the assign activity [`assign`] is not applied because the format of the attributes defined in `copy/to` is not supported.

To fetch the `to` element of the `assign` activity, the attributes of the `copy/to` element must be defined in a fetchable combination.

For details about the fetchable combinations, see *Appendix E. Support Range of BPEL Used by Linking with an High Level Design Tool*.

Output conditions

When the attributes of the `copy/to` element are defined using an unfetchable combination

Action

Import the BPEL file and then redefine the assign settings for the `assign` activity.

(8) Activity (invoke)

The attribute `partnerLink` of the element [`invoke`] is not supported and, hence, not applied.

When the BPEL file is imported, the `partnerLink` attribute is not fetched.

Output conditions

When the `partnerLink` attribute is defined

Action

On the Cosminexus Service Platform, the `partnerLink` attribute of the `invoke` element is equivalent to the service adapter.

If a service adapter corresponding to the `partnerLink` attribute is already created, specific action need not be taken.

The applicable service or operation does not exist in the repository; therefore, the service name {*service-name*} specified in `portType` of the invoke activity [`invoke`] is not applied.

If the service adapter corresponding to the `portType` attribute and `operation` attribute does not exist in the repository, the `portType` attribute is not fetched.

Output conditions

- When the service adapter corresponding to the `portType` attribute does not exist in the repository
- When the service adapter corresponding to the `portType` attribute does not have an operation corresponding to the `operation` attribute

Action

Create the corresponding service adapter and re-set the service name of the `invoke` activity.

The applicable service or operation does not exist in the repository; therefore, the operation name {*operation-name*} specified in `operation` of the invoke activity [`invoke`] is not applied.

If the service adapter corresponding to the portType attribute and operation attribute does not exist in the repository, the operation attribute is not fetched.

Output conditions

- When the service adapter corresponding to the portType attribute does not exist in the repository
- When the service adapter corresponding to the portType attribute does not have an operation corresponding to the operation attribute

Action

Create the corresponding service adapter and re-set the operation name of the invoke activity.

The value {*variable-name*} of the attribute inputVariable specified in the invoke activity [invoke] is not applied because the corresponding operation {*operation-name*} does not exist in the repository.

If the service adapter corresponding to the portType attribute and operation attribute does not exist in the repository, the inputVariable attribute is not fetched.

Output conditions

- When the service adapter corresponding to the portType attribute does not exist in the repository
- When the service adapter corresponding to the portType attribute does not have an operation corresponding to the operation attribute

Action

Create the corresponding service adapter and reset the allocation variable for the request message of the invoke activity.

The value {*variable-name*} of the attribute outputVariable specified in the invoke activity [invoke] is not applied because the corresponding operation {*operation-name*} does not exist in the repository.

If the service adapter corresponding to the portType attribute and operation attribute does not exist in the repository, the outputVariable attribute is not fetched.

Output conditions

- When the service adapter corresponding to the portType attribute does not exist in the repository
- When the service adapter corresponding to the portType attribute does not have an operation corresponding to the operation attribute

Action

Create the corresponding service adapter and reset the allocation variable for the response message of the invoke activity.

The value {*variable-name*} of the attribute outputVariable specified in the invoke activity [invoke] is not applied because the operation {*operation-name*} of the service {*service-name*} in the repository is asynchronous.

If the communication method of the operation for the invocation destination service is asynchronous, the response message is not required; therefore, the outputVariable attribute is not fetched.

Output conditions

When the communication method of the operation for the invocation destination service is asynchronous and the outputVariable attribute is defined

Action

The BPEL file and WSDL file used as input might not be consistent. Check whether the asynchronous communication method of operation is acceptable.

The attribute faultName of the element [invoke/catch] is not supported and, hence, not applied.

The Cosminexus Service Platform does not support fault catching using the faultName attribute.

Output conditions

When the faultName attribute is defined

Action

Import the BPEL file and then re-set fault catching using a business process editor.

The attribute faultMessageType of the element [invoke/catch] is not supported and, hence, not applied.

The Cosminexus Service Platform does not support fault catching using the faultMessageType attribute.

Output conditions

When the faultMessageType attribute is defined

Action

Import the BPEL file and then re-set fault catching using a business process editor.

The attribute faultElement of the element [invoke/catch] is not supported and, hence, not applied.

The Cosminexus Service Platform does not support fault catching using the faultElement attribute.

Output conditions

When the faultElement attribute is defined

Action

Import the BPEL file and then re-set fault catching using a business process editor.

The element [correlations] is not supported and, hence, not applied. Also, if lower attributes and elements exist, these attributes and elements are also not applied.

When the BPEL file is imported, the correlations element specifying the correlation set is not fetched.

Do not specify the correlation set in the high level design tools.

Output conditions

When the correlations element is defined

Action

Import the BPEL file and then specify the correlation set using the business process editor.

(9) Activity (flow)

The element [flow/links] is not supported and, hence, not applied. Also, if lower attributes and elements exist, these attributes and elements are also not applied.

When the BPEL file is imported, the links element specifying the link is not fetched.

Do not specify the link in the high level design tools.

Output conditions

When the links element is defined

Action

Import the BPEL file and then specify the link using the business process editor.

(10) Activity (throw)

The attribute faultName of the element [throw] is not supported and, hence, not applied.

The Cosminexus Service Platform does not support fault throwing using the faultName attribute.

Output conditions

When the faultName attribute is defined

Action

Import the BPEL file and then set up fault throwing using the business process editor.

(11) Activity (general)

The name of the activity [someElement] is repeated and, therefore, replaced by {*new-name*}.

On the Cosminexus Service Platform, the activity name (name attribute) cannot be repeated.

If the activity name is repeated when the BPEL file is imported, a number is added to the end of the activity name and then the activity is fetched. The number is sequentially specified from 1.

Output conditions

When the activity name (name attribute) is repeated

Action

The activity name is used for the output of the execution log for the operating environment and the execution environment log. As and when required, change the activity name.

The element [someElement] is not supported and, hence, replaced by an empty activity. Also, if lower attributes and elements exist, these attributes and elements are also not applied.

The Cosminexus Service Platform does not support the activities described in [someElement].

When the BPEL file is imported, [someElement] is replaced by an empty activity and then fetched.

Output conditions

When an unsupported activity is defined

Action

Replace the processing by an activity supported on the Cosminexus Service Platform.

(12) Handler (compensation handler)

The element [compensationHandler] is not supported and, hence, not applied. Also, if lower attributes and elements exist, these attributes and elements are also not applied.

The Cosminexus Service Platform does not support the compensationHandler element.

Output conditions

When the compensationHandler element is defined

Action

Replace the processing by an activity supported on the Cosminexus Service Platform.

(13) Handler (event handler)

The element [eventHandler] is not supported and, hence, not applied. Also, if lower attributes and elements exist, these attributes and elements are also not applied.

The Cosminexus Service Platform does not support the eventHandler element.

Output conditions

When the eventHandler element is defined

Action

Replace the processing by an activity supported on the Cosminexus Service Platform.

(14) Handler (termination handler)

The element [terminationHandler] is not supported and, hence, not applied. Also, if lower attributes and elements exist, these attributes and elements are also not applied.

The Cosminexus Service Platform does not support the terminationHandler element.

Output conditions

When the terminationHandler element is defined

Action

Replace the processing by an activity supported on the Cosminexus Service Platform.

(15) Handler (fault handler)

FaultHandlers [faultHandlers] are not applied because catch/catchAll is not defined.

The fault handlers in which one or more catch or catchAll is not defined are not fetched.

Output conditions

When one or more catch element or catchAll element are not defined in the faultHandler element

Action

Does not affect the execution of the business process. Specific action need not be taken.

The attribute faultName of the element [faultHandlers/catch] is not supported and, hence, not applied.

The Cosminexus Service Platform does not support fault catching using the faultName attribute.

Output conditions

When the faultName attribute is defined

Action

Import the BPEL file and then re-set fault catching using the business process editor.

The attribute faultMessageType of the element [faultHandlers/catch] is not supported and, hence, not applied.

The Cosminexus Service Platform does not support fault catching using the faultMessageType attribute.

Output conditions

When the faultMessageType attribute is defined

Action

Import the BPEL file and then re-set fault catching using the business process editor.

The attribute faultElement of the element [faultHandlers/catch] is not supported and, hence, not applied.

The Cosminexus Service Platform does not support fault catching using the faultElement attribute.

Output conditions

When the faultElement attribute is defined

Action

Import the BPEL file and then re-set fault catching using the business process editor.

(16) Others

The value no specified in the attribute suppressJoinFailure of the element [someElement] is replaced by yes.

The Cosminexus Service Platform does not support the suppressJoinFailure attribute. Therefore, the bpws:joinFailure fault is always controlled and the operation is the same as when "yes" is specified for the suppressJoinFailure attribute.

Note that in the case of BPEL 1.1 and BPEL 2.0, the default value of the suppressJoinFailure attribute is "no".

Output conditions

- When the suppressJoinFailure attribute is not specified
- When "no" is specified for the suppressJoinFailure attribute

Action

When the BPEL file is imported, the specification of the `suppressJoinFailure` attribute is replaced by "yes". Therefore, the execution results of the business process might differ. Open the BPEL file in the business process editor and review whether the business process is defined for the intended details.

Note

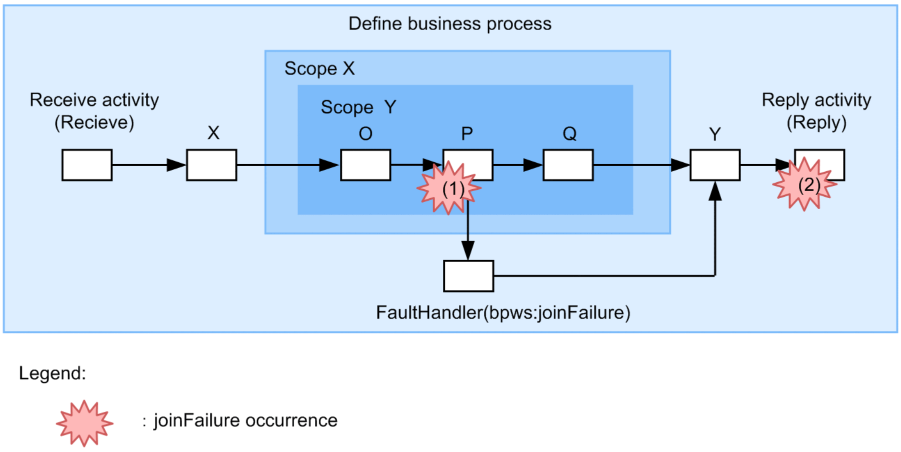
The operations of the business process differ depending on the value of the `suppressJoinFailure` attribute.

On the Cosminexus Service Platform, operations are always executed with the `suppressJoinFailure` attribute as "yes"; therefore, if `joinCondition` of the activity is not fulfilled, the business process goes into an interrupted state that cannot be restarted immediately.

In an environment where the `suppressJoinFailure` attribute is operated as "no", if `joinCondition` is not fulfilled, the `bpws:joinFailure` fault is thrown, but the processing continues according to the defined business process.

The following figure shows the relationship between the specification of the `suppressJoinFailure` attribute and `bpws:joinFailure`:

Figure C–4: Relationship between the specification of the suppressJoinFailure attribute and bpws:joinFailure



When the `suppressJoinFailure` attribute is "no":

When `joinFailure` occurs at (1), `joinFailure` is propagated to the upper scope. In this example, `joinFailure` is propagated in the order of scope Y scope X process, but error processing of `joinFailure` is executed in `FaultHandler` of scope X and the processing continues.

When `joinFailure` occurs at (2), an attempt is made to propagate to the upper level of the process, but there is nothing to catch `joinFailure`; therefore, the business process stops. In this case, the business process goes into a restartable interrupted state.

When the `suppressJoinFailure` attribute is "yes":

When `joinFailure` occurs at (1), the propagation of `joinFailure` is controlled and the business process goes into a non-restartable interrupted state.

When `joinFailure` occurs at (2) as well, the propagation of `joinFailure` is controlled and the business process goes into a non-restartable interrupted state.

The attribute `queryLanguage` of the element `[someElement]` is not supported and, hence, not applied.

The Cosminexus Service Platform does not support the `queryLanguage` attribute. XPath1.0 is always used as the query language. XPath1.0 for the Cosminexus Service Platform has a partially unique extension.

Note that in the case of BPEL 1.1 and BPEL 2.0, the default query value is XPath1.0.

Output conditions

When the `queryLanguage` attribute is defined

Action

If a query is not used in the high level design tools, specific action need not be taken.

If a query language other than XPath1.0 is used, import the BPEL file and then correct the query specification using the business process editor.

The attribute expressionLanguage of the element [someElement] is not supported and, hence, not applied.

The Cosminexus Service Platform does not support the expressionLanguage attribute. XPath1.0 is always used as the expression language. XPath1.0 for the Cosminexus Service Platform has a partially unique extension.

Note that in the case of BPEL 1.1 and BPEL 2.0, the default expression value is XPath1.0.

Output conditions

When the expressionLanguage attribute is defined

Action

If an expression is not used in the high level design tools, specific action need not be taken.

If a expression other than XPath1.0 is used, import the BPEL file and then correct the specification of the expression using the business process editor.

An empty activity was added because no activity was defined in the element [someElement].

If one or more child elements are not defined for a BPEL element (such as while activity) that can have a child element, an empty activity is added as a child element when the BPEL file is imported.

Output conditions

When the activity element of [someElement] does not have a child element

Action

Does not affect the execution of the business process. Specific action need not be taken.

The element [someElement] is not supported and, hence, not applied. Also, if lower attributes and elements exist, these attributes and elements are also not applied.

The Cosminexus Service Platform does not support the BPEL elements described by [someElement].

Output conditions

When unsupported elements are defined

Action

Replace the processing by functionality supported on the Cosminexus Service Platform.

The attribute [someAttribute] of the element [someElement] is not supported and, hence, not applied.

The Cosminexus Service Platform does not support the BPEL attributes described by [someAttribute].

Output conditions

When unsupported attributes are defined

Action

Replace the processing by functionality supported on the Cosminexus Service Platform.

D. Examples of System Development Using High Level Design Tools

INTENTIONALLY DELETED

D.1 Designing the business process overview

INTENTIONALLY DELETED

D.2 Reviewing the service overview interface

INTENTIONALLY DELETED

D.3 Detailing the business process

INTENTIONALLY DELETED

D.4 Output the business process

INTENTIONALLY DELETED

D.5 Reviewing the detailed interface of the service

INTENTIONALLY DELETED

D.6 Creating the service adapter

INTENTIONALLY DELETED

D.7 Importing the business process

INTENTIONALLY DELETED

D.8 Adding user-defined reception interfaces

INTENTIONALLY DELETED

D.9 Registering message schemas

INTENTIONALLY DELETED

D.10 Adding the message transformation and system exception processing

INTENTIONALLY DELETED

E. Support Range of BPEL Used by Linking with an High Level Design Tool

The Cosminexus Service Platform supports the import of BPEL1.1 and BPEL2.0. The support range of a BPEL file and rules for converting to a business process definition are different for BPEL1.1 and BPEL2.0.

This section explains (for BPEL1.1 and BPEL2.0) the support range of a BPEL file and the rules for converting to a business process definition, when you import the BPEL file created with an high level design tool.

! Important note

Among the elements and attributes of the BPEL file, the elements and attributes that are not described hereafter are not converted to the business process definitions.

The following table describes which elements of the BPEL file are converted to which definition contents of a business process:

Table E–1: Relation between the elements of a BPEL file and business process definitions

Element before conversion (BPEL file)		Definition contents after conversion (business process)	Reference for conversion method	
			BPEL1.1	BPEL2.0
`process` element	`variables` element	Variable	*Appendix E.1(1)(b)*	*Appendix E.2(1)(b)*
	`correlationSets` element	Correlation set	*Appendix E.1(1)(c)*	*Appendix E.2(1)(c)*
	`faultHandlers` element	Fault handling	*Appendix E.1(1)(d)*	*Appendix E.2(1)(d)*
	`receive` element	Receive activity	*Appendix E.1(2)(a)*	*Appendix E.2(2)(a)*
	`reply` element	Reply activity	*Appendix E.1(2)(b)*	*Appendix E.2(2)(b)*
	`invoke` element	Invoke Service Activity	*Appendix E.1(2)(c)*	*Appendix E.2(2)(c)*
	`assign` element	Assign activity	*Appendix E.1(2)(d)*	*Appendix E.2(2)(d)*
	`empty` element	Empty activity	*Appendix E.1(2)(e)*	*Appendix E.2(2)(e)*
	`throw` element	Throw activity	*Appendix E.1(2)(f)*	*Appendix E.2(2)(f)*
	`scope` element	Scope activity	*Appendix E.1(3)(a)*	*Appendix E.2(3)(a)*
	`while` element	While activity	*Appendix E.1(3)(b)*	*Appendix E.2(3)(b)*
	`switch` element[#1]	Start switch activity End switch activity	*Appendix E.1(3)(c)*	--
	`if` element[#2]		--	*Appendix E.2(3)(c)*
	`flow` element	Start flow activity End flow activity	*Appendix E.1(3)(d)*	*Appendix E.2(3)(d)*
	`sequence` element	Sequence activity	*Appendix E.1(3)(e)*	*Appendix E.2(3)(e)*
	`wait` element	Standby activity	*Appendix E.1(2)(g)*	*Appendix E.2(2)(g)*
	`compensate` element[#3]	Empty activity	--	--
	`terminate` element[#1#3]			
	`pick` element[#3]			

Element before conversion (BPEL file)		Definition contents after conversion (business process)	Reference for conversion method	
			BPEL1.1	BPEL2.0
process element	extensionActivity element[#2#3]	Empty activity	--	--
	rethrow element[#2#3]			
	exit element[#2#3]			
	validate element[#2#3]			
	compensateScope element[#2#3]			
	forEach element[#2#3]			
	repeatUntil element[#2#3]			

Legend:

--: There is no reference.

#1

These elements can be defined only for BPEL1.1.

#2

These elements can be defined only for BPEL2.0.

#3

Because these elements are not supported, they are converted to an empty activity during the import.

Tip

- The definition contents of the connection that connects each activity within a business process are decided from the structure of BPEL file and definition of each element.
- When you create a business process by importing a BPEL file, the Invoke Java Activity and Data Transformation Activity are not defined.

The following subsection explains the details of the relation between each element of the BPEL file and contents of the business process definition for BPEL1.1 and BPEL2.0.

E.1 Importing business process definitions for BPEL1.1

This section describes the support range of BPEL files and rules for converting to business process definitions for BPEL1.1.

Service Platform does not support some elements and attributes defined in the BPEL file for BPEL1.1. The following table indicates the elements and attributes defined in the BPEL file for BPEL1.1 that are supported or not supported in Service Platform.

Table E–2: Elements and attributes defined in the BPEL file for BPEL1.1 that are supported or not supported in Service Platform

Category	Element	Lower element or attribute	Support range
Elements related to overall business process definitions	process element	name attribute	N
		targetNamespace attribute	N
		queryLanguage attribute	N
		expressionLanguage attribute	N

Category	Element	Lower element or attribute			Support range
Elements related to overall business process definitions	`process` element	`suppressJoinFailure` attribute			N
		`enableInstanceCompensation` attribute			N
		`abstractProcess` attribute			N
		`partnerLinks` element			N
		`partners` element			N
		`correlationSets` element			N
		`variables` element	`variable` element	`name` attribute	Y
				`messageType` attribute	Y
				`type` attribute	Y
				`element` attribute	N
		`faultHandlers` element	`catch` element	`faultName` attribute	N
				`faultVariable` attribute	Y
				activity[#1]	Y
			`catchAll` element	*activity*[#1]	Y
		`compensationHandler` element			N
		`eventHandlers` element			N
		activity[#1]			Y
Elements related to basic activities	`receive` element	`name` attribute			Y
		`joinCondition` attribute			N
		`suppressJoinFailure` attribute			N
		`partnerLink` attribute			N
		`portType` attribute			N
		`operation` attribute			Y
		`variable` attribute			Y
		`createInstance` attribute			C
		`target` element			N
		`source` element			N
		`correlations` element			N
	`reply` element	`name` attribute			Y
		`joinCondition` attribute			N
		`suppressJoinFailure` attribute			N
		`partnerLink` attribute			N
		`portType` attribute			N

Category	Element	Lower element or attribute			Support range
Elements related to basic activities	`reply` element	`operation` attribute			Y
		`variable` attribute			Y
		`faultName` attribute			Y
		`target` element			N
		`source` element			N
		`correlations` element			N
	`invoke` element	`name` attribute			Y
		`joinCondition` attribute			N
		`suppressJoinFailure` attribute			N
		`partnerLink` attribute			N
		`portType` attribute			Y
		`operation` attribute			Y
		`inputVariable` attribute			Y
		`outputVariable` attribute			Y
		`target` element			N
		`source` element			N
		`correlations` element			N
		`catch` element	`faultName` attribute		N
			`faultVariable` attribute		Y
			activity#1		Y
		`catchAll` element	*activity*#1		Y
	`assign` element	`name` attribute			Y
		`joinCondition` attribute			N
		`suppressJoinFailure` attribute			N
		`target` element			N
		`source` element			N
		`copy` element	`from` element	`variable` attribute	Y
				`expression` attribute	Y
				`part` attribute	N
				`partnerLink` attribute	N
				`endpointReference` attribute	N
				`property` attribute	N
				`opaque` attribute	N
				`query` attribute	N

Category	Element	Lower element or attribute			Support range
Elements related to basic activities	`assign` element	`copy` element	`from` element	Tag value#2	Y
			`to` element	`variable` attribute	Y
				`part` attribute	N
				`partnerLink` attribute	N
				`property` attribute	N
				`query` attribute	N
	`empty` element	`name` attribute			Y
		`joinCondition` attribute			N
		`suppressJoinFailure` attribute			N
		`target` element			N
		`source` element			N
	`throw` element	`faultName` attribute			N
		`faultVariable` attribute			Y
		`name` attribute			Y
		`joinCondition` attribute			N
		`suppressJoinFailure` attribute			N
		`target` element			N
		`source` element			N
	`wait` element	`name` attribute			Y
		`joinCondition` attribute			N
		`suppressJoinFailure` attribute			N
		`target` element			N
		`source` element			N
		`for` attribute			Y
		`until` attribute			Y
	`terminate` element#3	`name` attribute			Y
		`joinCondition` attribute			N
		`suppressJoinFailure` attribute			N
		`target` element			N
		`source` element			N
	`compensate` element#3	`name` attribute			Y
		`joinCondition` attribute			N
		`suppressJoinFailure` attribute			N
		`scope` attribute			N

Category	Element	Lower element or attribute			Support range
Elements related to basic activities	`compensate` element#3	`target` element			N
		`source` element			N
Elements related to structure activities	`scope` element	`name` attribute			Y
		`joinCondition` attribute			N
		`suppressJoinFailure` attribute			N
		`variableAccessSerializable` attribute			N
		`target` element			N
		`source` element			N
		`correlationSets` element			N
		`variables` element	`variable` element	`name` attribute	Y
				`messageType` attribute	Y
				`type` attribute	Y
				`element` attribute	N
		`faultHandlers` element	`catch` element	`faultName` attribute	N
				`faultVariable` attribute	Y
				activity#1	Y
			`catchAll` element	*activity*#1	Y
		`compensationHandler` element			N
		`eventHandlers` element			N
		activity#1			Y
	`while` element	`name` attribute			Y
		`joinCondition` attribute			N
		`condition` attribute			Y
		`suppressJoinFailure` attribute			N
		`target` element			N
		`source` element			N
		activity#1			Y
	`switch` element	`name` attribute			Y
		`joinCondition` attribute			N
		`suppressJoinFailure` attribute			N
		`target` element			N
		`source` element			N
		`case` element	`condition` attribute		Y

Category	Element	Lower element or attribute		Support range
Elements related to structure activities	`switch` element	`case` element	*activity*#1	Y
		`otherwise` element	*activity*#1	Y
	`flow` element	`name` attribute		Y
		`joinCondition` attribute		N
		`suppressJoinFailure` attribute		N
		`target` element		N
		`source` element		N
		`links` element		N
		activity#1		Y
	`sequence` element	`name` attribute		N
		`joinCondition` attribute		N
		`suppressJoinFailure` attribute		N
		`target` element		N
		`source` element		N
		activity#1		Y
	`pick` element#3	`name` attribute		Y
		`joinCondition` attribute		N
		`suppressJoinFailure` attribute		N
		`createInstance` attribute		N
		`target` element		N
		`source` element		N
		`onAlarm` element		N
		`onMessage` element		N

Legend:

Y: Supported.

C: Supported, but with restrictions.

N: Not supported.

#1

In practice, elements such as `receive`, `reply`, and `invoke` are entered in *activity*.

#2

The tag value of the `from` element becomes the conversion source.

#3

Because this element is an unsupported activity, it is incorporated as an empty activity to which only the `name` attribute is applied.

(1) Converting elements related to overall business process definitions

This subsection explains the conversion of the elements defined in the BPEL file that are the contents related to the overall business process (such as settings of the business process and variables used).

(a) Converting the process element

The `process` element and its lower elements and attributes are converted to the contents related to the overall business process definitions.

The following table describes the conversion details.

Table E–3: Converting the process element

Elements of the BPEL file		Business process definitions	
		Definition item	Explanation
`process` element	`variables` element	Variable	For details, see *(1)(b) Converting the variables element*.
	`correlationSets` element	Correlation set	For details, see *(1)(c) Converting the correlationSets element*.
	`faultHandlers` element	Fault handling	For details, see *(1)(d) Converting the faultHandlers element*.
	activity#	*Activity*	This element is converted to activities that constitute a business process.
--		Business process name	This item is set to the business process name specified in the dialog box for adding a business process definition.
--		Business process version	This item is set to 1.
--		Persistence	This item is set to the persistence settings (persistent or non-persistent) specified in the dialog box for adding business process definitions.

Legend:

--: No corresponding element. Definition information is set automatically, or the information specified in the dialog box for adding business process definitions is set.

#

In reality, elements such as `receive`, `reply`, and `invoke` are entered in *activity*.

For practice about the elements converted to each activity of a business process, see *(2) Converting elements related to basic activities* and *(3) Converting elements related to structure activities*.

(b) Converting the variables element

The `variables` element and its lower elements and attributes are converted to definitions of the variables set in the business process (or in the scope). After importing the definitions, you can change the definitions in the List Of Variables And Correlation Sets dialog box.

! Important note

In Service Platform, the variables defined in the scopes above the scope that contains fault handling can be defined as the allocated variables of the activities that constitute the fault handling within the scope.

For this reason, if the elements below the `scope` or `faultHandlers` element in the BPEL file use the variables in the same `scope` element, redefine the allocated variables in the business process after importing the BPEL file.

The table below describes the conversion details. The names in the *Item* column in the following table are the item names in the List Of Variables And Correlation Sets dialog box.

Table E–4: Converting the variables element

Lower element or attribute of the variables element		Business process definitions	
		Item	Explanation
`variable` element	`name` attribute	**Variable name**	This attribute is set to the name of the variable.

Lower element or attribute of the variables element		Business process definitions	
		Item	Explanation
`variable` element	`messageType` attribute	**Type**	If this attribute has been defined, the `type` attribute is converted to the string type (`string`). For this reason, after converting the attribute, you must change the variable type to the message type (`XML`), and register the message format.
	`type` attribute	**Type**	This attribute is set to the following variable types: • For boolean `boolean` is set. • For a type that can be represented by `double`# `numeric` is set. • For other types or undefined types `string` is set.
	--	**Part Specifications**	No value is set in part specifications of a variable. To use part specifications, specify settings in the List Of Variables And Correlation Sets dialog box after importing the definitions.

Legend:

--: No corresponding element.

#

The following types are applicable:

`int`, `short`, `byte`, `unsignedInt`, `unsignedShort`, `unsignedByte`, `float`, `double`

(c) Converting the correlationSets element

The definition information of the `correlationSets` element corresponds to a correlation set in business process definitions. No value is set even if you create a business process by importing a BPEL file.

To use a correlation set in a business process, after importing the business process, first define a correlation set in the List Of Variables And Correlation Sets dialog box. Then, allocate the correlation set by using the Allocating Correlation Set Group dialog box for the activity that uses the correlation set.

(d) Converting the faultHandlers element

The `faultHandlers` element and its lower elements and attributes are converted to the definitions of the fault handling within the business process. You can change the converted definitions in the Fault Handler dialog box.

! Important note

- If a `faultHandlers` element is defined immediately below the `process` element, a scope is created in the highest-level business process, and the `faultHandlers` element is defined as the fault handling in that scope. In such a case, the activities immediately below the `process` element are moved to the scope that has been created.
- If a `faultHandlers` element is defined below the `scope` element, fault handling is set in the upper scope.

The table below describes the conversion details. The names listed in the *Item* column in the following table are the item names in the Fault Handler dialog box.

Table E–5: Converting the faultHandlers element

Lower element or attribute of the faultHandlers element		Business process definitions	
		Item	Explanation
`catch` element	`faultVariable` attribute	**Allocated variable**	For details about the variable, see also the notes in *(1)(b) Converting the variables element*.

Lower element or attribute of the faultHandlers element		Business process definitions	
		Item	Explanation
`catch` element	*activity*#	*Activity*	This element is converted to activities that constitute a business process.
`catchAll` element	*activity*#	*Activity*	This element is converted to activities that constitute a business process.
	--	**Allocated variable**	`catch-all` is set at all times.
--		**Transition destination**	If the `catch` or `catchAll` element is defined, the transition destination of the fault handling is set automatically.

Legend:

--: No corresponding element. Definitions are set automatically.

#

In reality, elements such as `receive`, `reply`, and `invoke` are entered in *activity*.

(2) Converting elements related to basic activities

This subsection explains the conversion of the elements defined in the BPEL file that are converted to basic activities.

(a) Converting the receive element

The `receive` element and its lower elements and attributes are converted to the definition contents of the receive activity. After importing the definitions, you can change the definitions in the Receive Activity dialog box.

! Important note

If more than one `receive` element that meets the following conditions is defined, the second and subsequent `receive` elements are converted to empty activities:

- The `portType` attribute has the same value.
- The `operation` attribute has the same value.

The table below describes the conversion details. The names in the *Item* column in the following table are the item names in the Receive Activity dialog box.

Table E‒6: Converting the receive element

Lower element or attribute of the receive element	Business process definitions	
	Item	Explanation
`operation` attribute	**Operation name**	If a `receive` element with a different `portType` attribute value and the same `operation` attribute value is already defined, *n* (*n* is an integer value of 1 or more) is appended at the end of the operation name.
`variable` attribute	**Allocated variable**	For details about the variable, see also the notes in *(1)(b) Converting the variables element*.
`createInstance` attribute	**Instance generation**	This attribute is converted to `yes`, regardless of the attribute value.
`name` attribute	**Activity name**	An activity name is set. If an activity with the same name is already defined in the business process, *n* (*n* is an integer value of 1 or more) is appended at the end of the activity name.
--	**Communication model**	**Sync** or **Async** is set according to the contents of the business process definition.

Legend:

--: No corresponding element. Definitions are set automatically.

(b) Converting the reply element

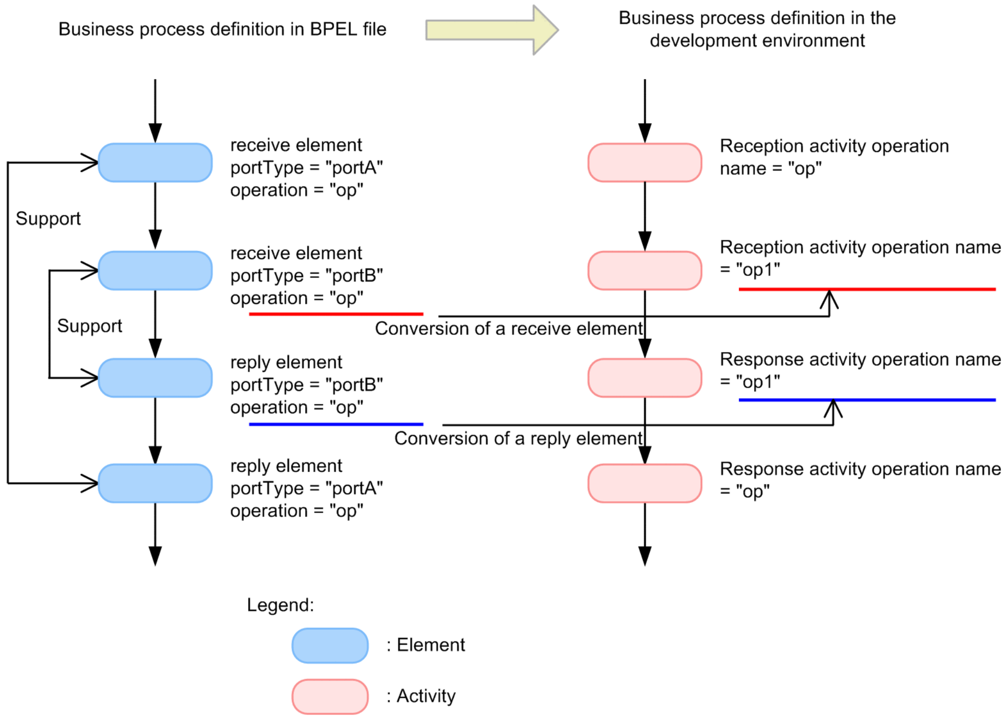
The `reply` element and its lower elements and attributes are converted to the definition contents of the reply activity. After importing the definitions, you can change the definitions in the Reply Activity dialog box.

The table below describes the conversion details. The names in the *Item* column in the following table are the item names in the Reply Activity dialog box.

Table E–7: Converting the reply element

Lower element or attribute of the reply element	Business process definitions	
	Item	Explanation
`operation` attribute	**Operation name**	If *n* (*n* is an integer value of 1 or more) is appended at the end of the operation name of a `receive` element where the `portType` attribute value and the `operation` attribute value match (see the `operation` attribute in *Table E-6*), the same value *n* is also appended at the end of the operation name of the `reply` element. *Figure E-1* shows the conversion of a `receive` element and a `reply` element where the `portType` attribute value and the `operation` attribute value match.
`variable` attribute	**Allocated variable**	For details about the variable, see also the notes in *(1)(b) Converting the variables element*.
`faultName` attribute	**Fault name**	The fault name is converted to a local name if a prefix is assigned.
`name` attribute	**Activity name**	An activity name is set. If an activity with the same name is already defined in the business process, *n* (*n* is an integer value of 1 or more) is appended at the end of the activity name.

Figure E–1: Conversion of a receive element and a reply element where the portType attribute value and the operation attribute value match


Business process definition in BPEL file
Business process definition in the development environment
receive element
portType = "portA"
operation = "op"
Support
receive element
portType = "portB"
operation = "op"
Support
reply element
portType = "portB"
operation = "op"
reply element
portType = "portA"
operation = "op"
Reception activity operation name = "op"
Reception activity operation name = "op1"
Conversion of a receive element
Response activity operation name = "op1"
Conversion of a reply element
Response activity operation name = "op"
Legend:
: Element
: Activity

(c) Converting the invoke element

The `invoke` element and its lower elements and attributes are converted to the definition contents of the invoke service activity. Furthermore, if a `catch` element or `catchAll` element is defined as a lower element of the `invoke` element, these elements are converted to a fault handler (an activity connected with a fault connection) for the invoke service activity. After importing the definitions, you can change the definitions in the Invoke Service Activity dialog box or Fault Handler dialog box.

The table below describes the conversion details. The names listed in the *Item* column in the following table are the item names in the Invoke Service Activity dialog box or Fault Handler dialog box.

Table E–8: Converting the invoke element

Lower element or attribute of the invoke element	Business process definitions	
	Item	Explanation
`portType` attribute	**Service name**	This attribute is converted only when the corresponding service components and operations exist in the repository. The service name is converted to a local name.
`operation` attribute	**Operation name**	This attribute is converted only when the corresponding service components and operations exist in the repository.
`inputVariable` attribute	**Body allocated variable for request message**	This attribute is set only when the corresponding service components and operations exist in the repository. For details about the variable, see also the notes in *(1)(b) Converting the variables element*.
`outputVariable` attribute	**Body allocated variable for reply message**	This attribute is set when the service components and operations exist in the repository, and the communication type is synchronous. For details about the variable, see also the notes in *(1)(b) Converting the variables element*.

Lower element or attribute of the invoke element		Business process definitions	
		Item	Explanation
`name` attribute		**Activity name**	An activity name is set. If an activity with the same name is already defined in the business process, *n* (*n* is an integer value of 1 or more) is appended at the end of the activity name.
`catch` element	`faultVariable` attribute	**Allocated variable**	This element is an item in the Fault Handler dialog box. For details about the variable, see also the notes in *(1)(b) Converting the variables element*.
	activity#	*Activity*	This element is converted to activities that constitute a business process.
`catchAll` element	*activity*#	*Activity*	This element is converted to activities that constitute a business process.
	--	**Allocated variable**	This element is an item in the Fault Handler dialog box. `catch-all` is set at all times.

Legend:

--: No corresponding element. Definitions are set automatically.

#

In practice, elements such as `receive`, `reply`, and `invoke` are entered in *activity*.

(d) Converting the assign element

The `assign` element and its lower elements and attributes are converted to the definition contents of the assign activity. After importing the definitions, you can change the definitions in the Assign Activity dialog box or Assign Activity sub-dialog box.

The table below describes the conversion details. The names listed in the *Item* column in the following table are the item names in the Assign Activity dialog box or Assign Activity sub-dialog box.

Table E–9: Converting the assign element

Lower element or attribute of the assign element			Business process definitions	
			Item	Explanation
`name` attribute			**Activity name**	An activity name is set. If an activity with the same name is already defined in the business process, *n* (*n* is an integer value of 1 or more) is appended at the end of the activity name.
`copy` element	`from` element	`variable` attribute	**Name** (under **Variable** in the **Copy source** frame)	This attribute is an item in the Assign Activity sub-dialog box. For details about the variable, see also the notes in *(1)(b) Converting the variables element*.
		`expression` attribute	**Value** (under **Expression** in the **Copy source** frame)	This attribute is an item in the Assign Activity sub-dialog box. This attribute is not converted if the `variable` attribute has been defined.
		Tag value#	**Value** (under **Expression** in the **Copy source** frame)	This attribute is an item in the Assign Activity sub-dialog box. If the `variable` and `expression` attributes are not defined, the tag value is set as the value of the expression.
	`to` element	`variable` attribute	**Variable name** (in the **Copy destination** frame)	This attribute is an item in the Assign Activity sub-dialog box.

Lower element or attribute of the assign element			Business process definitions	
			Item	Explanation
`copy` element	`to` element	`variable` attribute	**Variable name** (in the **Copy destination** frame)	For details about the variable, see also the notes in *(1)(b) Converting the variables element*.

#

The tag value of the `from` element becomes the conversion source.

(e) Converting the empty element

The `empty` element is converted to the definition contents of the empty activity. After importing the definitions, you can change the definitions in the Empty Activity dialog box.

The table below describes the conversion details. The names in the *Item* column in the following table are the item names in the Empty Activity dialog box.

Table E‒10: Converting the empty element

Lower element or attribute of the empty element	Business process definitions	
	Item	Explanation
`name` attribute	**Activity name**	An activity name is set. If an activity with the same name is already defined in the business process, *n* (*n* is an integer value of 1 or more) is appended at the end of the activity name.

(f) Converting the throw element

The `throw` element and its lower elements and attributes are converted to the definition contents of the throw activity. After importing the definitions, you can change the definitions in the Throw Activity dialog box.

The table below describes the conversion details. The names in the *Item* column in the following table are the item names in the Throw Activity dialog box.

Table E‒11: Converting the throw element

Lower element or attribute of the throw element	Business process definitions	
	Item	Explanation
`faultVariable` attribute	**Allocated variable**	For details about the variable, see also the notes in *(1)(b) Converting the variables element*.
`name` attribute	**Activity name**	An activity name is set. If an activity with the same name is already defined in the business process, *n* (*n* is an integer value of 1 or more) is appended at the end of the activity name.

(g) Converting the wait element

The `wait` element is converted to the definition contents of the standby activity. After importing the definitions, you can change the definitions in the Wait Activity dialog box.

The table below describes the conversion details. The names in the *Item* column in the following table are the item names in the Wait Activity dialog box.

Table E‒12: Converting the wait element

Lower element or attribute of the wait element	Business process definitions	
	Item	Explanation
`name` attribute	**Activity name**	An activity name is set. If an activity with the same name is already defined in the business process, *n* (*n* is an integer

Lower element or attribute of the wait element	Business process definitions	
	Item	Explanation
`name` attribute	**Activity name**	value of 1 or more) is appended at the end of the activity name.
`for` attribute#	**Expression**	This attribute sets information about the standby interval. The **For** radio button is selected.
`until` attribute#	**Expression**	This attribute sets information about the standby time period. The **Until** radio button is selected.

#

If both the `for` attribute and the `until` attribute are defined, the `for` attribute is enabled and the `until` attribute is disabled. If neither the `for` attribute nor the `until` attribute is defined, the **For** radio button is selected.

(3) Converting elements related to structure activities

This subsection explains the conversion of the elements defined in the BPEL file that are converted to structure activities.

(a) Converting the scope element

The `scope` element and its lower elements and attributes are converted to the definition contents of the scope activity. After importing the definitions, you can change the definitions in the Scope Activity dialog box.

Furthermore, the variables, correlation sets, fault handling, and transaction control information defined within the `scope` element can be changed in the List Of Variables And Correlation Sets dialog box and Fault Handler dialog box.

The following table describes the conversion details.

Table E–13: Converting the scope element

Lower element or attribute of the scope element	Business process definitions	
	Item	Explanation
`name` attribute	Activity name	An activity name is set. If an activity with the same name is already defined in the business process, *n* (*n* is an integer value of 1 or more) is appended at the end of the activity name.
--	Transaction control	This item specifies whether to execute transaction control. **Commit after each activity** is selected by default.
`variables` element	Variable	This element sets information about the variables used in the scope. For details about variables, see *(1)(b) Converting the variables element*.
`correlationSets` element	Correlation set	This element sets information about the correlation sets used in the scope. For details about correlation sets, see *(1)(c) Converting the correlationSets element*.
`faultHandlers` element	Fault handling	This item sets information about the fault handling in the scope. For details about fault handling, see *(1)(d) Converting the faultHandlers element*.
activity#	*Activity*	This element is converted to activities that constitute a business process.

Legend:

--: No corresponding element. Definitions are set automatically.

#

In practice, elements such as receive, reply, and invoke are entered in *activity*.

(b) Converting the while element

The while element and its lower elements and attributes are converted to the definition contents of the while activity. After importing the definitions, you can change the definitions in the While Activity dialog box and Condition Setting dialog box.

The table below shows the conversion details. The names listed in the *Item* column in the following table are the item names in the While Activity dialog box or Condition Setting dialog box.

Table E–14: Converting the while element

Lower element or attribute of the while element	Business process definitions	
	Item	Explanation
name attribute	**Activity name**	An activity name is set. If an activity with the same name is already defined in the business process, *n* (*n* is an integer value of 1 or more) is appended at the end of the activity name.
condition attribute	**Condition Expression**	Values are set in the conditional expression in the Condition Setting dialog box.
activity#	*Activity*	This element is converted to activities that constitute a business process.
--	**Max Loop Count**	This item is automatically set to 100.

Legend:

--: No corresponding element. Definitions are set automatically.

#

In practice, elements such as receive, reply, and invoke are entered in *activity*.

(c) Converting the switch element

The switch element and its lower elements and attributes are converted to the definition contents of the switch start activity. After importing the definitions, you can change the definitions in the Switch Activity dialog box and Condition Setting dialog box.

! Important note

A switch end activity is set automatically when the elements below the switch element are converted to a switch start activity. In such a case, _end is added to the name of the switch start activity to assign a name to the switch end activity.

The table below describes the conversion details. The names listed in the *Item* column in the following table are the item names in the Switch Activity dialog box or Condition Setting dialog box.

Table E–15: Converting the switch element

Lower element or attribute of the switch element		Business process definitions	
		Item	Explanation
name attribute		**Activity name**	An activity name is set. If an activity with the same name is already defined in the business process, *n* (*n* is an integer value of 1 or more) is appended at the end of the activity name.
case element	*activity*#	*Activity*	This element is converted to activities that constitute a business process.
	condition attribute	**Condition Expression**	Values are set in the conditional expression in the Condition Setting dialog box.
	--	**Condition name**	This attribute is an item in the Switch Activity dialog box.

Lower element or attribute of the switch element		Business process definitions	
		Item	Explanation
`case` element	--	**Condition name**	This item is automatically set to condition *n*. *n* indicates a unique integer within the switch.
	--	**Priority**	This attribute is an item in the Switch Activity dialog box. Priorities are assigned according to the occurrence order of the conditions within the `switch` element.
`otherwise` element	*activity*#	*Activity*	This element is converted to activities that constitute a business process.
	--	**Priority**	This attribute is an item in the Switch Activity dialog box. This item is set to the default value.
--		**Transition destination**	This attribute is an item in the Switch Activity dialog box. This item is set according to the definitions of the `case` and `otherwise` elements.

Legend:

--: No corresponding element. Definitions are set automatically.

#

In practice, elements such as `receive`, `reply`, and `invoke` are entered in *activity*.

(d) Converting the flow element

The `flow` element and its lower elements and attributes are converted to the definition contents of the flow start activity. After importing the definitions, you can change the definitions in the Flow Start Activity dialog box.

! Important note

- The link definitions (definitions below the `links` element in the BPEL file) are not converted.
- A flow end activity is set automatically when the elements below the `flow` element are converted to a flow start activity. In such a case, `_end` is added to the name of the flow start activity to assign a name to the flow end activity.

The table below describes the conversion details. The names in the *Item* column in the following table are the item names in the Flow Activity dialog box.

Table E–16: Converting the flow element

Lower element or attribute of the flow element	Business process definitions	
	Item	Explanation
`name` attribute	**Activity name**	An activity name is set. If an activity with the same name is already defined in the business process, *n* (*n* is an integer value of 1 or more) is appended at the end of the activity name.
activity#	*Activity*	This element is converted to activities that constitute a business process.

#

In practice, elements such as `receive`, `reply`, and `invoke` are entered in *activity*.

(e) Converting the sequence element

The `sequence` element is converted to the definition contents of the sequence activity. You cannot modify the definition contents.

The `sequence` element does not have any lower attributes to be converted.

In Service Platform, only the sequence elements specified at the positions described in *5.6.17 Sequence Activity* are converted to business process definitions.

E.2 Importing a business process definition of BPEL2.0

This section describes the support range of a BPEL file and rules for converting to a business process definition for BPEL2.0.

The Cosminexus Service Platform does not support some elements and attributes defined in a BPEL file for BPEL2.0. The following table describes whether or not the elements and attributes defined in a BPEL file in BPEL2.0 are supported in Cosminexus Service Platform:

Table E–17: Support range of elements and attributes defined in a BPEL file (For BPEL2.0)

Classificati on	Element	Lower element or attribute			Support range
Elements related to the overall business process definitions	process element	name attribute			N
		targetNamespace attribute			N
		queryLanguage attribute			N
		expressionLanguage attribute			N
		suppressJoinFailure attribute			N
		exitOnStandardFault attribute			N
		partnerLinks element			N
		correlationSets element			N
		variables element	documentation element		N
			variable element	name attribute	Y
				messageType attribute	Y
				type attribute	Y
				element attribute	N
				documentation element	N
				From element	N
		faultHandlers element	documentation element		N
			catch element	faultName attribute	N
				faultVariable attribute	Y
				faultMessageType attribute	N
				faultElement attribute	N
				documentation element	N
				activity#1	Y
			catchall element	documentation element	N
				activity#1	Y
		eventHandlers element			N
		import element			N

Classificati on	Element	Lower element or attribute	Support range
Elements related to the overall business process definitions	`process` element	`messageExchanges` element	N
		`documentation` element	N
		activity#1	Y
Elements related to the basic activity	`receive` element	`name` attribute	Y
		`suppressJoinFailure` attribute	N
		`partnerLink` attribute	N
		`portType` attribute	N
		`operation` attribute	Y
		`variable` attribute	Y
		`createInstance` attribute	C
		`messageExchange` attribute	N
		`targets` element	N
		`sources` element	N
		`documentation` element	N
		`correlations` element	N
		`fromParts` element	N
	`reply` element	`name` attribute	Y
		`suppressJoinFailure` attribute	N
		`partnerLink` attribute	N
		`portType` attribute	N
		`operation` attribute	Y
		`variable` attribute	Y
		`faultName` attribute	Y
		`messageExchange` attribute	N
		`targets` element	N
		`sources` element	N
		`documentation` element	N
		`correlations` element	N
		`toParts` element	N
	`invoke` element	`name` attribute	Y
		`suppressJoinFailure` attribute	N
		`partnerLink` attribute	N
		`portType` attribute	Y

Classificati on	Element	Lower element or attribute			Support range
Elements related to the basic activity	`invoke` element	`operation` attribute			Y
		`inputVariable` attribute			Y
		`outputVariable` attribute			Y
		`targets` element			N
		`sources` element			N
		`documentation` element			N
		`correlations` element			N
		`toParts` element			N
		`fromParts` element			N
		`catch` element	`faultName` attribute		N
			`faultVariable` attribute		Y
			`faultMessageType` attribute		N
			`faultElement` attribute		N
			`documentation` element		N
			activity#1		Y
		`catchall` element	`documentation` element		N
			activity#1		Y
	`assign` element	`name` attribute			Y
		`suppressJoinFailure` attribute			N
		`validate` attribute			N
		`targets` element			N
		`sources` element			N
		`documentation` element			N
		`copy` element	`keepSrcElementName` attribute		N
			`ignoreMissingFromData` attribute		N
			`documentation` element		N
			`from` element	`variable` attribute	Y
				`expressionLanguage` attribute	N
				`part` attribute	N
				`partnerLink` attribute	N
				`endpointReference` attribute	N
				`property` attribute	N
				`documentation` element	N
				`literal` element	Y

Classification	Element	Lower element or attribute				Support range
Elements related to the basic activity	`assign` element	`copy` element	`from` element	`query` element	`queryLanguage` attribute	N
			`to` element	`variable` attribute		Y
				`part` attribute		N
				`partnerLink` attribute		N
				`expressionLanguage` attribute		N
				`property` attribute		N
				`documentation` element		N
				`query` element	`queryLanguage` attribute	N
	`empty` element	`name` attribute				Y
		`suppressJoinFailure` attribute				N
		`targets` element				N
		`sources` element				N
		`documentation` element				N
	`throw` element	`name` attribute				Y
		`suppressJoinFailure` attribute				N
		`faultName` attribute				N
		`faultVariable` attribute				Y
		`targets` element				N
		`sources` element				N
		`documentation` element				N
	`wait` element	`name` attribute				Y
		`suppressJoinFailure` attribute				N
		`targets` element				N
		`sources` element				N
		`documentation` element				N
		`for` element				Y
		+		`expressionLanguage` attribute		N
		`until` element				Y
		+		`expressionLanguage` attribute		N
	`compensate` element#2	`name` attribute				Y
		`suppressJoinFailure` attribute				N
		`targets` element				N
		`sources` element				N

Classification	Element	Lower element or attribute		Support range
Elements related to the basic activity	`compensate` element#2	`documentation` element		N
	`extensionActivity` element#2	*anyElementQName*#3	`name` attribute	Y
			`suppressJoinFailure` attribute	N
			anyAttribute#3	N
			`targets` element	N
			`sources` element	N
			`documentation` element	N
			any#3	N
	`rethrow` element#2	`name` attribute		Y
		`suppressJoinFailure` attribute		N
		`targets` element		N
		`sources` element		N
		`documentation` element		N
	`exit` element#2	`name` attribute		Y
		`suppressJoinFailure` attribute		N
		`targets` element		N
		`sources` element		N
		`documentation` element		N
	`validate` element#2	`name` attribute		Y
		`suppressJoinFailure` attribute		N
		`variables` attribute		N
		`targets` element		N
		`sources` element		N
		`documentation` element		N
	`compensateScope` element#2	`name` attribute		Y
		`suppressJoinFailure` attribute		N
		`target` attribute		N
		`targets` element		N
		`sources` element		N
		`documentation` element		N
Elements related to the structure activity	`scope` element	`name` attribute		Y
		`suppressJoinFailure` attribute		N
		`exitOnStandardFault` attribute		N

Classification	Element	Lower element or attribute			Support range
Elements related to the structure activity	scope element	isolated attribute			N
		targets element			N
		sources element			N
		documentation element			N
		correlationSets element			N
		variables element	documentation element		N
			variable element	name attribute	Y
				messageType attribute	Y
				type attribute	Y
				element attribute	N
				documentation element	N
				From element	N
		faultHandlers element	documentation element		N
			catch element	faultName attribute	N
				faultVariable attribute	Y
				faultMessageType attribute	N
				faultElement attribute	N
				documentation element	N
				activity#1	Y
			catchall element	documentation element	N
				activity#1	Y
		compensationHandler element			N
		eventHandlers element			N
		terminationHandle element			N
		partnerLinks element			N
		messageExchanges element			N
		activity#1			Y
	while element	name attribute			Y
		suppressJoinFailure attribute			N
		targets element			N
		sources element			N
		documentation element			N
		condition element			Y
		+		expressionLanguage attribute	N

Classification	Element	Lower element or attribute		Support range
Elements related to the structure activity	`while` element	*activity*#1		Y
	`if` element	`name` attribute		Y
		`suppressJoinFailure` attribute		N
		`targets` element		N
		`sources` element		N
		`documentation` element		N
		`condition` element		Y
		+	`expressionLanguage` attribute	N
		`elseif` element	`documentation` element	N
			`condition` element	Y
			activity#1	Y
		`else` element	`documentation` element	N
			activity#1	Y
		activity#1		Y
	`flow` element	`name` attribute		Y
		`suppressJoinFailure` attribute		N
		`targets` element		N
		`sources` element		N
		`documentation` element		N
		`links` element		N
		activity#1		Y
	`sequence` element	`name` attribute		Y
		`suppressJoinFailure` attribute		N
		`targets` element		N
		`sources` element		N
		`documentation` element		N
		activity#1		Y
	`pick` element#2	`name` attribute		Y
		`suppressJoinFailure` attribute		N
		`createInstance` attribute		N
		`targets` element		N
		`sources` element		N
		`documentation` element		N
		`onAlarm` element		N

Classification	Element	Lower element or attribute	Support range
Elements related to the structure activity	`pick` element#2	`onMessage` element	N
	`forEach` element#2	`name` attribute	Y
		`suppressJoinFailure` attribute	N
		`counterName` attribute	N
		`parallel` attribute	N
		`targets` element	N
		`sources` element	N
		`documentation` element	N
		`startCounterValue` element	N
		`finalCounterValue` element	N
		`completionCondition` element	N
		`scope` element	N
	`repeatUntil` element#2	`name` attribute	Y
		`suppressJoinFailure` attribute	N
		`targets` element	N
		`sources` element	N
		`documentation` element	N
		`condition` element	N

Legend:

Y: Supported.

C: Supported, but with restrictions.

N: Not supported.

#1

In reality, elements such as the `receive` element, the `reply` element, and the `invoke` element are entered in *activity*.

#2

Because this is an unsupported activity, this will be incorporated as an empty activity in which only the `name` attribute is applied.

#3

In reality, an element expressing the extension activity is entered in *anyElementQName*. *anyAttribute* indicates an attribute specific to the extension activity, while *any* indicates an element specific to the extension activity.

(1) Converting elements related to the overall business process definitions

Among the elements defined in the BPEL file, the conversion of the contents related to the overall business process (such as settings of the business process and variables used) is explained below.

(a) process element conversion

The process element and its lower elements and attributes are converted to the contents related to the overall business process definitions.

The table below describes the conversion details.

Table E–18: process element conversion

Element of the BPEL file		Definition contents of the business process	
		Definition content	Explanation
process element	variables element	Variable	For details, see *Appendix E.2(1)(b) variables element conversion*.
	correlationSets element	Correlation set	For details about the variables, see *Appendix E.2(1)(c) correlationSets element conversion*.
	faultHandlers element	Fault handling	For details about the variables, see *Appendix E.2(1)(d) faultHandlers element conversion*.
	activity#	*Activity*	Converted to the activity constituting the business process.
--		Business process name	The business process name specified in the dialog box for adding a business process definition is set.
--		Business process version	1 is set.
--		Persistence	The persistence existence specified in the dialog box for adding a business process definition is set.

Legend:

--: No corresponding element. Either the definition contents are set automatically, or contents specified in the dialog box for adding the business process definition are set.

#

In reality, elements such as the receive element, the reply element, and the invoke element are entered in *activity*.

For details about the elements converted to each activity of a business process, see *Appendix E.2(2) Converting elements related to the Basic Activity* and *Appendix E.2(3) Converting elements related to the Structure Activity*.

(b) variables element conversion

The variables element and its lower elements and attributes are converted to definitions of the variables set in the business process (or in scope). After importing, you can change the definition contents using the List Of Variables And Correlation Sets dialog box.

! Important note

In the Cosminexus Service Platform, the variables defined in the scopes above the scope that contains the fault handler, can be defined as the allocated variables of the activities constituting the fault handler within the scope.

As a result, if the variables present in the same scope element are being used as variables in the elements below the scope/faultHandlers element of the BPEL file, redefine the assigned variables in the business process, after importing.

The table below describes the conversion details. The names indicated in Item are the item names of the List Of Variables And Correlation Sets dialog box.

Table E–19: variables element conversion

Lower elements or attributes of the variables element		Definition contents of the business process	
		Item	Explanation
variable element	name attribute	Variable name	The name of the variable is set.
	messageType attribute	Type	If this attribute is defined, the type attribute is converted to the string type (string). As a result, after conversion you must change the type of variable to the message type (messageType), and register the message format.

Lower elements or attributes of the variables element		Definition contents of the business process	
		Item	Explanation
`variable` element	`type` attribute	Type	The following variable types are set: • For boolean `boolean` is set. • For a type# that can be represented by double `numeric` is set. • For other than above or when the type is not defined `string` is set.
	--	Part specifications	Value is not set in part specifications of a variable. When you want to use part specifications, perform the setting with the List Of Variables And Correlation Sets dialog box, after importing.

Legend:

--: No corresponding element.

#

The following types are applicable:
int, short, byte, unsignedInt, unsignedShort, unsignedByte, float, double

(c) correlationSets element conversion

The definition contents of the `correlationSets` element correspond to Correlation set in the business process definition. The value will not be set even when you import the BPEL file and create a business process.

When you want to use a correlation set in the business process, you define the correlation set with the List Of Variables And Correlation Sets dialog box after importing, and then allocate the correlation set with the Allocating Correlation Set Group dialog box of the activity that uses the correlation set.

(d) faultHandlers element conversion

The `faultHandlers` element and its lower elements and attributes are converted to the definition contents of the fault handler within the business process. You can change the definition contents after conversion using the Fault Handler dialog box.

! Important note

- If the `faultHandlers` element is defined immediately below the process element, scope is created in the highest business process, and the `faultHandlers` element is defined as the fault handler of that scope. In such a case, the activities immediately below the `process` element moves to the scope that is created.
- If the `faultHandlers` element is defined below the scope element, the fault handler is set in the upper scope.

The table below describes the conversion details. The names indicated in Item are the item names of the Fault Handler dialog box.

Table E–20: faultHandlers element conversion

Lower elements or attributes of the faultHandlers element		Definition contents of the business process	
		Item	Explanation
`catch` element	`faultVariable` attribute	Allocated variable	For details about the variable, also see the notes in *Appendix E.2(1)(b) variables element conversion.*
	activity#	*Activity*	Converted to the activity constituting the business process.
`catchall` element	*activity*#	*Activity*	Converted to the activity constituting the business process.
	--	Allocated variable	`catch-all` is set always.

Lower elements or attributes of the faultHandlers element	Definition contents of the business process	
	Item	Explanation
--	Transition destination	If the catch element or catchAll element is defined, the transition destination of the fault handler is set automatically.

Legend:

--: No corresponding element. The definition contents are set automatically.

#

In reality, elements such as the `receive` element, the `reply` element, and the `invoke` element are entered in *activity*.

(2) Converting elements related to the Basic Activity

Of the elements defined in the BPEL file, the conversion of the elements converted to the Basic Activity is explained below.

(a) receive element conversion

The `receive` element and its lower elements and attributes are converted to the definition contents of the Receive Activity. After importing, you can change the definition contents with the Receive dialog box.

! Important note

If more than one `receive` element fulfilling the following conditions is defined, the second `receive` element onwards will be converted to an empty activity:

- The `portType` attribute is the same
- The `operation` attribute is the same

The table below describes the conversion details. The names indicated in Item are the item names of the Receive dialog box.

Table E–21: receive element conversion

Lower elements or attributes of the receive element	Definition contents of the business process	
	Item	Explanation
`operation` attribute	Operation name	If a receive element with a different `portType` attribute and the same operation attribute is already defined, n (n is an integer value of 1 or more) is added at the end of the operation name.
`variable` attribute	Allocated variable	For details about the variable, also see the notes in *Appendix E.2(1)(b) variables element conversion*.
`createInstance` attribute	Instance generation	Converted to yes, without any concern with the attribute value.
`name` attribute	Activity name	The name of the activity is set. If an activity with the same name is already defined in the business process, *n* (*n* is an integer value of 1 or more) will be added at the end of the activity name.
--	Communication model	`Sync` or `Async` is set based on the contents of the business process definition.

Legend:

--: No corresponding element. The definition contents are set automatically.

(b) reply element conversion

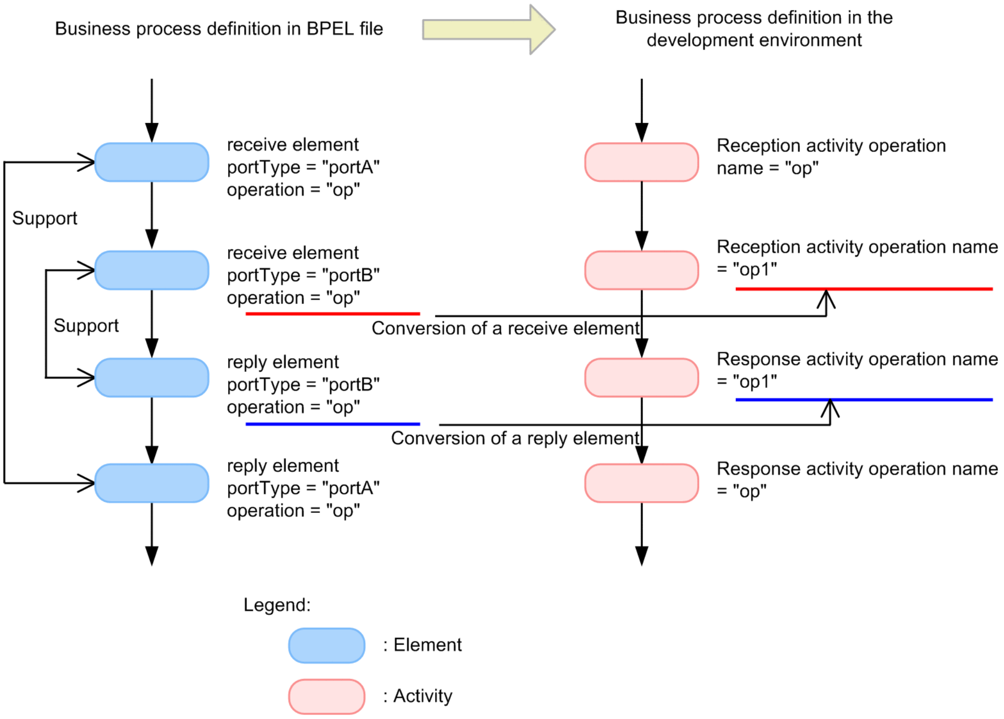
The `reply` element and its lower elements and attributes are converted to definition contents of the Reply Activity. After importing, you can change the definition contents with the Reply dialog box.

The table below describes the conversion details. The names indicated in Item are the item names of the Reply dialog box.

Table E–22: reply element conversion

Lower elements or attributes of the reply element	Definition contents of the business process	
	Item	Explanation
`operation` attribute	Operation name	If *n* (*n* is an integer value of 1 or more) is added at the end of the operation name of a `receive` element with the same `portType` attribute and the `operation` attribute (see the `operation` attribute in *Table E-21*), the same *n* will be added at the end of the operation name of the `reply` element as well. The conversion of the `receive` element and the `reply` element with the same `portType` attribute and the `operation` attribute is shown in *Figure E-2*.
`variable` attribute	Allocated variable	For details about the variable, also see the notes in *Appendix E.2(1)(b) variables element conversion*.
`faultName` attribute	Fault name	Converted to a local name, if a prefix is attached.
`name` attribute	Activity name	The name of the activity is set. If an activity with the same name is already defined in the business process, *n* (*n* is an integer value of 1 or more) will be added at the end of the activity name.

Figure E–2: Conversion of receive element and reply element with the same portType attribute and operation attribute


Business process definition in BPEL file
Business process definition in the development environment
receive element
portType = "portA"
operation = "op"
Support
receive element
portType = "portB"
operation = "op"
Support
reply element
portType = "portB"
operation = "op"
reply element
portType = "portA"
operation = "op"
Reception activity operation name = "op"
Reception activity operation name = "op1"
Conversion of a receive element
Response activity operation name = "op1"
Conversion of a reply element
Response activity operation name = "op"
Legend:
: Element
: Activity

(c) invoke element conversion

The invoke element and its lower elements and attributes are converted to definition contents of the Invoke Service Activity. Furthermore, if the catch element or catchAll element is defined as a lower element of the invoke element, these elements are converted to the fault handler (activity connected with the fault connection) in the Invoke Service Activity. After importing, you can change the definition contents with the Invoke Service Activity dialog box or Fault Handler dialog box.

The table below describes the conversion details. The names indicated in Item are the item names of the Invoke Service Activity dialog box or Fault Handler dialog box.

Table E–23: invoke element conversion

Lower elements or attributes of the invoke element		Definition contents of the business process	
		Item	Explanation
portType attribute		Service name	Converted only when the corresponding service components and operations exist in the repository. Converted to a local name.
operation attribute		Operation name	Converted only when the corresponding service components and operations exist in the repository.
inputVariable attribute		Request message variable	Set when the service components and operations exist in the repository. For details about the variable, also see the notes in *Appendix E.2(1)(b) variables element conversion*.
outputVariable attribute		Reply message variable	Set when the service components and operations exist in the repository, and the communication type is synchronous. For details about the variable, also see the notes in *Appendix E.2(1)(b) variables element conversion*.
name attribute		Activity name	The name of the activity is set. If an activity with the same name is already defined in the business process, *n* (*n* is an integer value of 1 or more) will be added at the end of the activity name.
catch element	faultVariable attribute	Allocated variable	An item of the Fault Handler dialog box. For details about the variable, also see the notes in *Appendix E.2(1)(b) variables element conversion*.
	activity#	*Activity*	Converted to the activity configuring a business process.
catchall element	*activity*#	*Activity*	Converted to the activity configuring a business process.
	--	Allocated variable	An item of the **Fault Handler** dialog box. catch-all is set always.

Legend:

--: No corresponding element. The definition contents are set automatically.

#

In reality, elements such as the receive element, the reply element, and the invoke element are entered in *activity*.

(d) assign element conversion

The assign element and its lower elements and attributes are converted to the definition contents of the Assign Activity. After importing, you can change the definition contents using the Assign Activity dialog box or Assign Activity sub dialog box.

The table below describes the conversion details. The names indicated in Item are the item names of the Assign Activity dialog box or Assign Activity sub dialog box.

Table E–24: assign element conversion

Lower elements or attributes of the assign element			Definition contents of the business process	
			Item	Explanation
`name` attribute			Activity name	The name of the activity is set. If an activity with the same name is already defined in the business process, *n* (*n* is an integer value of 1 or more) will be added at the end of the activity name.
`copy` element	`from` element	`variable` attribute	Name (variable of the copy source)	An item of the Assign Activity sub dialog box. The value set in the **Assign Activity** sub dialog box differs depending on the specification of the `variable` attribute, the `literal` element, and the `query` element. For details, see *Table E-25*. For details about the variable, also see the notes in *Appendix E.2(1)(b) variables element conversion*.
		`literal` element	Value (expression of the copy source)	An item of the **Assign Activity** sub dialog box. The value set in the **Assign Activity** sub dialog box differs depending on the specification of the `variable` attribute, the `literal` element, and the `query` element. For details, see *Table E-25*.
		`query` element	Value (expression of the copy source)	An item of the **Assign Activity** sub dialog box. The value set in the **Assign Activity** sub dialog box differs depending on the specification of the `variable` attribute, the `literal` element, and the `query` element. For details, see *Table E-25*.
	`to` element	`variable` attribute	Variable name (copy destination)	An item of the Assign Activity sub dialog box. For details about the variable, also see the notes in *Appendix E.2(1)(b) variables element conversion*.

The following table describes the values set in the **Assign Activity** sub dialog box according to the specification of the `variable` attribute, the `literal` element, and the `query` element:

Table E–25: Values set in the Assign Activity sub dialog box

Item No.	Attributes of the from element under the assign element			Value set in the Assign Activity sub dialog box
	variable attribute	literal element	query element	
1	N	N	N	No value is set.
2	Y	N	N	The value specified in the `variable` attribute is set in **Name** in **Variable** of the **Copy source**.
3	N	Y	N	The value specified in the `literal` element is set in **Value** in **Expression** of the **Copy source**.
4	N	N	Y	The value specified in the `query` element is set in **Value** in **Expression** of the **Copy source**.
5	Y	Y	N	The value specified in the `variable` attribute is set in **Name** in **Variable** of the **Copy source**.
6	Y	N	Y	The value specified in the `query` element is set in **Value** in **Expression** of the **Copy source**.
7	N	Y	Y	The assign element is not converted (an error occurs).
8	Y	Y	Y	The assign element is not converted (an error occurs).

Legend:

Y: Indicates that a value is set.

N: Indicates that no value is set.

(e) empty element conversion

The empty element is converted to definition contents of the Empty Activity. After importing, you can change the definition contents with the Empty dialog box.

The table below describes the conversion details. The names indicated in Item are the item names of the Empty Activity dialog box.

Table E–26: empty element conversion

Lower elements or attributes of the empty element	Definition contents of the business process	
	Item	Explanation
`name` attribute	Activity name	The name of the activity is set. If an activity with the same name is already defined in the business process, *n* (*n* is an integer value of 1 or more) will be added at the end of the activity name.

(f) throw element conversion

The `throw` element and its lower elements and attributes are converted to the definition contents of the Fault Handler Activity. After importing, you can change the definition contents with the Fault Handler dialog box.

The table below describes the conversion details. The name indicated in Item is the item name of the Fault Handler dialog box.

Table E–27: throw element conversion

Lower elements or attributes of the throw element	Definition contents of the business process	
	Item	Explanation
`faultVariable` attribute	Allocated variable	For details about the variable, also see the notes in *Appendix E.2(1)(b) variables element conversion*.
`name` attribute	Activity name	The name of the activity is set. If an activity with the same name is already defined in the business process, *n* (*n* is an integer value of 1 or more) will be added at the end of the activity name.

(g) Transforming the wait element

The `wait` element is transformed into the definition contents of the standby activity. After importing this element, you can change the definition contents in the standby activity dialog box.

The following table describes the details of transformation. The names described in the *Item* column in the table are the item names from the standby activity dialog box.

Table E–28: Transforming the wait element

Lower elements or attributes of the wait element	Definition contents of the business process	
	Item	Explanation
`name` attribute	Activity name	The activity name is set up. If an activity with the same name is already defined in the business process, *n* (n is an integer value of 1 or more) is added at the end of the activity name.
`for` element	Expression	Information about the standby interval. `Interval` becomes `ON`.
`until` element	Expression	Information about the standby time period. `Time Period` becomes `ON`.

(3) Converting elements related to the Structure Activity

Of the elements defined in the BPEL file, the conversion of the elements converted to the Basic Activity is explained below.

(a) scope element conversion

The `scope` element and its lower elements and attributes are converted to definition contents of the Scope Activity. After importing, you can change the definition contents with the Scope Activity dialog box.

Furthermore, the variables, correlation sets, and information about the fault handlers defined within the scope element can be changed with the List Of Variables And Correlation Sets dialog box and Fault Handler dialog box.

The table below describes the conversion details.

Table E–29: scope element conversion

Lower elements or attributes of the scope element	Definition contents of the business process	
	Item	Explanation
`name` attribute	Activity name	The name of the activity is set. If an activity with the same name is already defined in the business process, *n* (*n* is an integer value of 1 or more) will be added at the end of the activity name.
`variables` element	Variable	Information about the variables used in scope. For details about the variables, see *Appendix E.2(1)(b) variables element conversion*.
`correlationSets` element	Correlation set	Information about the correlation sets used in scope. For details about the correlation set information, see *Appendix E.2(1)(c) correlationSets element conversion*.
`faultHandlers` element	Fault handling	Information about the fault handlers used in scope. For details about the fault handler information, see *Appendix E.2(1)(d) faultHandlers element conversion*.
activity#	*Activity*	Converted to the activity configuring a business process.

#

In reality, elements such as the `receive` element, `reply` element, and `invoke` element are entered in *activity*.

(b) while element conversion

The while element and its lower elements and attributes are converted to definition contents of the While Activity. After importing, you can change the definition contents with the While Activity dialog box and Condition Setting dialog box.

The table below describes the conversion details. The names indicated in Item are the item names of the While Activity dialog box or Condition Setting dialog box.

Table E–30: while element conversion

Lower elements or attributes of the while element	Definition contents of the business process	
	Item	Explanation
`name` attribute	Activity name	The name of the activity is set. If an activity with the same name is already defined in the business process, *n* (*n* is an integer value of 1 or more) will be added at the end of the activity name.
`condition` element	Condition Expression	A value is set in the condition expression of the **Condition Setting** dialog box.
activity#	*Activity*	Converted to the activity constituting the business process.

Lower elements or attributes of the while element	Definition contents of the business process	
	Item	Explanation
--	Maximum Loop Count	`100` is set automatically.

Legend:

--: No corresponding element. The definition contents are set automatically.

#

In reality, elements such as the `receive` element, the `reply` element, and the `invoke` element are entered in *activity*.

(c) if element conversion

The if element and its lower elements and attributes are converted to definition contents of the Switch Start Activity. After importing, you can change the definition contents with the ConBranch Activity dialog box and Condition setting dialog box.

! Important note

The Switch End Activity is set automatically when the elements below the if element are converted to the Switch Start Activity. In such a case, _end is added to the name of the Switch Start Activity to set the name of the Switch End Activity.

The table below describes the conversion details. The names indicated in Item are the item names of the Switch Activity dialog box or Condition Setting dialog box.

Table E–31: if element conversion

Lower elements or attributes of the if element		Definition contents of the business process	
		Item	Explanation
`name` attribute		Activity name	The name of the activity is set. If an activity with the same name is already defined in the business process, *n* (*n* is an integer value of 1 or more) will be added at the end of the activity name.
`condition` element		Condition Expression	A value is set in the condition expression of the **Condition Setting** dialog box.
activity#		*Activity*	Converted to the activity constituting the business process.
--		Priority	An item of the Switch Activity dialog box. `1` is set at all times.
`elseif` element	`condition` element	Condition Expression	A value is set in the condition expression of the **Condition Setting** dialog box.
	activity#	*Activity*	Converted to the activity configuring a business process.
	--	Condition name	An item of the Switch Activity dialog box. condition n is set automatically. n indicates a unique integer within switch.
	--	Priority	An item of the Switch Activity dialog box. Set in the order of occurrence starting from `2`.
`else` element	*activity*#	*Activity*	Converted to the activity configuring a business process.
	--	Priority	An item of the Switch Activity dialog box. The default value is set.

Legend:

--: No corresponding element. The definition contents are set automatically.

#

In reality, elements such as the `receive` element, the `reply` element, and the `invoke` element are entered in *activity*.

(d) flow element conversion

The flow element and its lower elements and attributes are converted to the definition contents of the Flow Start Activity. After importing, you can change the definition contents using the Flow Start Activity dialog box.

! Important note

- The link definitions (definitions below the links element in the BPEL file) are not converted.
- The Flow End Activity is set automatically when the elements below the flow element are converted to the Flow Start Activity. In such a case, _end is added to the name of the Flow Start Activity to set the name of the Flow End Activity.

The table below describes the conversion details. The names indicated in Item are the item names of the Flow Activity dialog box.

Table E–32: flow element conversion

Lower elements or attributes of the flow element	Definition contents of the business process	
	Item	Explanation
`name` attribute	Activity name	The activity name is set up. If an activity with the same name is already defined in the business process, *n* (n is an integer value of 1 or more) is added at the end of the activity name.
activity#	*Activity*	Converted to the activity constituting the business process.

#

In reality, elements such as the `receive` element, `reply` element, and `invoke` element are entered in *activity*.

(e) sequence element conversion

The `sequence` element is converted to definition contents of the Sequence Activity. You cannot modify the definition contents.

The `sequence` element does not have any lower attributes to be converted.

With the Cosminexus Service Platform, only the sequence elements described at the positions corresponding to *5.6.17 Sequence Activity* are converted to a business process definition.

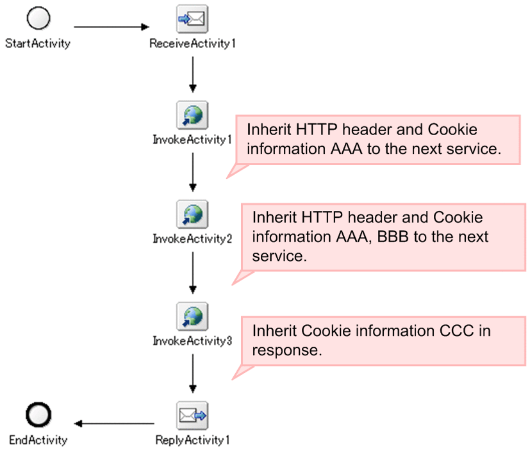
F. Inheriting HTTP header and Cookie information in which service adapter is used

This section describes method to inherit HTTP header and Cookie information in the business process, by using Inherit HTTP header function of the service adapter.

For overview of Inherit HTTP header function, see "*2.2.5 Inherit HTTP header in case of Web service(SOAP communication)*" in the "*Service Platform Function Guide*"

Following section describes the method of defining a business process with example of inheriting HTTP header and Cookie information in the business process shown in the following figure:

Figure F–1: FigureBusiness process used for inheriting Cookie information


StartActivity
ReceiveActivity1
InvokeActivity1
Inherit HTTP header and Cookie information AAA to the next service.
InvokeActivity2
Inherit HTTP header and Cookie information AAA, BBB to the next service.
InvokeActivity3
Inherit Cookie information CCC in response.
EndActivity
ReplyActivity1

Flow for defining a business process is as follows:

1. Defining activities
2. Creating schema of header variable
3. Creating header variables
4. Setting up allocation variables in Invoke service activity
5. Mapping the transformation source variable and transformation destination variable

(1) Defining activities

Schedule an activity in business process definition screen. For details on scheduling and connecting an activity, see "*5.4 Deploying and Linking Activities*".

(2) Creating schema of header variable

Create a schema of header variable for HTTP request and schema of header variable for HTTP response. The created schema use a template file provided by Service Platform, according to the following objectives:

- To inherit as a batch, without editing Cookie information
- To inherit by dividing for each Cookie name, in configurable status

When sending and receiving Cookie of " JSESSIONID" by using SOAP1.1 mode, change c4web.application.app_maintainsession of the client definition file to true. For details on the client definition file, see "*10.3 Setting up the client definition file*" in the "*Application Server SOAP Application Development Guide*"

(a) Schema of header variable for HTTP request

Create a schema of header variable for HTTP request. Tilted part indicates change locations.

- **Definition of the HTTP header request part (for handling Cookie information in a batch)**

  Template file storing destination:

  <Service Platform installation directory>\CSC\schema\soap\soap_http_header_request1.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- All Rights Reserved. Copyright (C) 2012, Hitachi, Ltd. -->
<xsd:schema elementFormDefault="qualified"
  targetNamespace="http://www.hitachi.co.jp/soft/xml/cosminexus/csc/soap/http/
header_request"
  xmlns:hrc="http://www.hitachi.co.jp/soft/xml/cosminexus/csc/soap/http/header_request"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:include schemaLocation="soap_cookie_request1.xsd#1"/>
  <xsd:element name="HTTPHeader_request">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="Cookies" type="hrc:Cookie_types#2" minOccurs="0" maxOccurs="1"/>
        <xsd:element name="HTTPHeader" minOccurs="0" maxOccurs="1">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:any namespace="##any" processContents="skip"/>
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

#1

Change to the include target schema file name that defines Cookie elements.

#2

Change according to the include target name attribute.

- **Definition to be used when handling Cookie information part in a batch(part included in Cookies elements)**

  Template file storage destination:

  <Service Platform installation directory>\CSC\schema\soap\soap_cookie_request1.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- All Rights Reserved. Copyright (C) 2012, Hitachi, Ltd. -->
<xsd:schema elementFormDefault="qualified"
  targetNamespace="http://www.hitachi.co.jp/soft/xml/cosminexus/csc/soap/http/
header_request"
  xmlns:hrc="http://www.hitachi.co.jp/soft/xml/cosminexus/csc/soap/http/header_request"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:complexType name="Cookie_types#">
    <xsd:sequence>
      <xsd:any namespace="##any" processContents="skip" minOccurs="0"
maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```

#

Describe the name attribute of Cookie name to be received.

- **Definition of HTTP header request part(for handling Cookie information individually)**

  Template file storing destination:

  <Service Platform installation directory>\CSC\schema\soap\soap_http_header_request2.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- All Rights Reserved. Copyright (C) 2012, Hitachi, Ltd. -->
<xsd:schema elementFormDefault="qualified"
  targetNamespace="http://www.hitachi.co.jp/soft/xml/cosminexus/csc/soap/http/
header_request"
  xmlns:hrc="http://www.hitachi.co.jp/soft/xml/cosminexus/csc/soap/http/header_request"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:include schemaLocation="soap_cookie_request2.xsd#1"/>
  <xsd:element name="HTTPHeader_request">
    <xsd:complexType>
```

```
      <xsd:sequence>
        <xsd:element name="Cookies" type="hrc:Cookie_types#2" minOccurs="0" maxOccurs="1"/>
        <xsd:element name="HTTPHeader" minOccurs="0" maxOccurs="1">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:any namespace="##any" processContents="skip"/>
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

#1

Change to the include target schema file name that defines Cookie elements.

#2

Change according to include target name attribute.

- **Definition to be used for individually handling Cookie information part(part included in Cookies element)**

  Template file storage destination:

  <Service Platform installation directory>\CSC\schema\soap\soap_cookie_request2.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- All Rights Reserved. Copyright (C) 2012, Hitachi, Ltd. -->
<xsd:schema elementFormDefault="qualified"
  targetNamespace="http://www.hitachi.co.jp/soft/xml/cosminexus/csc/soap/http/
header_request"
  xmlns:hrc="http://www.hitachi.co.jp/soft/xml/cosminexus/csc/soap/http/header_request"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:complexType name="Cookie_types#1">
    <xsd:sequence>
      <xsd:element name="Cookie" type="hrc:Cookie_type" minOccurs="0"
maxOccurs="unbounded"#2 />
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="Cookie_type">
    <xsd:simpleContent>
      <xsd:extension base="xsd:string">
        <xsd:attribute name="name" type="xsd:string" use="optional"/>
        <xsd:attribute name="path" type="xsd:string" use="optional"/>
        <xsd:attribute name="host" type="xsd:string" use="optional"/>
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>
</xsd:schema>
```

#1

Describe name attribute of Cookie name to be received.

#2

Set up the maximum count and minimum count of elements that can be defined, in maxOccurs attribute and minOccurs attribute.

Following table describes the contents of schema for HTTP request header settings:

Table F–1: TableContents of schema for HTTP request header settings

#	Tag name	Type	Occurrence count	Description
1	HTTPHeader_request	-	One time	-
2	+<Cookies> | |	-	0 times or 1 time	Acquires and stores Cookie information stored in the HTTP request header.
3	+<HTTPHeader>	-	0 times or 1 time	Acquires and stores the extension header stored in the HTTP requester header. Ignores the extension header existing in HTTP request header.

Legend:

-: Corresponding description does not exist.

(b) Schema of header variable for HTTP response

Create a schema of header variable for HTTP response. Tilted part indicates change locations.

- **Definition of HTTP header response part(for handling Cookie information in a batch)**

  Template file storing destination:

  <Service Platform installation directory>\CSC\schema\soap\soap_http_header_response1.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- All Rights Reserved. Copyright (C) 2012, Hitachi, Ltd. -->
<xsd:schema elementFormDefault="qualified"
  targetNamespace="http://www.hitachi.co.jp/soft/xml/cosminexus/csc/soap/http/
header_response"
  xmlns:hrc="http://www.hitachi.co.jp/soft/xml/cosminexus/csc/soap/http/header_response"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:include schemaLocation="soap_cookie_response1.xsd#1"/>
  <xsd:element name="HTTPHeader_response">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="Cookies" type="hrc:Cookie_types#2" minOccurs="0" maxOccurs="1"/>
        <xsd:element name="HTTPHeader" minOccurs="0" maxOccurs="1">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:any namespace="##any" processContents="skip"/>
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

#1

Change to the include target schema file name that defines Cookie elements.

#2

Change according to include target name attribute.

- **Definition to be used for handling Cookie information part in a batch(part included in Cookies element)**

  Template file storing destination:

  <Service Platform installation directory>\CSC\schema\soap\soap_cookie_response1.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- All Rights Reserved. Copyright (C) 2012, Hitachi, Ltd. -->
<xsd:schema elementFormDefault="qualified"
  targetNamespace="http://www.hitachi.co.jp/soft/xml/cosminexus/csc/soap/http/
header_response"
  xmlns:hrc="http://www.hitachi.co.jp/soft/xml/cosminexus/csc/soap/http/header_response"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:complexType name="Cookie_types#">
    <xsd:sequence>
      <xsd:any namespace="##any" processContents="skip" minOccurs="0"
maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```

#

Describe the name attribute of Cookie name to be received.

- **Definition of HTTP header response part(for individually handling Cookie information)**

  Template file storing destination:

  <Service Platform installation directory>\CSC\schema\soap\soap_http_header_response2.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- All Rights Reserved. Copyright (C) 2012, Hitachi, Ltd. -->
<xsd:schema elementFormDefault="qualified"
  targetNamespace="http://www.hitachi.co.jp/soft/xml/cosminexus/csc/soap/http/
header_response"
  xmlns:hrc="http://www.hitachi.co.jp/soft/xml/cosminexus/csc/soap/http/header_response"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:include schemaLocation="soap_cookie_response2.xsd#1"/>
  <xsd:element name="HTTPHeader_response">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="Cookies" type="hrc:Cookie_types#2" minOccurs="0" maxOccurs="1"/>
        <xsd:element name="HTTPHeader" minOccurs="0" maxOccurs="1">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:any namespace="##any" processContents="skip"/>
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

#1

Change to the include target schema file name that defines Cookie elements.

#2

Change according to the include target name attribute.

- **Definition to be used for individually handling the Cookie information part(part included in Cookies element)**

Template file storing destination:

<Service Platform installation directory>\CSC\schema\soap\soap_cookie_response2.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- All Rights Reserved. Copyright (C) 2012, Hitachi, Ltd. -->
<xsd:schema elementFormDefault="qualified"
  targetNamespace="http://www.hitachi.co.jp/soft/xml/cosminexus/csc/soap/http/
header_response"
  xmlns:hrc="http://www.hitachi.co.jp/soft/xml/cosminexus/csc/soap/http/header_response"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:complexType name="Cookie_types#1">
  <xsd:sequence>
    <xsd:element name="Cookie" type="hrc:Cookie_type" minOccurs="0"
maxOccurs="unbounded"#2 />
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="Cookie_type">
    <xsd:simpleContent>
      <xsd:extension base="xsd:string">
        <xsd:attribute name="name" type="xsd:string" use="optional"/>
        <xsd:attribute name="path" type="xsd:string" use="optional"/>
        <xsd:attribute name="host" type="xsd:string" use="optional"/>
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>
</xsd:schema>
```

#1

Describe the name attribute of Cookie name to be received.

#2

Set up the maximum count and minimum count of elements that you can define, in maxOccurs attribute and minOccurs attribute.

Following table describes the contents of schema for HTTP response header settings

Table F–2: TableContents of schema for HTTP response header settings

#	Tag name	Type	Occurrence count	Description
1	HTTPHeader_response	-	1 time	-
2	+<Cookies> | |	-	0 times or 1 time	Acquires and stores Cookie information stored in HTTP response header. Set-Cookie attribute in the HTTP response header is ignored.

#	Tag name	Type	Occurrence count	Description
2	|	-	0 times or 1 time	Acquires and stores Cookie information stored in HTTP response header. Set-Cookie attribute in the HTTP response header is ignored.
3	+<HTTPHeader>	-	0 times or 1 time	Acquires and stores the extension header stored in HTTP response header. Extension header existing in the HTTP response header is deleted.

Legend:

-: Corresponding description does not exist.

(3) Creating header variables

Create header variable for HTTP request and header variable for HTTP response. Procedure for creating is as follows:

1. Double click **Variable/correlation set** icon on the canvas on the business process definition screen.
   Variable/correlation set list dialog is displayed. For details on the variable/correlation set list dialog, see "*1.4.1 Variable/correlation set list dialog*" in the "*Service Platform Reference Guide*".
2. Select **Variables list** from the list.
3. Enter any name in the variable name.
4. From **Type** drop down list, select **XML**.
5. Click **...** button to set up schema of header variable for HTTP request created in "*(2) Creating schema of header variable*".
6. Insert check mark in the [Specify part] check box.
7. Click **Add row** button and specify part name, specification expression and type.
8. Click **Add** button.
   Added variables are displayed in the variables list.
9. Create header variable for HTTP response, by the procedure similar to step 3.~8..
10. Click **OK** button.

(4) Setting up allocation variables in Invoke service activity

Set up variables for created request message and response message, in the header allocation variables of Invoke service activity.

1. Double click the Invoke service activity "Invoke service 1" in the business process definition screen.
   Invoke service activity dialog is displayed. For details on the Invoke service activity dialog, see "*1.4.9 Invoke service activity dialog*" in the "*Service Platform Reference Guide*".
2. Click **Settings** button of **Header allocation variables** of **Request message**.
   Header allocation variable dialog for request message is displayed.
3. Click **Add** button.
   Line for setting up the allocation variables is displayed.
4. Click **Allocation variables** column and select the allocation variables for the request corresponding to Invoke service 1.
5. Click **Root element** column and select the root element.
6. Check the namespace of allocation variables displayed in **Namespace** column.
7. Click **OK** button.
8. Set up allocation variables for request, in header allocation variables of **Response message**, with the procedure similar to step 2.~7.
9. Set up allocation variable for request and response also in Invoke service activity "Invoke service 2" and "Invoke service 3", with the procedure similar to step1.~8.

(5) Mapping the transformation source variable and transformation destination variable

Perform mapping of transformation source schema and transformation destination schema in the data transformation definition screen.

Perform mapping of allocation variable set in response message of Invoke service 1 and allocation variable set in the request message of Invoke service 2, for inheriting the Cookie information between Invoke service 1 and Invoke service 2.

Similarly, perform mapping of allocation variable set in the response message in Invoke service 2 and allocation variable set in the request message of Invoke service 3, for inheriting Cookie information between Invoke service 2 and Invoke service 3.

Also, perform mapping of allocation variable set in response message of Invoke service 3 and allocation variable set in response 1, for inheriting the Cookie information between Invoke service 3 and response 1.

Procedure for mapping is as follows:

1. Right click the data transformation activity "Data transformation 1" and select **Start mapping definition**.
2. Perform mapping of the node of transformation source variable and node of transformation destination variable.
   Edit the inherited data by scheduling each function.
3. Perform mapping from Invoke service 2 to invoke service 3 and Invoke service 3 to response 1, with the procedure similar to step1.~2.
4. Right click the appropriate locations in transformation source schema tree viewer, mapping viewer or transformation destination schema tree viewer, in the data transformation definition screen and select **Validate**.
   Validation is executed. If validation result has no problem, you can execute the business process.

G. Emulating the Service Requester

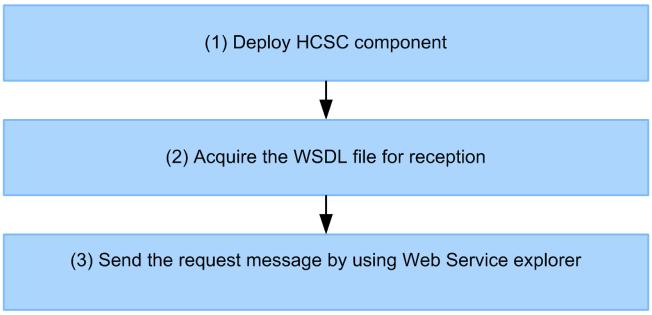
When you test the HCSC components, you can use the Web Service Explorer included in Eclipse as the service requester. This appendix describes how to emulate the service requester when the Web Service Explorer is used for testing.

Note that the Web Service Explorer uses SOAP (HTTP) as the protocol; therefore, the receptions that can be sent are standard reception (synchronous reception (Web service)) and user-defined reception. For details about the Web Service Explorer, see Eclipse Help.

G.1 Flow of service requester emulation

This section describes the flow of service requester emulation using the Web Service Explorer. The examples of emulation are described in *Appendix G.2 How to emulate the service requester* in accordance with this flow.

Figure G–1: Flow of service requester emulation


(1) Deploy HCSC component
(2) Acquire the WSDL file for reception
(3) Send the request message by using Web Service explorer

(1) Deploying the HCSC components

Deploy the HCSC components you want to test in such a way that the request messages can be sent from the Web Service Explorer.

(2) Acquiring the WSDL file for reception

Acquire the WSDL file of the reception to be sent and copy the file into any Eclipse project. For details about how to acquire the standard reception WSDL file, see *8.2.2 Acquiring the WSDL*.

For the user-defined reception, copy the WSDL file used for creating the user-defined reception. However, the service location values are corrected in the user-defined reception WSDL. For details about setting up the service location values, see *8.7.2 Editing a WSDL*.

(3) Sending the request messages using Web Service and Explorer

Start the Web Service Explorer and send the request messages. To send request messages from the Web Service Explorer, select the operation for sending the messages and set up the required parameters.

G.2 How to emulate the service requester

This section describes the procedure of testing with the Web Service Explorer using the Product Arrangement Sample Program that comes with the Cosminexus Service Platform as an example. For details about the sample program, see the manual *Cosminexus Service Platform Sample Program Guide*.

(1) Deploying the HCSC components

Deploy the Web services and the HCSC components of the Product Arrangement Sample Program in such a way that the request messages can be sent from the Web Service Explorer.

(2) Acquiring the reception WSDL file

The user-defined reception is used in the Product Arrangement Sample Program. This WSDL file for the user-defined reception is available in the following location:

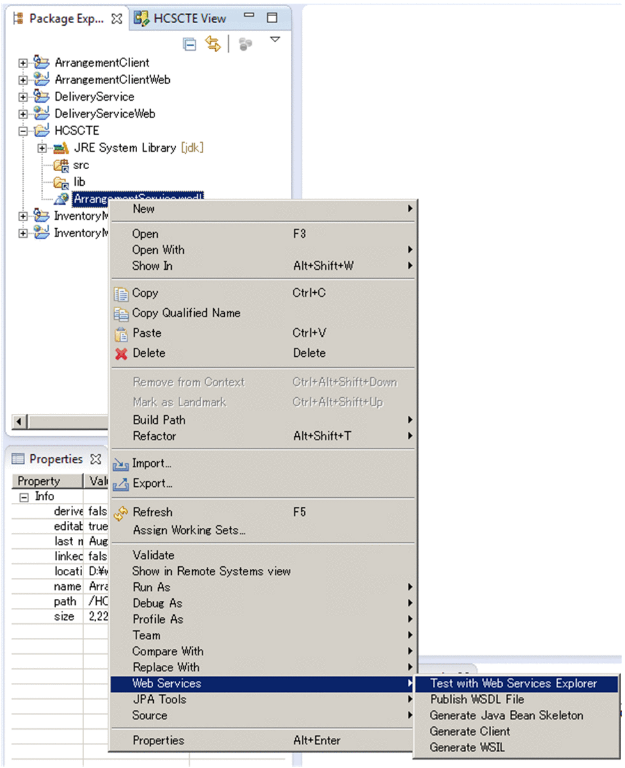
uCosminexus-Service-Architect-installation-directory`\CSCTE\Samples\ Product Arrangement\Service \WSDL\ArrangementService.wsdl`

Copy this file into any Eclipse project.

(3) Sending the request messages using the Web Service Explorer

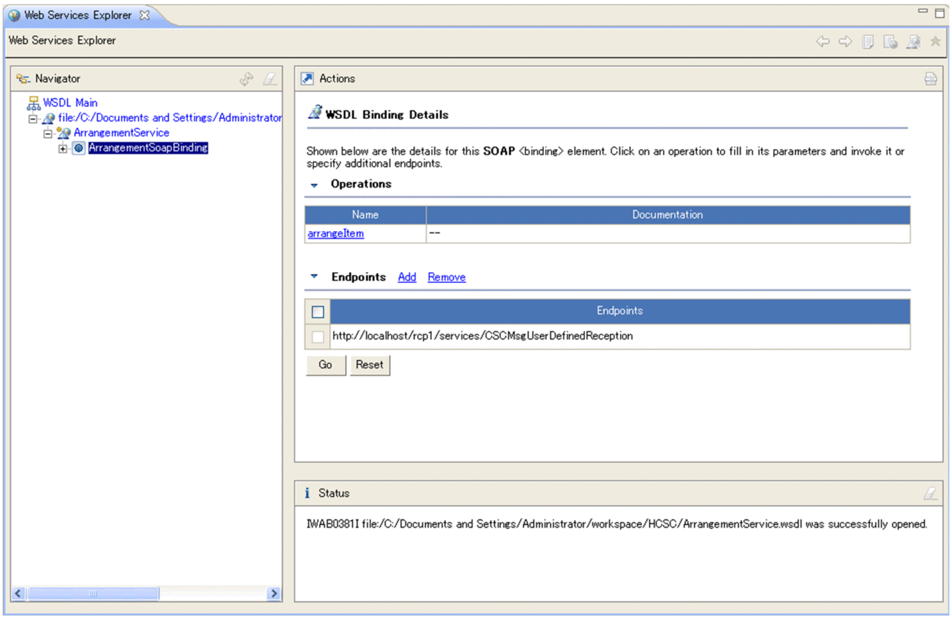
The request message is sent as follows from the Web Service Explorer:

1. Right click the copied WSDL file and choose **Web Services** and then **Testing in Web Services Explorer**.



The Web Service Explorer starts and the window for selecting the operations appears.

2. Select the operation you want to send.
   Select `arrangeItem`.



WSDL Binding Details

Shown below are the details for this **SOAP** <binding> element. Click on an operation to fill in its parameters and invoke it or specify additional endpoints.

Operations

Name	Documentation
arrangeItem	--

Endpoints Add Remove

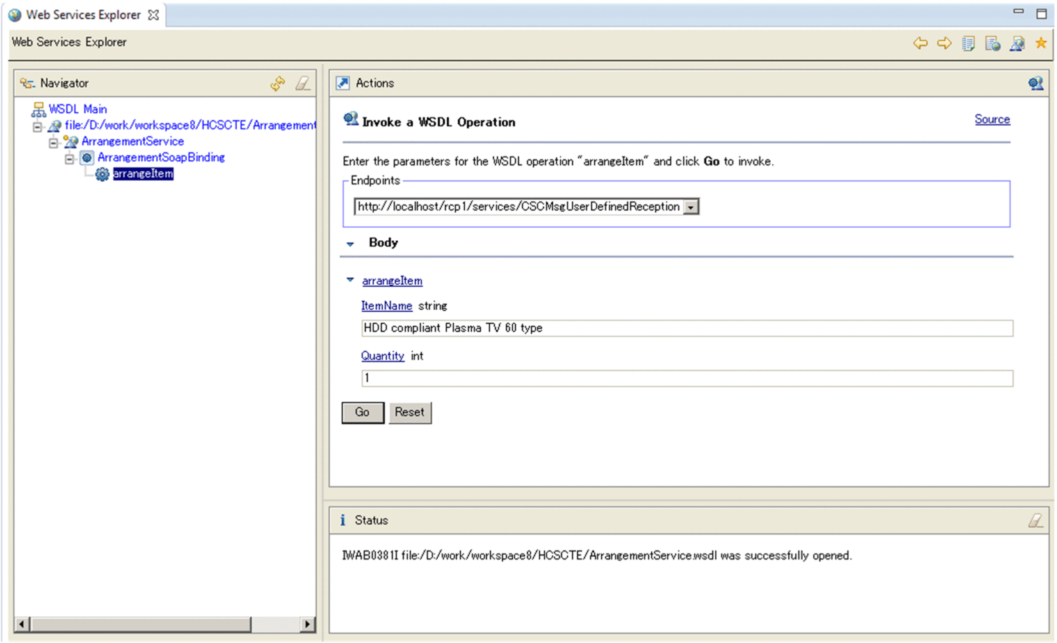
□	Endpoints
□	http://localhost/rcp1/services/CSCMsgUserDefinedReception

IWAB0381I file:/C:/Documents and Settings/Administrator/workspace/HCSC/ArrangementService.wsdl was successfully opened.

3. Set up the parameters for the `arrangeItem` operation.

   Set up the following parameters:

   - `ItemName`: HDD compliant Plasma TV?60 type (? is a one-byte space)
   - `Quantity`: 1



Invoke a WSDL Operation

Enter the parameters for the WSDL operation "arrangeItem" and click **Go** to invoke.

Endpoints: http://localhost/rcp1/services/CSCMsgUserDefinedReception

Body

arrangeItem

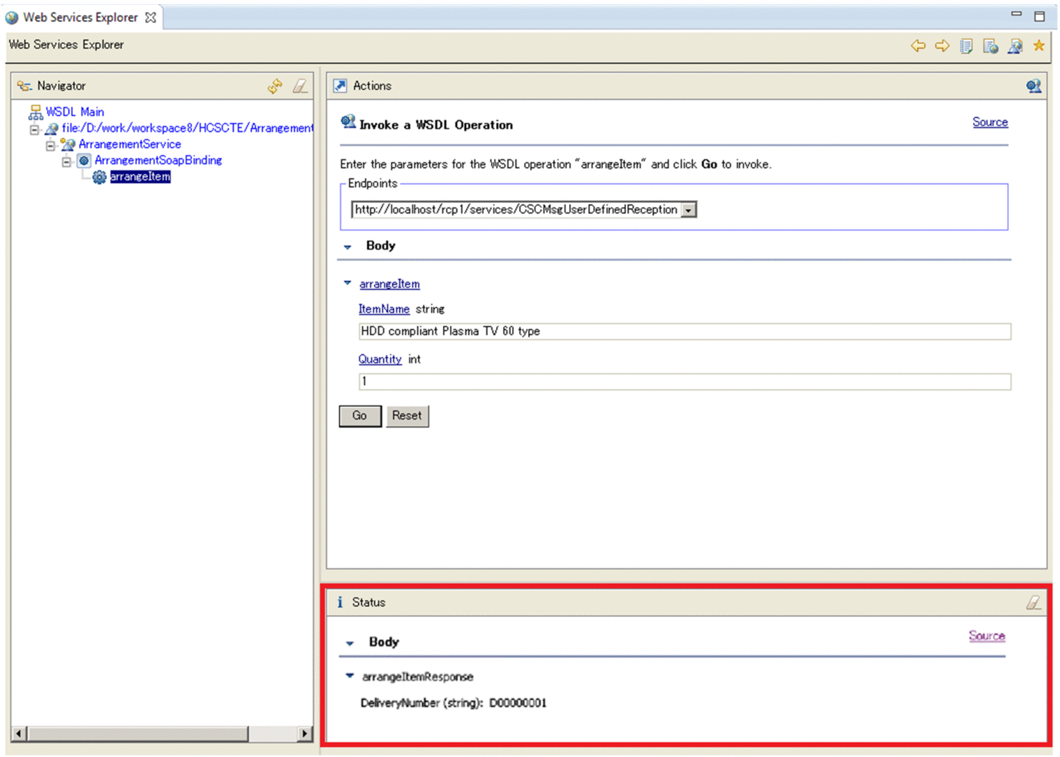
ItemName string: HDD compliant Plasma TV 60 type

Quantity int: 1

IWAB0381I file:/D:/work/workspace8/HCSCTE/ArrangementService.wsdl was successfully opened.

4. Click the **Go** button.

   The sending result (response message or error) appears in the lower right pane.



Invoke a WSDL Operation

Enter the parameters for the WSDL operation "arrangeItem" and click **Go** to invoke.

Endpoints: http://localhost/rcp1/services/CSCMsgUserDefinedReception

arrangeItem

ItemName string: HDD compliant Plasma TV 60 type

Quantity int: 1

arrangeItemResponse

DeliveryNumber (string): D00000001

H. Component common UOC

The property file describes the association between component and common UOC class.

H.1 Property file of component common UOC class

File name

cscmsg_uoc.properties

Description format

Description format is property file format of J2SE..

(UOC class invoking source)=(class path)

Description example

```
ADP1=sample.AdpUocClass
ADP2=sample.AdpUocClass
RCP1=sample.package.RcpUocClass
adapter.standard.soap=sample.SoapUocClass
adapter.custom.ftp=sample.FtpAdpUocClass
reception.urecp.ftp=sample.FtpRcpUocClass
ADP3=
```

H.2 Method for specifying the component common UOC class

Method for specifying the component common UOC class is as follows:

- When you specify the class to be invoked only with specific components

  Specify UOC class for individual components.

  Set reception ID of reception and service ID of service adapter in (UOC class invoking source). In this case, even if you have specified UOC class in the reception type or adapter type, only the UOC class specified in individual component is invoked.

- When you want to invoke the same UOC class in the specified component type

  Specify UOC class in the reception type or adapter type.

  Specify invoking source type ID in the invoking source of UOC class.

  Following table describes the invoking source type ID of reception.

Table H–1: TableInvoking source type ID list (reception)

Reception		Invoking source type ID
Standard reception	Standard reception (Web service)	reception.standard.soap
	Standard reception (SessionBean)	reception.standard.ejb
	Standard reception(MDB(WS-R))	reception.standard.wsr
	Standard reception (MDB(DB queue))	reception.standard.dbq
User-defined reception	SOAP reception	reception.urecp.soap
	TP1/RPC reception	reception.urecp.tp1
	FTP reception	reception.urecp.ftp
	HTTP reception	reception.urecp.http
	Message Queue reception	reception.urecp.mq
	Custom reception	reception.urecp.rcpfw

Following table describes the invoking source type ID of the service adapter.

Table H–2: TableInvoking source type ID list (service adapter)

Service adapter	Invoking source type ID
SOAP adapter	adapter.soap
SessionBean adapter	adapter.ejb
MDB (WS-R) adapter	adapter.wsr
MDB (DB queue) adapter	adapter.dbq
DB adapter	adapter.db
TP1 adapter	adapter.tp1
File adapter	adapter.ff
Object Access adapter	adapter.oa
Message Queue adapter	adapter.mq
FTP adapter	adapter.ftp
File operation adapter	adapter.fop
Mail adapter	adapter.mail
HTTP adapter	adapter.http
General custom adapter	adapter.custom

- When UOC class is specified in reception type or adapter type, but UOC class is not invoked from a specific service

  Makes the class path of the specific service as blank. UOC class is not invoked from that service.

  Example) ADP3=

Operations regarding description of property file for component common UOC are as follows:

- You must use ISO-8859-1 character coding.
- Data till linefeed is considered as value.
- Lines starting with # are considered as comments.
- Leading and trailing spaces of class path are ignored.
- You cannot add character string such as comment, after the class path. If you add, entire text including the added comment is parsed as the class path and this may lead to occurrence of exception such as class is not found.
- If you specify multiple class paths for 1 invoking source, only one of the class paths is activated.
- When you set a single class path for multiple invoking sources, single class instance is used in the concerned invoking source.
- UOC is not invoked for the component for which class path of UOC has not been set.
- If class corresponding to the already set class path does not exist, UOC is not invoked.
- As component dependency existence is not checked, instances of UOC class are created, even when you specify service ID or reception ID of non-existing service adapter.
- Changes in property file are reflected after you restart HCSC server.

H.3 API for UOC class

An interface exist for each timing of invoking UOC class and UOC class is executed when the interface is added. You can map to multiple invoking timings, by adding multiple interfaces.

Following interfaces are executed in UOC class:

(1) Interfaces

When exception occurs in UOC class, throws that exception by wrapping in CSCUocSystemException.

(a) UOC class common interface

Package

jp.co.Hitachi.soft.csc.msg.uoc

Interface name

ComponentCommonUoc

Description

This is a common interface for defining constants to be used in component common UOC.

Constants

Following table describes constants.

Constant	Description
COMPO_TYPE	Key for acquiring the code for discriminating wither it is service adapter or reception
COMPO_KIND	Key for acquiring the component type
COMPO_ID	Key for acquiring the component ID
TELEGRAM_DATA	Key for acquiring API for message acquisition
MONITOR_SEND_DATA	Key for acquiring Map for setting the data to be sent to monitoring from the parameter
DESTINATION_DATA	Key for acquiring the address information# of the invoked service

#

You can acquire the information only in case of user-defined reception (excluding SOAP reception).

For details on how to use the constant, see "*4.8.1 Overview of component common UOC*" in the "*Service Platform Function Guide*".

(b) Interface for the request process

Package

jp.co.Hitachi.soft.csc.msg.uoc

Interface name

ComponentRequestUoc

Description

This is the interface of UOC class invoked at the time of request processing.

#request

Description related to the operations of #request is as follows:

Argument

Map<String,Object>
(object in which data passed to UOC class is consolidated)

Return value

None

Exception

CSCUocSystemException

(c) Interface for response processing

Package

jp.co.Hitachi.soft.csc.msg.uoc

Interface name

ComponentResponseUoc

Description

This is the interface of UOC class invoked at the time of response processing.

#response

Description related to operations of #response is as follows:

Argument

Map<String,Object>

(Object in which data to be passed to UOC class is consolidated)

Return value

None

Exception

CSCUocSystemException

(2) Exception class

Exception is thrown from UOC class by wrapping in the following class:

Package

jp.co.Hitachi.soft.csc.msg.uoc

Exception class name

CSCUocSystemException

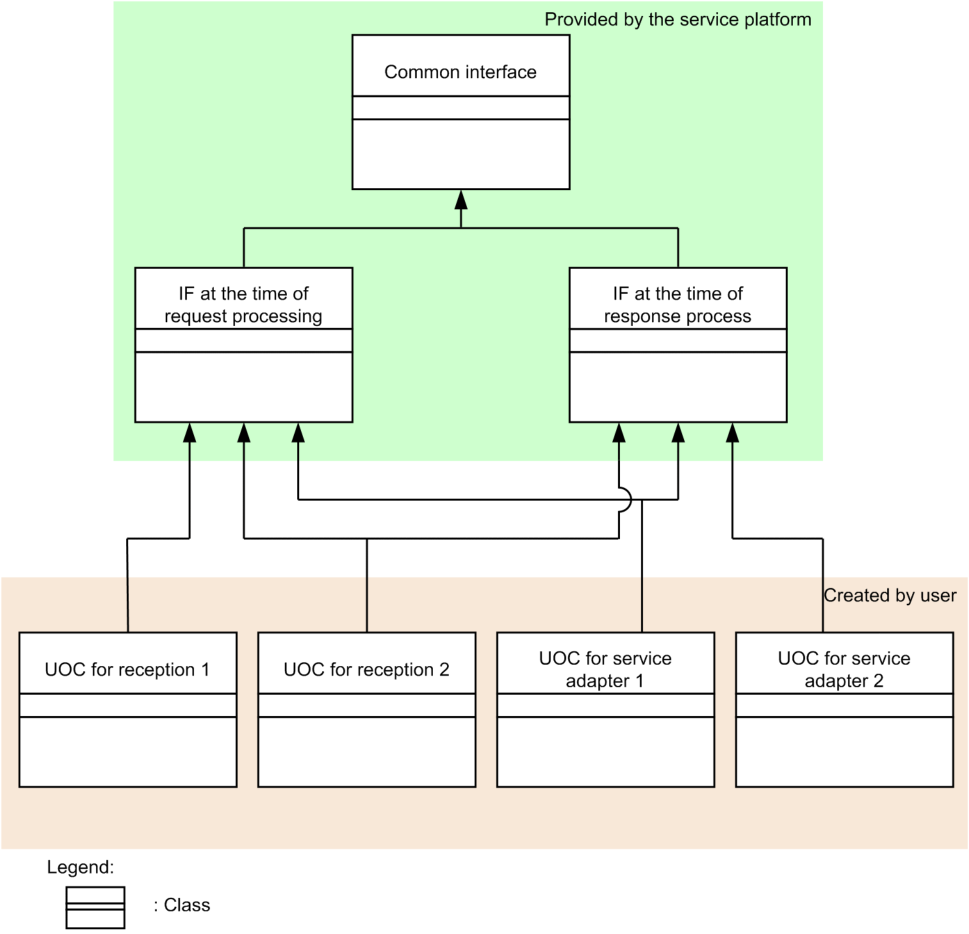
Constructor

```
public CSCUocSystemException()
public CSCUocSystemException(String errorMessage )
```

(3) Class structure example

Following figure shows the example of class structure.

Figure H–1: Figure Class structure example


Provided by the service platform
Common interface
IF at the time of request processing
IF at the time of response process
Created by user
UOC for reception 1
UOC for reception 2
UOC for service adapter 1
UOC for service adapter 2
Legend:
: Class

H.4 API for message acquisition

Details of APIL for acquiring message set through parameter in the UOC class are as follows:

Package

jp.co.Hitachi.soft.csc.msg.uoc

Class name

CSCMsgTelegramManager

Constants

Following table describes constants:

Constant	Description
static final int NONE = 0	Indicates that the body message does not exist.
static final int TYPE_XML = 1	Indicates that the body message is XML.
static final int TYPE_BINARY = 2	Indicates that the body message is binary.

#getBody

Description related to operations of #getBody is as follows:

Argument

None

Return value

byte[](body message)

Description

Acquires the body message.

If body message does not exist, null is returned.

#getHeader

Description related to operations of #getHeader is as follows:

Argument

String(namespace), String(element name)

Return value

byte[](header message)

Description

Acquires header message.

If header message has not been set, null is returned.

#setBody

Description related to operations of #setBody is as follows:

Argument

byte[](body message)

Return value

None

Description

Updates the body message.

#setHeader

Description related to operations of #setHeader is as follows:

Argument

String(namespace), String(element name), byte[](header message)

Return value

None

Description

Updates header message.

#getTelegramType

Description related to operations of #getTelegramType is as follows:

Argument

None

Return value

int (message type)

0:none(message does not exist)

1:XML

2:binary

Description

Acquires types of the body message.

H.5 Method of specifying jar file of UOC class

For jar file in which UOC class is consolidated, add the storage location in usrconf.cfg.

Specification example

```
add.class.path=C:\uocjar\uocClass2.jar
```

H.6 Notes

This section describes nodes when designing UOC class.

- As UOC class operates as extension of the container extended library of J2EE server, there are limitations of the available functions.

  For details, see "*14.6 Limitations when using container extended library and server start/stop hook function*" in the "*Application Server Common Container Functionality Guide*".

  UOC class is read in the system class loader and hence if you can read and write the external file with full path specification, if not in scope of limitations, mentioned above.
- When you crate UOC class, specify public in access settings modifier of the class.

  When it is not public(not writing the access settings modifier=package private), IllegalAccessException occurs and you cannot send UOC data.
- Perform process considering the multi-thread access.

  UOC class is invoked from multiple threads and hence you must implement thread safe. You must release the appropriate resources as well, considering the cases of insufficient memory.
- In UOC class of standard reception (SessionBean), you must consider the input of null in the message at the time of request processing.

  When you re-execute the business process, standard reception (SessionBean) is invoked, with message is null status.
- You must add cscmsg.jar to class path, when compiling. Interface used is included in cscmsg.jar.

I. Character code conversion using character code conversion UOC

When using the character code that is not supported in Service Platform, you can handle the non-supported character code by creating the character code conversion UOC for processing the concerned character code.

! Important note

You can define only 1 character code conversion UOC in HCSC server.

This section describes about the character code conversion UOC.

I.1 Developing jar file of character code conversion UOC

Develop the character code conversion UOC by using a jar file provided by Service Platform.

Interface and exception class defined in the jar file required for character code conversion UOC is stored in the following location:

```
<Service Platform installation directory>\CSC\lib\cscdt_uoc.jar
```

! Important note

- Store the jar file of the developed character code conversion UOC, in any directory. When you compile the character code conversion UOC, include cscdt_uoc.jar in the class path.
- Resources secured at the time of executing character code conversion UOC are retained even after the end of the process and when this poses a high load on the entire system, OutOfMemoryError might occur due to reason like Java heap insufficiency. Therefore, you must execute process for releasing appropriate resources when OutOfMemoryError occurs or error processing such as rollback.

I.2 Settings for using the character code conversion UOC

This section describes the settings required for using character code conversion UOC.

(1) Definition of property (self-defined file)

The character code conversion UOC can divide the character code conversion processes by acquiring the custom reception or self-defined file. Always specify name of self-defined file as "csc_owncodeconvert.properties".

Specify the self-defined file in format such as "<key value>= <specified value>".

Specify any value as the key value and specified value.

(2) Settings for using jar file of the character code conversion UOC

Following settings are required for using the character code conversion UOC created in development environment, in the execution environment.

- Add a class path in option definition file (usrconf.cfg) for J2EE server
- Add a class name in the system properties file (usrconf.properties)

(a) Adding a class path

Procedure for adding a class path in option definition file (usrconf.cfg) for J2EE server, is as follows:

1. Select **Logical server environment settings** from the management portal.
2. From Server view, select [Logical J2EE server]-[J2EE server]-[<Logical server name>].
3. Specify a class path in **Extension parameter** of **Container** tab.
   Specify the class path in following format:

```
add.class.path=<path of jar file>
```

<Path of jar file>

Specify jar file of character code conversion UOC, in absolute path.

(b) Registering a jar file

Procedure for registering a class name in user property file (usrconf.properties) for J2EE server is as follows:

1. Select Logical server environment settings] from the management portal.
2. From the Server View, select [Logical J2EE server]-[J2EE server]-[<Logical server name>].
3. Specify class name (including package name) in the [Properties] of [JVM] tab.
   Specify class name (including package name) in the following format.

```
csc.dt.ownCodeConverter.className=<class name of character code conversion UOC>
```

<Class name of character code conversion UOC>

Specify class of character code conversion UOC with the fully qualified name.

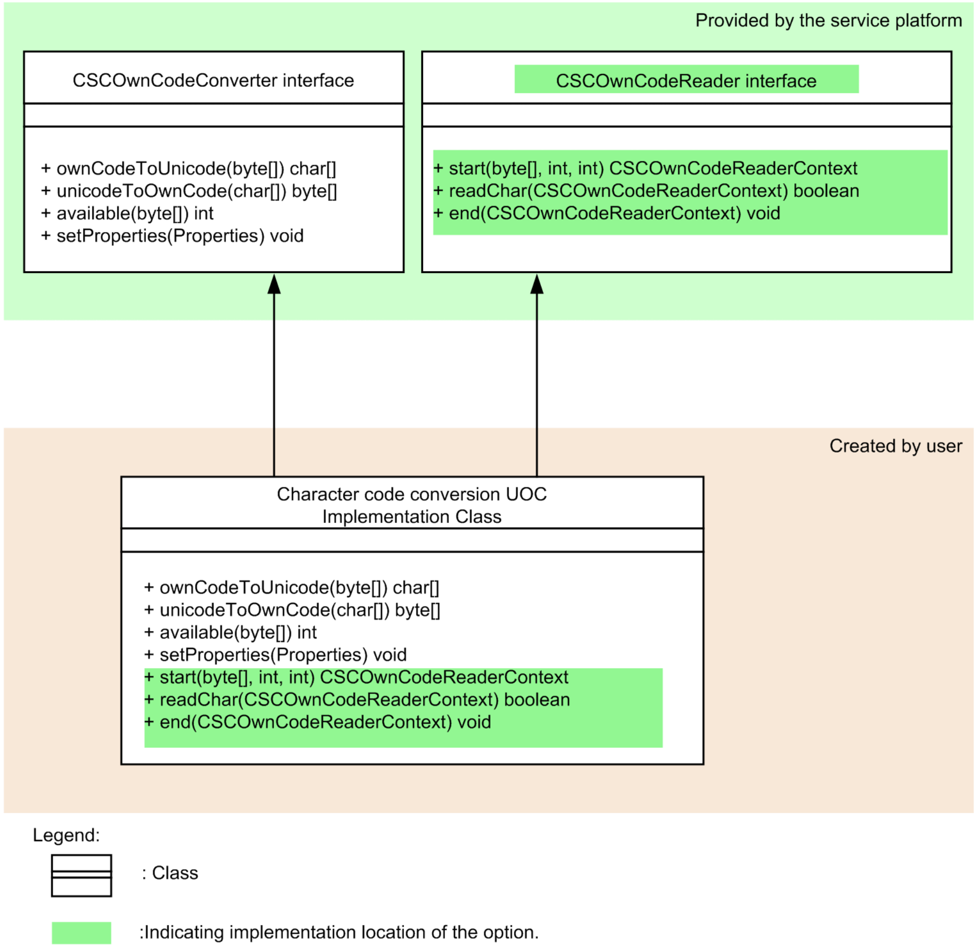
I.3 CSCOwnCodeConverter interface

(1) Interface

CSCOwnCodeConverter interface is as follows:

Create the implementation class of following interface, when developing the character code conversion UOC.

Figure I–1: Figurecharacter code conversion UOC class


Provided by the service platform
CSCOwnCodeConverter interface
+ ownCodeToUnicode(byte[]) char[]
+ unicodeToOwnCode(char[]) byte[]
+ available(byte[]) int
+ setProperties(Properties) void
CSCOwnCodeReader interface
+ start(byte[], int, int) CSCOwnCodeReaderContext
+ readChar(CSCOwnCodeReaderContext) boolean
+ end(CSCOwnCodeReaderContext) void
Created by user
Character code conversion UOC
Implementation Class
+ ownCodeToUnicode(byte[]) char[]
+ unicodeToOwnCode(char[]) byte[]
+ available(byte[]) int
+ setProperties(Properties) void
+ start(byte[], int, int) CSCOwnCodeReaderContext
+ readChar(CSCOwnCodeReaderContext) boolean
+ end(CSCOwnCodeReaderContext) void
Legend:
: Class
:Indicating implementation location of the option.

Interface name

CSCOwnCodeConverter interface

Description

This is interface for implementing character code conversion UOC.

Package name of CSCOwnCodeConverter is jp.co.Hitachi.soft.csc.dt.uoc.CSCOwnCodeConverter.

Format

```
public class OwnCodeConverter implements CSCOwnCodeConverter
{
  public void setProperties(final Properties properties)
           throws CSCOwnCodeConverterException;
  public char[] ownCodeToUnicode(final byte[] inBuffer)
           throws CSCOwnCodeConverterException;
  public byte[] unicodeToOwnCode(final char[] inBuffer)
           throws CSCOwnCodeConverterException;
  public int available(final byte[] inBuffer)
           throws CSCOwnCodeConverterException;
}
```

Methods

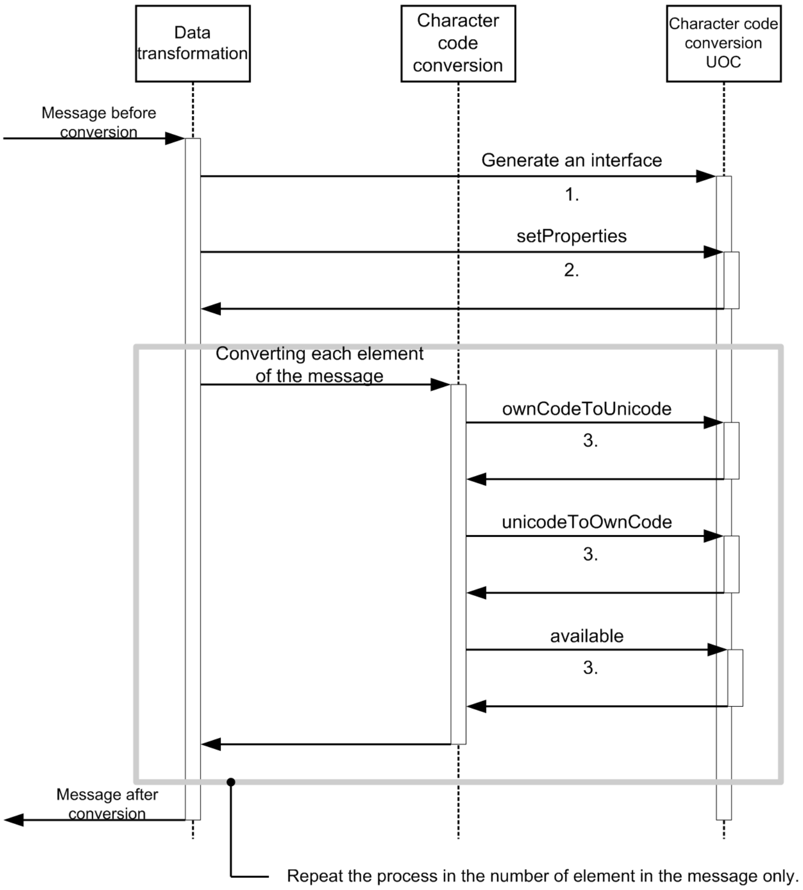
Following table describes the methods of CSCOwnCodeConverter interface:

Method name	Description
setProperties method	This is a method for passing definition contents of self-defined file to the character code conversion UOC.
ownCodeToUnicode method	This is method for converting the character string of self-defined character code to Unicode.

Method name	Description
unicodeToOwnCode method	This is the method for converting the character string of Unicode(UTF-16) to self-defined character code.
Available method	This is the method of returning bytes count of character string that can be converted at the time of character code conversion.

Following figure shows the order of invoking each method from the character code conversion UOC.

Figure I–2: FigureOrder of invoking each method from character code conversion UOC


Data transformation
Character code conversion
Character code conversion UOC
Message before conversion
Generate an interface
1.
setProperties
2.
Converting each element of the message
ownCodeToUnicode
3.
unicodeToOwnCode
3.
available
3.
Message after conversion
Repeat the process in the number of element in the message only.

1. Generating an instance

   After receiving the message before conversion, generate an instance of character code conversion UOC, depending on data transformation.

2. setProperties method

   Pass the definition contents of the self-defined file to character code conversion UOC. This method is invoked only once at the time of starting character code conversion UOC.

3. ownCodeToUnicode method, unicodeToOwnCode method, available ,method

   The respective methods are invoked from character code conversion at the time of executing character code conversion UOC. Method execution order depends on the message format.

   - ownCodeToUnicode method

     This method is used to convert character string of self-defined character code to Unicode.

   - unicodeToOwnCode method

     This method is used to convert the character string of Unicode(UTF-16) to the self-defined character code.

   - available method

     This method is used to return the bytes count of character string that can be converted at the time of character code conversion.

(a) setProperties method

Description

Passes the definition contents of the self-defined file to character code conversion UOC.

Format

```
public void setProperties(final Properties properties)
         throws CSCOwnCodeConverterException;
```

Parameter

properties:

Definition contents of self-defined file are stored.

Notes

In the following cases, arguments of setProperties method always become null. Implement such that error does not occur for character code conversion UOC, even when the argument of setProperties method are null.

- When self-defined file does not exist (not defined)
- When executing character code conversion UOC from reception other than custom reception (reception for which self-defined file cannot be defined)
- When executing character code conversion UOC from adapter for which self-defined file cannot be defined

Exception

CSCOwnCodeConverterException:

Entire data transformation process is aborted as an error occurred during the character code conversion process.

Return value

None

(b) ownCodeToUnicode method

Description

This method converts the character string of self-defined character code to Unicode.

Format

```
public char[] ownCodeToUnicode(final byte[] inBuffer)
         throws CSCOwnCodeConverterException;
```

Parameter

inBuffer:

Buffer of the byte array that stores the character string of self-defined character code is stored. As this buffer is read-only, you cannot edit the same.

Exception

CSCOwnCodeConverterException:

Entire data transformation process was aborted as error occurred during the character code conversion process.

Return value

Returns the buffer of character array that stores the character string converted to Unicode.

(c) unicodeToOwnCode method

Description

Coverts the character string in Unicode(UTF-16) to self-defined character code.

Format

```
public byte[] unicodeToOwnCode(final char[] inBuffer)
         throws CSCOwnCodeConverterException;
```

Parameter

inBuffer:

Buffer of character array that stores the character string of Unicode is stored.

Since this buffer is read-only you cannot edit the same.

Exception

CSCOwnCodeConverterException:

Entire data transformation process was aborted, as error occurred during the character code conversion process.

Return value

Returns the byte array buffer that stores the character string converted to the self-defined character code.

(d) available method

Description

Returns the byte count of character string that can be converted at the time of character code conversion.

Format

```
public int available(final byte[] inBuffer)
         throws CSCOwnCodeConverterException;
```

Parameter

inBuffer:

Buffer of the byte array that stores the character string of self-defined character code is stored. Since this buffer is read-only, you cannot edit the same.

Exception

CSCOwnCodeConverterException:

Entire data transformation process was aborted, as error occurred during the character code conversion process.

Return value

Stores the byte count of character string that can be successfully converted. If any character cannot be converted, stores the byte count till character just before the concerned character.

(2) Exception class

Exception class that occurs at the time of developing character code conversion UOC is as follows:

Class name

CSCOwnCodeConverterException class

Description

This exception is sent when error occurs during the character code conversion process.

When this exception occurs, entire data transformation process is aborted.

(3) Implementation example

Implementation example of CSCOwnCodeConverter interface is as follows:

```
public class OwnConvertUoc implements CSCOwnCodeConverter {

   // Code conversion option
   HJCOption option = null;
   // Code conversion result
   HJCResult result = null;
   // Encode
   String encode = null;

   // Constructor
   public void OwnConvertUoc() {
      result = new HJCResult();
      option = new HJCOption();
   }

   // Receive self-defined file in property format of Java
   public void setProperties(final Properties properties)
```

```
                  throws CSCOwnCodeConverterException {

      if ( properties == null ) {
         // self-defined file has not been registered
         // self-defined file is executed from the non-defined reception/
service adapter
         // Since there is a possibility of occurrence, it is not considered
as error
         encode = "";
         return;
      }

      encode = properties.getProperty("encode");

      if ( encode == null ) {
         // When key does not exist
         String message = "Character code is not specified.";
         throw new CSCOwnCodeConverterException(message);
      }
   }

   // Convert the self-defined code to Unicode
   public char[] ownCodeToUnicode(final byte[] inBuffer)
                    throws CSCOwnCodeConverterException {

      if ( encode.equals("SJIS") ) {
         // Convert from SJIS(MS932) to Unicode
         // Use code conversion
         HJCString inStr = new HJCString( inBuffer );
         try {
            HJCConverters.cs_ms932tounicode(inStr, result, option);
         } catch ( Exception e ){
            throw new CSCOwnCodeConverterException(e);
         }
         return result.getStrResult().toString().toCharArray();
      }
      else if ( encode.equals("KEIS") ) {
         // Convert from KEIS to Unicode
         // Use code conversion
         HJCString inStr = new HJCString( inBuffer );
         try {
            HJCConverters.cs_keistounicode(inStr, result, option);
         } catch ( Exception e ){
            throw new CSCOwnCodeConverterException(e);
         }
         return result.getStrResult().toString().toCharArray();
      }
      else {
         // Character code not considered as target
         String message = "This is a character code not considered as target.";
         throw new CSCOwnCodeConverterException(message);
      }
   }

   // Convert Unicode to self-defined code
   public byte[] unicodeToOwnCode(final char[] inBuffer)
                  throws CSCOwnCodeConverterException {
      // (omitted)
   }

   // Returns the byte count, that can be successfully converted
   public int available(final byte[] inBuffer)
                  throws CSCOwnCodeConverterException {
      // (omitted)
   }
}
```

! Important note

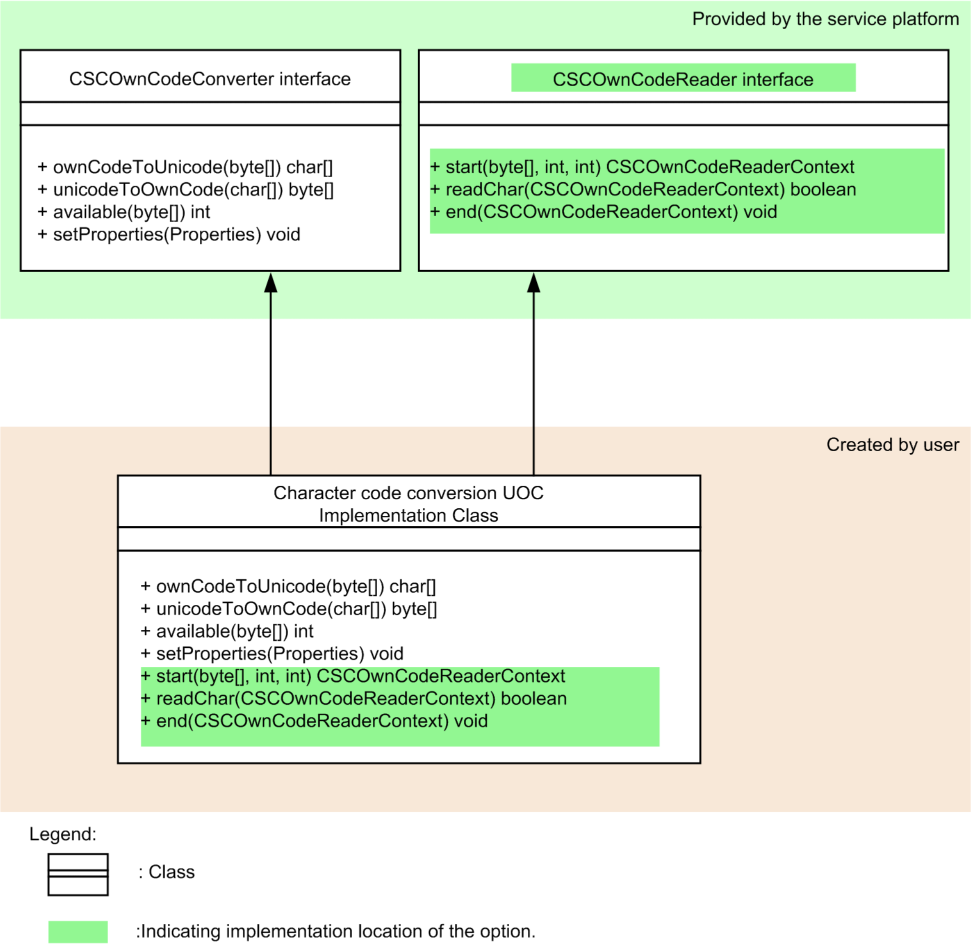
For implementation same as example, you must purchase code conversion separately.

I.4 CSCOwnCodeReader interface

(1) Interface

Following figure shows CSCOwnCodeReader interface:

Figure I–3: FigureCSCOwnCodeReader interface


Provided by the service platform
CSCOwnCodeConverter interface
+ ownCodeToUnicode(byte[]) char[]
+ unicodeToOwnCode(char[]) byte[]
+ available(byte[]) int
+ setProperties(Properties) void
CSCOwnCodeReader interface
+ start(byte[], int, int) CSCOwnCodeReaderContext
+ readChar(CSCOwnCodeReaderContext) boolean
+ end(CSCOwnCodeReaderContext) void
Created by user
Character code conversion UOC
Implementation Class
+ ownCodeToUnicode(byte[]) char[]
+ unicodeToOwnCode(char[]) byte[]
+ available(byte[]) int
+ setProperties(Properties) void
+ start(byte[], int, int) CSCOwnCodeReaderContext
+ readChar(CSCOwnCodeReaderContext) boolean
+ end(CSCOwnCodeReaderContext) void
Legend:
: Class
:Indicating implementation location of the option.

Interface name

CSCOwnCodeReader interface

Description

This is the interface for reading the character string of self-defined character code.

Package of CSCOwnCodeReader is jp.co.Hitachi.soft.csc.dt.uoc.CSCOwnCodeReader.

In the binary character string reading process, when you parse the separator without converting the character code, you can relax the restrictions of conversion in which bytes count or character count is changed and improve the conversion performance.

Implementation of this interface is optional. If you do not implement, the separator is parsed by executing the character code conversion.

As the instances of this interface are shared in multiple threads, you must set thread safe at the time of implementation.

Format

```
package jp.co.Hitachi.soft.csc.dt.uoc ;

import jp.co.Hitachi.soft.csc.dt.uoc.CSCOwnCodeConverterException ;

public interface CSCOwnCodeReader {
        CSCOwnCodeReaderContext start( byte[] data, int offset, int length )
              throws CSCOwnCodeConverterException ;

        boolean readChar( CSCOwnCodeReaderContext context )
              throws CSCOwnCodeConverterException ;

        void end( CSCOwnCodeReaderContext context ) ;
}
```

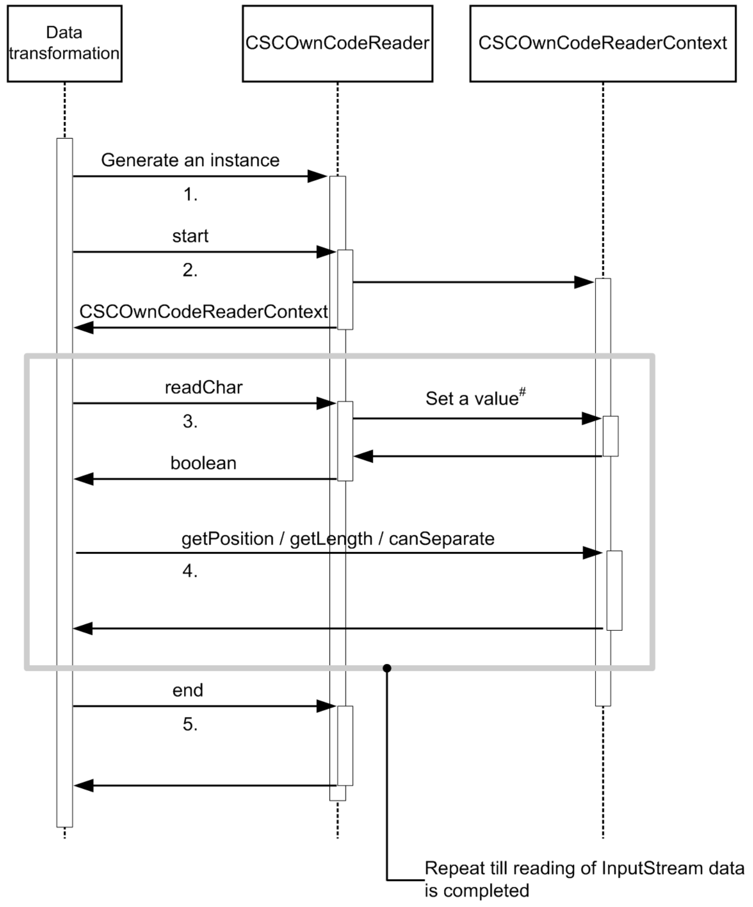
Method

Following table describes the methods of CSCOwnCodeReader interface.

Method name	Description
start method	This is the method to start the reading of character string of self-defined character code.
readChar method	This is the method to read 1 character of self-defined character code characters.
end method	This is the method to end the reading of character string of self-defined character code.

When data transformation target is variable length character string and separator has been set in the binary format definition, data transformation executes the separator parsing process. Following figure shows the order of invoking each method of CSCOwnCodeReader and CSCOwnCodeReaderContext.

Figure I–4: FigureOrder of invoking each method of CSCOwnCodeReader and CSCOwnCodeReaderContext



#
Interface is not defined because the value between the classes implemented by the user is already delivered.

1. Generating an instance

Generate an instance of CSCOwnCodeReader, by data transformation.

2. CSCOwnCodeReader#start method

   Generate instance of self-thread dedicated CSCOwnCodeReaderContext. From here onwards, delivery of value with CSCOwnCodeReader is executed through instance of CSCOwnCodeReaderContext saved by the self thread.

3. CSCOwnCodeReader#readChar method

   When executing readChar, data transformation process passes the instance of CSCOwnCodeReaderContext to the argument. In readChar process, parsing result is set to the instance of CSCOwnCodeReaderContext. The set parsing result is used for parsing the separator in the data transformation process.

4. getPosition method, getLength method, canSeparate method

   Respective methods are invoked from the data transformation. Execution order of method depends on the message format.

   - getPosition method

     This method returns the current character position.

   - getLength method

     This method returns the current character length.

   - canSeparate method

     This method returns whether to consider the current character as the separator parsing target.

5. CSCOwnCodeReader#end method

   If the process to release implementation class of CSCOwnCodeReaderContext is required, execute end method.

(a) start method

Description

Starts the reading of character string of self-defined character code.

Format

```
public CSCOwnCodeReaderContext start( byte[] data, int offset, int length )
```

Parameter

data:

This is reading target data.

offset:

This is reading start position.

length:

This is length from the reading start position.

Exception

CSCOwnCodeConverterException:

Entire data transformation process was aborted, as error occurred during the character code conversion process.

Return value

This is instance of the implementation class of CSCOwnCodeReaderContext. You cannot use this instance in any other thread.

(b) readChar method

Description

Reads 1 character of self-defined character code character.

You can acquire character position and length by CSCOwnCodeReaderContext#getPosition, CSCOwnCodeReaderContext#getLength.

Format

```
public boolean readChar( CSCOwnCodeReaderContext context )
```

Parameter

context:

This is instance of implementation class of CSCOwnCodeReaderContext, which is returned in #start.

Exception

CSCOwnCodeConverterException:

Entire data transformation process was aborted, as error occurred during the character code conversion process.

Return value

When upper limit for reading is exceeded and character cannot be converted, returns false. Otherwise, returns true.

(c) end method

Description

Ends the reading of character string of the self-defined character code.

This method is always called regardless of success or failure (error) in reading process.

Format

```
public void end( CSCOwnCodeReaderContext context )
```

Parameter

context:

This is instance of implementation class of CSCOwnCodeReaderContext, which is returned in #start.

Exception

CSCOwnCodeConverterException:

Entire data transformation process was aborted, as error occurred during the character code conversion process.

Return value

None

(2) Exception class

The exception class that occurs at the time of developing character code conversion UOC is as follows:

Class name

CSCOwnCodeConverterException class

Description

This is the exception, which is sent when error occurs in character code conversion process.

When this exception occurs, entire data transformation process is aborted.

(3) Implementation example (MS932)

Implementation example (MS932) of CSCOwnCodeReader interface is as follows:

```
public class CSCOwnCodeReaderImpl implements CSCOwnCodeConverter,
CSCOwnCodeReader {

    private static final String UNICODE = "ISO-10646-UCS-2" ;

    private final HJCOption option ;

    private static byte[] charSizeTable = initCharSizeTable() ;

    private static byte[] initCharSizeTable() {

        final byte[] objTable = new byte[0x100] ;

        for ( int i = 0; i <= 0xff; i++ ) {
            if ( i <= 0x80
                || (i >= 0xA0 && i <= 0xDF)
                || (i >= 0xFD && i <= 0xFF) ) {
                objTable[i] = 1 ;
            }
            else {
                objTable[i] = 2 ;
            }
        }

        return objTable ;
    }


    public CSCOwnCodeReaderImpl() {

        option = new HJCOption() ;
        try {
            // Unicode is Big Endian
            option.enableOption( HJCOption.COP_BIGENDIAN ) ;
        }
        catch ( Exception e ) {
            e.printStackTrace() ;
        }
    }


    @Override
    public void setProperties( Properties properties )
        throws CSCOwnCodeConverterException {

        String codetablepath = null ;
        if ( properties != null ) {
            codetablepath = properties.getProperty( "codetablepath" ) ;
        }

        try {
            if ( codetablepath == null ) {
                option
                    .setTablePath( "C:\\Program Files\\HITACHI\\Cosminexus\\CSC
\\lib\\external\\table" ) ;
            }
            else {
                option.setTablePath( codetablepath ) ;
            }
        }
        catch ( Exception e ) {
            e.printStackTrace() ;
        }
    }


    @Override
    public int available( byte[] inBuffer ) throws
CSCOwnCodeConverterException {

        if ( inBuffer == null ) {
            final String message = "You cannot convert blank character
string" ;
            throw new CSCOwnCodeConverterException( message ) ;
        }

        int retInt = -1 ;
        final HJCResult result = new HJCResult() ;
        final HJCString inStr = new HJCString( inBuffer ) ;

        try {
            HJCConverters.cs_ms932tounicode( inStr, result, option ) ;
        }
        catch ( Exception e ) {
            e.printStackTrace() ;
            throw new CSCOwnCodeConverterException( e ) ;
        }

        final byte[] resultData = result.getStrResult().getBytes() ;
        final int resultState = result.getConvertState() ;
        if ( resultState == HJCConvertState.CST_NORMAL ) {
            // Conversion ended successfully
            if ( resultData != null ) {
                retInt = result.getResultLength() ;
            }
        }
        else {
            // Conversion ended abnormally
            final byte[] bytes = new byte[result.getResultLength() - 1] ;
            System.arraycopy(
                inBuffer,
                0,
                bytes,
                0,
                result.getResultLength() - 1 ) ;
            retInt = available( bytes ) ;
        }
        return retInt ;
    }

    @Override
    public char[] ownCodeToUnicode( byte[] inBuffer )
        throws CSCOwnCodeConverterException {

        char[] retChar = null ;
        final HJCResult result = new HJCResult() ;
        final HJCString inStr = new HJCString( inBuffer ) ;

        try {
            HJCConverters.cs_ms932tounicode( inStr, result, option ) ;
            final byte[] resultData = result.getStrResult().getBytes() ;
            final String retstr = new String( resultData, UNICODE ) ;
            retChar = retstr.toCharArray() ;
        }
        catch ( Exception e ) {
            e.printStackTrace() ;
            throw new CSCOwnCodeConverterException( e ) ;
        }

        return retChar ;
    }


    @Override
    public byte[] unicodeToOwnCode( char[] inBuffer )
        throws CSCOwnCodeConverterException {

        byte[] retByte = null ;
        final String data = new String( inBuffer ) ;
        final HJCResult result = new HJCResult() ;

        try {
            final HJCString inStr = new HJCString( data.getBytes( UNICODE ) ) ;
            HJCConverters.cs_unicodetoms932( inStr, result, option ) ;
            retByte = result.getStrResult().getBytes() ;
        }
        catch ( Exception e ) {
            e.printStackTrace() ;
            throw new CSCOwnCodeConverterException( e ) ;
        }

        return retByte ;
    }


    @Override
    public CSCOwnCodeReaderContext start( byte[] data, int offset, int length )
        throws CSCOwnCodeConverterException {

        return new CSCOwnCodeReaderContextImpl( Arrays.copyOfRange(
            data,
            offset,
            length ) ) ;
    }
    @Override
    public boolean readChar( CSCOwnCodeReaderContext context )
        throws CSCOwnCodeConverterException {

        final CSCOwnCodeReaderContextImpl contextImpl =
(CSCOwnCodeReaderContextImpl)context ;

        final int offset = contextImpl.getNextPosition() ;
        contextImpl.setPosition( offset ) ;

        final byte[] data = contextImpl.getData() ;

        // Acquire maximum length of input data (not the size of character)
        final int maxLength = data.length ;
        if ( offset >= maxLength ) {
            // Current position is outside the range of input data
            return false ;
        }

        final int len = charSizeTable[data[offset] & 0xff] ;

        contextImpl.setLength( len ) ;

        final int next = offset + len ;
        contextImpl.setNextPosition( next ) ;

        if ( next > maxLength ) {
            // Conversion result is error
            // Occurs when input data is invalid
            // Returns false and aborts the parsing
            return false ;
        }

        return true ;
    }

    @Override
    public void end( CSCOwnCodeReaderContext context )
        throws CSCOwnCodeConverterException {

        // No process is performed as resources to be released do not exist
    }

}
```

(4) Implementation example(IBM Kanji code)

Implementation example (IBM Kanji code) of CSCOwnCodeReader interface is as follows:

```
public class CSCOwnCodeReaderImpl
    implements
        CSCOwnCodeConverter,
        CSCOwnCodeReader {

    private static final String UNICODE = "ISO-10646-UCS-2" ;

    private static final byte SHIFT_SINGLEBYTE = (byte)0x0f ;

    private static final byte SHIFT_MULTIBYTE = (byte)0x0e ;

    private final HJCOption option ;


    public CSCOwnCodeReaderImpl() {

        option = new HJCOption() ;
        try {
            // Unicode is Big Endian
            option.enableOption( HJCOption.COP_BIGENDIAN ) ;
            // EBCDIC
            option.enableOption( HJCOption.COP_EBCDIC ) ;
        }
        catch ( Exception e ) {
            e.printStackTrace() ;
        }
    }

    @Override
    public void setProperties( Properties properties )
        throws CSCOwnCodeConverterException {

        String codetablepath = null ;
        if ( properties != null ) {
            codetablepath = properties.getProperty( "codetablepath" ) ;
        }

        try {
            if ( codetablepath == null ) {
                option
                    .setTablePath( "C:\\Program Files\\HITACHI\\Cosminexus\\CSC
\\lib\\external\\table" ) ;
            }
            else {
                option.setTablePath( codetablepath ) ;
            }
        }
        catch ( Exception e ) {
            e.printStackTrace() ;
        }
    }

    @Override
    public int available( byte[] inBuffer ) throws
CSCOwnCodeConverterException {

        if ( inBuffer == null ) {
            final String message = "You cannot convert blank character
string." ;
            throw new CSCOwnCodeConverterException( message ) ;
        }

        int retInt = -1 ;
        final HJCResult result = new HJCResult() ;
        final HJCString inStr = new HJCString( inBuffer ) ;

        try {
            HJCConverters.cs_ibmtounicode( inStr, result, option ) ;
        }
        catch ( Exception e ) {
            e.printStackTrace() ;
            throw new CSCOwnCodeConverterException( e ) ;
        }

        final byte[] resultData = result.getStrResult().getBytes() ;
        final int resultState = result.getConvertState() ;
        if ( resultState == HJCConvertState.CST_NORMAL ) {
            // Conversion ends successfully
            if ( resultData != null ) {
                retInt = result.getResultLength() ;
            }
        }
        else {
            // Conversion ends abnormally
            final byte[] bytes = new byte[result.getResultLength() - 1] ;
            System.arraycopy(
                inBuffer,
                0,
                bytes,
                0,
                result.getResultLength() - 1 ) ;
            retInt = available( bytes ) ;
        }
        return retInt ;
    }
    @Override
    public char[] ownCodeToUnicode( byte[] inBuffer )
        throws CSCOwnCodeConverterException {

        char[] retChar = null ;
        final HJCResult result = new HJCResult() ;
        final HJCString inStr = new HJCString( inBuffer ) ;

        try {
            HJCConverters.cs_ibmtounicode( inStr, result, option ) ;
            final byte[] resultData = result.getStrResult().getBytes() ;
            final String retstr = new String( resultData, UNICODE ) ;
            retChar = retstr.toCharArray() ;
        }
        catch ( Exception e ) {
            e.printStackTrace() ;
            throw new CSCOwnCodeConverterException( e ) ;
        }

        return retChar ;
    }
    @Override
    public byte[] unicodeToOwnCode( char[] inBuffer )
        throws CSCOwnCodeConverterException {

        byte[] retByte = null ;
        final String data = new String( inBuffer ) ;
        final HJCResult result = new HJCResult() ;

        try {
            final HJCString inStr = new HJCString( data.getBytes( UNICODE ) ) ;
            HJCConverters.cs_unicodetoibm( inStr, result, option ) ;
            retByte = result.getStrResult().getBytes() ;
        }
        catch ( Exception e ) {
            e.printStackTrace() ;
            throw new CSCOwnCodeConverterException( e ) ;
        }

        return retByte ;
    }


    @Override
    public CSCOwnCodeReaderContext start( byte[] data, int offset, int length )
        throws CSCOwnCodeConverterException {

        final byte[] tempData = Arrays.copyOfRange( data, offset, length ) ;

        // Start parsing from position of offset
        return new CSCOwnCodeReaderContextImpl( tempData ) ;
    }


    @Override
    public boolean readChar( CSCOwnCodeReaderContext context )
        throws CSCOwnCodeConverterException {

        final CSCOwnCodeReaderContextImpl contextImpl =
(CSCOwnCodeReaderContextImpl)context ;

        // Current position
        final int position = contextImpl.getNextPosition() ;

        // Byte string of parsing target character string
        final byte[] data = contextImpl.getData() ;

        // Upper limit of parsing is till end of input data
        final int maxLength = data.length ;

        // Parse 1 by 1 byte from current position
        for ( int i = position; i < maxLength; i++ ) {

            // Parse the next character
            if ( data[i] == SHIFT_SINGLEBYTE ) {
                // Transit to single byte mode
                contextImpl.setLength( 1 ) ;
                contextImpl.setCanSeparate( true ) ;
                continue ;
            }
            else if ( data[i] == SHIFT_MULTIBYTE ) {
                // Transit to multi-byte mode
                contextImpl.setLength( 2 ) ;
                contextImpl.setCanSeparate( false ) ;
                continue ;
            }

            contextImpl.setPosition( i ) ;
            contextImpl.setNextPosition( i + contextImpl.getLength() ) ;

            if ( contextImpl.getNextPosition() > maxLength ) {
                // Data is not sufficient
                return false ;
            }

            // Parsing is successful
            return true ;
        }

        contextImpl.setPosition( maxLength ) ;
        contextImpl.setNextPosition( maxLength ) ;

        return false ;
    }


    @Override
    public void end( CSCOwnCodeReaderContext context )
        throws CSCOwnCodeConverterException {

        // Performs nothing as the resources to be parsed do not exist
    }

}
```

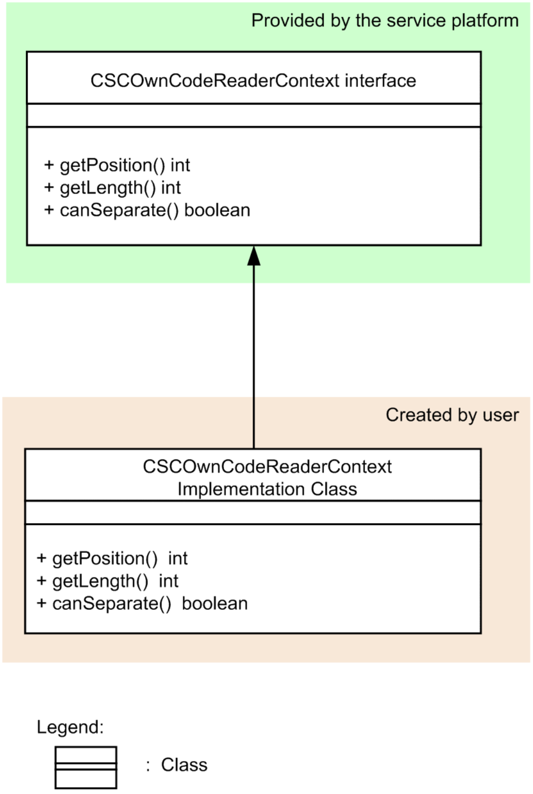
I.5 CSCOwnCodeReaderContext interface

(1) Interface

Reading information interface is as follows:

Create the implementation class of following interface, when developing character code conversion UOC.

Figure I–5: FigureCSCOwnCodeReaderContext implementation class



Interface name

CSCOwnCodeReaderContext interface

Description

This is reading information interface.

Package name of CSCOwnCodeReaderContext is jp.co.Hitachi.soft.csc.dt.uoc.CSCOwnCodeReaderContext.

Uses the parsing result of CSCOwnCodeReader to pass to the data transformation process.

Generates 1 instance in 1 thread. In CSCOwnCodeReader#readChar, parsing result of character string must be set to the implementation class of CSCOwnCodeReaderContext. The set information is referenced in the data transformation process.

Format

```
package jp.co.Hitachi.soft.csc.dt.uoc ;

public interface CSCOwnCodeReaderContext {

      int getPosition() ;

      int getLength() ;

      boolean canSeparate() ;
}
```

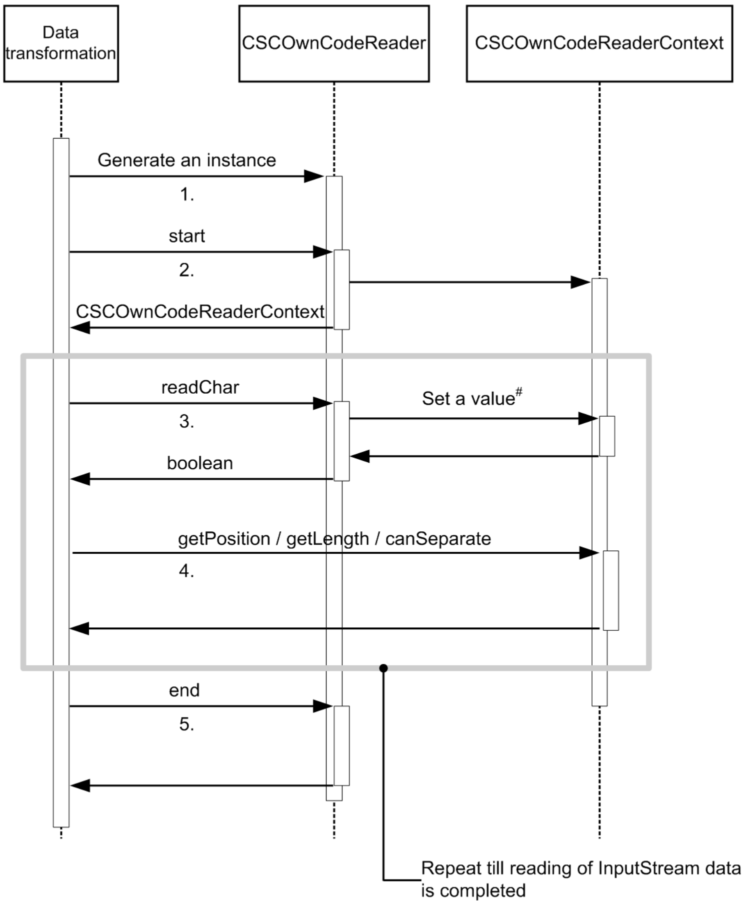
Methods

Following table describes the methods of CSCOwnCodeReaderContext interface.

Method name	Description
getPosition method	This method returns the current character position or size of data to be read.
getLength method	This method returns the current character length.
canSeparate method	This method returns whether to consider the current character as the separator parsing target.

When data transformation target is variable length character string and separator has been set in the binary format definition, data transformation executes the separator parsing process. Following figure shows the order of invoking each method of CSCOwnCodeReader and CSCOwnCodeReaderContext.

Figure I–6: FigureOrder of invoking each method of CSCOwnCodeReader and CSCOwnCodeReaderContext



#
Interface is not defined because the value between the classes implemented by the user is already delivered.

1. Generating an instance
   Generate an instance of CSCOwnCodeReader by data transformation.
2. CSCOwnCodeReader#start method
   Generate an instance of self-thread dedicated CSCOwnCodeReaderContext. Here onwards, deliver of values with CSCOwnCodeReader is executed by instance of CSCOwnCodeReaderContext saved by self-thread.
3. CSCOwnCodeReader#readChar method
   At the time of executing readChar, data transformation process passes the instance of CSCOwnCodeReaderContext to the argument. In process of readChar, parsing result is set in the instance of CSCOwnCodeReaderContext. The set parsing result is used for parsing the separator in the data transformation process.
4. getPosition method, getLength method, canSeparate method
   The respective methods are invoked from data transformation. Method execution order depends on the message format.
   - getPosition method
     This method returns the current character position.
   - getLength method
     This method returns the current character length.

- canSeparate method
  This method returns whether to consider the current character as the separator parsing target.

5. CSCOwnCodeReader#end method
   When parsing process of implementation class of CSCOwnCodeReaderContext is required, execute end method.

(a) getPosition method

Description

When result of readChar is true, returns the current character position.

When result of readChar is false, returns the size of data that can be read.

Format

```
public int getPosition()
```

Parameter

None

Exception

None

Return value

Current character position and size (unit is byte) of the data that can be read. Position of offset passed by CSCOwnCodeReader#start is considered as 0.

(b) getLength method

Description

Returns the current character length.

Format

```
public int getLength()
```

Parameter

None

Exception

None

Return value

Returns the current character length.

(c) canSeparate method

Description

Returns whether to consider the current character as the separator parsing target. When false is returned, even if the byte string of separator match with the current character, it is not considered as separator.

Format

```
public boolean canSeparate()
```

Parameter

None

Exception

None

Return value

Returns true when separator is to be parsed.

Returns false when separator is not to be parsed.

(2) Exception class

Exception class that occurs at the time of developing character code conversion UOC is as follows:

Class name

CSCOwnCodeConverterException class

Description

This exception is sent when error occurs during the character code conversion process.

When this exception occurs, entire data transformation process is aborted.

(3) Implementation example(MS932)

Implementation example (MS932) of CSCOwnCodeReaderContext interface is as follows:

```
public class CSCOwnCodeReaderContextImpl implements CSCOwnCodeReaderContext {

    private final byte[] data ;

    private int position = 0 ;

    private int next = 0 ;

    private int length = 0 ;


    public CSCOwnCodeReaderContextImpl(
        final byte[] data ) {

        this.data = data ;
    }


    @Override
    public int getPosition() {

        return position ;
    }


    @Override
    public int getLength() {

        return length ;
    }


    @Override
    public boolean canSeparate() {

        // MS932 does not have shift (escape sequence) status and
        // no limitation for occurrence of separator. Hence, always returns
true
        return true ;
    }


    public byte[] getData() {

        return data ;
    }


    public void setPosition( int position ) {

        this.position = position ;
    }


    public void setLength( int length ) {

        this.length = length ;
    }


    public int getNextPosition() {
        return this.next ;
    }

    public void setNextPosition( int position ) {

        this.next = position ;
    }
}
```

(4) Implementation example (IBM Kanji code)

Implementation example (IBM Kanji code) of CSCOwnCodeReaderContext interface is as follows:

```
public class CSCOwnCodeReaderContextImpl implements CSCOwnCodeReaderContext {

    private final byte[] data ;

    private int position = 0 ;

    private int length = 1 ;

    private int next = 0 ;

    private boolean canSeparate = true ;

    public CSCOwnCodeReaderContextImpl( final byte[] data ) {

        this.data = data ;
    }

    @Override
    public int getPosition() {

        return position ;
    }

    @Override
    public int getLength() {

        return length ;
    }

    @Override
    public boolean canSeparate() {

        return canSeparate ;
    }

    public byte[] getData() {

        return data ;
    }

    public void setPosition( int position ) {

        this.position = position ;
    }

    public void setLength( int length ) {

        this.length = length ;
    }
```

```
        public void setCanSeparate( boolean canSeparate ) {
            this.canSeparate = canSeparate ;
        }

        public int getNextPosition() {
            return this.next ;
        }

        public void setNextPosition( int position ) {
            this.next = position ;
        }
    }
```

J. Examples of transforming the format of data acquired by using the database adapter

Values in the data acquired by using the database adapter cannot be referenced easily by using a column name as a key. However, you can easily handle such data if you transform the data into a format where column names become element names.

This appendix uses examples to explain how to transform the data acquired with the database adapter into a format where column names become element names.

J.1 Examples

The database contents, executed SQL formats, SQL operation definition file, and formats after transformation used in the examples are as follows:

Database contents

Database reference name: `DB_SERVER1`

Schema name: `DBA`

Table name: `Order_Table`

Table configuration:

Order_number (INTEGER)	Customer_code (CHAR)	Product_code (CHAR)	Number_of_orders (INTEGER)
1	AA001	0001	5
2	AB002	0001	1
3	AA001	0102	3
4	XA005	0103	1
5	AA001	0105	1

Executed SQL format

```
OPERATION1:SELECT * FROM DBA.Order_table WHERE val1 val2 val3
```

SQL operation definition file

```
<?xml version="1.0" encoding="UTF-8" ?>
<DBadapter_SQL_OPERATION>
 <DATABASE_DATA>
 <DB_NAME>DB_SERVER1</DB_NAME>
 <DB_TYPE>HIRDB</DB_TYPE>
 </DATABASE_DATA>
 <SQL_DATA>
 <OPERATION1>
 SELECT * FROM DBA.Order_table WHERE <val1 dba_inf="column"/>
 <val2 dba_inf="preset"/>
 <val3 dba_inf="data" data_type="CHAR"/>
 </OPERATION1>
 </SQL_DATA>
</DBadapter_SQL_OPERATION>
```

For details about each item in the SQL operation definition file, see the descriptions related to the creation of an SQL operation definition file in *3.3.5 Defining Database Adapters* in the *Service Platform Reception and Adapter Definition Guide*. For the formats of data that can be acquired by using the database adapter, see the descriptions related to response message format in *3.3.5 Defining Database Adapters* in the *Service Platform Reception and Adapter Definition Guide*.

XML schema of the post-transformation formats

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema"
 targetNamespace="http://www.example.org/DBA_Sample1"
 xmlns:tns="http://www.example.org/DBA_Sample1"
 elementFormDefault="qualified">
```

```
  <complexType name="DBA_Sample1Type">
  <sequence>
  <element name="record" type="tns:RecordType"
  maxOccurs="unbounded" minOccurs="0"/>
  </sequence>
  </complexType>

  <complexType name="RecordType">
  <sequence>
  <element name="Order_number" type="int" maxOccurs="unbounded" minOccurs="0"/>
  <element name="Customer_code" type="string" maxOccurs="unbounded" minOccurs="0"/>
  <element name=" Product_code" type="string" maxOccurs="unbounded" minOccurs="0"/>
  <element name="Number_of_orders" type="int" maxOccurs="unbounded" minOccurs="0"/>
  </sequence>
  </complexType>

  <element name="Order_table" type="tns:DBA_Sample1Type"/>
 </schema>
```

J.2 Format transformation methods

To transform the formats of data acquired using the database adapter:

1. Data settings at transformation source and transformation destination

   To transform the formats of the data acquired using the database adapter, use the data transformation definition. Before you begin data format transformation, set up the data transformation source and transformation destination in the Data Transformation Definition screen.

2. Acquiring the cid attribute value of the column name

   Use the trim node function to acquire the column name ID (`cid` attribute value).

3. Settings for the loop node function

   Specify settings for the loop node function such that all the values acquired using the database adapter can be assigned to the transformation destination.

4. Assigning the values to the column name element at the transformation destination

   Compare the column name ID and the value ID. If the IDs are the same, assign the value to the transformation destination.

5. Validating the defined contents

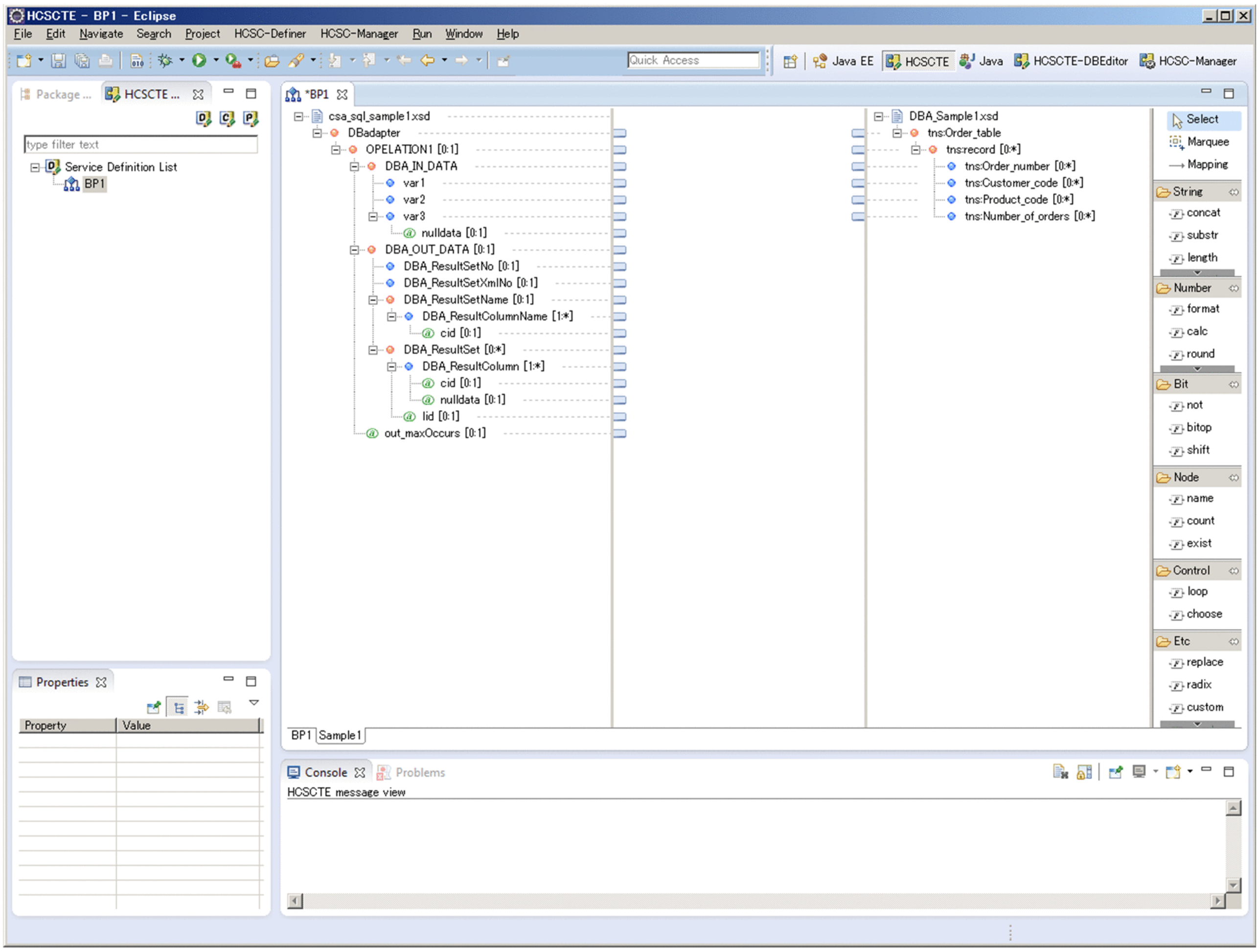
   Validate the defined contents and confirm that the mapping is consistent.

The following is a description of each of the steps:

(1) Data settings at transformation source and transformation destination

To transform the formats of the data acquired using the database adapter, use the data transformation definition.

In the Data Transformation Definition screen, set up the data acquired using the database adapter in the transformation source schema tree viewer and the XML schema with the post-transformation format in the transformation destination schema tree viewer.



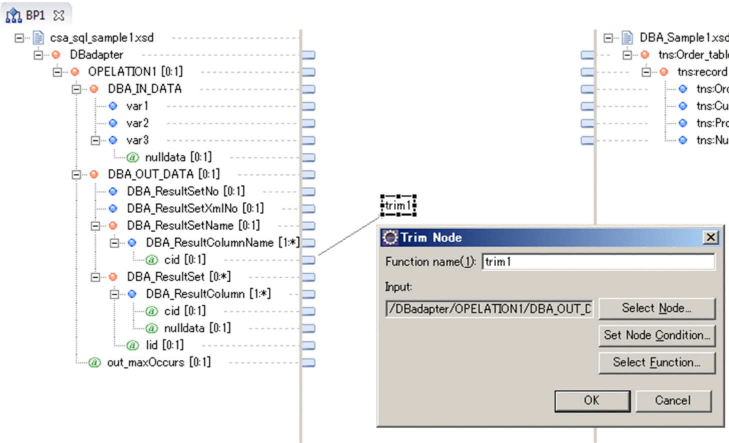
(2) Acquiring the cid attribute value of the column name

In the `DBA_ResultSetName` element of the response message, the column name and the ID (`cid` attribute) are stored in pairs. Use the trim node function to acquire the `cid` attribute corresponding to the column name specified in the node conditions.

To acquire the `cid` attribute value of the column name:

1. Define the trim node function for the `cid` attribute under the `DBA_ResultColumnName` element.

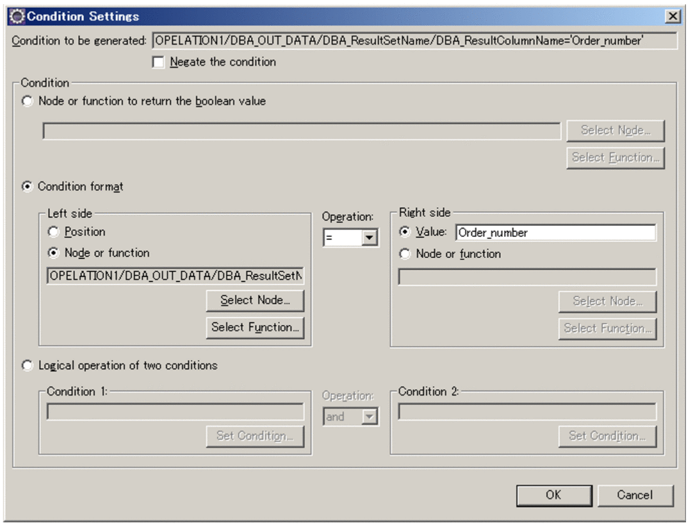
   For details about how to define the trim node function, see *6.5.5 Removing Spaces from a String*.



Function name(1): trim1

Input: /DBadapter/OPELATION1/DBA_OUT_D

2. In the Trim Node dialog box, click the Set Node Conditions button.
   The **Set Node Conditions** dialog box appears.
3. In the **Set Node Conditions** dialog box, click the Set Conditions button.
   The **Set Conditions** dialog box appears.
4. Set up the node conditions.
   Specify the settings as follows:
   - Left-hand side: `DBA_ResultColumnName` element
   - Right-hand side: Column name (here, Order_number)



Condition Settings

Condition to be generated: OPELATION1/DBA_OUT_DATA/DBA_ResultSetName/DBA_ResultColumnName='Order_number'

☐ Negate the condition

Condition

○ Node or function to return the boolean value

● Condition format

Left side: ○ Position ● Node or function: OPELATION1/DBA_OUT_DATA/DBA_ResultSetN

Operation: =

Right side: ● Value: Order_number ○ Node or function

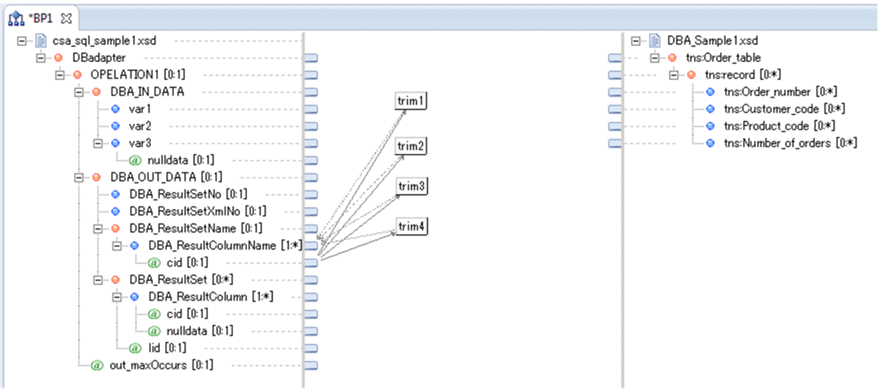
○ Logical operation of two conditions

Condition 1:

Operation: and

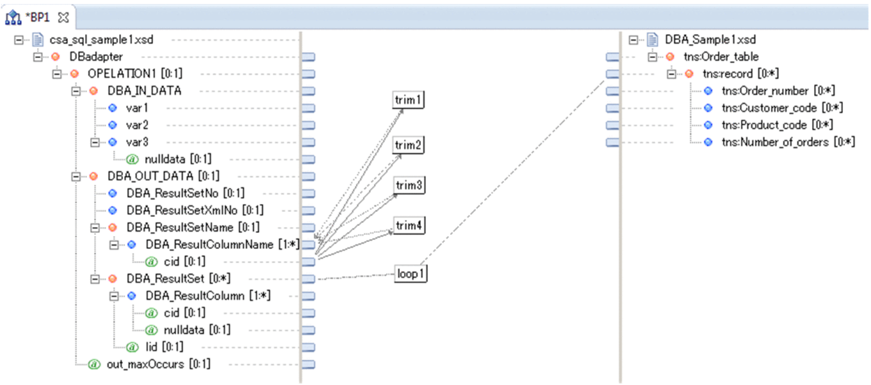
Condition 2:

5. Similarly define the elements for the Customer_code, Product_code, and the Number_of_orders.



(3) Settings for the loop node function

Map the loop node function from the `DBA_ResultSet` element of the transformation source to the `record` element of the transformation destination. For details about how to define the loop node function, see *6.5.16 Mapping Looping*.



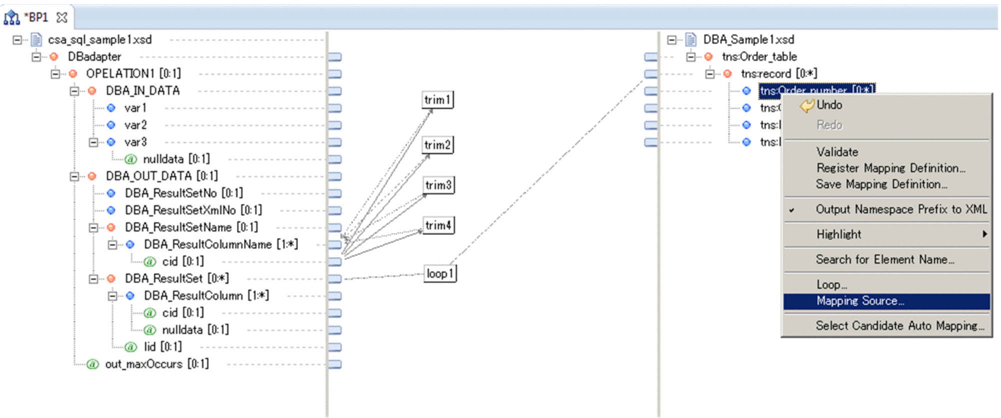
(4) Assigning the values to the column name element at the transformation destination

Assign a value to each row name element at the transformation destination. In this case, set up the conditions such that the trim node function defined in *(2) Acquiring the cid attribute value of the column name* matches the `cid` attribute under the `DBA_ResultColumn` element.

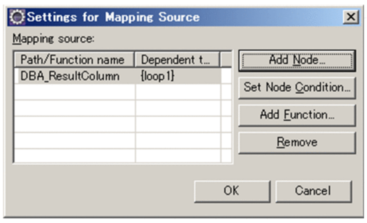
To assign a value to each row name element of the transformation destination:

1. Right click the transformation destination element (here, Order_number) and choose **Mapping Source**.

   The **Set Mapping Source** dialog box appears.



2. In the **Mapping Source**, select `DBA_ResultColumn`.


Settings for Mapping Source

Mapping source:

Path/Function name	Dependent t...
DBA_ResultColumn	{loop1}

3. In the **Set Mapping Source** dialog box, click the **Set Node Conditions** button.

   The **Set Node Conditions** dialog box appears.

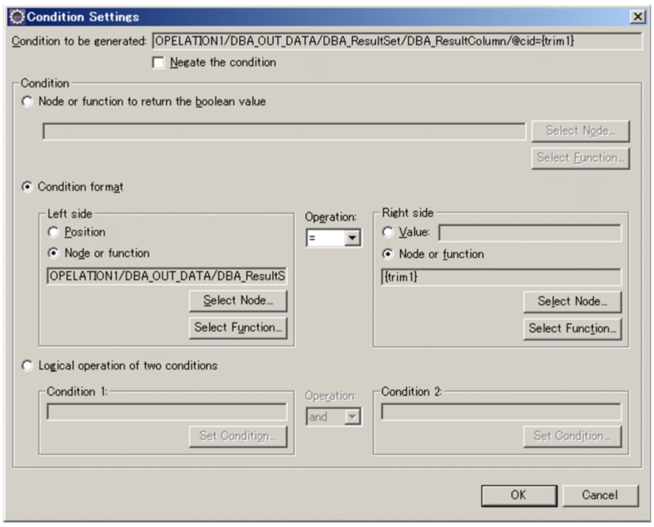
4. In the **Set Node Conditions** dialog box, click the **Set Conditions** button.

   The **Set Conditions** dialog box appears.

5. Set up the node conditions.

   Specify the settings as follows:

   - Left-hand side: `cid` attribute under the `DBA_ResultColumn` element
   - Right-hand side: Trim node function defined to correspond to the transformation destination column name



Condition Settings

Condition to be generated: OPELATION1/DBA_OUT_DATA/DBA_ResultSet/DBA_ResultColumn/@cid={trim1}

□ Negate the condition

Condition

○ Node or function to return the boolean value

● Condition format

Left side

○ Position

● Node or function

OPELATION1/DBA_OUT_DATA/DBA_ResultS

Operation: =

Right side

○ Value:

● Node or function

{trim1}

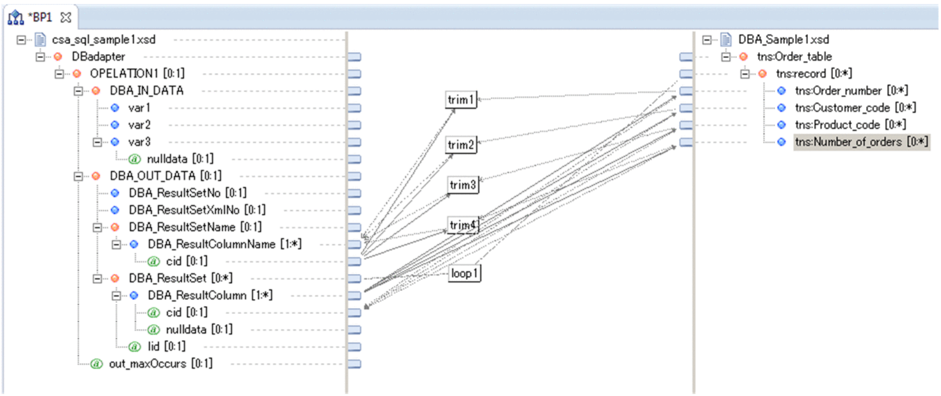
○ Logical operation of two conditions

Condition 1:

Operation: and

Condition 2:

6. Similarly define the elements for the Customer_code, Product_code, and the Number_of_orders.



(5) Validating the defined contents

Validate whether the defined mapping is consistent. For details about the validation method, see *5.10.2 Validation Method*.

K. Auto mapping of data acquired by DB adapter

This section describes the method of automatically mapping the data acquired by DB adapter, in the format in which column name serves as element name.

Use this method for automatically implementing "*Appendix J. Examples of transforming the format of data acquired by using the database adapter*". However, take note of the following points, while defining.

- In "*Appendix J. Examples of transforming the format of data acquired by using the database adapter*", mapping is done with cid attribute as key. However, in this method, auto mapping is performed with transformation source column number as key.
- As repeat function is not set automatically, you must set the function manually.

Flow of mapping is as follows:

1. Create the column definition file, based on the SQL operation definition file, request message and response message.
2. Perform mapping to the data transformation definition screen, by using the created column definition file.

The respective activities are described as follows:

(1) Creating the column definition file

The column definition file is CSV format file in which Position()[transformation source column number] of the search result of DB adapter is associated with transformation destination element name of XML schema.

(a) Format

Format of the column definition file is as follows:

```
Transformation source string number, transformation destination element name [linefeed]
  :
```

Following table describes the syntax of the column definition file:

Table K–1: TableSyntax of the column definition file

Item	Syntax of the column definition file
Column definition file	(Line data linefeed code)*
Line data	Space* [transformation source string number blank*] [',' blank* transformation destination element name blank*]
Linefeed code	'\r\n' | '\r' | '\n'
Blank	' ' | '\t'
Transformation source string number	('0' | '1' | '2' | '3' | '4' | '5' | '6' | '7' | '8' | '9') +
Transformation destination element name	(MS932 character - (',' | ':' | blank | linefeed code))+

(b) Definition example

This section describes the definition example of column definition file when acquiring the following table structure data with DB adapter, by considering the contents of database of "*Appendix J.1 Examples*" as an example.

Table K–2: TableTable structure (first row indicates column name)

Order_number (INTEGER)	Customer_code (CHAR)	Product_code (CHAR)	Number_of_orders (INTEGER)
1	AA001	0001	5

Order_number (INTEGER)	Customer_code (CHAR)	Product_code (CHAR)	Number_of_orders (INTEGER)
2	AB002	0001	1
3	AA001	0102	3
4	XA005	0103	1
5	AA001	0105	1

This table stricture data is acquired from DB adapter, as described in the following table.

Table K–3: TableData acquired from DB adapter (first row indicates acquisition order (column number))

1	2	3	4
1	AA001	0001	5
2	AB002	0001	1
3	AA001	0102	3
4	XA005	0103	1
5	AA001	0105	1

As described above, in case of data acquired from DB adapter, you cannot reference the value with column name as key. Therefore, you must transform to the format where column name is the element name.

Accordingly, use the column definition file to define which column number of the data acquired from the DB adapter is to be mapped with which element name.

Definition example of the column definition file

```
1,Order_number
2,Customer_code
3,Product_code
4,Number_of_orders
```

First row of this column definition file indicates that the DB adapter maps the data in first column to the **Order_number**.

Thus, When you perform auto mapping by using this column definition file, the data acquired from the DB adapter is transformed as follows and you can reference the value by considering the element name (column name) as key.

Table K–4: TableTable structure (first row indicates the column name)

Order_number	Customer_code	Product_code	Number_of_orders
1	AA001	0001	5
2	AB002	0001	1
3	AA001	0102	3
4	XA005	0103	1
5	AA001	0105	1

(c) Notes

This section describes the points to be considered when creating the column definition file.

- You can use any file name and extension.
- Available character code is MS932.
- Transformation destination element name is case sensitive.

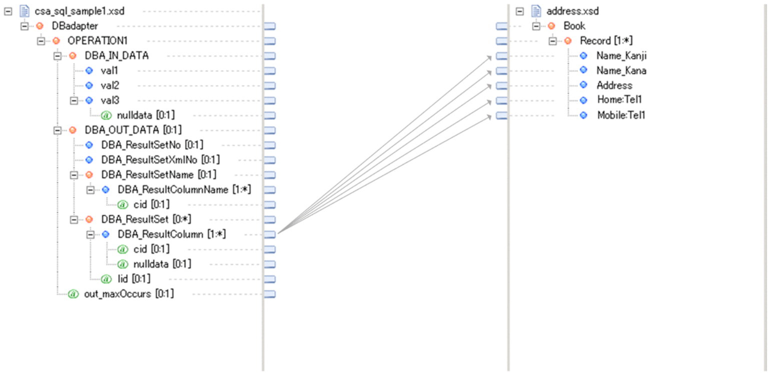
- When you omit the transformation source column number, transformation destination element name or both, definition of that row is ignored and a warning message is output.
- Maximum length of the character string of transformation source column number is 1,024 characters.
- When you define the same transformation destination element name in duplication, element defined at first is ignored. In that case, a warning message is displayed.
- In the auto mapping using the column definition file, evaluation is performed only with the transformation destination element name of the column definition file and namespace is not differentiated.

  For example, if you define as follows, in the column definition file and mapping is performed in the schema in which element name is duplicated as "Home:Tel1" "Mobile:Tel1", mapping is performed in both the elements, as shown in the following figure:

  Contents of the column definition file

```
1,Name_Kanji
2,Name_Kana
3,Address
4,Tel1
```

Figure K–1: FigureExample of mapping in case of same element name having different namespace



(2) Mapping in the data transformation definition screen

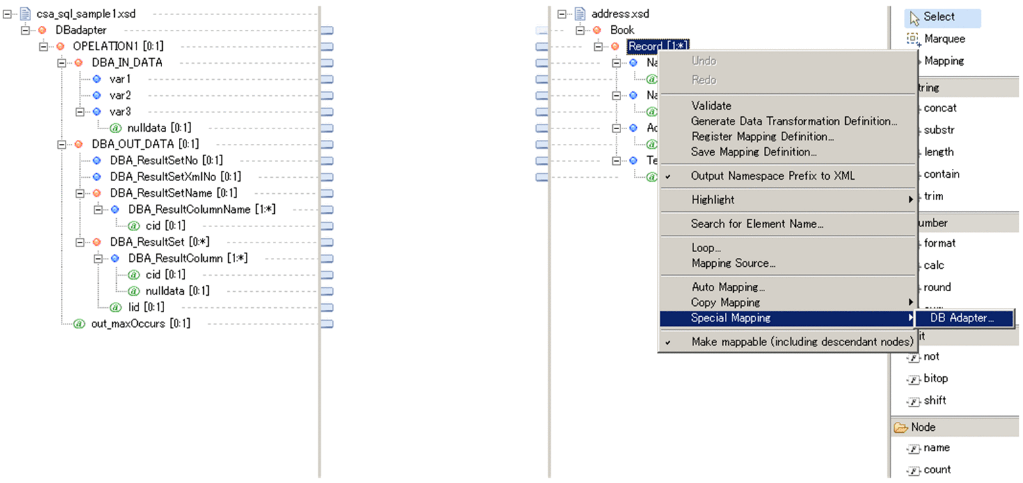
Use the data transformation definition to perform automatic mapping to the format in which column name is the element name.

In the transformation source schema tree view of the data transformation definition screen, set the data acquired with DB adapter and in the transformation destination schema tree viewer, set the XML schema having format after conversion.

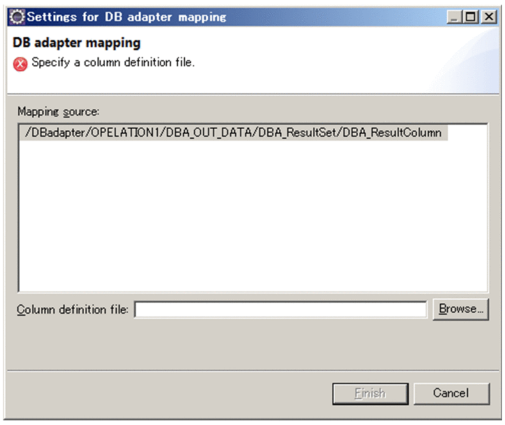
Method of mapping is as follows:

1. Select parent element (complex contents element) of the element that is transformation destination mapping target.
2. Right click when the parent element is in selected status.

   Popup menu is displayed.



3. From popup menu, select [Special mapping]-[DB adapter].
   DB adapter mapping settings dialog is displayed.


Settings for DB adapter mapping

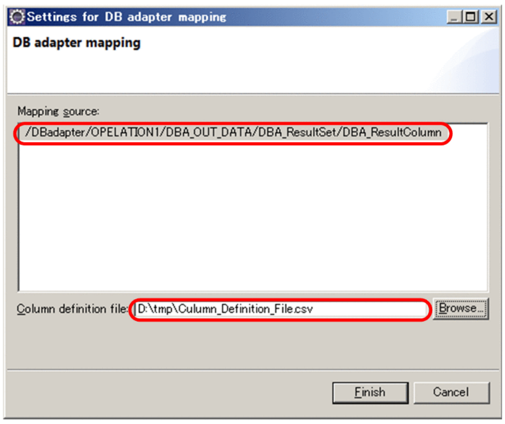
DB adapter mapping

Specify a column definition file.

Mapping source:
/DBadapter/OPELATION1/DBA_OUT_DATA/DBA_ResultSet/DBA_ResultColumn

Column definition file:

4. Select the mapping source and specify the column definition file.

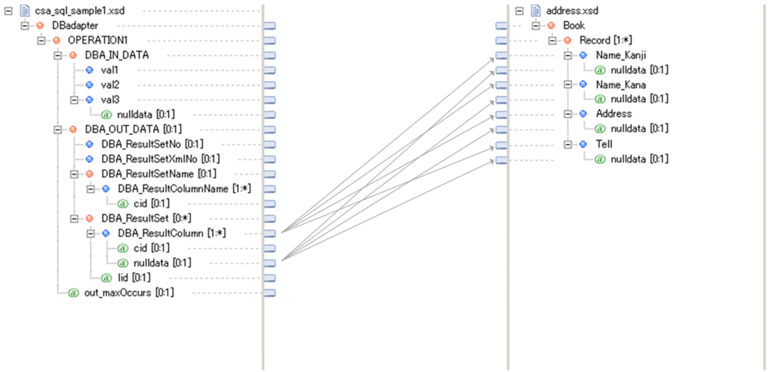

Settings for DB adapter mapping

DB adapter mapping

Mapping source:
/DBadapter/OPELATION1/DBA_OUT_DATA/DBA_ResultSet/DBA_ResultColumn

Column definition file: D:\tmp\Culumn_Definition_File.csv

5. Click **Finish** button.
   Auto mapping is performed.



If the element name under the node for which you selected **DB adapter** in the popup menu, and the transformation destination element name of the column definition file do not match, or the matching element is not the mapping target, warning message is displayed.

6. Perform mapping of Repeat function from transformation source element to transformation destination element.

   For details on how to define the repeat function, see "*6.5.16 Mapping Looping*".

L. Customizing WSDL using the external binding file

You can customize the WSDL parsing method by using the external binding file. Flow of WSDL customizing using the external binding file is as follows:

1. Create the external binding file

   You can use any file name.

   For details on how to create the external binding file, see "*15.2 Customizing the mapping from WSDL to Java*" in the "*Application Server Web Service Development Guide*".

2. Set up the created external binding file through WSDL customized binding setting page.

   For details on the WSDL customized binding setting page, see (Data transformation settings) in the "*Service Platform Reference Guide*".

   When external binding file is used in the version prior to 09-00 and version is upgraded to 09-50 onwards in the overwrite installation, Soecuft external binding file that you want to use (<Service Platform installation directory>\CSCTE\config\tools\custom\custombinding.xml) in the WSDL custom binding settings page.

Parse WSDL file based on the stored external binding file. You can use the customized WSDL file when creating the service adapter or user-defined reception.

M. Changing IBM kanji code character set

When you select IBM kanji code as a character code, in binary format definition file, by default it is set as follows:

Table M–1: TableDefault when you select IBM kanji code in the character code

Character code type	Code system
IBM_CODE+EBCDIC(LATIN)	IBM kanji code + standard English lower case characters set
IBM_CODE+EBCDIC(KANA)	IBM kanji code + standard Katakana character set

When setting the character code of binary format definition file, use Format dialog. For details on Format dialog, see "*1.3.1 Format dialog*" in the "*Service Platform Reference Guide*".

For details on character code, see "*Appendix A Character code support table*" in the "*Service Platform Reference Guide*".

M.1 Procedure for changing the character set of IBM kanji code

Overwrite all the table files stored in the following copy source directories, to the copy destination directories, after stopping HCSC server.

Table M–2: TableChanging IBM kanji code

Change contents	Copy source	Overwrite destination
Change from standard character set to extended character set (for Windows)	<Service Platform installation directory>\CSC\lib\external\table \ibmE	<Service Platform installation directory>\CSC\lib\external \table
Changing from extended character set to standard character set (for Windows)	<Service Platform installation directory>\CSC\lib\external\table \ibmB	<Service Platform installation directory>\CSC\lib\external \table
Changing from standard character set to extended character set (for UNIX)	/opt/Cosminexus/CSC/lib/external/table/ibmE	/opt/Cosminexus/CSC/lib/external/table
Changing from extended character set to standard character set (for UNIX)	/opt/Cosminexus/CSC/lib/external/table/ibmB	/opt/Cosminexus/CSC/lib/external/table

N. Glossary

Terminology used in this manual

For the terms used in the manual, see *Application Server and BPM/ESB Platform Terminology Guide*.

Index