

uCosminexus Service Platform

# Reception and Adapter Definition Guide

3020-3-Y44-40(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

All Borland brand and product names are trademarks or registered trademarks of Borland Software Corporation in the United States and other countries.

IOP is a trademark of Object Management Group, Inc. in the United States.

Microsoft is either a registered trademark or trademark of Microsoft Corporation in the United States and/or other countries.

OMG, CORBA, IOP, UML, Unified Modeling Language, MDA and Model Driven Architecture are either registered trademarks or trademarks of Object Management Group, Inc. in the United States and/or other countries.

Oracle and Java are registered trademarks of Oracle Corporation and/or its affiliates.

SOAP is an XML-based protocol for sending messages and making remote procedure calls in a distributed environment.

UNIX is a registered trademark of The Open Group in the United States and other countries.

W3C is a trademark (registered in numerous countries) of the World Wide Web Consortium.

WebSphere is a trademark of International Business Machines Corporation in the United States, other countries, or both.

Windows, Windows Server, and Windows Vista are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

The other company names and product names are either trademarks or registered trademarks of the respective companies.

Eclipse is an open development platform for tools integration provided by Eclipse Foundation, Inc., an open source community for development tool providers.

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

## ■ Microsoft product name abbreviations

This manual uses the following abbreviations for Microsoft product names:

Abbreviation		Full name or meaning
Excel		Microsoft(R) Excel
		Microsoft(R) Office Excel
Internet Explorer 6		Microsoft(R) Internet Explorer(R) 6
Internet Explorer 7		Windows(R) Internet Explorer(R) 7
Internet Explorer 8		Windows(R) Internet Explorer(R) 8
Internet Explorer 9		Windows(R) Internet Explorer(R) 9
Internet Explorer 10		Windows(R) Internet Explorer(R) 10
Internet Explorer 11		Windows(R) Internet Explorer(R) 11
Windows	Windows Server 2008	Windows Server 2008 x86
		Windows Server 2008 x64
	Windows Server 2008 R2	Microsoft(R) Windows Server(R) 2008 Standard 32-bit
		Microsoft(R) Windows Server(R) 2008 Enterprise 32-bit
	Windows Server 2008 x64	Microsoft(R) Windows Server(R) 2008 Standard
		Microsoft(R) Windows Server(R) 2008 Enterprise
Windows Server 2008 R2	Microsoft(R) Windows Server(R) 2008 R2 Standard	
	Microsoft(R) Windows Server(R) 2008 R2 Enterprise	

Abbreviation		Full name or meaning	
Windows	Windows Server 2008	Windows Server 2008 R2	Microsoft(R) Windows Server(R) 2008 R2 Datacenter
	Windows Server 2012	Windows Server 2012 Standard	Microsoft(R) Windows Server(R) 2012 Standard
		Windows Server 2012 R2 Standard	Microsoft(R) Windows Server(R) 2012 R2 Standard
		Windows Server 2012 Datacenter	Microsoft(R) Windows Server(R) 2012 Datacenter
		Windows Server 2012 R2 Datacenter	Microsoft(R) Windows Server(R) 2012 R2 Datacenter
	Windows Vista	Windows Vista Business	Microsoft(R) Windows Vista(R) Business (32-bit)
		Windows Vista Enterprise	Microsoft(R) Windows Vista(R) Enterprise (32-bit)
		Windows Vista Ultimate	Microsoft(R) Windows Vista(R) Ultimate (32-bit)
	Windows 7	Windows 7 x86	Microsoft(R) Windows(R) 7 Professional (32-bit)
			Microsoft(R) Windows(R) 7 Enterprise (32-bit)
			Microsoft(R) Windows(R) 7 Ultimate (32-bit)
		Windows 7 x64	Microsoft(R) Windows(R) 7 Professional (64-bit)
			Microsoft(R) Windows(R) 7 Enterprise (64-bit)
			Microsoft(R) Windows(R) 7 Ultimate (64-bit)
	Windows 8	Windows 8 x86	Windows(R) 8 Pro (32-bit)
			Windows(R) 8 Enterprise (32-bit)
		Windows 8 x64	Windows(R) 8 Pro (64-bit)
			Windows(R) 8 Enterprise (64-bit)
	Windows 8.1	Windows 8.1 x86	Windows(R) 8.1 Pro (32-bit)
			Windows(R) 8.1 Enterprise (32-bit)
Windows 8.1 x64		Windows(R) 8.1 Pro (64-bit)	
		Windows(R) 8.1 Enterprise (64-bit)	

■ Issued

Oct. 2015: 3020-3-Y44-40(E)

■ Copyright

All Rights Reserved. Copyright (C) 2015, Hitachi, Ltd.



# Preface

---

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



# Contents

<b>1</b>	<b>Overview of Development of Receptions and Service Adapters</b>	<b>1</b>
1.1	Positioning of receptions and service adapters in the whole system	2
1.2	Receptions and service adapters that can be used	4
1.2.1	Receptions that can be used	4
1.2.2	Service adapters that can be used	5
<b>2</b>	<b>Defining User-Defined Reception</b>	<b>7</b>
2.1	Overview of user-defined receptions	8
2.1.1	Overview of SOAP receptions	8
2.1.2	Overview of TP1/RPC receptions	9
2.1.3	Overview of FTP receptions	9
2.1.4	Overview of HTTP receptions	9
2.1.5	Overview of Message Queue receptions	9
2.1.6	Overview of Custom receptions	9
2.2	Defining a SOAP reception	10
2.2.1	Work flows for defining a SOAP reception	10
2.2.2	Adding a SOAP reception	15
2.3	Defining TP1/RPC reception	18
2.3.1	Flow of defining TP1/RPC reception	18
2.3.2	Adding TP1/RPC reception	18
2.3.3	Editing definition of TP1/RPC reception	18
2.4	Defining FTP reception	19
2.4.1	Flow of defining FTP reception	19
2.4.2	Creating message format of FTP reception	20
2.4.3	Creating definition file of FTP reception	27
2.4.4	Adding FTP reception	28
2.5	Defining HTTP reception	31
2.5.1	Flow of defining HTTP reception	31
2.5.2	Creating message format of HTTP reception	32
2.5.3	Creating definition file of HTTP reception	38
2.5.4	Adding HTTP reception	39
2.5.5	Setting up an activity	40
2.6	Defining Message Queue reception	43
2.6.1	Flow of defining Message Queue reception	43
2.6.2	Creating message format of Message Queue reception	44
2.6.3	Creating of Message Queue reception definition file	46
2.6.4	Adding Message Queue reception	47

2.7	Creating WSDL	49
2.7.1	Example of a Business Process Used for Creating WSDL	49
2.7.2	Examples of dynamically changing the connection-destination information of the service adapter	65
2.7.3	Notes on creating WSDL of a user-defined reception	69
2.8	Checking User-Defined Reception Contents	71
2.9	Saving a user-defined reception	72
2.10	Validating a User-Defined Reception	73
2.10.1	Validation Contents	73
2.10.2	Validation Method	73
2.11	Changing the information of a user-defined reception	75
2.12	Deleting a User-Defined Reception	76

## 3

3	Defining Adapters	77
3.1	Workflow for Defining Service Adapters	78
3.2	Adding Service Adapters	81
3.2.1	Adding a new SOAP adapter	81
3.2.2	Adding a new Session Bean adapter	82
3.2.3	Adding a new MDB (WS-R) adapter	82
3.2.4	Adding a new MDB (DB queue) adapter	82
3.2.5	Adding a new database adapter	83
3.2.6	Adding a new TP1 adapter	83
3.2.7	Adding a new file adapter	83
3.2.8	Adding a new Object Access adapter	83
3.2.9	Adding a new Message Queue adapter	84
3.2.10	Adding a new FTP adapter	84
3.2.11	Adding a new file operation adapter	84
3.2.12	Adding a new mail adapter	85
3.2.13	Adding a new HTTP adapter	85
3.2.14	Adding a new custom adapter	86
3.2.15	Using an Already Defined Adapter to Add an Adapter	86
3.3	Defining the Contents of Service Adapters	87
3.3.1	Defining SOAP adapters	87
3.3.2	Defining Service Adapters (SessionBean)	90
3.3.3	Defining Service Adapters (MDB (WS-R))	94
3.3.4	Defining Service Adapters (MDB (DB Queue))	97
3.3.5	Defining Database Adapters	98
3.3.6	Defining a TP1 adapter	123
3.3.7	Defining file adapters	123
3.3.8	Defining an Object Access adapter	139
3.3.9	Defining Message Queue adapters	139
3.3.10	Defining FTP adapters	153

3.3.11 Defining file operations adapters	178
3.3.12 Defining mail adapters	189
3.3.13 Defining HTTP adapters	205
3.3.14 Defining custom adapters	218
3.3.15 List of Settings in Adapter Definition	227
3.4 Saving Adapters	252
3.5 Editing Adapters	253
3.6 Validating Adapters	254
3.6.1 Validation Method	254
3.6.2 Displaying the Validation Contents	254
3.7 Deleting Adapters	255

## Appendixes 257

A. Custom Reception	258
A.1 Custom reception overview	258
A.2 Custom reception development	260
A.3 APIs of the custom reception framework	264
A.4 Custom reception definition	274
A.5 Custom reception operations	277
A.6 Custom reception tuning	282
A.7 Method of notifying error in custom reception	283
A.8 Acquiring the failure information (Custom reception)	287
B. Custom Adapter Development Framework	295
B.1 APIs of the custom adapter development framework	295
B.2 Definition files of the custom adapter development framework	311
B.3 Sample program of the custom adapter development framework	317
B.4 Custom adapter definition screen	324
B.5 Environment Settings when using a custom adapter	330
B.6 Troubleshooting	330
C. Defining the DB adapter using the DB adapter definition support function	334
C.1 Overview of Overview of the DB adapter definition support function	334
C.2 Defining Defining the DB adapter	334
C.3 Exporting and importing at the time of distributed development	350
D. Example for setting up file adapter	352
D.1 About the sample	352
D.2 DB adapter settings	354
D.3 Sample for reading XML data	356
D.4 Sample for reading CSV data	365
D.5 Sample for writing XML data	376
D.6 Sample for writing CSV data	387

E. Example for defining file operations adapter	401
E.1 When converting from the XML format to the binary format with the split processing method	401
E.2 When converting from the binary format to the XML format with the split processing method	404
F. Example for setting up business processes using HTTP reception	409
F.1 Setup example of HTTP reception (in standard mode)	409
F.2 Setup example of HTTP reception (in pass-through mode)	417
G. Example for setting up business processes using HTTP reception and HTTP adapter	420
G.1 Setup example 1 (downloading the file from the HTTP server to the HTTP client)	420
G.2 Setup example 2 (dynamically changing the connection destination of the HTTP adapter for each request)	444
H. Security settings required in FTP linkage	452
H.1 Secure connection using FTPS (FTP adapter)	452
H.2 Encryption and authentication by secure protocols (FTP inbound adapter)	453
H.3 Definition items in the property file at the time of setup (FTP inbound adapter)	454
I. Security settings required in the HTTP adapter	456
J. Glossary	458

## Index

459

# *1*

## Overview of Development of Receptions and Service Adapters

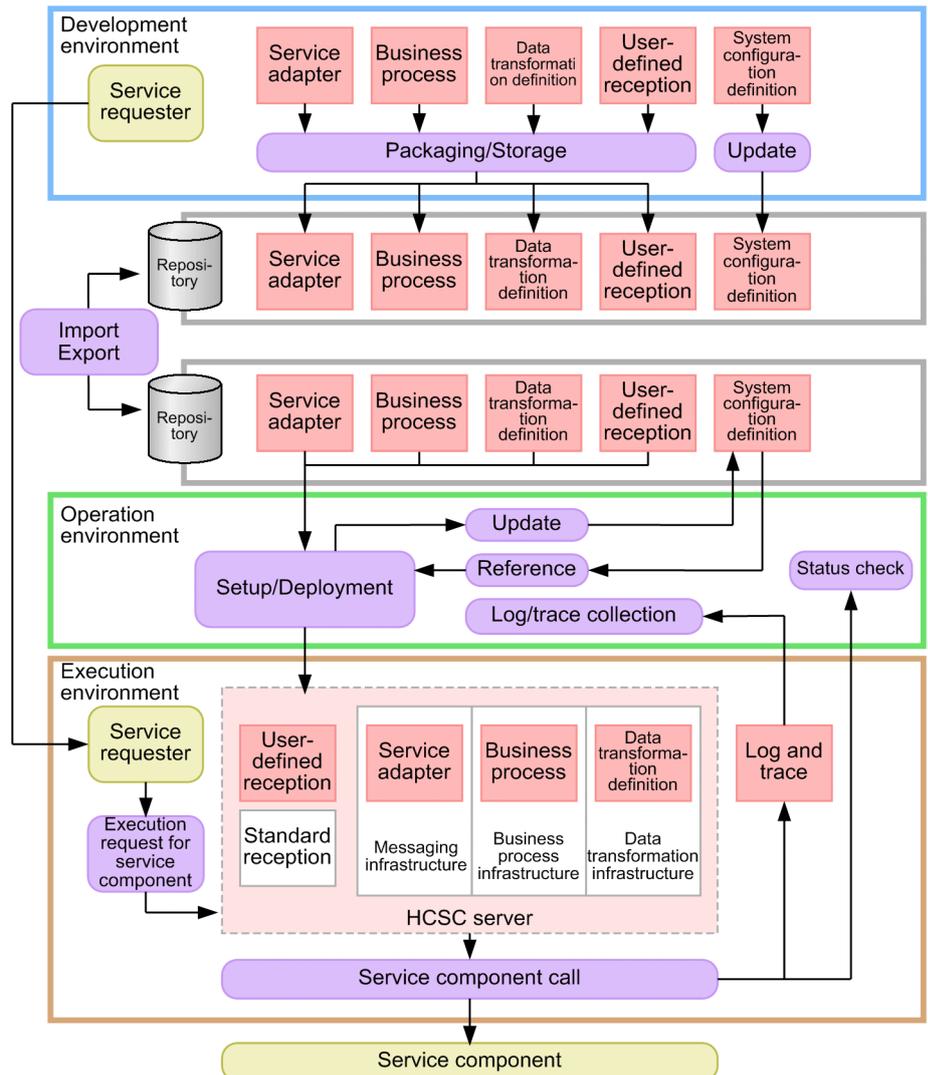
This chapter gives an overview of development of receptions and service adapters.

# 1.1 Positioning of receptions and service adapters in the whole system

The service platform receives a request message from a service requester via a reception, sends the request message to the service component defined in the service adapter, and then calls the corresponding service component.

The following figure shows the positioning of receptions and service adapters in the whole system (development environment, operating environment, and execution environment) of the service platform.

Figure 1–1: Positioning of receptions and service adapters



Legend:  
 : Data to be created or stored     $\longrightarrow$  : Data flow  
 : Operation to be executed     : Program prepared by the user

Define receptions and service adapters in the development environment. The HCSC component that contains the defined receptions and service adapters and the deployment definitions are stored in the repository, and the data is sent to the operating environment. The data sent to the operating environment is deployed to the execution environment.

This manual describes the definitions of receptions and service adapters, which must be performed in the *development environment*. For details about other operations that are performed in the *development environment*, see the *Service*

*Platform Basic Development Guide*. For details about operations that are performed in the *operation environment* and *execution environment*, see the *Service Platform Setup and Operation Guide*.

## 1.2 Receptions and service adapters that can be used

This section describes the receptions and service adapters that can be used in the service platform.

### 1.2.1 Receptions that can be used

There are two types of receptions: standard receptions, which are provided as the HCSC server functions, and user-defined receptions, for which the user can define interfaces as desired.

The following table describes the types of receptions that can be used in the service platform.

Table 1–1: Types of available receptions

Category	Type	Purpose
Standard reception	Standard reception (Web service)	Used when a system using a Web service (SOAP communication) receives a call request for a service component.
	Standard reception (Session Bean)	Used when a system using Session Bean receives a call request for a service component.
	Standard reception (MDB (WS-R))	Used when a system using MDB (WS-R) receives a call request for a service component.
	Standard reception (MDB (DB queue))	Used when a system using MDB (DB queue) receives a call request for a service component.
User-defined reception	SOAP reception	Used when a system using a Web service (SOAP communication) receives a call request for a service component.
	TP1/RPC reception	Used to receive a call request for a service component, from a service requester in an existing OpenTP1 system.
	FTP reception	Used to receive an execution request from an FTP client via the FTP inbound adapter. <sup>#1</sup>
	HTTP reception	Used to receive a call request for a service component, from an HTTP client by using HTTP communication.
	Message Queue reception	Used to receive an execution request from a JMS provider (IBM WebSphere MQ system) via the MQ resource adapter.
	Custom reception	Used to receive an execution request via an arbitrary protocol. <sup>#2</sup>

#1

Linkage with the FTP adapter is required to perform file transfer between an FTP client and the FTP server on the service platform.

#2

To receive an execution request via an arbitrary protocol, the user-defined reception processing must be run on the custom reception framework.

To use standard receptions, you need to create a service requester suitable for each type of standard reception, and create and analyze a message that corresponds to the input/output format of the service component. For details about how to create a service requester and message when a standard reception is used, see *Chapter 8. Creating Service Requesters* in the *Service Platform Basic Development Guide*.

To use a user-defined reception, define an interface in the development environment, and deploy the interface to the HCSC server. After deploying the interface to the HCSC server, start the user-defined reception. Then, a request message from a service requester can be received.

For details about the definitions necessary for using user-defined receptions, see *Chapter 2. Defining User-Defined Reception*.

## 1.2.2 Service adapters that can be used

The service platform provides various service adapters, for various purposes (such as for other systems, databases, or files). Also, general custom adapters, which are developed by using the custom adapter development framework, are provided to support systems using protocols that are not supported by the provided service adapters.

The table below describes the types of service adapters that can be used in the service platform.

For details about the definitions of individual service adapters, see *Chapter 3. Defining Adapters*.

Table 1–2: Types of available service adapters

Type	Purpose
SOAP adapter	Used to call a service component by using a Web service (SOAP communication).
Session Bean adapter	Used to call a service component by using Session Bean.
MDB (WS-R) adapter	Used to call a service component created by MDB (Message Driven Bean), by using WS-R (WS-Reliability).
MDB (DB queue) adapter	Used to call an asynchronous service component by using the DB queue.
DB adapter	Used when a database operation is used as a service component.#
TP1 adapter	Used to call a service component in an existing OpenTP1 system.
File adapter	Used to directly perform input/output operations on a file in the local disk of the HCSC server.
Object Access adapter	Used to call a service component of an existing TPBroker system (Object Wrapper system).
Message Queue adapter	Used to send and receive messages to and from an existing message queue (IBM WebSphere MQ system).
FTP adapter	Used to send and receive files between the service platform and the FTP server.
File operation adapter	Used to perform layout conversion, character code conversion, replication, deletion, compression, and expansion of files.
Mail adapter	Used to call the mail server that supports the SMTP protocol as a service component on the service platform.
HTTP adapter	Used to call a Web service (RESTful Web service) that is opened to the public in the REST style.
General custom adapter	Used to call a service component of a system using a protocol that is not supported by the service adapters provided with the service platform.

#

You need to define the database to be operated and the SQL to be executed.



# 2

## Defining User-Defined Reception

This chapter explains how to define a user-defined reception.

## 2.1 Overview of user-defined receptions

A user-defined reception can receive, from a service requester, a request in a format that corresponds to the input and output format of each service component or business process.

The following are the types of user-defined receptions:

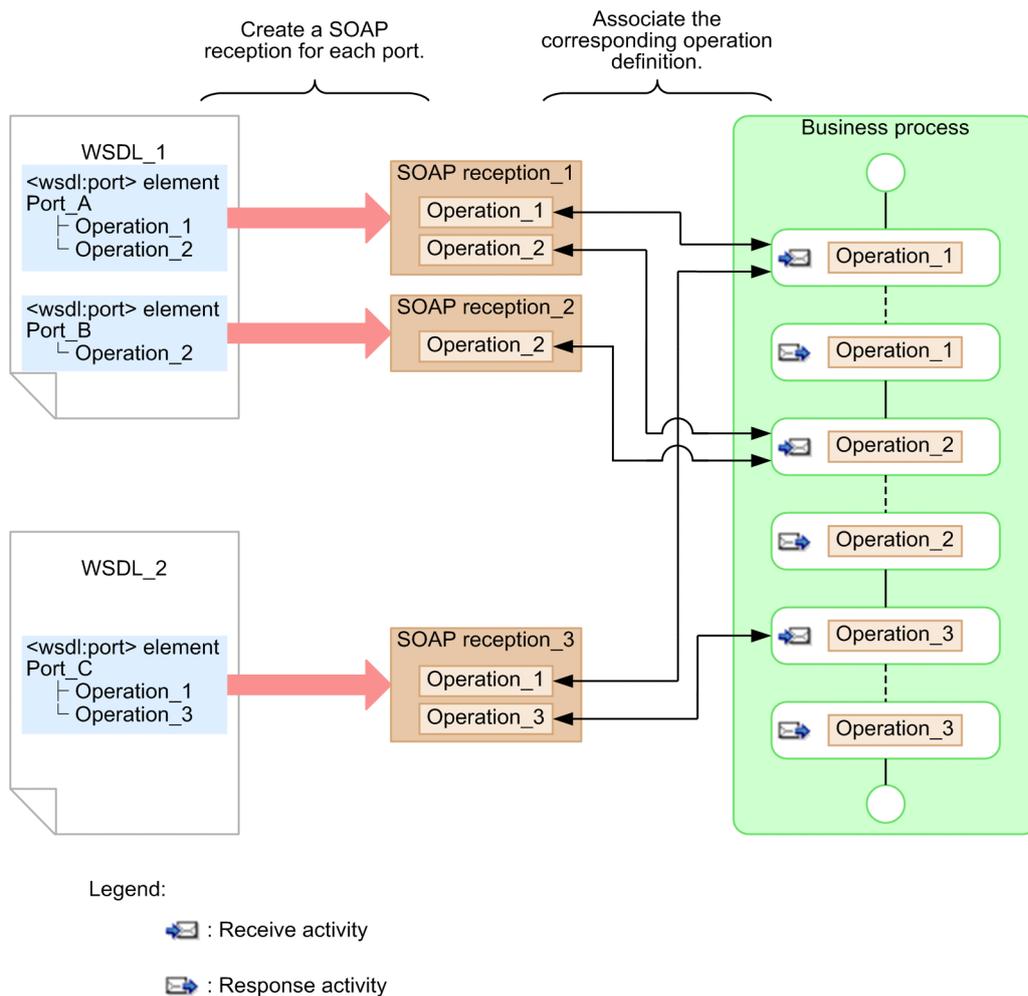
- SOAP reception
- TP1/RPC reception
- FTP reception
- HTTP reception
- Message Queue reception
- Custom reception

### 2.1.1 Overview of SOAP receptions

A SOAP reception uses a WSDL-format file specified by the user as the interface, and uses a Web service (SOAP) to communicate with the service requester. The HCSC component for which a SOAP reception can receive a request is a business process only.

The following figure shows the relationships between SOAP receptions, WSDLs, and business process definitions.

Figure 2–1: Relationships between SOAP receptions, WSDLs, and business process definitions



As shown in Figure 2-1, a SOAP reception is created for each WSDL port (`wsdl:port` element). One SOAP reception contains all operations in the `wsdl:port` element.

`Operation_2` in Figure 2-1 is contained in the two `wsdl:port` elements (`Port_A` and `Port_B`). Thus, when the same operation is contained in different `wsdl:port` elements, the same operation can be used by multiple SOAP receptions. In Figure 2-1, `Operation_2` is used by `SOAP_reception_1` and `SOAP_reception_2`.

## 2.1.2 Overview of TP1/RPC receptions

INTENTIONALLY DELETED

## 2.1.3 Overview of FTP receptions

An FTP reception receives a request from an FTP client when file transfer is performed between the FTP client and the FTP server. User-definable FTP receptions are provided in the service platform.

For details about FTP receptions, see *2.4 Defining FTP reception*.

## 2.1.4 Overview of HTTP receptions

An HTTP reception is an interface that can receive a connection request from an HTTP client without needing to go through a Web front system or SOAP reception.

For details about HTTP receptions, see *2.5 Defining HTTP reception*.

## 2.1.5 Overview of Message Queue receptions

A Message Queue reception receives an execution request from the queue of a JMS provider (IBM WebSphere MQ system) via the MQ resource adapter. The service platform provides Message Queue receptions as user-defined receptions.

For details about Message Queue receptions, see *2.6 Defining Message Queue reception*.

## 2.1.6 Overview of Custom receptions

A Custom reception receives a request from a service requester via an arbitrary interface and communication protocol. The user must implement the interface and the communication protocol when using Custom receptions.

For details about Custom reception, see *Appendix A. Custom Reception*.

## 2.2 Defining a SOAP reception

---

This section describes how to define a SOAP reception, including how to add a SOAP reception and how to create a WSDL.

### 2.2.1 Work flows for defining a SOAP reception

There are two work flows for defining a SOAP reception.

Defining a SOAP reception before defining a business process

If both of the following conditions are satisfied, first create a SOAP reception based on the WSDL, and then define the receive activity and reply activity for the business process.

- An interface with the service requester has been defined.
- A WSDL has been created that conforms with the specified interface.

In this case, the operations of the receive and reply activities and the message definitions must be defined on the business process so that they are suitable for WSDL interface input when the SOAP reception is created.

Defining a SOAP reception after defining a business process

If a business process has already been defined, define a SOAP reception that is suitable for the defined business process. Before adding a SOAP reception, you need to create a WSDL that is suitable for the message definition for the operations of the receive and reply activities for the business process and the allocated variables.

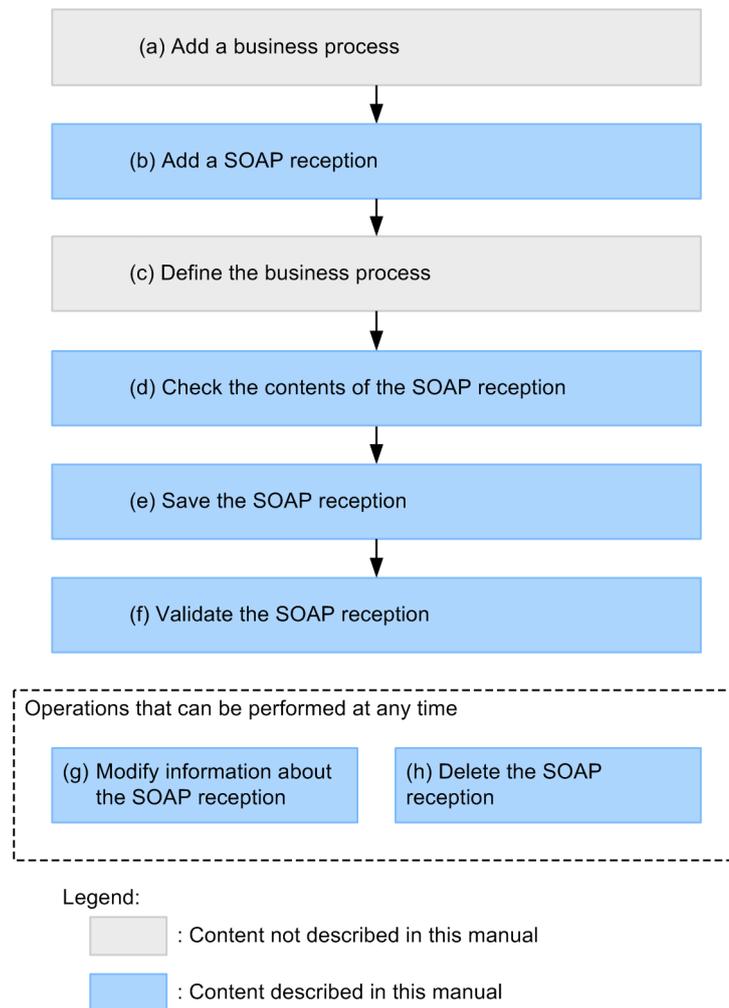
Define a SOAP reception so that the operations defined in the receive and reply activities for the business process, and the allocated variables can be called.

The following describes the work flows for defining a SOAP reception according to the above two definitions.

#### (1) Defining a SOAP reception before defining a business process

The following figure shows the work flow, in which you define a SOAP reception before defining a business process, and define the receive and reply activities (that are suitable for the interface) for the business process.

Figure 2–2: Work flow for defining a SOAP reception (when defining a SOAP reception before defining a business process)



The following describes the operations for defining a SOAP reception before defining a business process.

#### (a) Adding a business process

Create a business process for which a SOAP reception is to be defined. Define the business process so that the interface will be suitable for the contents of the SOAP reception to be defined in (b). Therefore, in this stage, you do not have to arrange activities, define variables, or enter information in the dialog box used to define detailed definitions of individual activities.

For details about how to add a business process, see *5.2.1 Adding New Business Processes* in the manual *Service Platform Basic Development Guide*.

#### (b) Adding a SOAP reception

In the Add User Defined Reception wizard, enter the created WSDL, and add a SOAP reception.

For details about how to add a new SOAP reception, see *2.2.2 Adding a SOAP reception*.

#### (c) Defining a business process

Define the receive and reply activities for all operations defined in a SOAP reception.

The procedure below defines (in the receive and reply activities) a variable with the message format defined in the SOAP reception, and sets it as the allocated variable. (The following procedure is for the receive activity. The same procedure applies in the case of reply activities.)

## 2. Defining User-Defined Reception

1. In the Receive Activity dialog box, set the operation name defined in the SOAP reception.
2. Click the **Edit** button under **Body allocated variable**.  
The List Of Variables And Correlation Sets dialog box appears.
3. Set the variable name.
4. Click the **Take In** button.  
The Take In Message Format dialog box appears. In the Take In Message Format dialog box, the following information is displayed:  
**Reception name**  
The name of the SOAP reception that has the operation specified in step 1 is set.  
**Operation name**  
The value specified in step 1 is set.  
**Message type**  
A request message is selected for a receive activity, and a response message is selected for a reply activity.  
**Message format name**  
A variable name is set.  
If the operation name specified in step 1 does not exist in the SOAP reception, a blank is displayed for the reception name and operation name. If this occurs, return to step 1, and specify the correct operation name.  
For the message format name, the variable name is set by default. Rename it as necessary.
5. Click the **OK** button.  
The Take In Message Format dialog box closes, and you are now able to make operations on the List Of Variables And Correlation Sets dialog box.
6. In the List Of Variables And Correlation Sets dialog box, click the **Add** button.
7. In the List Of Variables And Correlation Sets dialog box, add a variable to be allocated to the message header.  
For details about how to add a variable, see the description of how to define a variable in 5.5.1 *Defining Variables* in the manual *Service Platform Basic Development Guide*.
8. In the List Of Variables And Correlation Sets dialog box, select the added variable from **Variable List**.
9. In the List Of Variables And Correlation Sets dialog box, click the **OK** button.  
The List Of Variables And Correlation Sets dialog box closes.
10. In the Receive Activity dialog box, click the **Set** button under **Header allocated variable**.  
The Header allocated variable dialog box appears.
11. In the Header allocated variable dialog box, click the **Add** button.
12. In the Header allocated variable dialog box, select the added variable from the allocated variables.
13. In the Header allocated variable dialog box, click the **OK** button.

For details about the Receive Activity dialog box, see 1.4.7 *Receive Activity Dialog* in the manual *Service Platform Reference Guide*. 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 *Service Platform Reference Guide*. For details about the Take In Message Format dialog box, see 1.4.5 *Take In Message Format Dialog* in the manual *Service Platform Reference Guide*.

---

### Tip

If an operation of the SOAP reception contained a fault, for the business process, define a reply activity for which the fault name is specified. For **Fault name** in the Reply Activity dialog box, specify the value displayed for **Fault name** of **Fault message** in the user-defined reception definition window, and then assign a variable as shown in steps 2 to 9.

If multiple faults are contained in an operation of the SOAP reception, add as many reply activities as the contained faults, and set the corresponding fault names. In addition, assign variables for individual reply activities as shown in steps 2 to 9.

---

### (d) Checking the contents of a SOAP reception

Check the contents of the SOAP reception that has been added. The contents of the SOAP reception must match the contents of the business process definition.

For details about how to check a SOAP reception, see 2.8 *Checking User-Defined Reception Contents*.

**(e) Saving a SOAP reception**

Save the defined SOAP reception to the repository.

For details about how to save a SOAP reception, see *2.9 Saving a user-defined reception*.

**(f) Validating a SOAP reception**

You can validate consistency between the defined SOAP reception and the business process definition.

For details about how to validate a SOAP reception, see *2.10 Validating a User-Defined Reception*.

**(g) Changing the information of a SOAP reception**

For a defined SOAP reception, you can change the reception name, reception ID, and context root, as necessary.

For details about how to change the information of a SOAP reception, see *2.11 Changing the information of a user-defined reception*.

**(h) Deleting a SOAP reception**

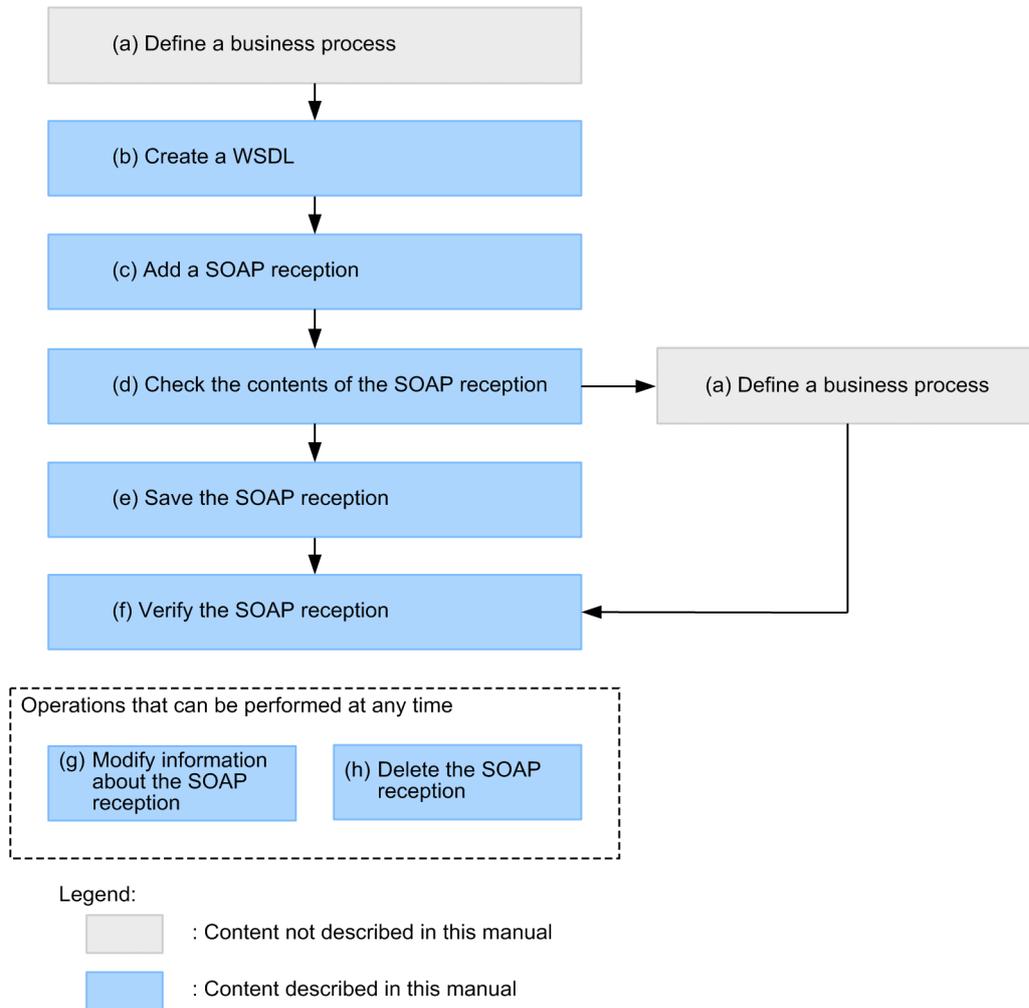
You can delete an unnecessary SOAP reception, as necessary.

For details about how to delete a SOAP reception, see *2.12 Deleting a User-Defined Reception*.

**(2) Defining a SOAP reception after defining a business process**

The following figure shows the work flow for defining a SOAP reception that is suitable for the interface of the defined business process.

Figure 2–3: Work flow for defining a SOAP reception (after defining a business process)



The following describes the operations for defining a SOAP reception after defining a business process.

**(a) Defining a business process**

Before defining a SOAP reception, you need to define a business process. Even if the business process has not yet been completely defined, arrange the receive and reply activities that form the interface with the service requester, and set the operation name and allocated variable.

For details about how to define a business process, see 5.3 *Defining Business Process Contents* in the manual *Service Platform Basic Development Guide*.

**(b) Creating a WSDL**

Create a WSDL to be used when a SOAP reception is defined.

For details about how to create a WSDL, see 2.7 *Creating WSDL*.

**(c) Adding a SOAP reception**

In the Add User Defined Reception wizard, add a SOAP reception.

For details about how to add a new SOAP reception, see 2.2.2 *Adding a SOAP reception*.

**(d) Checking the contents of a SOAP reception**

Check the contents of the SOAP reception that has been added. The contents of the SOAP reception must match the contents of the business process definition.

For details about how to check a SOAP reception, see *2.8 Checking User-Defined Reception Contents*.

(e) Saving a SOAP reception

Save the defined SOAP reception to the repository.

For details about how to save a SOAP reception, see *2.9 Saving a user-defined reception*.

(f) Validating a SOAP reception

You can validate consistency between the defined SOAP reception and the business process definition.

For details about how to validate a SOAP reception, see *2.10 Validating a User-Defined Reception*.

(g) Changing the information of a SOAP reception

For a defined SOAP reception, you can change the reception name, reception ID, and context root, as necessary.

For details about how to change the information of a SOAP reception, see *2.11 Changing the information of a user-defined reception*.

(h) Deleting a SOAP reception

You can delete unnecessary SOAP receptions.

For details about how to delete a SOAP reception, see *2.12 Deleting a User-Defined Reception*.

## 2.2.2 Adding a SOAP reception

Add a SOAP reception. To add a new SOAP reception, use the Add User Defined Reception wizard. In the Add User Defined Reception wizard, you can define which WSDL file is to be used for the SOAP reception to be added, and which `wsdl:port` element in the WSDL is to be used.

---

**Tip**

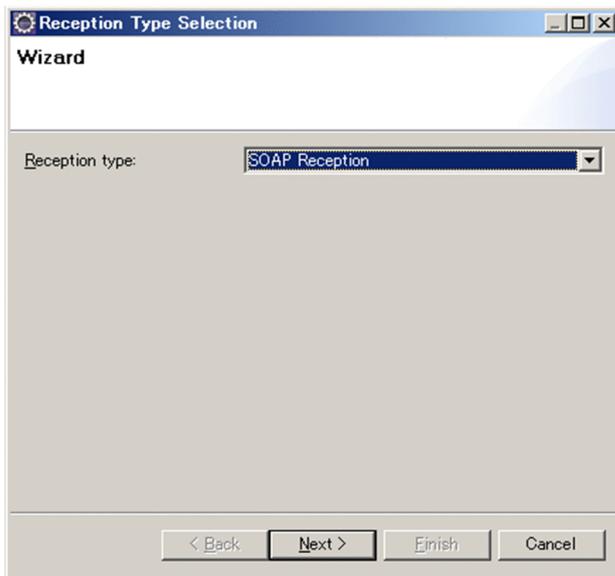
The definitions of the SOAP reception to be added need to match the definitions of the receive and reply activities of the business process. Match the definitions of the SOAP reception and the business process according to either of the following:

- Define the receive and reply activities for the business process, and then add a SOAP reception that is suitable for the operations and message definitions of the receive and reply activities.
  - Add a SOAP reception, and then change the definitions of the receive and reply activities for the business process so that the definitions are suitable for the contents of the SOAP reception.
- 

To add a new SOAP reception:

1. In **Service Definition List** in the tree view, select and then right-click the service (business process) to which a new SOAP reception is to be added.  
A pop-up menu of the service list appears.
2. From the pop-up menu, select **Add User Defined Reception**.  
The Reception Type Selection wizard appears.

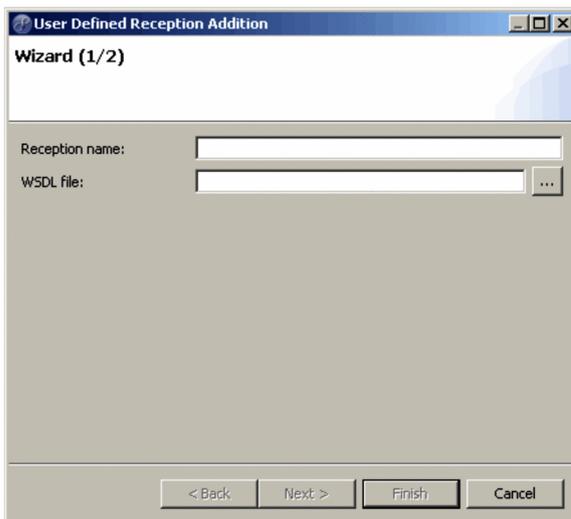
## 2. Defining User-Defined Reception



3. From the **Reception type** drop-down list, select **SOAP Reception**.

4. Click the **Next** button.

The Add User Defined Reception wizard (1/2) appears.



5. Specify the values for **Reception name** and **WSDL file**.

- **Reception name**

Specify the name of a SOAP reception, by using the NCName definition characters of the XML Schema. Specify 1 to 40 bytes. The reception name must be unique within the business process.

- **WSDL file**

Specify the WSDL file to be used for the SOAP reception, by using the absolute path. Do not specify the WSDL file, by using the relative path (e.g., `wsdlfile\wsdlfile.wsdl`) or UNC format (e.g., `\\mypc\wsdlfile\wsdlfile.wsdl`).

You can also select a WSDL file from the dialog box that is displayed after clicking the ... button.

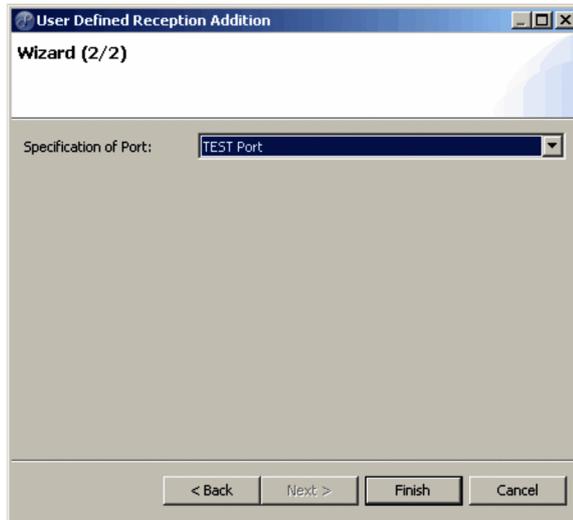
6. Click the **Next** button.

The Add User Defined Reception wizard (2/2) appears.

When you click the **Next** button, if the message `Failed to analyze the WSDL file` is displayed, there is an error in the WSDL file.

For details about the WSDL format when a Web service is used, see *2.6.1 Applicability of the service components that use Web service* in the manual *Service Platform Basic Development Guide*. For details about the WSDL

format when a user-defined reception is used, see 2.7.3 *Notes on creating WSDL of a user-defined reception* in this manual.



7. From the **Specification of Port** drop-down list, select the port to be used.

In the drop-down list, the port name (value of the `name` attribute of the `wsdl:port` element) that is defined in the WSDL file specified in step 5 is displayed.

8. Click the **Finish** button.

The SOAP reception is added to the business process, and the user-defined reception definition window appears.

For details about the user-defined reception definition window, see 1.2.6 *User-Defined Reception Definition Window* in the manual *Service Platform Reference Guide*.

#### ! Important note

If a warning message is displayed, check the contents of the displayed message by referring to the following parts in the manual *Application Server Messages*:

- If the displayed warning message begins with `KDCCC`:  
See 5.2 *Messages from KDCCC0001 to KDCCC9999*.
- If the displayed warning message begins with `KDJW`:  
See Chapter 10. *Messages from KD JW00000 to KD JW99999 (Messages Output by Cosminexus Component Container)*.

## 2.3 Defining TP1/RPC reception

---

INTENTIONALLY DELETED

### 2.3.1 Flow of defining TP1/RPC reception

INTENTIONALLY DELETED

### 2.3.2 Adding TP1/RPC reception

INTENTIONALLY DELETED

### 2.3.3 Editing definition of TP1/RPC reception

INTENTIONALLY DELETED

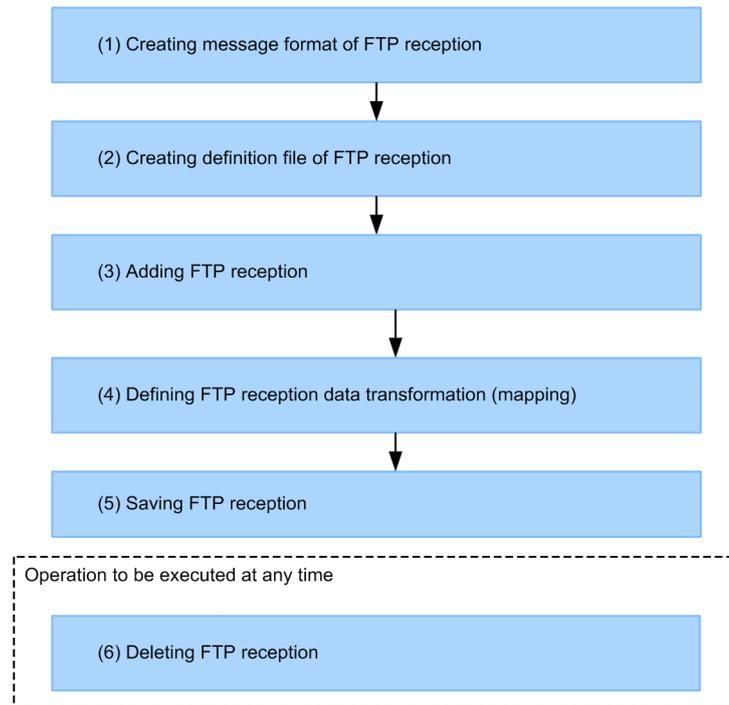
## 2.4 Defining FTP reception

This section gives details about defining FTP reception such as creating definition file for FTP reception and adding FTP reception.

### 2.4.1 Flow of defining FTP reception

The following figure shows the flow of defining FTP reception:

Figure 2-4: Flow of setting FTP reception



The activities to be performed when defining FTP reception are described here.

#### (1) Creating message format of FTP reception

Create the format of messages to be used in FTP reception.

For details on creating the message format of FTP reception, see [2.4.2 Creating message format of FTP reception](#).

#### (2) Creating definition file of FTP reception

Create the definition file to be used in FTP reception and store it at the specified location.

For details on creating the definition file of FTP reception, see [2.4.3 Creating definition file of FTP reception](#).

#### (3) Adding FTP reception

Add FTP reception using wizard and service adapter definition screen.

For details on the procedure for adding FTP reception, see [2.4.4 Adding FTP reception](#).

#### (4) Defining data transformation of FTP reception (mapping)

Define data transformation of message format definition files of the conversion source and conversion destination using the data transformation definition screen.

## 2. Defining User-Defined Reception

For details on defining data transformation, see 6. *Defining Data Transformation* in the manual *Service Platform Basic Development Guide*.

For details on the data transformation definition screen, see 1.2.5 *Data transformation definition screen* in the manual *Service Platform Reference Guide*.

### (5) Saving FTP reception

You must save the definition information of FTP reception in the repository, if necessary.

For details on saving FTP reception, see 2.9 *Saving a user-defined reception*.

### (6) Deleting FTP reception

You can delete the unwanted FTP reception, if necessary.

For details on deleting FTP reception, see 2.12 *Deleting a User-Defined Reception*.

## 2.4.2 Creating message format of FTP reception

The message format definition file of FTP reception uses the schema provided by the service platform. Therefore you need not create the message format definition file.

This subsection describes the types of message format used in FTP reception and contents of each type.

Storage location of the file is "*Installation directory of service platform\CSC\schema\ftprecp*".

### (1) Request message format of FTP reception

The operation-wise request message format of FTP reception is described here.

#### (a) PUT operation

The following table describes the request message format to be passed on to a business process from FTP reception of PUT operation. The file name is "urecp\_ftp\_put\_request.xsd". The name of the namespace is "http://www.hitachi.co.jp/soft/xml/cosminexus/csc/reception/ftp/put\_request".

Table 2-1: Request message format of FTP reception (PUT operation)

Tag name	Occurrence count #	Description
<request>	1 time	-
<ftpclient-ipaddr>	1 time	Specifies IP address of FTP client. It is set in text format of IP address.
<ftp-user>	1 time	Specifies the user name mentioned by FTP client in USER command. The information specified in the argument of USER command is set as-is.
<ftp-type>	1 time	Specifies information of the type mentioned by FTP client in TYPE command. The information specified in the argument of TYPE command is set as-is.
<transfer-type>	1 time	Specifies the transfer type. Either of the following types is set: <ul style="list-style-type: none"><li>• APPE Adding information in the file</li><li>• STOR Overwriting the file</li></ul>
<transfer-path>	1 time	Specifies path of the file to be transferred. From the path information specified by FTP client in APPE command or STOR command, the information of the file path excluding sorting definition part mentioned in the beginning is set.

Tag name	Occurrence count #	Description
<transfer-path>	1 time	"/" is set in the beginning without fail.
<file-name>	1 time	Specifies name of the intermediate file containing data of the file sent by FTP client.
<file-size>	1 time	Specifies size of the data written in the intermediate file sent by FTP client.
<request-id>	1 time	Specifies the request ID. The request ID generated by FTP reception is set.
<ftp-commands-before>	0 or 1 time	<p>Specifies the FTP command to be executed prior to transfer.</p> <p>It is set when the following FTP command is requested by FTP client:</p> <pre>SITE CSCTHR -B &lt;FTP command&gt;[;&lt;FTP command&gt; ...]</pre> <p>Here, "&lt;FTP command&gt;[;&lt;FTP command&gt; ...]" is set as-is.</p> <p>However, the FTP command specified in the argument of SITE CSCTHR command is not set, if it is not permitted in FTP command permission list definition function.</p> <p>When CSCTHR command is requested multiple times by FTP client, "&lt;FTP command&gt;[;&lt;FTP command&gt; ...]" is joined with semicolon (;) in the sequence in which SITE CSCTHR command is executed and it is added.</p> <p>If CSCTHR command is not requested even once, this tag is not output.</p>
<ftp-commands-after>	0 or 1 time	<p>Specifies the FTP command to be executed after transfer.</p> <p>It is set, when the following FTP command is requested by FTP client.</p> <pre>SITE CSCTHR -A &lt;FTP command&gt;[;&lt;FTP command&gt; ...]</pre> <p>Here, "&lt;FTP command&gt;[;&lt;FTP command&gt; ...]" is set as-is.</p> <p>However, the FTP command specified in the argument of SITE CSCTHR command is not set, if it is not permitted in the FTP command permission list definition function.</p> <p>When CSCTHR command is requested multiple times by FTP client, "&lt;FTP command&gt;[;&lt;FTP command&gt; ...]" is joined with semicolon (;) in the sequence in which SITE CSCTHR command is executed and it is added.</p> <p>If CSCTHR command is not requested even once, this tag is not output.</p>

## Legend:

-: Corresponding item does not exist.

## Note#

The operation to be performed when the specified occurrence count is exceeded is not guaranteed.

## (b) GET operation

The following table describes the request message format to be passed on to a business process from FTP reception of GET operation. The file name is "urecp\_ftp\_get\_request.xsd". The name of the namespace is "http://www.hitachi.co.jp/soft/xml/cosminexus/csc/reception/ftp/get\_request".

Table 2-2: Request message format of FTP reception (GET operation)

Tag name	Occurrence count #	Description
<request>	1 time	-
<ftpclient-ipaddr>	1 time	Specifies IP address of FTP client. It is set in text format of IP address.
<ftp-user>	1 time	Specifies the user name mentioned by FTP client in USER command.

## 2. Defining User-Defined Reception

Tag name	Occurrence count #	Description
<ftp-user>	1 time	The information specified in the argument of USER command is set-is.
<ftp-type>	1 time	Specifies information of the type mentioned by FTP client in TYPE command. The information specified in the argument of TYPE command is set as-is.
<transfer-type>	1 time	Specifies the transfer type. The following contents are set: <ul style="list-style-type: none"> <li>RETR Acquiring the file</li> </ul>
<transfer-path>	1 time	Specifies path of the file to be transferred. From the path information specified by FTP client in RETR command, the information of the file path excluding sorting definition part mentioned in the beginning is set. "/" should be set in the beginning.
<request-id>	1 time	Specifies the request ID. The request ID generated by FTP reception is set.
<ftp-commands-before>	0 or 1 time	Specifies the FTP command to be executed before transfer. It is set, when the following FTP command is requested by FTP client: SITE CSCTHR -B <FTP command>;<FTP command> ...] Here, "<FTP command>;<FTP command> ...]" is set as-is. However, the FTP command specified in the argument of SITE CSCTHR command is not set, if it is not permitted in the FTP command permission list definition function. When CSCTHR command is requested multiple times by FTP client, "<FTP command>;<FTP command> ...]" is joined with semicolon (;) in the sequence in which SITE CSCTHR command is executed and it is added. If CSCTHR command is not requested even once, this tag is not output.
<ftp-commands-after>	0 or 1 time	Specifies the FTP command to be executed after transfer. It is set, when the following command is requested by FTP client: SITE CSCTHR -A <FTP command>;<FTP command> ...] Here, "<FTP command>;<FTP command> ...]" is set as-is. However, the FTP command specified in the argument of SITE CSCTHR command is not set, if it is not permitted in the FTP command permission list definition function. When CSCTHR command is requested multiple times by FTP client, "<FTP command>;<FTP command> ...]" is joined with semicolon (;) in the sequence in which SITE CSCTHR command is executed and it is added. If CSCTHR command is not requested even once, this tag is not output.

### Legend:

-: Corresponding item does not exist.

### Note#

The operation to be performed when the specified occurrence count is exceeded is not guaranteed.

### (c) GETINFO operation

The following table describes the request message format to be passed on to a business process from FTP reception of GETINFO operation. The file name is "urecp\_ftp\_getinfo\_request.xsd". The name of the namespace is "http://www.hitachi.co.jp/soft/xml/cosminexus/csc/reception/ftp/getinfo\_request".

Table 2–3: Request message format of FTP reception (GETINFO operation)

Tag name	Occurrence count #	Description
<request>	1 time	-
<ftpclient-ipaddr>	1 time	Specifies IP address of FTP client. It is set in text format of IP address.
<ftp-user>	1 time	Specifies the user name mentioned by FTP client in USER command. The information specified in the argument of USER command is set as-is.
<getinfo-type>	1 time	Either of the following options is set as information acquisition type: <ul style="list-style-type: none"> <li>• LIST Acquires the list of file information.</li> <li>• NLST Acquires the list of file names.</li> </ul>
<getinfo-option>	0 or 1 time	Specifies the information about the information acquisition option. When all of the following conditions are fulfilled, a string same as the one set in the option definition file is set: <ul style="list-style-type: none"> <li>• The starting part of the argument specified by FTP client in LIST command or NLST command matches with the string set in the option definition file</li> <li>• A string is not continued after the starting part of the argument or there is space</li> </ul> Note that, this tag is not output, if either of the following conditions is applicable: <ul style="list-style-type: none"> <li>• If FTP client does not specify an argument in LIST command or NLST command</li> <li>• If the starting part of the argument specified by FTP client in LIST command or NLST command does not match with the string set in the option definition file.</li> </ul>
<getinfo-path>	0 or 1 time	Specifies the information about information acquisition path. When the <getinfo-option> tag is not output, the argument specified by FTP client in LIST command or NLST command is set as-is. When the <getinfo-option> tag is output, from the start of the argument specified by FTP client in LIST command or NLST command, the option information output to the <getinfo-option> tag and path information excluding the space after the option information are set. Note that, this tag is not output, if either of the following conditions is applicable: <ul style="list-style-type: none"> <li>• If FTP client does not specify an argument in LIST command or NLST command</li> <li>• If only option information is specified in the argument specified by FTP client in LIST command or NLST command</li> </ul>
<request-id>	1 time	Specifies the request ID. The request ID generated by FTP reception is set.
<ftp-commands-before>	0 or 1 time	Specifies the FTP command to be executed before executing list command. It is set, when the following FTP command is requested by FTP client: SITE CSCTHR -B <FTP command>[;<FTP command> ...] Here, "<FTP command>[;<FTP command> ...]" is set as-is. However, the FTP command specified in the argument of SITE CSCTHR command is not set, if it is not permitted in the FTP command permission list definition function. When CSCTHR command is requested multiple times by FTP client, "<FTP command>[;<FTP command> ...]" is joined with semicolon (;) in the sequence in which SITE CSCTHR command is executed and it is added. If CSCTHR command is not requested even once, this tag is not output.

## 2. Defining User-Defined Reception

Tag name	Occurrence count #	Description
<ftp-commands-after>	0 or 1 time	<p>Specifies the FTP command to be executed after execution of list command.</p> <p>It is set, when the following command is requested by FTP client:</p> <pre>SITE CSCTHR -A &lt;FTP command&gt;[;&lt;FTP command&gt; ...]</pre> <p>Here, "&lt;FTP command&gt;[;&lt;FTP command&gt; ...]" is set as-is.</p> <p>However, the FTP command specified in the argument of SITE CSCTHR command is not set, if it is not permitted in the FTP command permission list definition function.</p> <p>When CSCTHR command is requested multiple times by FTP client, "&lt;FTP command&gt;[;&lt;FTP command&gt; ...]" is joined with semicolon (;) in the sequence in which SITE CSCTHR command is executed and it is added.</p> <p>If CSCTHR command is not requested even once, this tag is not output.</p>

### Legend:

-: Corresponding item does not exist.

### Note#

The operation to be performed when the specified occurrence count is exceeded is not guaranteed.

## (2) Response message format of FTP reception

The operation-wise response message format of FTP reception is described here.

### (a) PUT operation

The following table describes the response message format to be passed on to FTP reception of PUT operation from a business process. The file name is "urecp\_ftp\_put\_response.xsd". The name of the namespace is "http://www.hitachi.co.jp/soft/xml/cosminexus/csc/reception/ftp/put\_response".

Table 2-4: Response message format of FTP reception (PUT operation)

Tag name	Occurrence count #	Description
<response>	1 time	-
<message>	0 or 1 time	<p>Specifies a message indicating end of a business process. Maximum character length that can be specified in a message is 1,024.</p> <p>If length of the message specified by you exceeds the maximum value, the last part of the message is truncated so that the message fits in 1,024 characters.</p> <p>If a message contains a character that cannot be used<sup>#2</sup>, such character is replaced with a question mark (?).</p> <p>If you omit the message, the following message is set:</p> <ul style="list-style-type: none"> <li>Value of the &lt;success&gt; tag is true Transfer complete.</li> <li>Value of the &lt;success&gt; tag is false Requested action aborted.</li> </ul>
<success>	0 or 1 time	<p>Specifies a flag indicating end of a business process. Specify either of the following flags:</p> <ul style="list-style-type: none"> <li>true Processing of the business process ended successfully.</li> <li>false An error occurred in the processing of the business process.</li> </ul> <p>If you omit the flag, true is set.</p>

Legend:

-: Corresponding item does not exist.

Note#1

The operation to be performed when the specified occurrence count is exceeded is not guaranteed.

Note#2

You can use only the following characters:

one-byte alphanumeric characters, one-byte space, one-byte tab, one-byte linefeed, exclamation mark (!), double-quotation mark ("), number sign (#), dollar sign (\$), percentage sign (%), ampersand (&), apostrophe ('), left parenthesis ( ( ), right parenthesis ( ) ), asterisk (\*), positive sign (+), comma ( , ), hyphen (-), period ( . ), slash ( / ), colon ( : ), semi-colon ( ; ), left-angle bracket (<), right-angle bracket (>), equal sign (=), question mark (?), at sign (@), left-square bracket ( [ ), right-square bracket ( ] ), back slash (\), circumflex accent (^), underscore ( \_ ), accent grave (`), left curly bracket ( { ), right curly bracket ( } ), pipeline ( | ), wave dash (~)

## (b) GET operation

The following table describes the response message format to be passed on to FTP reception of GET operation from a business process. The file name is "urecp\_ftp\_get\_response.xsd". The name of the namespace is "http://www.hitachi.co.jp/soft/xml/cosminexus/csc/reception/ftp/get\_response".

Table 2-5: Response message format of FTP reception (GET operation)

Tag name	Occurrence count#1	Description
<response>	1 time	-
<message>	0 or 1 time	Specifies a message indicating end of a business process. Maximum character length that can be specified in a message is 1,024. If length of the message specified by you exceeds the maximum value, the last part of the message is truncated so that the message fits in 1,024 characters. If a message contains a character that cannot be used <sup>#2</sup> , such character is replaced with a question mark (?). If you omit the message, the following message is set: <ul style="list-style-type: none"> <li>Value of the &lt;success&gt; tag is true Transfer complete.</li> <li>Value of the &lt;success&gt; tag is false Requested action not taken.</li> </ul>
<success>	0 or 1 time	Specifies a flag indicating end of a business process. Specify either of the following flags: <ul style="list-style-type: none"> <li>true Processing of the business process ended successfully.</li> <li>false An error occurred in the processing of the business process.</li> </ul> If you omit the flag, true is set.
<file-name>	0 or 1 time	Specifies name of the intermediate file in which data of the file to be resent to FTP client is mentioned. However, if value of the <success> tag is false, this setting is not required. Even if this setting is performed, it is ignored.
<file-size>	0 or 1 time	Specifies size of data of the file to be resent to FTP client. However, if value of the <success> tag is false, this setting is not required. Even if this setting is performed, it is ignored.

Legend:

-: Corresponding item does not exist.

Note#1

The operation to be performed when the specified occurrence count is exceeded is not guaranteed.

## Note#2

You can use only the following characters:

one-byte alphanumeric characters, one-byte space, one-byte tab, one-byte linefeed, exclamation mark (!), double-quotation mark ("), number sign (#), dollar sign (\$), percentage sign (%), ampersand (&), apostrophe ('), left parenthesis ( ( ), right parenthesis ( ) ), asterisk (\*), positive sign (+), comma ( , ), hyphen (-), period ( . ), slash (/), colon (:), semi-colon (;), left-angle bracket (<), right-angle bracket (>), equal sign (=), question mark (?), at sign (@), left-square bracket ([), right-square bracket (]), back slash (\), circumflex accent (^), underscore (\_), accent grave (`), left curly bracket ({), right curly bracket (}), pipeline (|), wave dash (~)

## (3) GETINFO operation

The following table describes the response message format to be passed on to FTP reception of GETINFO operation from a business process. The file name is "urecp\_ftp\_getinfo\_response.xsd". The name of the namespace is "http://www.hitachi.co.jp/soft/xml/cosminexus/csc/reception/ftp/getinfo\_response".

Table 2-6: Response message format of FTP reception (GETINFO operation)

Tag name	Occurrence count #1	Description
<response>	1 time	-
<reply-code>	0 or 1 time	<p>Specifies the reply code received from FTP server when information acquisition process is executed (or code based on this).</p> <p>When executing the information acquisition process using GETINFO operation of FTP adapter, it is recommended that you set the value of &lt;reply-code&gt; of response message of FTP adapter.</p> <p>Value that can be specified in this tag is a number between 200 to 299 or 400 to 599.</p> <p>If you set a number between 200 to 299, the reply code that is returned to FTP client is 266.</p> <p>If you set a number between 400 to 599, the set value is the reply code that is returned to FTP client.</p> <p>If value of the &lt;success&gt; tag is true and if you omit this tag, FTP reception assumes 226 in the reply code.</p> <p>If value of the &lt;success&gt; tag is false, this setting is not required. Even if this setting is performed, it is ignored.</p>
<message>	0 or 1 time	<p>Specifies the reply message received from FTP server when the information acquisition process is executed or the message indicating end of a business process.</p> <p>Maximum character length that can be specified in a message is 1,024. If length of the message specified by you exceeds the maximum value, the last part of the message is truncated so that the message fits in 1,024 characters.</p> <p>If a message contains a character that cannot be used<sup>#2</sup>, such character is replaced with a question mark (?).</p> <p>If you omit the message, the following message is set:</p> <ul style="list-style-type: none"> <li>Value of the &lt;success&gt; tag is true Transfer complete.</li> <li>Value of the &lt;success&gt; tag is false Requested action not taken.</li> </ul> <p>When executing the information acquisition process using GETINFO operation of FTP adapter, it is recommended that you set the value of &lt;reply-message&gt; of response message of FTP adapter.</p>
<success>	0 or 1 time	<p>Specifies the flag indicating end of a business process. Specify either of the following flags:</p> <ul style="list-style-type: none"> <li>true Processing of the business process ended successfully.</li> <li>false An error occurred in the processing of a business process.</li> </ul>

Tag name	Occurrence count #1	Description
<success>	0 or 1 time	If you omit this, true is set.
<file-name>	0 or 1 time	<p>Specifies name of the intermediate file in which data of the file to be resent to FTP client is mentioned.</p> <p>The information acquired in the following format must be mentioned in the file specified in this tag:</p> <ul style="list-style-type: none"> <li>• Character code: UTF-16</li> <li>• Linefeed code: CRLF</li> </ul> <p>However, when either of the following condition is applicable, this setting is not required. Even if this setting is performed, it is ignored.</p> <ul style="list-style-type: none"> <li>• If value of the &lt;success&gt; tag is false</li> <li>• If value of the &lt;success&gt; tag is true and value of &lt;reply-code&gt; is between 400 to 599</li> </ul>

## Legend:

-: Corresponding item does not exist.

## Note#1

The operation to be performed when the specified occurrence count is exceeded is not guaranteed.

## Note#2

You can use only the following characters:

one-byte alphanumeric characters, one-byte space, one-byte tab, one-byte linefeed, exclamation mark (!), double-quotation mark ("), number sign (#), dollar sign (\$), percentage sign (%), ampersand (&), apostrophe ('), left parenthesis ( ( ), right parenthesis ( ) ), asterisk (\*), positive sign (+), comma ( , ), hyphen (-), period ( . ), slash ( / ), colon ( : ), semi-colon ( ; ), left-angle bracket ( < ), right-angle bracket ( > ), equal sign ( = ), question mark ( ? ), at sign ( @ ), left-square bracket ( [ ), right-square bracket ( ] ), back slash ( \ ), circumflex accent ( ^ ), underscore ( \_ ), accent grave ( ` ), left curly bracket ( { ), right curly bracket ( } ), pipeline ( | ), wave dash ( ~ )

## 2.4.3 Creating definition file of FTP reception

This subsection describes types of definition file of FTP reception and creation method of each type.

### (1) Types of definition file

Types of the definition file to be created are as follows:

- FTP reception definition file  
This file defines the contents for which settings can be changed after FTP reception is set up.
- FTP execution permission list definition file  
This file defines the users having permission for executing a business process and the users not having this permission. Creation of this file is optional.
- FTP command permission list definition file  
This file defines the FTP commands, which are allowed to be executed before transfer or after transfer. Creation of this file is optional.
- List command option definition file  
This file defines the strings that can be specified as options in the argument of list command. Creation of this file is optional.
- FTP reception config file  
This file defines the contents whose setting can be changed after FTP reception is set up and the information whose setting might be required to be changed frequently. Creation of this file is optional.

### (2) Creating the definition file

The procedure for creating the definition file to be used in FTP reception is described here. Create a definition file other than list command option definition file using the template file provided by FTP reception.

#### (a) FTP reception definition file

When adding FTP reception, edit and use the template file provided by service platform (`cscurecpftp.properties`). Therefore, you need not create FTP reception definition file.

#### (b) FTP execution permission list definition file

1. Copy the template file (*Installation directory of service platform\CSC\config\ftprecv\templates\ftp\_permission\_allow.properties*) and store it in any directory.
2. Change the name of the template file, which is copied to any name.
3. Edit the definition contents and save them.

For details on the items that can be edited in FTP execution permission list definition file, see *FTP execution permission list definition file* in the manual *Service Platform Reference Guide*.

The FTP execution permission list definition file is reflected in the execution environment when starting FTP reception.

#### (c) FTP command permission list definition file (for FTP reception)

1. Copy the template file (*Installation directory of service platform\CSC\config\ftprecv\templates\ftp\_command\_allow.properties*) and store it in any directory.
2. Change the name of the template file, which is copied to any name.
3. Edit the definition contents and save them.

For details on the items that can be edited in the FTP command permission list definition file, see *FTP command permission list definition file* in the manual *Service Platform Reference Guide*.

The FTP command permission list definition file is reflected in the execution environment when starting FTP reception.

#### (d) List command option definition file

1. Create a text file with any name and store it in any directory.
2. Edit the definition contents and save them.

For details on the items that can be edited in the list command option definition file, see *List command option definition file* in the manual *Service Platform Reference Guide*.

The list command option definition file is reflected in the execution environment when starting FTP reception.

#### (e) FTP reception config file

1. Copy the template file (*Installation directory of service platform\CSC\config\ftprecv\templates\ftprecv\_config.properties*) and store it in the following directory:  
*Installation directory of service platform\CSC\config\ftprecv*
2. Change the name of the template file, which is copied to "<FTP reception ID>.properties".
3. Edit the definition contents and save them.

For details on the items that can be edited in the FTP reception config file, see *FTP reception config file* in the manual *Service Platform Reference Guide*.

The FTP reception config file is reflected in the execution environment when starting FTP reception.

## 2.4.4 Adding FTP reception

The procedure for adding new FTP reception is as follows:

1. From Eclipse menu, select [Window] - [View display] - [Others].  
[View display] dialog is displayed.
2. Select [HCSC-Definer] - [HCSCTE view] and click the [OK] button.  
Service definition list is displayed in tree view.
3. Right-click a business process in tree view and select "Add user-defined reception".  
Reception-wise selection wizard is displayed.
4. From reception-wise drop-down list, select "FTP reception".  
If you click the [Next] button, a dialog box is displayed for entering information required to add FTP reception.
5. Specify [Reception name].  
Specify FTP reception name within 1 to 40 bytes.
6. Click the [Finish] button.  
If you click the [Finish] button, the required file is created and it is saved in the repository. Service adapter definition screen (basic) is displayed.
7. Set the definition information of FTP reception.

The following table describes the setting items of FTP reception:

Table 2-7: Setting items of user-defined reception definition of FTP reception screen (basic)

Type	Item		Setting contents	Setting
User-defined reception information	Reception name		The set reception name is displayed.	A
	Reception ID		Reception ID is displayed. Change it, if necessary.	A
	Name of default operation		Select the operation to be invoked by default.	A
	Operation		Add the operation to be used from the following operations: <ul style="list-style-type: none"> <li>• PUT: Transfer a file from FTP client to FTP server</li> <li>• GET: transfer a file from FTP server to FTP client</li> <li>• GETINFO: Acquire information of FTP server and send it to FTP client</li> </ul>	A
Operation information	Communication model		Set "Synchronization".	A
Request message	Reception	Message format	The format definition file for the following request messages is set as per the setting of "Operation": <ul style="list-style-type: none"> <li>• In case of PUT: urecp_ftp_put_request.xsd<sup>#</sup></li> <li>• In case of GET: urecp_ftp_get_request.xsd<sup>#</sup></li> <li>• In case of GETINFO: urecp_ftp_getinfo_request.xsd<sup>#</sup></li> </ul>	N
	Service component	"Use" checkbox	Check this checkbox, if necessary.	Y
	Data transformation definition		When you specify message format of a service component, set the data transformation definition of request message from reception to a service component.	Y
Response message	Reception	Message format	The format definition file for the following response messages is set as per the setting of "Operation": <ul style="list-style-type: none"> <li>• In case of PUT: urecp_ftp_put_response.xsd<sup>#</sup></li> <li>• In case of GET: urecp_ftp_get_response.xsd<sup>#</sup></li> <li>• In case of GETINFO: urecp_ftp_getinfo_response.xsd<sup>#</sup></li> </ul>	N
	Service component	"Use" checkbox	Check this checkbox, if necessary.	Y

## 2. Defining User-Defined Reception

Type	Item	Setting contents	Setting
Response message	Data transformation definition	When you specify message format of a service component, set the data transformation definition of response message from a service component to reception.	Y

Legend:

- A: Set this item without fail.
- Y: Setting of this item is optional.
- N: Confirm the displayed contents.

Note#

The directory in which the file is to be stored is "*Installation directory of service platform\CSC\schema\ftp\prec*".  
For details on the setting items of user-defined reception definition screen other than the above-mentioned items, see *1.2.6 User-Defined Reception Definition Window* in the manual *Service Platform Reference Guide*.

8. Click the user-defined reception (Details) tab.

User-defined reception definition screen (details) is displayed.

9. Set the definition information of FTP reception.

The following table describes the setting items of FTP reception:

Table 2–8: Setting items of user-defined reception definition screen (details) of FTP reception

Type	Item	Setting contents	Setting
User-defined reception control information	Self-defined file	Confirm that the following self-defined file is set: <ul style="list-style-type: none"> <li>• <i>cscrecftp.properties</i><sup>#1</sup></li> </ul>	Y
	EAR file	Confirm that the following EAR file is set: <ul style="list-style-type: none"> <li>• <i>cscmsg_urecp_ftp.ear</i><sup>#2</sup></li> </ul>	N

Legend:

- Y: Setting of this item is optional.
- N: Confirm the displayed contents.

Note#1

*cscrecftp.properties*, which is being set, is a template file. Select *cscrecftp.properties*, if necessary and click the [Edit] button. Modify contents of self-defined file. For details on the property files, see *FTP reception definition file* in the manual *Service Platform Reference Guide*.

Note#2

The directory in which the file is to be stored is "*Installation directory of service platform\CSC\lib*".

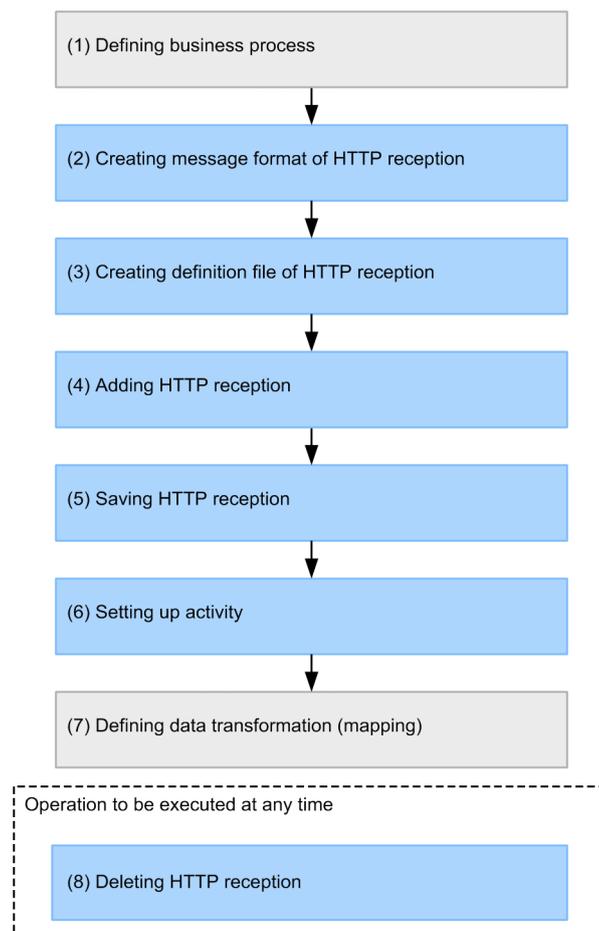
## 2.5 Defining HTTP reception

This section gives details about how to define HTTP reception such as creating message format of HTTP reception and adding the HTTP reception. When you want to confirm operations of HTTP reception based on the actual setting example, see *Appendix F Example for setting up business processes using HTTP reception*.

### 2.5.1 Flow of defining HTTP reception

The following figure shows the flow of defining HTTP reception:

Figure 2–5: Flow of defining HTTP reception



Activities to be performed when defining HTTP reception are described here.

#### (1) Defining a business process

Add a business process using the wizard and a defined business process. Define the added business in the business process definition screen. For details on the definition of a business process, see *5. Defining a Business Process* in the manual *Service Platform Basic Development Guide*.

Also, you must define a service adapter to invoke a service component from a business process. For details on defining a service adapter, see *3. Defining Adapters*.

#### (2) Creating message format of HTTP reception

Create the request message format and the response message format to be used in the HTTP reception.

## 2. Defining User-Defined Reception

For details on how to create a message format, see [2.5.2 Creating message format of HTTP reception](#).

### (3) Creating definition file of HTTP reception

Create the definition file to be used in the HTTP reception. The created definition file is reflected in the repository.

For details on how to create the definition file to be used in HTTP reception, see [2.5.3 Creating definition file of HTTP reception](#).

### (4) Adding HTTP reception

Add HTTP reception in the service definition list in tree view and define its contents.

For details on how to add and define the HTTP reception, see [2.5.4 Adding HTTP reception](#).

### (5) Saving HTTP reception

Save HTTP reception that is added and defined. For details on how to save HTTP reception, see [2.9 Saving a user-defined reception](#).

### (6) Setting an activity

Set the body variables and header variables belonging to request message format/response message format of HTTP reception in the activity of a business process.

For details on how to set an activity, see [2.5.5 Setting up an activity](#).

### (7) Defining data transformation (mapping)

If data transformation is required while receiving and sending a message, define data transformation. For details on how to define data transformation, see [6. Defining Data Transformation](#) in the manual *Service Platform Basic Development Guide*.

For details on the example of mapping in case of using HTTP reception, see [Appendix F Example for setting up business processes using HTTP reception](#).

### (8) Deleting HTTP reception

You can delete the unwanted HTTP reception, if necessary.

For details on deleting HTTP reception, see [2.12 Deleting a User-Defined Reception](#).

## 2.5.2 Creating message format of HTTP reception

This subsection describes types of message format to be used in HTTP reception, format and creation method of each type.

### (1) Types of message format

The following table describes the types of message format to be used in HTTP reception. You can create a message format using XML schema provided by service platform.

Table 2–9: Types of message format

Major classification	Minor classification	Description
Request message format	For header variable	Set the header information and URL information of HTTP request.
	For body variable (main)	Set the body information of HTTP request.
	For body variable (details)	It is referenced as a part of request message format (body variable (main)).

Major classification	Minor classification	Description
Response message format	For header variable	Set the header information of HTTP response.
	For body variable	Set the body information of HTTP response.

Create a message format other than the above-mentioned to use validation activity in a business process.

## (2) Message format

The request message format and response message format to be used in HTTP reception is described here.

### (a) urecp\_http\_header\_request.xsd(request message format(for header variable))

The following table describes the request message format of header variable, which is passed on to HTTP reception, when HTTP reception is invoked by a business process. The name of the namespace is "http://www.hitachi.co.jp/soft/xml/cosminexus/csc/reception/http/request".

Table 2–10: Request message format (for header variable)

Tag name	Type	Occurrence count	Description
<http-header-request>	-	1 time	-
<auth>	-	0 or 1 time	When HTTP header authentication is specified, this tag is created.
<username>	string	1 time	Name of the authenticated user is set. Information about the user name is decoded in plain text and it is set.
<method>	string	1 time	HTTP method used in HTTP reception (GET or POST) is set.
<url>	-	1 time	Location information and query string are set from the request URL.
<location>	string	1 time	Location information is set.
<query>	string	0 or 1 time	Request string is set.
<content-type>	string	0 or 1 time	Media type specified in Content-Type header is set. Charset attribute (character code) specified in Content-Type header is set in charset attribute of content-type tag. This attribute is not set, if charset attribute is not provided.
<request-id>	string	0 or 1 time	Request ID generated by HTTP reception is set.
<files>	-	0 or 1 time	List of intermediate files generated within the working folder is set based on the multi-part data received from HTTP client.
<file>	-	More than 0 time	The intermediate files generated within the working folder are set based on the multi-part data received from HTTP client.
<partID>	string	0 or 1 time	Identifier for differentiating each part is set. This part ID corresponds to name attribute of Content-Disposition header of each

## 2. Defining User-Defined Reception

Tag name		Type	Occurrence count	Description
	<partID>	string	0 or 1 time	part within multi-part data received by HTTP reception.
	<file-name>	string	0 or 1 time	File name of the file data received from HTTP client is set. This file name corresponds to filename attribute of Content-Disposition header of each part within multi-part data received by HTTP reception. In case of path specification, only the file name is set.
	<local-file-name>	string	1 time	Name of the intermediate file generated within the working folder is set. Name of the intermediate file is unique within the working folder.
	<content-type>	string	0 or 1 time	Media type of the file data received by HTTP reception is set. This content type corresponds to the value of Content-Type header of each part within the multi-part data received by HTTP reception. If Content-Type header of each part is omitted, this tag is not generated. Charset attribute (character code) specified in Content-Type header of each part is set in charset attribute of content-type tag. This attribute is not set, if charset attribute is not provided.
	<http-header>	any	0 or 1 time	The HTTP header information received by HTTP reception is converted into header variable and it is set. For details, see 2.13.4 HTTP reception request process in the manual Service Platform Overview.

Legend:

-: Corresponding item does not exist.

### (b) urecp\_http\_body\_request.xsd(request message format (for body variable))

The following table describes the request message format of the body variable passed on to HTTP reception, when HTTP reception is invoked by a business process. Name of each namespace is "http://www.hitachi.co.jp/soft/xml/cosminexus/csc/reception/http/request".

Table 2-11: Request message format (for body variable)

Tag name		Type	Occurrence count	Description
	<http-body-request>	-	1 time	-
	<parameter>	http-body-requestType	More than 0 time	The information of the request body is converted into body variable and it is set. For details, see 2.13.4 HTTP reception request process in the manual Service Platform Overview.

Legend:

-: Corresponding item does not exist.

## (c) urecp\_http\_header\_response.xsd(response message format (for header variable))

The following table describes the response message format of the header variable, which is passed on, when a business process is invoked by HTTP reception. Name of the namespace is "http://www.hitachi.co.jp/soft/xml/cosminexus/csc/reception/http/response".

Table 2-12: Response message format

Tag name	Type	Occurrence count	Description
<http-header-response>	-	1 time	-
<status-code>	string	0 or 1 time	Specify the status code of HTTP response.
<content-type>	string	0 or 1 time	Specify Content-Type header of the data indicated in the response message (for body variable) of HTTP reception. If null is specified, it is considered that this element is omitted.  If data type is multi-part, the content type is to be used as Content-Type header to be provided for response message (for body variable) converted into part.  Specify character code which is used as charset attribute of Content-Type header in charset attribute of content-type element. If null is specified, it is considered that this element is omitted. If charset attribute or any other attribute is specified in the value of content-type element, operation is not guaranteed.
<body-partID>	string	0 or 1 time	When sending multi-part data to HTTP client, specify the value of name attribute of Content-Disposition header to be provided for response message (for body variable) converted into part.  If you omit value of body-partID element and store response message (for body variable) in multi-part data, "csc-body-text" is set in the value of this name attribute.  If the data type is not multi-part, this value is ignored.
<ignore-bodymsg>	boolean	0 or 1 time	Ignore response message (body) and specify whether to return HTTP response to HTTP client.  If you specify true in the value of this element, HTTP response is returned to HTTP client and response message (body) is ignored. If false is specified, response message (body) is used.  If you do not specify this element or if you specify an invalid string in this element, the value specified in httprecp.response.ignore-bodymsg property of HTTP reception definition file is applied.  If you specify a value in both ignore-bodymsg element of response message (header) and httprecp.response.ignore-bodymsg property of HTTP reception definition file, specification of response message (header) gets a priority.
<files>	string	0 or 1 time	Specify a list of intermediate files to be read from the working folder and to be sent to HTTP client.
<file>	-	More than 0 time	Specify an intermediate file to be read from the working folder and to be sent to HTTP client.
<partID>	string	0 or 1 time	Specify an identifier for identifying each part, when the data type is multi-part.

## 2. Defining User-Defined Reception

Tag name	Type	Occurrence count	Description
<partID>	string	0 or 1 time	<p>This is a string to be provided for the value of name attribute of Content-Disposition header with each part unit of multi-part data in HTTP response.</p> <p>If this tag is omitted, the value of local-file-name tag is set.</p> <p>When the value of ignore-bodymsg element is "true" or httprecp.response.ignore-bodymsg property of HTTP reception definition file is "true" (setting to ignore response message (body)) and the intermediate file to be sent to HTTP client is 1, specification of this tag is ignored since it is not multi-part type.</p>
<file-name>	string	0 or 1 time	<p>Specify the file name, when it is to be downloaded by HTTP client.</p> <p>This is a string to be provided for the value of filename attribute of Content-Disposition header in each part of multi-part data in HTTP response.</p> <p>If you omit this tag, filename attribute is not set.</p>
<local-file-name>	string	1 time	<p>Specify name of the intermediate file within the working folder to be downloaded to HTTP client.</p> <p>File having name specified in this tag is read from the working folder and it is set in body part of HTTP response or body part of each part of multi-part data.</p> <p>If you omit this tag, a system exception occurs.</p>
<content-type>	string	0 or 1 time	<p>Specify media type of the file to be sent HTTP client.</p> <p>This is a value to be provided to Content-Type header in each part of multi-path data in HTTP response.</p> <p>If you omit this tag, the media type is not set. If you specify null, it is considered that specification of this element is omitted.</p> <p>Specify the character code to be used as charset attribute of Content-Type header in charset attribute of content-type element. If you specify null, it is considered that specification of this element is omitted. If you specify charset attribute or any other attribute in the value of content-type element, the operation is not guaranteed.</p>
<http-header>	any	0 or 1 time	<p>Specify the information of HTTP response header. Specify this tag as a child element of &lt;http-header&gt; by considering the element name as header field and the value as header value.</p> <p>If you do not specify, the information of HTTP response header definition file is used as HTTP response header.</p>

Legend:

-: Corresponding item does not exist.

### (d) urecp\_http\_dummy\_body\_response.xsd (response message format (for body variable))

The following table describes the response message format of the body variable to be passed on when HTTP reception invokes a business process. The name of the namespace is "http://www.hitachi.co.jp/soft/xml/cosminexus/csc/reception/http/response".

Note that this message format is the dummy format of the response message format (for body variable). Use this message format, when either the value of ignore-bodymsg element of response message (header) is "true" or the value of httprecp.response.ignore-bodymsg property of HTTP reception definition file is "true" (settings for ignoring response message (body)).

In case of settings for ignoring response message (body), if you do not set response message format (for body variable) in HTTP reception, the business process cannot return a variable, since the variable of response message (body) does not exist. Therefore, it is recommended that you set this dummy format in HTTP reception, create a variable of response message (body), and assign it to reply activity.

Table 2–13: Request message format (for body variable) (in case of setting for ignoring response message (body))

Tag name	Type	Occurrence count	Description
<http-body-response>	-	1 time	-
<any>	string	More than 0 time	-

Legend:

-: Corresponding item does not exist.

In case of setting for not ignoring the response message (body), the user must create response message format (for body variable) for HTTP response.

### (3) Method of creating message format

Method of creating message format to be used in HTTP reception is described here.

XML schema of the message format provided by service platform is stored in "*Installation directory of service platform\CSC\custom-reception\http\schema*". When editing this schema, first copy it at any location in each folder.

The following table describes relation of XML schema to be used with the message format:

Table 2–14: Relation of XML schemas to be used with the message format

Message format	XML schema to be used	Editing by user	Description
Request message format (for header variable)	urecp_http_header_request.xsd	Not required	Use XML schema file provided by service platform as-is.
Request message format (for body variable (main)) <sup>#</sup>	urecp_http_body_request.xsd	Not required	Use XML schema file provided by service platform as-is.
Request message format (for body variable (details)) <sup>#</sup>	urecp_http_body_detail_request.xsd	Optional	When using value of request message (body), edit the XML schema file provided by service platform partially and use them. When you do not use value of request message (body), do not use the XML schema file provided by service platform.
Response message format (for header variable)	urecp_http_header_response.xsd	Not required	Use the XML schema file provided by service platform as-is.
Response message format (for body variable) <sup>#</sup>	urecp_http_dummy_body_response.xsd	Not required	Use the XML schema file provided by service platform as-is. It is recommended that you this file in case of settings for ignoring response message (body).
	To be created by user	Required	If the option of "Do not ignore response message (body)" is set, the user must create the message format as per the contents of HTTP request.

Note<sup>#</sup>

When using HTTP reception in pass-through mode, the user must create this message format as per the contents of HTTP request and HTTP response.

## 2. Defining User-Defined Reception

The following table describes the method of creating XML schema of request message format (for body variable) and response message format (for body variable) (in case of to be created by user):

### (a) Creating request message format (for body variable)

1. Open XML schema file (urecp\_http\_body\_detail\_request.xsd) using XML editor and edit the part enclosed within the framework of the following figure.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema elementFormDefault="qualified"
targetNamespace="http://www.hitachi.co.jp/soft/xml/cosminexus/csc/reception/http/request"
xmlns:hrc="http://www.hitachi.co.jp/soft/xml/cosminexus/csc/reception/http/request"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<xsd:complexType name="http-body-requestType">
<xsd:sequence>
<!-- User Customize -->
<!-- <xsd:element name="key" type="xsd:string" minOccurs="0" /> -->
</xsd:sequence>
</xsd:complexType>
</xsd:schema>
```

Legend:

 : Edit location

Note the following points when editing the XML schema file:

- When adding an element, define it as a child element of `<http-body-request>`. Do not define the structure of a child element.
- If the key corresponding to the defined element is not present in the query string of HTTP request, the element defined in the request message is not created.
- If the key which is not defined as an element is present in the query string of HTTP request, the element corresponding to the key in the request message is created.

2. Save the XML schema file.

### (b) Creating response message format (for body variable)

1. Create any XML schema file using XML editor.

For details on the example of creating response message format (for body variable), see *Appendix F Example for setting up business processes using HTTP reception*.

2. Save this file in any directory as "urecp\_http\_body\_response.xsd".

## 2.5.3 Creating definition file of HTTP reception

This subsection describes types of a definition file used in HTTP reception.

The following table describes types of a definition file:

Table 2–15: Types of a definition file

Definition file	Description	Possibility of omission
HTTP response header definition file	Header definition of default HTTP response is to be specified in this file, if the header variable of response message is omitted.	Possible
HTTP reception definition file	The file is required to set the parameters required for using HTTP reception.	Possible <sup>#</sup>

Note#

If you omit HTTP reception definition file, default value of the template file of HTTP reception definition file is used.

For HTTP response header definition file and HTTP reception definition file, edit and use the template files (`scsurecphttp_header.properties` and `scsurecphttp.properties`) provided by service platform at the time of adding HTTP reception. Therefore, you need not create HTTP response header definition file and HTTP reception definition file.

## 2.5.4 Adding HTTP reception

The procedure for adding a new HTTP reception is as follows:

1. From Eclipse menu, select [Window] - [Display view] - [Others].  
[Display view] dialog box is displayed.
2. Select [HCSC-Definer] - [HCSCTE view] and click the [OK] button.  
Service definition list is displayed in tree view.
3. Right click on the business process in tree view and select "Add user-defined reception".  
Reception-type section wizard is displayed.
4. Select "HTTP reception" from the drop-down list of reception type.  
If you click the [Next] button, a dialog box for entering the information required for adding HTTP reception is displayed.
5. Specify [Reception name].  
Specify HTTP reception name within 1 to 40 bytes.
6. Click the [Finish] button.  
If you click the [Finish] button, the required file is created and it is saved in the repository. Service adapter definition screen (basic) is displayed.
7. Set up definition information of HTTP reception.

The following table describes the setting items of HTTP reception:

Table 2–16: Setting items of user-defined reception definition screen of HTTP reception (basic)

Classification	Item		Setting contents	Setting
User-defined reception information	Reception name		Reception name that is set up is displayed.	A
	Reception ID		Reception ID is displayed. Change it, if necessary.	A
	Name of default operation		Select the operation to be invoked by default.	A
	Operation		Specify name of any operation.	A
Operation information	Communication model		Set "Synchronization".	A
Request message	"Use any" checkbox		Do not check it.	-
	Reception	Message format	Specify <code>urecp_http_body_request.xsd<sup>#</sup></code> .	A
	Service component	"Use" checkbox	Do not check it.	-
Response message	[Use any] checkbox		Do not check it.	-
	Reception	Message format	<ul style="list-style-type: none"> <li>• In case of setting for ignoring response message (body) Specify any message format. It is recommended that you use <code>urecp_http_dummy_body_response.xsd</code> provided by service platform<sup>#</sup>.</li> <li>• In case of setting for not ignoring response message (body) Specify <code>urecp_http_body_response.xsd</code> created in "2.5.2(3)(b) Creating response message format (for body variable)".</li> </ul>	A

## 2. Defining User-Defined Reception

Classification	Item		Setting contents	Setting
Response message	Service component	"Use" checkbox	Do not check it.	-

Legend:

A: Set up this item without fail.

-: Not applicable.

Note#

The directory in which the file is to be stored is "*Installation directory of service platform\CSC\custom-reception\http\schema*".

For details on the setting items apart from those mentioned in user-defined reception definition screen, see *1.2.6 User-defined reception definition screen* in the manual *Service Platform Reference Guide*.

8. Click the user-defined reception (details) tab.

The user-defined reception definition screen (details) is displayed.

9. Set up the definition information of HTTP reception.

The following table describes the setting items of HTTP reception:

Table 2–17: Setting items of user-defined reception definition screen of HTTP reception (details)

Classification	Item	Setting contents	Setting
User-defined reception control information	Self-defined file	Confirm that the following self-defined files are set: <ul style="list-style-type: none"> <li>cscrephttp.properties<sup>#1</sup></li> <li>cscrephttp_header.properties<sup>#2</sup></li> </ul>	A
	EAR file	Confirm that the following EAR file is set: <ul style="list-style-type: none"> <li>cscmsg_urecp_http.ear<sup>#3</sup></li> </ul>	N

Legend:

A: Set up this item without fail.

N: Confirm the displayed contents.

Note#1

cscrephttp.properties, which is set up is a template file. Select cscrephttp.properties, if necessary, click the [Edit] button, and modify the contents of self-defined file. For details on cscrephttp.properties, see *HTTP reception definition file* in the manual *Service Platform Reference Guide*.

Note#2

cscrephttp\_header.properties, which is set up is a template file. Select cscrephttp\_header.properties, if necessary, click the [Edit] button, and modify the contents of self-defined file. For details on cscrephttp\_header.properties, see *HTTP response header definition file* in the manual *Service Platform Reference Guide*.

Note#3

The directory in which the file is to be stored is "*Installation directory of service platform\CSC\custom-reception\http\lib*".

### 2.5.5 Setting up an activity

This subsection describes how to set up receive activity and reply activity in business process definition screen after defining HTTP reception.

After defining the variables for request message and response message in activity settings, assign the variables defined in receive activity and reply activity.

#### (1) Defining a variable

Define the following variables in [List of variables and correlation set] dialog box.

- Variable of request message (header)
- Variable of response message (header)
- Variable of request message (body)

- Variable of response message (body)

Procedure for defining a variable is as follows:

(a) Variables of request message and response message (header)

1. Double click the [Variable and correlation set list] icon present on canvas of the business process definition screen.  
The [Variable and correlation set list] dialog box is displayed.
2. Specify any name in [Variable name].
3. Select [XML] from the [Types] drop-down list.
4. Click the [Browse] button of [Message format].  
A dialog box to specify the message format is displayed.
5. Specify the following files as per the variable:
  - Variable of request message (header)  
*Installation directory of service platform\CSC\custom-reception\http\schema\urecp\_http\_header\_request.xsd*
  - Variable of response message (header)  
*Installation directory of service platform\CSC\custom-reception\http\schema\urecp\_http\_header\_response.xsd*
6. In the [Variable and correlation set list], click the [Add] button.  
The variable is added.  
Add the variable of response message (header) in the procedure same as Operation 2. to Operation 6.

(b) Variables of request message (Body) response message (body)

1. Double click the [Variables and correlation sets] icon present on canvas of the business process definition screen.  
The [Variable and correlation set list] dialog box is displayed.
2. Specify any name in [Variable name].
3. Select [XML] from the [Types] drop-down list.
4. Click the [Capture] button of [Message format].  
The [Capturing message format] dialog box is displayed.
5. Check [Reception name] and select name of HTTP reception set up in the user-defined reception definition screen from the drop-down list.
6. Select the operation set up in HTTP reception from the [Operation name] drop-down list.
7. Perform following selections in the [Message type] drop-down list as per the variables:
  - For variables of request message (body)  
Select [Request message (body)].
  - For variables of response message (body)  
Select [response message (body)].
8. Enter any name in [Message format].
9. Click the [OK] button.
10. Click the [Add] button in the [Variable and correlation set list] dialog box.  
A variable is added.  
Add a variable of response message (body) using the procedure same as that used for Operation 2 to Operation 10.

## (2) Assigning a variable

The procedure for assigning a variable, which is added in receive activity and reply activity:

(a) For receive activity

1. Double click receive activity present on the canvas of business process definition screen.  
The [Receive activity] dialog box is displayed.

## 2. Defining User-Defined Reception

2. Enter any name in [Activity name].
3. Enter the operation name set in HTTP reception in [Operation name].
4. Select a variable of the defined request message (body) from the drop-down list of [Body-assigned variable].
5. Click the [Set] button of [Header-assigned variable].  
The [Header-assigned variable] dialog box is displayed.
6. Click the [Add] button.
7. Select a variable of the defined request message (header) from the drop-down list of [Assigned variable] cell.
8. Select the root-element from the drop-down list of [Root element] cell.
9. Click the [OK] button.  
The [Header-assigned variable] dialog box is closed.
10. Click the [OK] button in the [Receive activity] dialog box.

### (b) For reply activity

1. Double click reply activity present on the canvas of business process definition screen.  
The [Reply activity] dialog box is displayed.
2. Enter any name in [Activity name].
3. Enter the operation name set in HTTP reception in [Operation name].
4. Select a variable of the defined response message (body) from the drop-down list of [Body-assigned variable].
5. Click the [Set] button of [Header-assigned variable].  
The [Header-assigned variable] dialog box is displayed.
6. Click the [Add] button.
7. Select the defined response message (header) from the drop-down list of [Assigned variable].
8. Select the root element from the drop-down list of [Root element] cell.
9. Click the [OK] button.  
The [Header-assigned variable] dialog box is closed.
10. Click the [OK] button in the [Reply activity] dialog box.

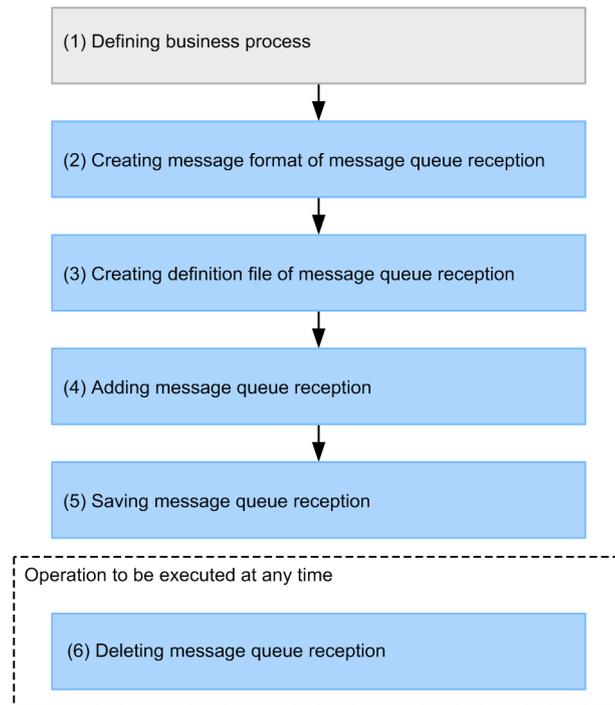
## 2.6 Defining Message Queue reception

This section gives details on how to define Message Queue reception such as creating message format of Message Queue reception and adding Message Queue reception.

### 2.6.1 Flow of defining Message Queue reception

The following figure shows the flow of defining Message Queue reception:

Figure 2–6: Flow of defining Message Queue reception



The activities to be performed when defining Message Queue reception are as follows:

#### (1) Defining a business process

Add a business process using a wizard required for adding a business process and a defined business process. Define the added business process on business process definition screen. For details on defining a business process, see 5. *Defining a Business Process*" in the manual *Service Platform Basic Development Guide*.

Also, you must define a service adapter to invoke a service component from a business process. For details on defining a service adapter, see 3. *Defining Adapters*.

#### (2) Creating message format of Message Queue reception

Create a message format to be used in Message Queue reception.

For details on how to create a message format, see 2.6.2 *Creating message format of Message Queue reception*.

#### (3) Creating definition of Message Queue reception

Create a definition file to be used in Message Queue reception. The created definition file is reflected in the repository.

For details on how to create a definition file to be used in Message Queue reception, see 2.6.3 *Creating of Message Queue reception definition file*.

#### (4) Adding Message Queue reception

Add Message Queue reception in the service definition list in tree view and define its contents.

For details on the procedure for adding and defining Message Queue reception, see *2.6.4 Adding Message Queue reception*.

#### (5) Saving Message Queue reception

Save the Message Queue reception which is added and defined. For details on the procedure for saving Message Queue reception, see *2.9 Saving a user-defined reception*.

#### (6) Deleting Message Queue reception

You can delete the unwanted Message Queue reception, if necessary.

For details on deleting Message Queue reception, see *2.12 Deleting a User-Defined Reception*.

### 2.6.2 Creating message format of Message Queue reception

The message format to be used in Message Queue reception and its creation method are described here.

#### (1) Message format

The JMS message to be handled in Message Queue reception consists of JMS header, property, and body.

The message format to be used in Message Queue reception is described here.

##### (a) Format of the header message

Set up the data specified in JMS header and property of JMS message in the header of the request message to be passed on to a business process from Message Queue reception.

The following table describes the request message format (for header variable) to be passed on to a business process from Message Queue reception. The name of the namespace is "http://www.hitachi.co.jp/soft/xml/cosminexus/csc/reception/mq/header".

Table 2–18: Request message format of the header message of Message Queue reception

Tag name	Type	Occurrence count	Description
<mq_header>	-	1 time	-
<jms_header>	string	1 time	Contents of JMS header are set.
<JMSTDestination>	string	1 time	Name of the queue in which the message is set is set.
<JMSTDeliveryMode>	string	1 time	Value indicating persistence of the message is set.
<JMSTMessageID>	string	1 time	A message identifier that uniquely identifies each message sent by the provider is set.
<JMSTTimestamp>	string	1 time	A value indicating time of sending a message is set. (unit: milliseconds)
<JMSTExpiration>	string	1 time	A value indicating expiration period of a message is set. (unit: milliseconds)
<JMSTRedelivered>	string	1 time	A value indicating whether the message is being redelivered or not is set.
<JMSTPriority>	string	1 time	Message priority is set.

Tag name	Type	Occurrence count	Description
<JMSReplyTo>	string	1 time	The address to be mentioned in "ReplyTo" is set. Do not use this information in Message Queue reception.
<JMSCorrelationID>	string	1 time	Correlation identifier required to associate the current message with another message is set.
<JMSType>	string	1 time	When a message is sent, the message type provided by an application is set.
<JMSHeaderExtension>	string	More than 0 time	The following values are set: Attribute contains "name" and name of JMS header is set. Value of JMS header set in name is set.in value.
<user_property>	-	1 time	Property contents set by the user are set.
<property>	string	More than 0 time	The following values are set. This tag is generated for the number part in the set property. Attribute contains "name" and the property name is set. Property value set in name is set in value.

Legend:

-: Corresponding item does not exist.

### (b) Format of body message

Pass on the main text stored in the body of a request message to a business process as a body message.

The types of a body message differ as per the interface of JMS message and settings of user-defined reception definition. The following table describes the message types of a body message:

Table 2–19: Message types of a body message

Interface name	Setting of user-defined reception definition (request message format is specified)	
	any is specified	any is not specified
BytesMessage	Binary message	Message in any format
TextMessage	XML message	Message in any format

## (2) Method of creating a message format

The method of creating a message format to be used in Message Queue reception is described here.

### (a) Creating header message

For the message format of header message of Message Queue reception, edit XML schema provided by service platform and use it. Therefore, you need not create XML schema of header message.

XML schema of the message format provided by service platform is stored in "*Installation directory of service platform\CSC\custom-reception\mq\schema*".

The contents of XML schema provided by service platform are as follows:

---

```

<?xml version="1.0" encoding="UTF-8"?>
<!-- All Rights Reserved. Copyright (C) 2014, Hitachi, Ltd. -->
<xsd:schema elementFormDefault="qualified"
  targetNamespace="http://www.hitachi.co.jp/soft/xml/cosminexus/csc/reception/mq/
header"
  xmlns:mhc="http://www.hitachi.co.jp/soft/xml/cosminexus/csc/reception/mq/header"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">

  <xsd:element name="mq_header">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="jms_header" maxOccurs="1" minOccurs="0">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="JMSDestination" maxOccurs="1" minOccurs="0" nillable="true"
type="xsd:string"/>
              <xsd:element name="JMSDeliveryMode" maxOccurs="1" minOccurs="0" nillable="true"
type="xsd:string"/>
              <xsd:element name="JMSMessageID" maxOccurs="1" minOccurs="0" nillable="true"
type="xsd:string"/>
              <xsd:element name="JMSTimestamp" maxOccurs="1" minOccurs="0" nillable="true"
type="xsd:string"/>
              <xsd:element name="JMSExpiration" maxOccurs="1" minOccurs="0" nillable="true"
type="xsd:string"/>
              <xsd:element name="JMSRedelivered" maxOccurs="1" minOccurs="0" nillable="true"
type="xsd:string"/>
              <xsd:element name="JMSPriority" maxOccurs="1" minOccurs="0" nillable="true"
type="xsd:string"/>
              <xsd:element name="JMSReplyTo" maxOccurs="1" minOccurs="0" nillable="true"
type="xsd:string"/>
              <xsd:element name="JMSCorrelationID" maxOccurs="1" minOccurs="0" nillable="true"
type="xsd:string"/>
              <xsd:element name="JMSType" maxOccurs="1" minOccurs="0" nillable="true"
type="xsd:string"/>
              <xsd:element name=" JMSHeaderExtension " minOccurs="0" maxOccurs="unbounded" >
                <xsd:complexType>
                  <xsd:simpleContent>
                    <xsd:extension base="xsd:string">
                      <xsd:attribute name="name" type="xsd:string" use="optional"/>
                    </xsd:extension>
                  </xsd:simpleContent>
                </xsd:complexType>
              </xsd:element>
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
        <xsd:element name="user_property" maxOccurs="1" minOccurs="0">
          <xsd:complexType>
            <xsd:sequence>
              <xsd:element name="property" minOccurs="0" maxOccurs="unbounded" >
                <xsd:complexType>
                  <xsd:simpleContent>
                    <xsd:extension base="xsd:string">
                      <xsd:attribute name="name" type="xsd:string" use="optional"/>
                    </xsd:extension>
                  </xsd:simpleContent>
                </xsd:complexType>
              </xsd:element>
            </xsd:sequence>
          </xsd:complexType>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>

```

---

### (b) Creating body message

Schema format is optional for the message format of body message of Message Queue reception. You must create the schema by matching it with the message type of body message. Note that you need not create schema, if the message format is optional (any is specified).

## 2.6.3 Creating of Message Queue reception definition file

Message Queue reception definition file is described here.

- Message Queue reception definition file

This file is required to set up the parameters required for using Message Queue reception. At the time of deployment of Message Queue reception, the contents set up in the Message Queue reception definition file are written in the application integration attribute file and this file is enabled when starting Message Queue reception.

For Message Queue reception definition file, edit and use the template file (cscurecpmq.properties) provided by service platform when adding Message Queue reception. Therefore, you need create Message Queue reception definition file.

An example of describing Message Queue reception definition file is as follows:

---

```
resource-adapter=WebSphere_MQ_Resource_Adapter
activation-config.destination=QUEUE.NAME
activation-config.username=user1
```

---

## 2.6.4 Adding Message Queue reception

The procedure for adding new Message Queue reception is as follows:

1. From Eclipse menu, select [Window] - [Display view] - [Others].  
The [Display view] dialog box is displayed.
2. Select [HCSC-Definer] - [HCSCTE view] and click the [OK] button.  
Service definition list is displayed in tree view.
3. Right click on a business process in tree view and select "Add user-defined reception".  
Reception-type selection wizard is displayed.
4. Select "Message Queue reception" from the drop-down list of reception type.  
If you click the [Next] button, a dialog box for entering the information required for adding Message Queue reception is displayed.
5. Specify [Reception name]  
Specify name of Message Queue reception within 1 to 40 bytes.
6. Click the [Finish] button.  
If you click the [Finish] button, the required file is created and it is saved in the repository. The service adapter definition screen (basic) is displayed.
7. Set up the definition information of Message Queue reception.  
The following table describes the setting items of Message Queue reception:

Table 2–20: Setting items of user-defined reception definition screen (basic) of Message Queue reception

Classification	Item	Setting contents	Setting
User-defined reception information	Reception name	The reception name, which is set up is displayed.	A
	Reception ID	The reception ID is displayed. Change it, if necessary.	A
	Name of default operation	Select the operation to be invoked by default.	A
	Operation	Select the operation type to be added from the following details and specify the operation name: <ul style="list-style-type: none"> <li>• Commit operation to be performed at the time of termination It is the standard pattern of Message Queue reception. Transaction with JMS provider is committed after the end of business process execution. The communication model of a business process is asynchronous.</li> <li>• Commit operation to be performed at any time It is the application pattern of Message Queue reception. Transaction with JMS provider is</li> </ul>	A

## 2. Defining User-Defined Reception

Classification	Item		Setting contents	Setting
User-defined reception information	Operation		committed after completion of reply activity process of a business process. The communication model of a business process is synchronous. If an exception occurs in the processing of a business process after completion of reply activity process, you must resend the request.	A
Operation information	Communication model		Set as "Synchronous" or "Asynchronous".	A
Request message	[use any] check box		Check it when using a request message of any format (body)	Y
	Reception	Message format	If you do not check the [Use any] check box, specify any message format file. If you check the [Use any] check box, you need not specify message format file.	Y
	Service component	[Use] check box	Do not check this check box.	-
Response message	[Use any] check box		Check it when using a response message of any format (body).	Y
	Reception	Message format	Specify any message format file.	Y
	Service component	[Use] check box	Do not check this check box.	-

Legend:

- A: Set up this item, if necessary.
- Y: Setting of this item is optional.
- : Not applicable.

8. Click the user-defined reception (details) tab.

User-defined reception definition screen (details) is displayed.

9. Set up the definition information of Message Queue reception.

The following table describes the setting items of Message Queue reception:

Table 2–21: Setting items of user-defined reception screen of Message Queue reception (details)

Classification	Item	Setting contents	Setting
User-defined reception control information	Self-defined file	Confirm that the following self-defined file is set: <ul style="list-style-type: none"> <li>• cscurecpmq.properties<sup>#1</sup></li> </ul>	A
	EAR file	Confirm that the following EAR file is set: <ul style="list-style-type: none"> <li>• cscmsg_urecp_mq.ear<sup>#2</sup></li> </ul>	N

Legend:

- A: Set up this item without fail.
- N: Confirm the displayed contents.

Note#1

cscurecpmq.properties, which is set up is a template file. Select cscurecpmq.properties, if necessary, click the [Edit] button and modify the contents of self-defined file. For details on cscurecpmq.properties, see *Message Queue reception definition file* in the manual *Service Platform Reference Guide*.

Note#2

The directory in which the file is to be stored is "*Installation directory of service platform*\CSC\custom-reception\mq\lib".

## 2.7 Creating WSDL

Create a WSDL file that defines the interface information used when calling a service component by using a user-defined reception.

The following describes how to create a WSDL file.

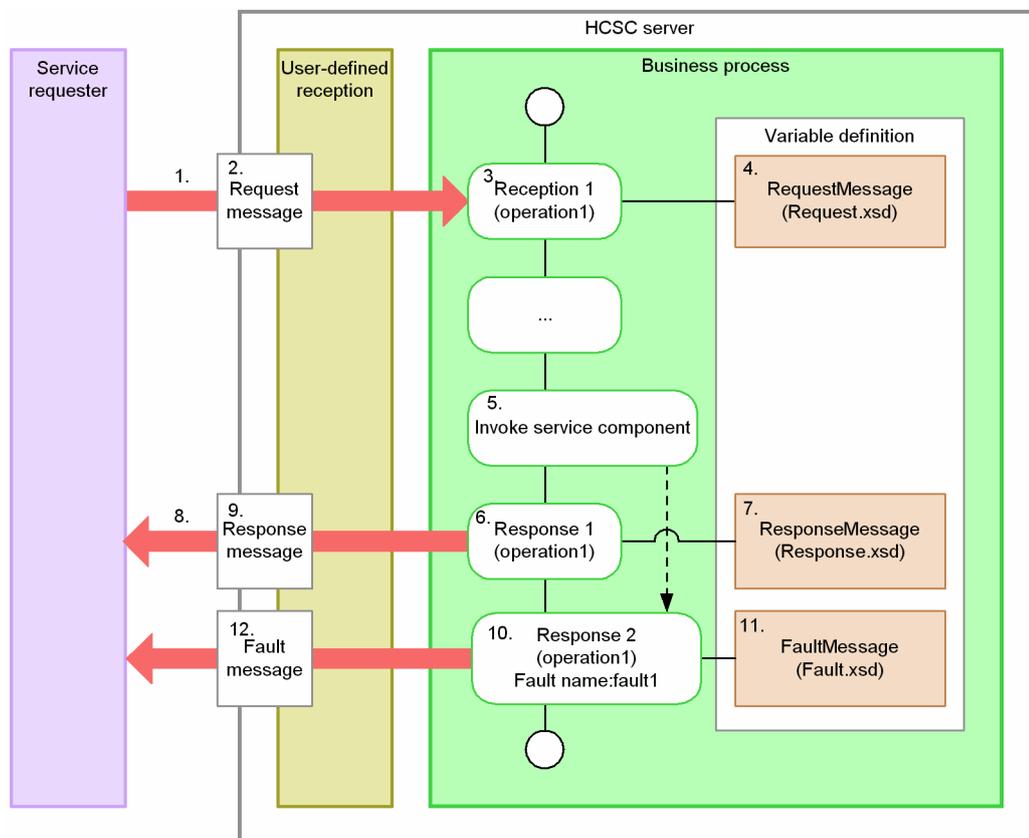
### 2.7.1 Example of a Business Process Used for Creating WSDL

This subsection explains the illustrated contents of a business process used for explaining how to create a WSDL file.

#### (1) Example of a business process

The following figure shows a business process to be used as an example for creating the WSDL.

Figure 2–7: Business process to be used as an example for creating the WSDL



1. The service requester sends a service component execution request to a business process.
2. An XML request message is sent with the service component execution request.
3. The business process receives the XML request message with the receive activity Reception 1. The operation name in this case is `operation1`.
4. In Reception 1, the variable `RequestMessage` is allocated and `Request.xsd` is specified as a message format that provides the request message.
5. The service component is invoked from the business process. If a fault occurs, the fault is returned in No. 10.
6. The business process configures the service component execution result as an XML response message, and returns the result with the reply activity Response 1.

7. In Response 1, the variable `ResponseMessage` is allocated and `Response.xsd` is specified as a message format that provides the response message.
8. The response to the service requester is returned.
9. The XML response message is sent as the response to the service requester.
10. If a fault occurs in No. 5, the fault `fault1` is returned from the reply activity Response 2.
11. In Response 2, the variable `FaultMessage` is allocated and `Fault.xsd` is specified as the message format that provides the fault message.
12. The fault message is sent to the service requester.

## (2) Example of message format contents

An example of message format used in the business process and shown in the Figure 8-4 is described below:

### Request.xsd

- XML schema source

This is the message format for a request message that provides a single element `<Request1>` having a string value.

---

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema targetNamespace="http://www.example.org/request"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="Request1" type="xsd:string" />
</xsd:schema>
```

---

- XML sent according to the provision of this message format

---

```
<Request1>Sample request message</Request1>
```

---

### Response.xsd

- XML schema source

This is the message format for a response message that provides a single element `<Response1>` having a boolean value.

---

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema targetNamespace="http://www.example.org/response"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="Response1" type="xsd:boolean" />
</xsd:schema>
```

---

- XML sent according to the provision of this message format

---

```
<Response1>>true</Response1>
```

---

### Fault.xsd

- XML schema source

This is a message format for the fault message that provides an element called `<Fault1>`, and also provides an element `<param1>` having int value as a child element.

---

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema targetNamespace="http://www.example.org/service"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:svc="http://www.example.org/service">
  <xsd:element name="Fault1" type="svc:SampleData"/>
  <xsd:complexType name="SampleData">
    <xsd:sequence>
      <xsd:element name="param1" type="xsd:int"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```

---

- XML sent according to the provision of this message format

---

```
<Fault1>
  <param1>123</param1>
</Fault1>
```

---

### (3) Example of Creating WSDL

This subsection explains an example of creating WSDL that is appropriate to the business process described in 2.7.1 *Example of a Business Process Used for Creating WSDL* by assuming the use of SOAP1.1. For details on the conditions and precautions related to the WSDL format, see 2.7.1 *Application Scopes of Service Components That Use Web Services* and 2.7.3 *Notes on creating WSDL of a user-defined reception*.

The example for creating WSDL is assumed to be `Sample_Reception.wsdl`. An overview of `Sample_Reception.wsdl` is explained below:

- The contents described in 2.7.1 *Example of a Business Process Used for Creating WSDL* are prerequisites.
- You can also create a user-defined reception for each operation, however, `Sample_Reception.wsdl` is the WSDL used when one user-defined reception contains all the operations.
- Hitachi recommends document style as the communication style for the WSDL to be created. Therefore, `Sample_Reception.wsdl` is the WSDL with a document style.

`Sample_Reception.wsdl`, an example of creating WSDL, is explained below:

---

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions
  targetNamespace="http://www.example.org/service"
  xmlns:svc="http://www.example.org/service"
  xmlns:req="http://www.example.org/request"
  xmlns:res="http://www.example.org/response"
  xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <wsdl:types>
    <xsd:schema targetNamespace="http://www.example.org/request">
      <xsd:element name="Request1" type="xsd:string" />
    </xsd:schema>
    <xsd:schema targetNamespace="http://www.example.org/response">
      <xsd:element name="Response1" type="xsd:boolean" />
    </xsd:schema>
    <xsd:schema targetNamespace="http://www.example.org/service">
      <xsd:element name="Fault1" type="svc:SampleData"/>
      <xsd:complexType name="SampleData">
        <xsd:sequence>
          <xsd:element name="param1" type="xsd:int"/>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:schema>
  </wsdl:types>
  <wsdl:message name="ReqMsg1">
    <wsdl:part element="req:Request1" name="reqParam"/>
  </wsdl:message>
  <wsdl:message name="ResMsg1">
    <wsdl:part element="res:Response1" name="resParam"/>
  </wsdl:message>
  <wsdl:message name="FltMsg1">
    <wsdl:part element="svc:Fault1" name="fltParam"/>
  </wsdl:message>
  <wsdl:portType name="SamplePortType">
    <wsdl:operation name="operation1">
      <wsdl:input message="svc:ReqMsg1"/>
      <wsdl:output message="svc:ResMsg1"/>
      <wsdl:fault message="svc:FltMsg1" name="Fault1"/>
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:binding name="SampleBinding" type="svc:SamplePortType">
    <soap:binding transport="http://schemas.xmlsoap.org/soap/http"/>
    <wsdl:operation name="operation1">
      <soap:operation soapAction=""/>
      <wsdl:input>
        <soap:body/>
      </wsdl:input>
      <wsdl:output>
        <soap:body/>
      </wsdl:output>
      <wsdl:fault name="Fault1">
        <soap:fault name="Fault1"/>
      </wsdl:fault>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:service name="SampleService">
    <wsdl:port binding="svc:SampleBinding" name="UserInfo">
      <soap:address location="http://localhost:80/SampleService/services/UserInfo" />
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

---

---

```

</wsdl:port>
</wsdl:service>
</wsdl:definitions>

```

---

This example `Sample_Reception.wsdl` is created as per the operations explained below from (1) to (8):

(a) Listing the operations and fault names

List the synchronous operations defined in the receive activity and reply activity of a business process. If a reply activity exists in which the fault name is defined, list the fault names as well.

In the creation example, the operation corresponds to `operation1`. Also, the fault name corresponds to `Fault1`.

(b) Listing the message format definition files for the allocated variable

List message format definition files for the allocated variables used in the receive activity and reply activity.

In the creation example, the files correspond to `Request.xsd`, `Response.xsd`, and `Fault.xsd`.

**! Important note**

- If there are multiple schemas with the same name space in the listed message format definition files, they must be consolidated into one schema.  
At this time, if the following conditions are satisfied, make changes so that the message format definition file for the allocated variable does not have duplicate names:
    - There are duplicate names for elements within the schema.
    - The contents of the elements are different.
 Note that if there are multiple message format definition files for an allocated variable (when an external XML file is referenced), make changes so that there is no duplication of elements including the external XML file schema.  
Furthermore, if the individual schema attributes have conflicting values, you consolidate the schema considering the impact caused by the change in values.
  - If there is a schema (chameleon schema) that does not have the `targetNamespace` attribute within the listed message format definition files, change to a schema with the `targetNamespace` attribute specified in the message format definition files for the allocated variable.
  - If the root element in the message format definition file for the listed fault messages is not a complex type, you change the root element in the message format definition file for the allocated variable, so that the root element becomes a complex type.
  - The name space of the fault message root element needs to match with the name space of the `wsdl:definitions` element.  
Therefore, if the following conditions are satisfied, you change the message format file of the allocated variables:
    - There are multiple fault messages.
    - The name spaces of the root element are different.
 If the name spaces for the same fault messages in different operations are different, you can create separate WSDL files for each operation, and create multiple user-defined receptions for providing support.
- 

(c) Creating the template for a WSDL file (defining the `wsdl:definitions` element)

Define an XML declaration and the `wsdl:definitions` element as the template for a WSDL file. In the `wsdl:definitions` element, add the attributes described in the following table.

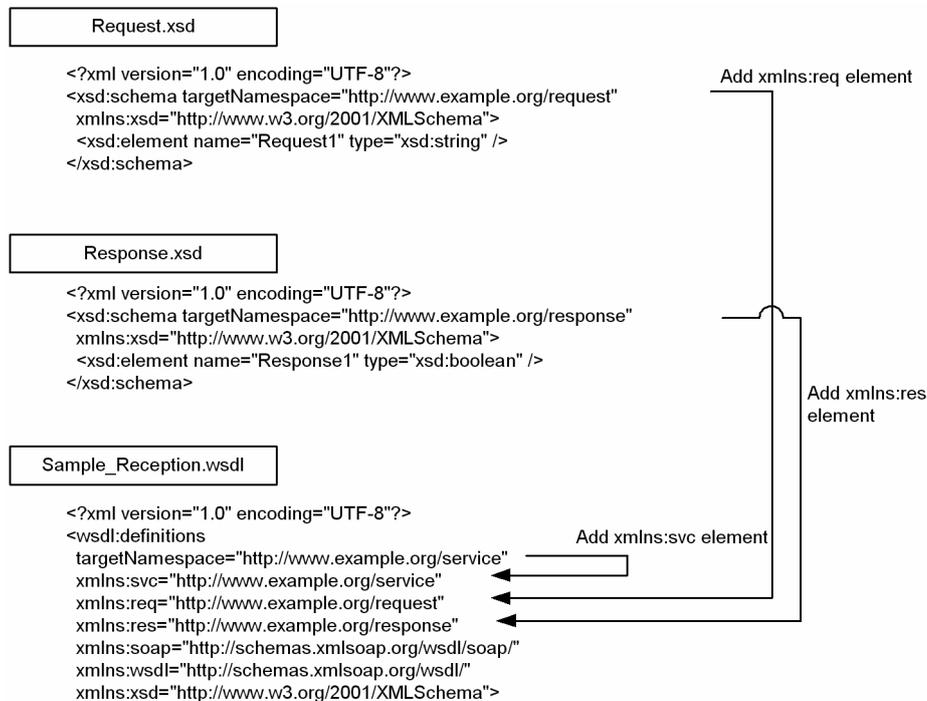
Table 2–22: Attributes to be added in the `wsdl:definitions` element

Attributes	Meaning	Value	Number
<code>targetNamespace</code>	Name space URI of the service component	You can define any URI. However, if a fault message exists, define URI same as the value of <code>targetNamespace</code> attribute that exists in the <code>xsd:schema</code> element of the message format definition file for the fault message.	1
<code>xmlns:svc</code>	Prefix declaration of a service component	Define the value specified in the <code>targetNamespace</code> attribute.	1

Attributes	Meaning	Value	Number
xmlns:req	Message format prefix declaration of a request message	Define URI same as the value of <code>targetNamespace</code> attribute that exists in the <code>xsd:schema</code> element of the message format definition file for the request message.	1
xmlns:res	Message format prefix declaration for a response message	Define URI same as the value of <code>targetNamespace</code> attribute that exists in the <code>xsd:schema</code> element of the message format definition file for the response message.	1
xmlns:soap	Prefix declaration of SOAP	Define the pre-defined value <code>http://schemas.xmlsoap.org/wsdl/soap/</code> .	1
xmlns:wSDL	Prefix declaration of WSDL	Define the pre-defined value <code>http://schemas.xmlsoap.org/wsdl/</code>	1
xmlns:xsd	Prefix declaration of XML schema	Define the pre-defined value <code>http://www.w3.org/2001/XMLSchema</code> .	1

The following figure shows an example for defining the `wSDL:definitions` element:

Figure 2–8: Definition example of the `wSDL:definitions` element



### When using multiple operations

In the creation example (`Sample_Reception.wsdl`) only one operation is used, but if multiple operations are used, multiple request messages and response messages are handled. If these messages use multiple different name spaces, you define individual prefix declarations in the `wSDL:definitions` element, as shown in the table below:

Table 2–23: Attribute to be added in the `wSDL:definitions` element (when multiple operations are used)

Attributes	Meaning	Value
xmlns:reqN(N>0)	Prefix declaration used when the request messages are divided into multiple name spaces	Define URI same as the value of <code>targetNamespace</code> attribute that exists in the <code>xsd:schema</code> element of the message format definition file for the request message.

## 2. Defining User-Defined Reception

Attributes	Meaning	Value
<code>xmlns:reqN(N&gt;0)</code>	Prefix declaration used when the request messages are divided into multiple name spaces	Example <code>http://www.example.org/request1</code>
<code>xmlns:resN(N&gt;0)</code>	Prefix declaration used when the response messages are divided into multiple name spaces	Define URI same as the value of <code>targetNamespace</code> attribute that exists in the <code>xsd:schema</code> element of the message format definition file for the response message.  Example <code>http://www.example.org/response1</code>

The contents are defined in the element attribute of the `wsdl:part` element that is a child element of the `wsdl:message` element. For details on the `wsdl:message` element definition, see "(e) Defining the `wsdl:message` element".

The following figure shows an example of definition when multiple operations are used:

Figure 2–9: Definition example of the `wsdl:definitions` element (When multiple operations are used)

```

Sample_Reception.wsdl
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions
  targetNamespace="http://www.example.org/service"
  xmlns:svc="http://www.example.org/service"
  xmlns:req1="http://www.example.org/request1"
  xmlns:req2="http://www.example.org/request2"
  :
  >
  <wsdl:types>
    <xsd:schema targetNamespace="http://www.example.org/request">
      <xsd:element name="Request1" type="xsd:string" />
    </xsd:schema>
    <xsd:schema targetNamespace="http://www.example.org/response">
      <xsd:element name="Response1" type="xsd:boolean" />
    </xsd:schema>
    <xsd:schema targetNamespace="http://www.example.org/service">
      <xsd:element name="Fault1" type="svc:SampleData"/>
      <xsd:complexType name="SampleData">
        <xsd:sequence>
          <xsd:element name="param1" type="xsd:int"/>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:schema>
  </wsdl:types>

  <wsdl:message name="ReqMsg1">
    <wsdl:part name="reqParam" element="req1:Request1" />
  </wsdl:message>
  <wsdl:message name="ReqMsg2">
    <wsdl:part name="reqParam" element="req2:Request2" />
  </wsdl:message>

```

Prefix declaration

Use prefix

### (d) Defining the `wsdl:types` element

Define the `wsdl:types` element, its lower elements, and attributes.

1. Define the `wsdl:types` element as a child element of the `wsdl:definitions` element defined in (c) *Creating the template for a WSDL file (defining the `wsdl:definitions` element)*.
2. Classify the `xsd:schema` element in message format definition files listed in (2) into the `targetNamespace` attribute, and define one `xsd:schema` element for each `targetNamespace` values (b) *Listing the message format definition files for the allocated variable*.
3. Add all the attributes that exist in the `xsd:schema` element of the message format definition file as attributes for the `xsd:schema` element defined in step 2.
4. Add all the child elements that exist in the `xsd:schema` element of the message format definition file as the child elements for the `xsd:schema` element defined in step 2.
5. If there are multiple fault messages and name spaces are different, change the name space of fault message into the value of `targetNamespace` attribute in the `wsdl:definitions` element.

The following table describes the elements and attributes to be defined in the `wSDL:types` element:

Table 2–24: Elements and attributes to be defined in the `wSDL:types` element

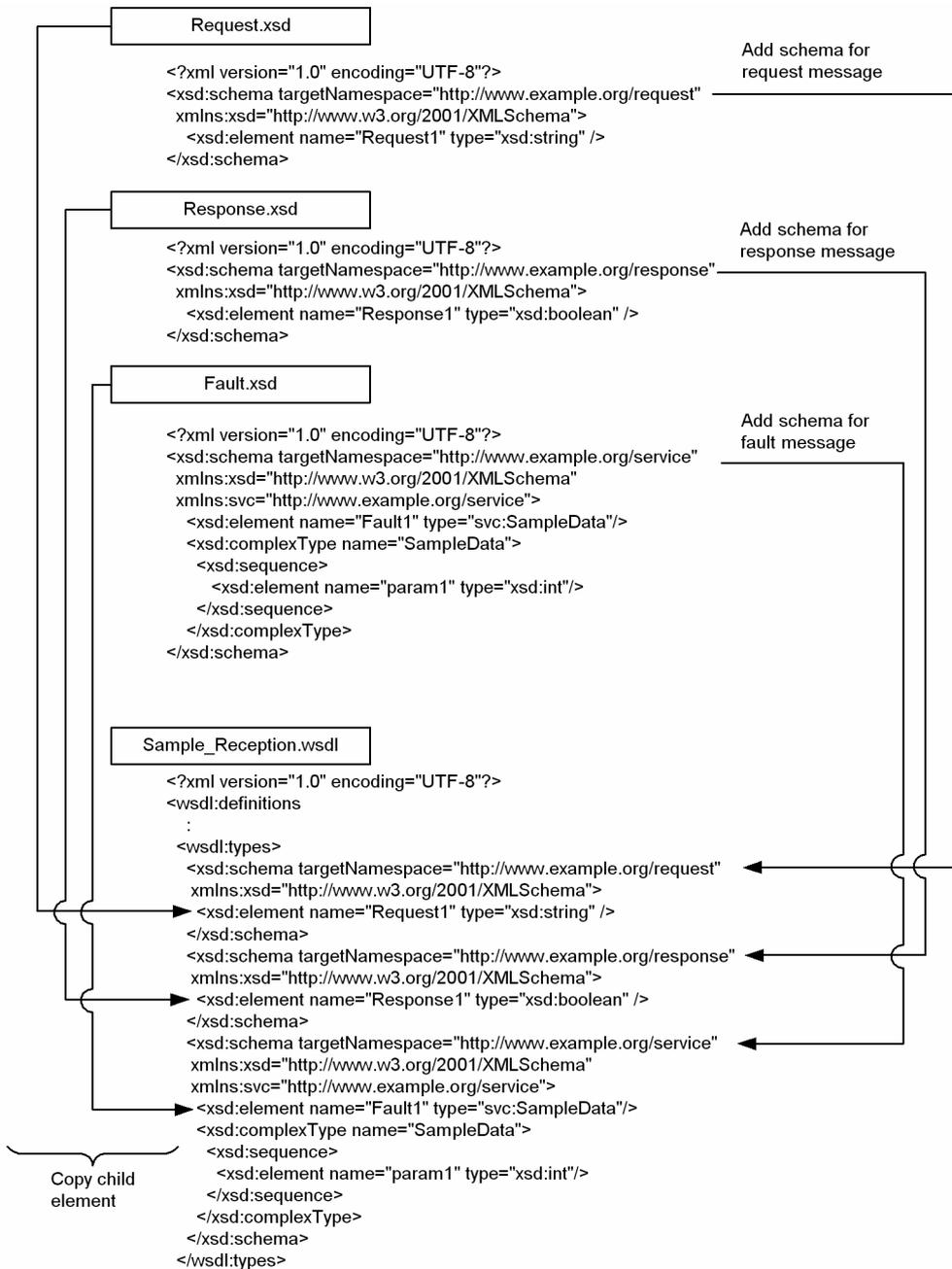
Element		Attributes	Meaning	Value	Number
<code>wSDL:types</code>		--	Type definition element	--	1
	<code>xsd:schema</code>	--	Schema element	--	From 1 up to the number of name spaces used in the message format definition file
		<code>targetNamespace</code>	Name space attribute	Define the value same as the <code>targetNamespace</code> of the message.	1
		Attribute of <code>xsd:schema</code>	Attributes	Define the value same as the message attribute.	From 1 up to the number of attributes
	Child element of <code>xsd:schema</code>	--	Message contents	Copy all the child elements of the <code>xsd:schema</code> element that the message contains.	From 1 up to the number of child elements of the <code>xsd:schema</code> element in the message formats using the same name space

Legend:

--: Not applicable

The following figure shows an example for defining the `wSDL:types` element:

Figure 2–10: Definition example of the wsdl:types element



(e) Defining the wsdl:message element

Define the `wsdl:message` element, its lower elements and attributes:

1. Define the `wsdl:message` element as the child element of the `wsdl:definitions` element that is defined in (c) *Creating the template for a WSDL file (defining the wsdl:definitions element)*.  
 Define the same number of `wsdl:message` elements as the number of request messages, response messages, and fault messages that exist in the operations defined in WSDL.  
 There is one operation in the creation example. Since this operation has one request message, one response message, and one fault message, three `wsdl:message` elements are defined.
2. Define the name attribute in the `wsdl:message` element.

Define a value different from the name attribute of other `wsdl:message` elements so that the name attribute value is a unique value in the file.

In the creation example, `ReqMsg1` is defined in the `operation1` request message and `ResMsg1` is defined in the response message. The fault messages are `FltMsgN` (N is 1 or more) as per the numbers.

3. Define `wsdl:part` elements as child elements of the `wsdl:message` element, one by one.

4. Define the name attribute and element attribute in the `wsdl:part` element.

Specify any string in the name attribute.

Select a root element from a child element of the `xsd:schema` element copied in (4), and specify *(d) Defining the wsdl:types element* in the element attribute value element. Use the prefix and root element names described in the Table 2-22 and specify the name in `QName`.

The following table describes the elements and attributes to be defined in the `wsdl:message` element:

Table 2–25: Elements and attributes to be defined under the `wsdl:message` element

Element	Attributes	Meaning	Value	Number
<code>wsdl:message</code>	--	Message definition element	--	1
	<code>name</code>	Message name attribute	Define a unique string different from the name attribute of other <code>wsdl:message</code> elements.	1
<code>wsdl:part</code>	--	Message part definition element	--	1
	<code>name</code>	Message part name attribute	Define any string <sup>#</sup> .	1
	<code>element</code>	Element reference attribute	Define the root element of applicable message format file.	1

Legend:

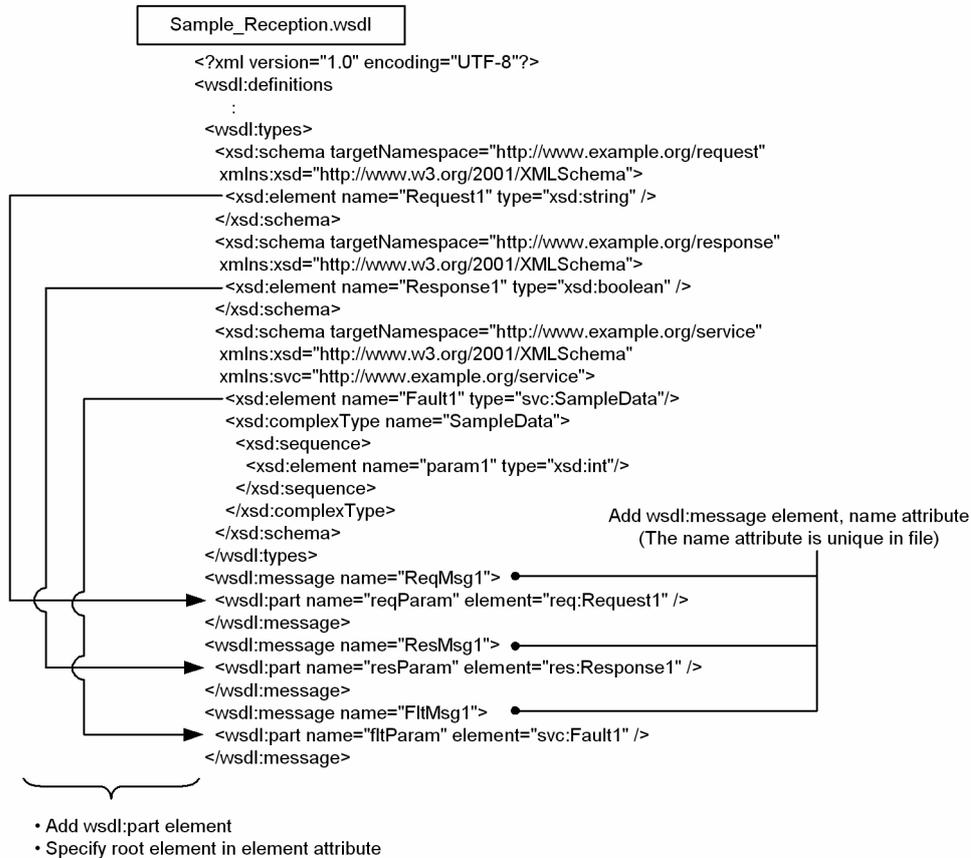
--: Not applicable

#

There are limited character types that can be used. For details about the character types that can be used, see "2.7.3(2) Characters that can be used for WSDL definition".

The following figure shows an example for defining the `wsdl:message` element:

Figure 2–11: Definition example of the wsdl:message element



## (f) Defining the wsdl:portType element

Define the `wsdl:portType` element, its lower elements, and attributes.

1. Define one `wsdl:portType` element as the child element of the `wsdl:definitions` element defined in (c) *Creating the template for a WSDL file (defining the wsdl:definitions element)*.
2. Define the name attribute in the `wsdl:portType` element.  
You can specify any string in the name attribute value.
3. Define only the number of operations listed up by (a) *Listing the operations and fault names* for the `wsdl:operation` element.  
Define the `wsdl:operation` element for all the operations listed in (1).
4. Define the name attribute in the `wsdl:operation` element.  
Specify the operation name for the operation listed in (a) *Listing the operations and fault names* in the name attribute value.  
In the creation example, define one `wsdl:operation` element and assume the name attribute as `operation1`.
5. Define the `wsdl:input` element, `wsdl:output` element, and `wsdl:fault` element as the child elements of the `wsdl:operation` element.  
Define the child elements in the order of `wsdl:input` element, `wsdl:output` element, and `wsdl:fault` element.  
If the reply activity for the fault message does not exist, you can omit the `wsdl:fault` element. If there are multiple reply activities for the fault message, define only for the number of activities that exist.  
In the creation example, `wsdl:input` element, `wsdl:output` element, and `wsdl:fault` element are defined one by one.
6. Define the message attribute in the `wsdl:input` element and `wsdl:output` element.

In the message attribute value, specify the name attribute value of the `wsdl:message` element defined in (e) *Defining the wsdl:message element* with QName.

7. Define the name attribute and message attribute in the `wsdl:fault` element.

Specify a fault name listed in (a) *Listing the operations and fault names* in the name attribute value.

In the message attribute value, specify the name attribute value of the `wsdl:message` element defined in (e) *Defining the wsdl:message element* with QName.

The following table describes the elements and attributes to be defined in the `wsdl:portType` element:

Table 2–26: Elements and attributes to be defined in the `wsdl:portType` element

Element	Attributes	Meaning	Value	Number
<code>wsdl:portType</code>	--	Port type definition element	--	1
	name	Port type name attribute	Define any string <sup>#</sup> .	1
<code>wsdl:operation</code>	--	Operation definition element	--	From 1 up to number of operations
	name	Operation name attribute	Define the operation name.	1
<code>wsdl:input</code>	--	Request message definition element	--	1
	Message	Message reference attribute	Specify the name attribute value of the <code>wsdl:message</code> element defined in (e) <i>Defining the wsdl:message element</i> with QName.	1
<code>wsdl:output</code>	--	Response message definition element	--	1
	Message	Message reference attribute	Specify the name attribute value of the <code>wsdl:message</code> element defined in (e) <i>Defining the wsdl:message element</i> with QName.	1
<code>wsdl:fault</code>	--	Fault message definition element	--	From 1 up to number of faults
	name	Fault message name attribute	Define fault name.	1
	Message	Fault message reference attribute	Specify the name attribute value of the <code>wsdl:message</code> element defined in (e) <i>Defining the wsdl:message element</i> with QName.	1

Legend:

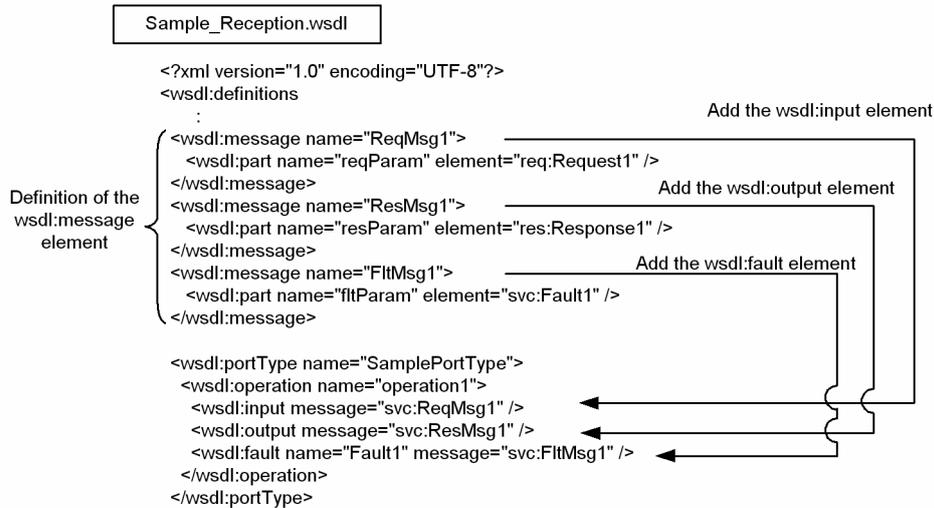
--: Not applicable

#

There are limited character types that can be used. For details about the character types that can be used, see "2.7.3(2) Characters that can be used for WSDL definition".

The following figure shows an example for defining the `wsdl:portType` element:

Figure 2–12: Definition example of the wsdl:portType element

**(g) Defining the wsdl:binding element**

Define the wsdl:binding element, its lower elements, and attributes.

1. Define the wsdl:binding element as a child element of the wsdl:definitions element defined in (c) *Creating the template for a WSDL file (defining the wsdl:definitions element)*.
2. Define the name attribute and type attribute in the wsdl:binding element.
 

You can specify any string in the name attribute value.

In the type attribute value, specify the name attribute value of wsdl:portType element defined in (f) *Defining the wsdl:portType element* with QName.
3. Define the soap:binding element as a child element of the wsdl:binding element.
4. Define the transport attribute in the soap:binding element.
 

The transport attribute value is a fixed value. It is defined as `http://schemas.xmlsoap.org/soap/http`.
5. Define only the number of operations listed up by (a) *Listing the operations and fault names* for the wsdl:operation element.
 

Define the wsdl:operation element for all the operations listed in (1).
6. Define the name attribute in the wsdl:operation element.
 

In the name attribute value, specify the value same as the name attribute value of the wsdl:operation element that is the child element of the wsdl:portType element defined in (f) *Defining the wsdl:portType element*.
7. Define the soap:operation element as a child element of the wsdl:operation element.
8. Specify the soapAction attribute in the soap:operation element.
 

The value of the soapAction attribute is set to null.
9. Define the wsdl:input element, wsdl:output element, and wsdl:fault element as the child elements of the wsdl:operation element.
 

Define the child elements in the order of wsdl:input element, wsdl:output element, and wsdl:fault element.

If the reply activity for the fault message does not exist, you can omit the wsdl:fault element. If there are multiple reply activities for the fault message, define only for the number of activities that exist.

In the creation example, wsdl:input element, wsdl:output element, and wsdl:fault element are defined one by one.
10. Define the soap:body element as a child element of the wsdl:input element and wsdl:output element.
 

Value is not specified in the soap:body element.

11. Define the name attribute in the `wsdl:fault` element.

In the name attribute value, specify a value same as the name attribute value of the `wsdl:fault` element that is a child element of the `wsdl:operation` element defined in (f) *Defining the wsdl:portType element*.

12. Define the `soap:fault` element as a child element of the `wsdl:fault` element.

13. Define the name attribute in the `soap:fault` element.

In the name attribute value, specify a value same as the name attribute of the `wsdl:fault` element that is the parent element.

The following table describes the elements and attributes to be defined in the `wsdl:binding` element:

Table 2–27: Elements and attributes to be defined in the `wsdl:binding` element

Element	Attributes	Meaning	Value	Number
<code>wsdl:binding</code>	--	Binding definition element	--	1
	name	Binding name attribute	Define any string <sup>#</sup> .	1
	type	Port type reference attribute	Specify the name attribute value of <code>wsdl:portType</code> defined in (f) <i>Defining the wsdl:portType element</i> in QName.	1
<code>soap:binding</code>	--	SOAP binding definition element	--	1
	transport	SOAP binding transmission type definition attribute	Define <code>http://schemas.xmlsoap.org/soap/http</code> (fixed value).	1
<code>wsdl:operation</code>	--	Operation definition element	--	From 1 up to number of operations
	name	Operation name attribute	Define the operation name. Specify the value same as the name attribute value of the <code>wsdl:fault</code> element that is a child element of the <code>wsdl:operation</code> element defined in (f) <i>Defining the wsdl:portType element</i> .	1
<code>soap:operation</code>	--	SOAP binding operation definition element	--	1
	soapAction	SOAPAction header definition attribute	--	1
<code>wsdl:input</code>	--	Request message definition element	--	1
<code>soap:body</code>	--	SOAP binding Body element definition element	--	1
<code>wsdl:output</code>	--	Response message definition element	--	1
<code>soap:body</code>	--	SOAP binding Body element definition element	--	1
<code>wsdl:fault</code>	--	Fault message definition element	--	From 1 up to number of faults
	name	Fault message name attribute	Define fault name.	1

## 2. Defining User-Defined Reception

Element	Attributes	Meaning	Value	Number
soap:fault	--	SOAP binding Fault element definition element	--	1
	name	SOAP binding name attribute	Define fault name.	1

Legend:

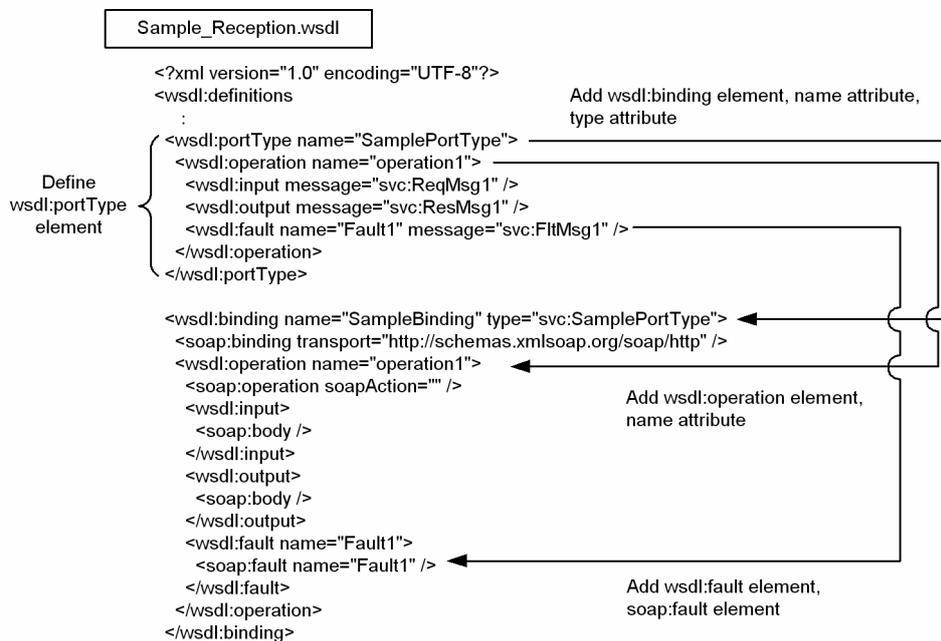
--: Not applicable

#

There are limited character types that can be used. For details about the character types that can be used, see "2.7.3(2) Characters that can be used for WSDL definition".

The following figure shows an example for defining the `wSDL:binding` element:

Figure 2–13: Definition example of the `wSDL:binding` element



### To use the style attribute and use attribute

You can also use the style attribute and use attribute in the `wSDL:binding` element to clearly specify document or literal.

Figure 2–14: Definition example of the wsdl:binding element (When the style attribute and use attribute are used)

Sample\_Reception.wsdl

```

<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions
:
  <wsdl:binding name="SampleBinding" type="svc:SamplePortType">
    <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http" />
    <wsdl:operation name="operation1">
      <soap:operation soapAction="" />
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal" />
      </wsdl:output>
      <wsdl:fault name="Fault1">
        <soap:fault name="Fault1" />
      </wsdl:fault>
    </wsdl:operation>
  </wsdl:binding>

```

Explicitly display style="document"

Explicitly display use="literal"

#### (h) Defining the wsdl:service element

Define the `wsdl:service` element, its lower elements, and attributes.

1. Define the `wsdl:service` element as a child element of the `wsdl:definitions` element defined in (c) *Creating the template for a WSDL file (defining the wsdl:definitions element)*.
2. Define the name attribute in the `wsdl:service` element.  
You can specify any string in the name attribute value.
3. Define one `wsdl:port` element as the child element of the `wsdl:service` element.
4. Define the name attribute and binding attribute in the `wsdl:port` element.  
You can specify any string in the name attribute value.  
As the binding attribute value, specify the name attribute value of the `wsdl:binding` element defined in (g) *Defining the wsdl:binding element* with QName.
5. Define one `soap:address` element as a child element of the `wsdl:port` element.
6. Define the location attribute in the `soap:address` element.  
As the location attribute value, `http://localhost:80/SampleService/services/PortName` is specified as a temporary value.  
Specify the value same as the name attribute defined in step 4. in `PortName`.

#### Tip

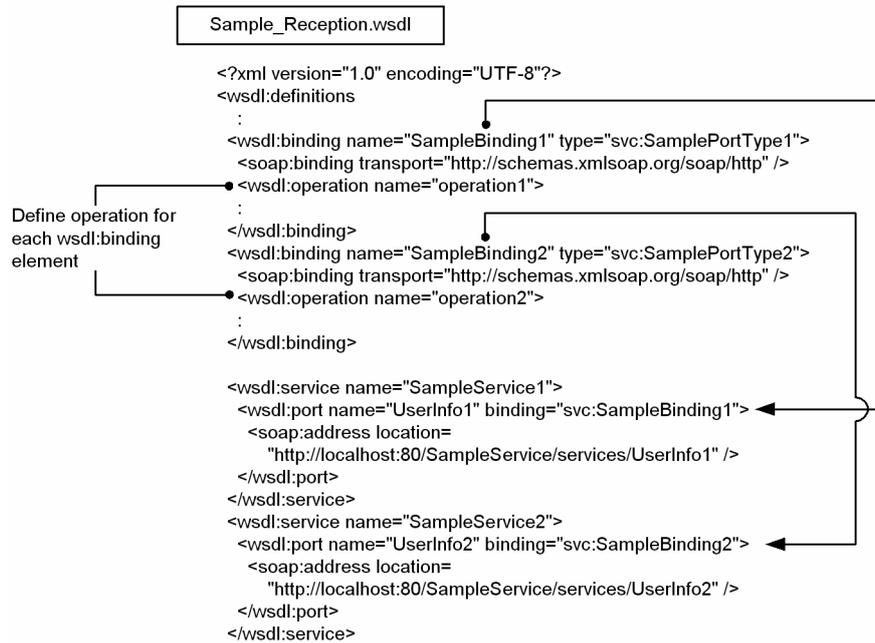
At the stage when WSDL is created, the service location value (location attribute value of the `soap:address` element in the `wsdl:port` element) is not yet determined. Therefore, specify a temporary value. This value does not cause an error in the SOAP Communication Infrastructure and JAX-WS engine provided by Cosminexus.

The service location value is fixed after specifying the deployment definition of the HCSC component in which the user definition file is included. Edit WSDL, set up the fixed service location value, and then create the service requester based on WSDL. For details on editing WSDL, see 8.7.2 *Editing a WSDL*.

The following table describes the elements and attributes to be defined in the `wsdl:service` element:



Figure 2–16: Definition example of the wsdl:service element (When multiple wsdl:port elements are used)



## 2.7.2 Examples of dynamically changing the connection-destination information of the service adapter

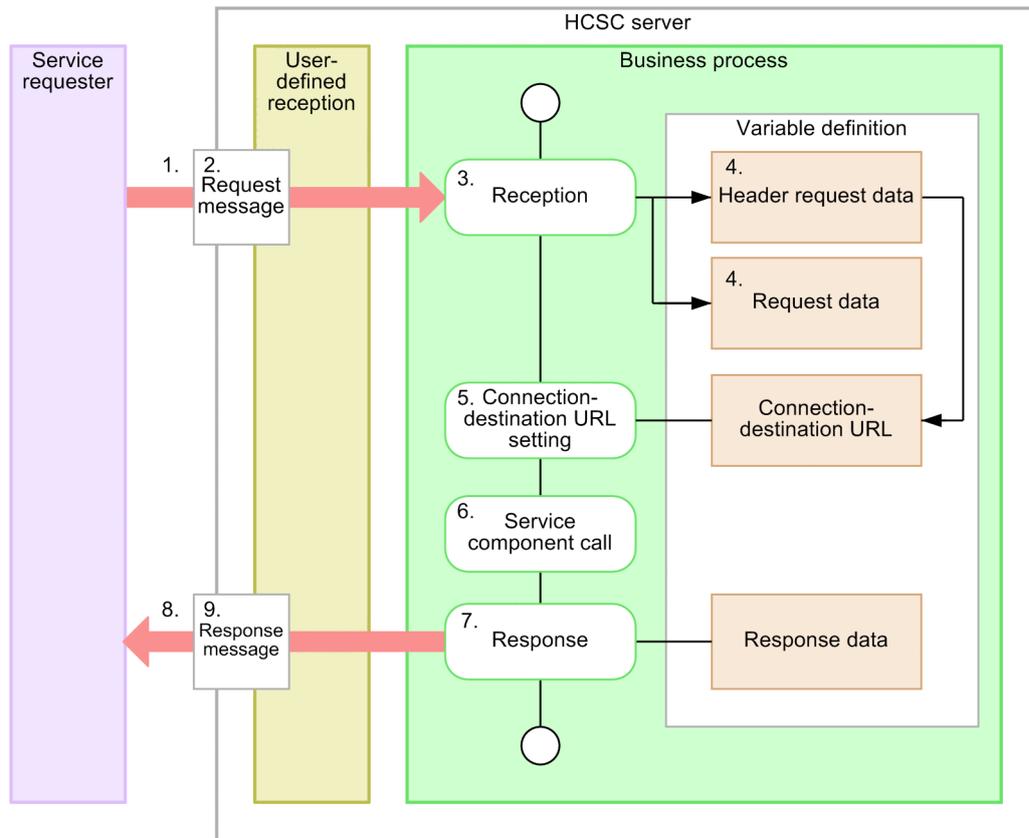
You can dynamically change the connection-destination information of the service adapter, by using the connection-destination information file.

The following describes examples of using the connection-destination information file when SOAP communication is performed by using a SOAP reception and service adapter.

### (1) Example of a business process

The following figure shows a business process as an example of how to use the connection-destination information file.

Figure 2–17: Example of a business process that uses the connection-destination information file



1. The service requester requests the business process to execute the service component.
2. The connection-destination URL is set in the header message of the request message that is sent from the service requester.
3. The business process receives the XML request message via the receive activity.
4. The business process generates header request data and request data.
5. The connection-destination URL of the header request data is set for the connection-destination URL variable in the data transformation activity or other activities.
6. The business process obtains the connection-destination URL from the connection-destination URL variable, and then calls the service component.
7. The business process configures an XML response message from the execution result of the service component, and returns it via the reply activity.
8. The response is returned to the service requester.
9. The XML response message is sent as the response to the service requester.

## (2) Example of a WSDL that generates a SOAP reception

The following shows an example of creating a WSDL that generates a SOAP reception.

In this example, the URL of the connection-destination service component is sent contained in the SOAP header.

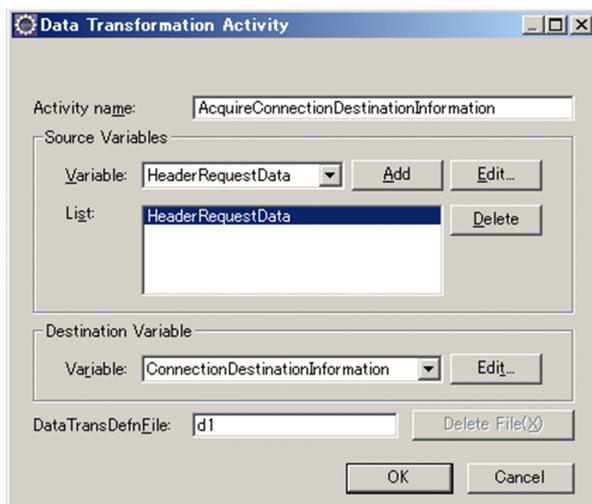


(3) Example configuration of the data transformation activity

The following shows an example configuration of the data transformation activity that is used by the business process shown in Figure 2-17 Example of a business process that uses the connection-destination information file.

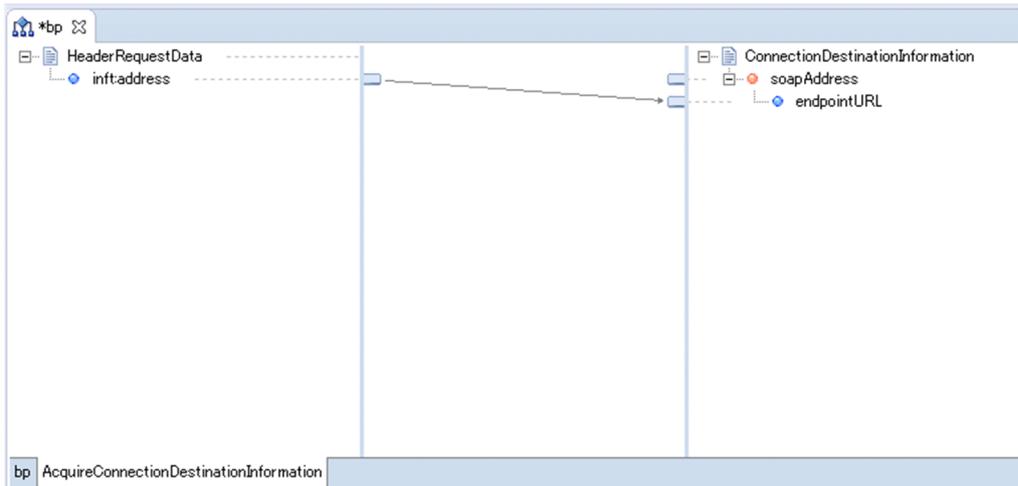
In this example, *HeaderRequestData* is specified as the source, and *ConnectionDestinationInformation* is specified as the target.

Figure 2-18: Example configuration of the data transformation activity



Define the transformation mapping as shown in the following figure.

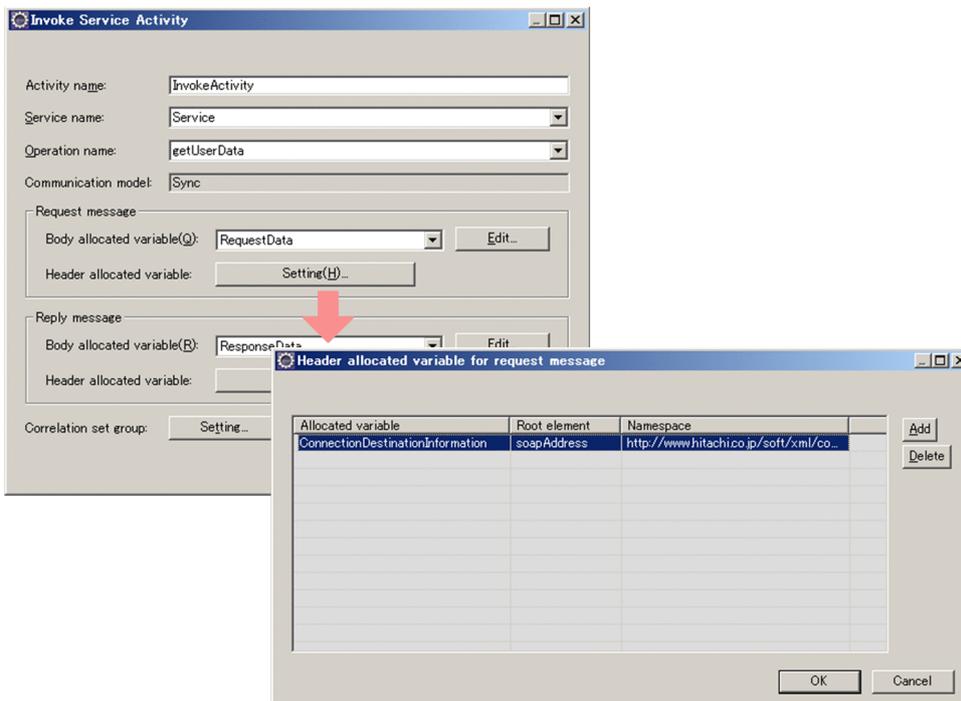
Figure 2–19: Example configuration of transformation mapping



#### (4) Configuring the connection-destination information

The following shows an example configuration of the connection-destination information that is used by the business process shown in Figure 2-17 Example of a business process that uses the connection-destination information file. Note that the connection-destination information must be set for the header-allocated variable of the invoke service activity. The information set here is the information that can be viewed on the service adapter.

Figure 2–20: Example configuration of connection-destination information (invoke service activity settings)



The connection-destination information files for individual service adapters are stored in the following locations:

- Storage location for the connection-destination information file for the DB adapter  
*installation-directory-of-the-service-platform\CSC\schema\connection\connection\_db\_adapter.xsd.*
- Storage location for the connection-destination information file for the SOAP adapter

*installation-directory-of-the-service-platform\CSC\schema\connection\connection\_soap\_adapter.xsd.*

- Storage location for the connection-destination information file for the TP1 adapter

*installation-directory-of-the-service-platform\CSC\schema\connection\connection\_tp1\_adapter.xsd.*

#### ! Important note

When you dynamically change the connection-destination information of the service adapter, use the connection-destination information file provided by the service platform. If you do not use the provided connection-destination information file, operations are not guaranteed to work.

## 2.7.3 Notes on creating WSDL of a user-defined reception

This subsection contains notes on creating a WSDL of a SOAP reception and the characters that can be used for defining the WSDL when the SOAP1.1 mode is used.

For notes on using the SOAP1.1/1.2 combined mode, see the explanation of defining and using WSDL (SOAP1.1/1.2 combined mode) in 2.6.1 *Applicability of the service components that use Web service* in the manual *Service Platform Basic Development Guide*.

### (1) Notes on creating a WSDL (for a SOAP reception)

Note the following when you create a WSDL for a SOAP reception:

- Specify a valid value for the name space (`targetNamespace`) for all XML schemata under the `wSDL:types` element.
- Define the XML schema within the supported range of the XML schema described in the manual *Application Server SOAP Application Development Guide*.
- You cannot specify the `wSDL:import` element.
- If the `style` attribute is `document`, do not specify the `namespace` attribute for the `soap:body` element.
- If the `style` attribute is `rpc`, specify the `namespace` attribute for the `soap:body` element.
- You cannot specify the `soap:header` element.
- The value for the `name` attribute of the `soap:fault` element must be the same as the value for the `name` attribute of the parent `wSDL:fault` element. Do not specify the `namespace` attribute.

### (2) Characters that can be used for WSDL definition

When defining a WSDL, you can use unreserved characters defined by RFC3986 as well as Japanese characters. The following shows the characters that can be used for defining a WSDL, and contains notes on using the characters.

#### (a) For "document/literal"

You can use one-byte alphanumeric characters (A to Z, a to z, 0 to 9), underscore (`_`), and the following characters:

- Two-byte *hiragana* and *katakana* characters
- Two-byte Greek and Russian characters
- Repetition symbols ( `>` , `<` , `々` )
- Chinese characters (included in the first and second standards of JIS X 0208)

For attributes where the data type of WSDL is `anyURI`, you can also use the following characters:

- Hyphens (`-`)
- Periods (`.`)
- Tildes (`~`)

## 2. Defining User-Defined Reception

### (b) For "rpc/literal"

You can use one-byte alphanumeric characters (A to Z, a to z, 0 to 9), and underscore (`_`). For attributes where the data type of WSDL is `anyURI`, you can also use the following characters:

- Hyphens (`-`)
- Periods (`.`)
- Tildes (`~`)

## 2.8 Checking User-Defined Reception Contents

---

Add a user-defined reception, and check the defined contents. You can check the contents of user-defined reception with the User-Defined Reception Definition screen. The User-Defined Reception Definition screen opens, after a user-defined reception is added using the User-defined Reception Addition Wizard.

Whenever you want to open the User-Defined Reception Definition screen, follow the procedures given below:

1. Double click the business process containing the user-defined reception in the service definition list of the tree view.  
The User-defined Reception Definition screen opens.
2. Click the tab that displays the reception name of user-defined reception in the lower part of the Business Process Definition screen canvas.  
The window display switches to the User-Defined Reception Definition screen.

For details about the User-Defined Reception Definition screen, see the manual *Service Platform Overview*.

In the User-Defined Reception Definition screen, check the definition contents of user-defined reception and also confirm their consistency with the definition of the receive activity and reply activity. If there is any inconsistency, change the definition of the receive activity or reply activity.

## 2.9 Saving a user-defined reception

---

You can save a user-defined reception during or after editing in the User-Defined Reception Definition screen. The contents of the user-defined reception are saved in the repository.

The procedure for saving a user-defined reception is as follows:

**Method 1**

Choose **File** and **Save** in the Eclipse menu.

**Method 2**

Choose **File** and **Save all** in the Eclipse menu.

**Method 3**

Click the **Save** icon in the Eclipse toolbar.

**Method 4**

Click the **Ctrl** key and the **S** key in the User-Defined Reception Definition screen.

Note that if the User-Defined Reception Definition screen is closed and the user-defined reception is defined, if the user-defined reception is not saved, the save resource dialog box appears. You can save the definition of the unsaved user-defined reception in the save resource dialog box.

**! Important note**

If invalid data is entered, the user-defined reception might not be saved. In such cases, take action according to the displayed message.

---

## 2.10 Validating a User-Defined Reception

In the uCosminexus Service Architect, you can validate whether the contents of the created user-defined reception are appropriate.

### 2.10.1 Validation Contents

The following table describes the validation content and corrective actions in the validation for a user-defined reception:

Table 2–29: Validation contents and actions for a user-defined reception

Item No.	Validated content	Corrective action to be taken
1	Is an operation name that matches the operation name of the user-defined reception defined in the business process?	Check the operation name of the receive activity of the business process and the user-defined reception and if necessary, revise to define a matching operation name.
2	Does the communication model (synchronous/asynchronous) of the User-Defined Reception Operations match with the communication model (synchronous/asynchronous) of the business process operations?	Check the definition of the receive activity for the business process and the user-defined reception and if necessary, revise to match with the communication model (synchronous/asynchronous) of the operation.
3	Is the type of the allocated variable of the receive activity and reply activity (including those for fault messages) corresponding the user-defined reception, a message type?	Check the business process definition, and if necessary revise and change the allocated variable of the relevant activity to a variable that corresponds to user-defined reception.
4	Does the reply activity (not including those for fault messages) of the response message for user-defined reception, exist?	Check the business process definition, and if necessary revise and define the reply activity corresponding to the User-Defined Reception Operations.
5	Does a fault name that matches the fault name for the User-Defined Reception Operations exist in the reply activity of the business process?	Check the business process and user-defined reception definitions, and if necessary revise and take any one of the following measures: <ul style="list-style-type: none"> <li>• Define the reply activity corresponding to the fault name in the User-Defined Reception Operations.</li> <li>• Delete the unnecessary fault definitions from the User-Defined Reception Operations. In this case, delete the user-defined reception once, and then re-create from the WSDL file from where the fault definition was deleted.</li> </ul>
6	Is a fault name that does not exist in the User-Defined Reception Operations defined in the reply activity of the business process?	Check the business process and user-defined reception definitions, and if necessary revise and take any one of the following measures: <ul style="list-style-type: none"> <li>• Delete the reply activity corresponding to the fault that is not defined in the User-Defined Reception Operations.</li> <li>• Add the fault definition in the User-Defined Reception Operations. In this case, delete the user-defined reception once, and then re-create from the WSDL file where the fault definition was added.</li> </ul>

### 2.10.2 Validation Method

The user-defined reception is validated when the business process is validated. If you validate the business process, the user-defined reception in the business process to be validated, is validated.

For details about how to validate a business process, see *5.10.2 Validation Method* in the manual *Service Platform Basic Development Guide*.

## 2. Defining User-Defined Reception

Furthermore, the validation result of the user-defined reception is displayed in the console view same as the validation result of the business process. For details on the display of validation results, see *5.10.3 Displaying the Validation Contents* in the manual *Service Platform Basic Development Guide*.

## 2.11 Changing the information of a user-defined reception

---

You can change the reception name, reception ID, and context root in the User-Defined Reception Definition screen.

The procedure for changing information of a user-defined reception is as follows:

1. In the tree view service definition list, double-click the user-defined reception.  
The User-Defined Reception Definition screen appears.
2. Enter the post-change value in the **Reception name**, **Reception ID** and **Context root** text boxes of **User-defined reception information**.

For details about the User-Defined Reception Definition screen and values that can be entered, see the manual *Service Platform Reference Guide*.

## 2.12 Deleting a User-Defined Reception

---

Delete a user-defined reception. Note that you cannot delete the following user-defined receptions:

- User-defined reception in the published business processes
- User-defined reception created in a previous version of the business process

### (1) Simultaneous deletion of multiple user-defined receptions

The procedure to delete multiple user-defined receptions simultaneously is explained below:

1. In the tree view service definition list, select the business process for deleting the user-defined reception, and right click.

The Service List pop-up menu opens.

2. From the popup menu, choose Delete User-Defined Reception.

The Delete User-Defined Reception dialog box opens. For details about the Delete User-Defined Reception dialog box, see the manual *Cosminexus Service Platform Overview*.

3. Select the user-defined receptions to be deleted.

4. Click OK.

The specified user-defined reception is deleted from the business process.

#### Important note

If you delete a business process, the user-defined reception included in the deleted business process is also deleted.

---

### (2) Deleting a single user-defined reception

The procedure to delete a single user-defined reception is as follows:

#### Method 1

1. In the tree view service definition list, right click the user-defined reception to be deleted.

The popup menu of the user-defined reception to be deleted appears.

2. From the popup menu, choose **Delete User-Defined Reception**.

The confirmation dialog box appears.

3. Click the **Yes** button.

The specified user-defined reception is deleted from the business process.

#### Method 2

1. In the tree view service definition list, choose the user-defined reception to be deleted.

2. Click the **Delete** key.

The confirmation dialog box appears.

3. Click the **Yes** button.

The specified user-defined reception is deleted from the business process.

# 3

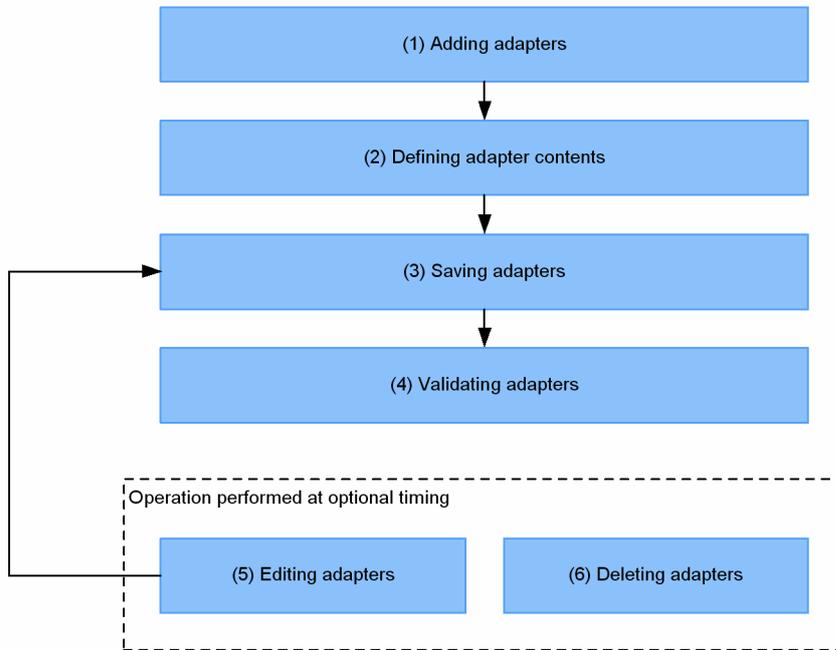
## Defining Adapters

This chapter describes the definitions of various service adapters.

## 3.1 Workflow for Defining Service Adapters

The following figure shows the workflow for defining service adapters.

Figure 3–1: Workflow of defining service adapters



The following describes tasks for defining service adapters.

### (1) Adding service adapters

To add a service adapter, perform either of the following:

When adding a new service adapter

Use the wizard to add a new service adapter. There are two types of adapters: standard service adapters, and custom adapters, which are available to users as required.

The wizard used to add a new service adapter differs depending on the type of service component (such as Web service, Session Bean, or MDB (WS-R or DB queue)) to be called by the service adapter.

For details about how to add a new service adapter, see *3.2.1 Adding a new SOAP adapter*.

The following are the types of service adapters:

- SOAP adapter
- Session Bean adapter
- MDB (WS-R) adapter
- MDB (DB queue) adapter
- DB adapter
- TP1 adapter
- File adapter
- Object Access adapter
- Message Queue adapter
- FTP adapter
- File operation adapter
- Mail adapter

- HTTP adapter
- Custom adapter

For details about how to add individual service adapters, see the following locations:

Type	Reference location
SOAP adapter	<i>3.2.1 Adding a new SOAP adapter</i>
Session Bean adapter	<i>3.2.2 Adding a new Session Bean adapter</i>
MDB (WS-R) adapter	<i>3.2.3 Adding a new MDB (WS-R) adapter</i>
MDB (DB queue) adapter	<i>3.2.4 Adding a new MDB (DB queue) adapter</i>
DB adapter	<i>3.2.5 Adding a new database adapter</i>
TP1 adapter	<i>3.2.6 Adding a new TP1 adapter</i>
File adapter	<i>3.2.7 Adding a new file adapter</i>
Object Access adapter	<i>3.2.8 Adding a new Object Access adapter</i>
Message Queue adapter	<i>3.2.9 Adding a new Message Queue adapter</i>
FTP adapter	<i>3.2.10 Adding a new FTP adapter</i>
File operation adapter	<i>3.2.11 Adding a new file operation adapter</i>
Mail adapter	<i>3.2.12 Adding a new mail adapter</i>
HTTP adapter	<i>3.2.13 Adding a new HTTP adapter</i>
Custom adapter	<i>3.2.14 Adding a new custom adapter</i>

When adding a defined service adapter

You can add a service adapter by copying an already defined service adapter. If you copy an already defined service adapter, a new service adapter with the same definitions will be added. You can also edit the definitions of the new copy.

For details about how to add a service adapter by using an already defined service adapter, see *3.2.15 Using an Already Defined Adapter to Add an Adapter*.

## (2) Defining the contents of service adapters

Use the Service Adapter Settings window to define the contents of service adapters.

For details about the definitions of service adapters and how to define them, see *3.3 Defining the Contents of Service Adapters*. For details about the Service Adapter Settings window, see *1.2.2 Service Adapter Definition Window* in the manual *Service Platform Reference Guide*.

## (3) Saving service adapters

You need to save the definitions of the edited service adapters to the repository, as necessary.

For details about how and when to save the definitions of service adapters, see *3.4 Saving Adapters*.

## (4) Validating service adapters

Validate the consistency of defined service adapters. You can validate service adapters at any timing (for example, during, or after the definition of a service adapter).

For details about how to validate service adapters, see *3.6 Validating Adapters*.

## (5) Editing service adapters

You can edit the contents of defined service adapters as necessary.

For details about how to edit service adapters, see *3.5 Editing Adapters*.

### (6) Deleting service adapters

You can delete unnecessary service adapters as necessary.

For details about how to delete service adapters, see [3.7 \*Deleting Adapters\*](#).

## 3.2 Adding Service Adapters

---

Add service adapters.

To add a new service adapter, use the wizard for adding adapters. You can also add a new service adapter by using an already created service adapter.

### ! Important note

If a warning message is displayed, check the contents of the displayed message with reference to the following parts in the manual *Application Server Messages*:

- When the displayed warning message begins with KDCCC:  
See *5.2 Messages from KDCCC0001 to KDCCC9999*.
  - When the displayed warning message begins with KDJW:  
See *Chapter 10. Messages from KDJW00000 to KDJW99999 (Messages Output by Cosminexus Component Container)*.
- 

### 3.2.1 Adding a new SOAP adapter

To add a new SOAP adapter:

1. From the Eclipse menu, select **Window, Show View**, and then **Other**.  
The Show View dialog box appears.
2. Select **HCSC-Definer** and then **HCSCTE View**, and click the **OK** button.  
The Service Definition List appears in the tree view.
3. Right-click the Service Definition List in the tree view, and select **Add Service Adapter**.  
A dialog box appears, in which you can set the service type used by the service adapter to be added.
4. From the drop-down list, select **Web Services**.
5. Click the **Next** button.  
A dialog box for entering the information necessary for adding a SOAP adapter appears.
6. Specify the values for **Service name** and **WSDL file**.  
To specify **WSDL file**, select either of the following:
  - **File**  
Specify the path to the WSDL file.  
Select the **File** radio button, and then specify the absolute path to a file whose extension is `.wsdl`. Do not specify WSDL files by using a relative path (for example, `wsldir\wsdlfile.wsdl`) or UNC format (for example, `\\mypc\wsldir\wsdlfile.wsdl`).
  - **URL**  
Specify the WSDL by using a URL.  
Select the **URL** radio button, and then specify the URL to be referenced in the text box.
7. Click the **Next** button.  
A dialog box for specifying the port appears.  
When you click the **Next** button, if the message `Failed to analyze the WSDL file` is displayed, there is an error in the WSDL file.  
For details about the WSDL format, see *2.6.1 Applicability of the service components that use Web services* in the manual *Service Platform Basic Development Guide*.
8. From the drop-down list, select the port.
9. Click the **Finish** button.  
Clicking the **Finish** button creates the necessary files and saves them to the repository.  
The Service Adapter Settings window appears, in which you can define SOAP adapters. For details about how to specify the settings in the Service Adapter Settings window, see *3.3.15(1) For SOAP adapters*.

## 3.2.2 Adding a new Session Bean adapter

To add a new Session Bean adapter:

1. From the Eclipse menu, select **Window, Show View**, and then **Other**.  
The Show View dialog box appears.
2. Select **HCSC-Definer** and then **HCSCTE View**, and click the **OK** button.  
The Service Definition List appears in the tree view.
3. Right-click the Service Definition List in the tree view, and select **Add Service Adapter**.  
A dialog box appears, in which you can set the service type used by the service adapter to be added.
4. From the drop-down list, select **SessionBean**.
5. Click the **Next** button.  
A dialog box for entering the information necessary for adding a Session Bean adapter appears.
6. Enter the values for **Service name** and **EAR file**.  
For the value for **EAR file**, specify the absolute path to a file whose extension is `.ear`.
7. Click the **Next** button.  
A dialog box for specifying the Session Bean appears.
8. From the drop-down list, select a Session Bean.
9. Click the **Finish** button.  
Clicking the **Finish** button creates the necessary files and saves them to the repository.  
The Service Adapter Settings window appears, in which you can define Session Bean adapters. For details about how to specify the settings in the Service Adapter Settings window, see *3.3.15(2) For the SessionBean adapters*.

## 3.2.3 Adding a new MDB (WS-R) adapter

1. From the service definition list in the tree view, choose and right-click **Add Service Adapter**.  
A dialog box for specifying the service type to be used by the service adapter being added opens.
2. On the drop-down list, choose **MDB/WS-R** or **MDB/DBQueue**.
3. Click **Next**.  
A dialog box for entering the information necessary for adding a service adapter opens.
4. Enter a service name.
5. Click **Finish**.  
The required files are created and saved in a repository.  
The Service Adapter Definition screen opens.

## 3.2.4 Adding a new MDB (DB queue) adapter

1. From the service definition list in the tree view, choose and right-click **Add Service Adapter**.  
A dialog box for specifying the service type to be used by the service adapter being added opens.
2. On the drop-down list, choose **MDB/WS-R** or **MDB/DBQueue**.
3. Click **Next**.  
A dialog box for entering the information necessary for adding a service adapter opens.
4. Enter a service name.
5. Click **Finish**.  
The required files are created and saved in a repository.  
The Service Adapter Definition screen opens.

### 3.2.5 Adding a new database adapter

To add a new DB adapter:

1. On the Eclipse menu, choose Window, Display View, and then Other.
2. The View Display dialog box opens.
3. Choose HCSC-Definer, HCSCTE View, and click OK.  
The service definition list is displayed in the tree view.
4. From the service definition list in the tree view, choose and right-click Add Service Adapter.  
A dialog box for specifying the service type to be used by the service adapter being added opens.
5. On the drop-down list, choose Custom adapter.  
When you click the **Next** button, a dialog box will appear for entering the information required for adding the service adapter.
6. Enter a service name.  
The *service name* is the identification information required when invoking the HCSC server from service requesters.
7. Enter the EAR file.  
For the EAR file, specify an absolute path to a file with the `.ear` extension.  
The EAR file to be specified is `Cosminexus-installation-directory\CSC\lib\cscdba.ear`.
8. Click Finish.  
When you click the **Finish** button, the required files are created and saved in the repository.  
The Service Adapter Definition screen opens. Define the service adapter in this window.

### 3.2.6 Adding a new TP1 adapter

INTENTIONALLY DELETED

### 3.2.7 Adding a new file adapter

To add a new file adapter:

1. From the Eclipse menu, select **Window, Show View**, and then **Other**.  
The Show View dialog box appears.
2. Select **HCSC-Definer** and then **HCSCTE View**, and click the **OK** button.  
The Service Definition List appears in the tree view.
3. Right-click the Service Definition List in the tree view, and select **Add Service Adapter**.  
A dialog box appears, in which you can set the service type used by the service adapter to be added.
4. From the **Service component type:** drop-down list, select **File adapter**.  
Clicking the **Next** button displays a dialog box for entering the information necessary for adding the file adapter.
5. Enter the value for **Service name**.  
For **Service name**, enter any service name.
6. Click the **Finish** button.  
Clicking the **Finish** button creates the necessary files and saves them to the repository.  
The Service Adapter Settings window appears, in which you can define the file adapter. For details about how to specify the settings in the Service Adapter Settings window, see 3.3.15(6) *For file adapters*.

### 3.2.8 Adding a new Object Access adapter

INTENTIONALLY DELETED

### 3.2.9 Adding a new Message Queue adapter

To add a new Message Queue adapter:

1. From the Eclipse menu, select **Window, Show View**, and then **Other**.  
The Show View dialog box appears.
2. Select **HCSC-Definer** and then **HCSCTE View**, and click the **OK** button.  
The Service Definition List appears in the tree view.
3. Right-click the Service Definition List in the tree view, and select **Add Service Adapter**.  
A dialog box appears, in which you can set the service type used by the service adapter to be added.
4. From the **Service component type:** drop-down list, select **Message Queue adapter**.  
Clicking the **Next** button displays a dialog box for entering the information necessary for adding the Message Queue adapter.
5. Enter the value for **Service name**.  
For **Service name**, enter any service name.
6. Click the **Finish** button.  
Clicking the **Finish** button creates the necessary files and saves them to the repository.  
The Service Adapter Settings window appears, in which you can define the Message Queue adapter. For details about how to specify the settings in the Service Adapter Settings window, see 3.3.15(7) *For Message Queue adapters*.

### 3.2.10 Adding a new FTP adapter

To add a new FTP adapter:

1. From the Eclipse menu, select **Window, Show View**, and then **Other**.  
The Show View dialog box appears.
2. Select **HCSC-Definer** and then **HCSCTE View**, and click the **OK** button.  
The Service Definition List appears in the tree view.
3. Right-click the Service Definition List in the tree view, and select **Add Service Adapter**.  
A dialog box appears, in which you can set the service type used by the service adapter to be added.
4. From the **Service component type:** drop-down list, select **FTP adapter**.  
Clicking the **Next** button displays a dialog box for entering the information necessary for adding the FTP adapter.
5. Enter the value for **Service name**.  
For **Service name**, enter any service name.
6. Click the **Finish** button.  
Clicking the **Finish** button creates the necessary files and saves them to the repository.  
The Service Adapter Settings window appears, in which you can define the FTP adapter. For details about how to specify the settings in the Service Adapter Settings window, see 3.3.15(8) *For FTP adapters*.

### 3.2.11 Adding a new file operation adapter

To add a new file operation adapter:

1. From the Eclipse menu, select **Window, Show View**, and then **Other**.  
The Show View dialog box appears.
2. Select **HCSC-Definer** and then **HCSCTE View**, and click the **OK** button.  
The Service Definition List appears in the tree view.
3. Right-click the Service Definition List in the tree view, and select **Add Service Adapter**.  
A dialog box appears, in which you can set the service type used by the service adapter to be added.
4. From the **Service component type:** drop-down list, select **File operation adapter**.

Clicking the **Next** button displays a dialog box for entering the information necessary for adding the file operation adapter.

5. Enter the value for **Service name**.

For **Service name**, enter any service name.

6. Click the **Finish** button.

Clicking the **Finish** button displays the Service Adapter Settings window, in which you can define the file operation adapter. For the settings for individual operations, see the following locations:

- For file transformation operations:  
See 3.3.15(9)(a) *File transformation operation*.
- For file replication operations:  
See 3.3.15(9)(b) *File replication operation*.
- For file and folder deletion operations:  
See 3.3.15(9)(c) *File and folder deletion operation*.
- For file compression operations:  
See 3.3.15(9)(d) *File compression operation*.
- For file extraction operations:  
See 3.3.15(9)(e) *File extraction operation*.

### 3.2.12 Adding a new mail adapter

To add a new mail adapter:

1. From the Eclipse menu, select **Window, Show View**, and then **Other**.

The Show View dialog box appears.

2. Select **HCSC-Definer** and then **HCSCTE View**, and click the **OK** button.

The Service Definition List appears in the tree view.

3. Right-click the Service Definition List in the tree view, and select **Add Service Adapter**.

A dialog box appears, in which you can set the service type used by the service adapter to be added.

4. From the **Service component type**: drop-down list, select **Mail adapter**.

Clicking the **Next** button displays a dialog box for entering the information necessary for adding the mail adapter.

5. Enter the value for **Service name**.

For **Service name**, enter any service name.

6. Click the **Finish** button.

Clicking the **Finish** button creates the necessary files and saves them to the repository.

The Service Adapter Settings window appears, in which you can define the mail adapter. For details about how to specify the settings in the Service Adapter Settings window, see 3.3.15(10) *For mail adapters*.

### 3.2.13 Adding a new HTTP adapter

To add a new HTTP adapter:

1. From the Eclipse menu, select **Window, Show View**, and then **Other**.

The Show View dialog box appears.

2. Select **HCSC-Definer** and then **HCSCTE View**, and click the **OK** button.

The Service Definition List appears in the tree view.

3. Right-click the Service Definition List in the tree view, and select **Add Service Adapter**.

A dialog box appears, in which you can set the service type used by the service adapter to be added.

4. From the **Service component type**: drop-down list, select **HTTP adapter**.

Clicking the **Next** button displays a dialog box for entering the information necessary for adding the HTTP adapter.

5. Enter the value for **Service name**.

For **Service name**, enter any service name.

6. Click the **Finish** button.

Clicking the **Finish** button creates the necessary files and saves them to the repository.

The Service Adapter Settings window appears, in which you can define the HTTP adapter. For details about how to specify the settings in the Service Adapter Settings window, see *3.3.15(11) For HTTP adapters*.

## 3.2.14 Adding a new custom adapter

To add a new custom adapter:

1. On the Eclipse menu, choose Window, Display View, and then Other.

The View Display dialog box opens.

2. Choose HCSC-Definer, HCSCTE View, and click OK.

The service definition list is displayed in the tree view.

3. From the service definition list in the tree view, choose and right-click Add Service Adapter.

A dialog box for specifying the service type to be used by the service adapter being added opens.

4. On the drop-down list, choose Custom adapter.

When you click the **Next** button, a dialog box appears to enter the information required for adding custom adapters.

5. Enter a service name.

The *service name* is the identification information required when invoking the HCSC server from service requesters.

6. Enter the EAR file.

For the EAR file, specify an absolute path to a file with the `.ear` extension.

The EAR file that is to be specified is the file created in *3.3.14(6) Creating an EAR file*.

7. Click Finish.

When you click the **Finish** button, the required files are created and saved in the repository.

The Service Adapter Definition screen opens. You define the service adapter in this window.

## 3.2.15 Using an Already Defined Adapter to Add an Adapter

You can copy an already defined adapter to add adapters.

1. From the service definition list in the tree view, choose and right-click the adapter to be copied.

The **Service List** pop-up menu opens.

2. On the pop-up menu, click Copy.

A copy of the choosed adapter is created. A different service name and service ID are automatically assigned to the copied adapter to prevent any conflict within the system.

## 3.3 Defining the Contents of Service Adapters

---

A request received from a service requester can be sent to a service component by specifying a service adapter deployed in a cluster (or single HCSC server). The service component requires a service adapter that supports that service component. This section shows how to define service adapters.

### 3.3.1 Defining SOAP adapters

This subsection describes how to define SOAP adapters. You can define SOAP adapters in the service adapter settings screen.

#### (1) Service component message format

For Web service, the message format definition file created from WSDL will be the message format definition file of a service component. Therefore, you need not set the service component message format along with the request and response messages.

For details on how to create a service component message format, see *4.3.2 Creating a Service Component Message (for Web Services)* in the manual *Service Platform Basic Development Guide*.

#### (2) Data transformation

If the message format of the service requester and the message format of the service component invoked by the service adapter are different, data transformation is required.

When performing data transformation for a request message, specify a message entered from the service requester in the standard message using the service adapter settings screen, and then define data transformation in the data transformation definition screen. When performing data transformation for a response message, specify the standard message in the message returned from the service component, and then define data transformation in the data transformation definition screen.

For details on data transformation, see *6. Defining Data Transformation* in the manual *Service Platform Basic Development Guide*.

#### Important note

Note

Select the following elements in the root element of a message format to define data transformation in SOAP adapters:

- For document/literal  
Request message: Element indicating element attribute of <wsdl:message><wsdl:part> that is referenced by <wsdl:operation><wsdl:input>  
Response message: Element indicating element attribute of <wsdl:message><wsdl:part> that is referenced by <wsdl:operation><wsdl:output>
- For rpc/literal  
Request message: Element with the name same as "Operation name"  
Response message: Element with the name same as "Operation name + "Response"".

#### (3) Specifying the basic authentication

When the target service component is on a Web server that requires user authentication, you must set up the basic authentication (a user name and a password).

#### (4) Displaying the service adapter settings screen

Display the service adapter settings screen to define a service adapter. The service adapter settings screen is displayed, when you select and double-click the applicable service adapter from the service definition list in the tree view.

## (5) Operations to be performed in the service adapter settings screen

The procedure for defining SOAP adapters is as follows:

1. Display the service adapter settings screen.  
For details on how to display the service adapter settings screen, see 3.3.1(4) *Displaying the service adapter settings screen*.
2. In **Service component control information**, edit **Service name**, **Service ID**, **Address**, and **Maximum instances**, as necessary.
3. From the drop-down list of **Operation** of the service component control information, select the operation to be edited.
4. Check the contents displayed in the operation information.
5. Check the **Convert a system exception into a fault message** check box of the service component control information, if necessary.
6. Perform steps from 7 to 25 for request messages.
7. Perform the following operations:

**When you define SOAP header information in a request message**

Perform steps from 8 to 14, and then proceed to step 15.

**When you do not define SOAP header information in a request message**

Proceed to step 15.

8. Click the [Header] tab of request message.
9. Select the root element of the header information from the drop-down list of [Root element].  
Name space of the root element selected in [Name space] is displayed. Name of the message format file used as header information in [Message format] is displayed. Add the root element of header information, if necessary.
10. Perform the following operations:

**When you add the root element of header information**  
Follow the steps from 11 to 13, and then proceed to step 14.

**When you do not add the root element of header information**  
Proceed to step 14.
11. Click the [Add] button of [Root element].  
The [Select message format file] dialog box is displayed.
12. Select the message format used as the header information, and then click the [Open] button.  
The [Add header information] dialog box is displayed.
13. Select the root element of the header information to be added from the drop-down list of [Root element], and then click the [OK] button.  
The added root element is added in [Root element] of the [Header] tab of request message. The root element is to be added at the end of the drop-down list.
14. Click the [Display] button of message format.  
The message format to be used as the header information is displayed. Check the message format, if necessary.
15. Click the [Body] tab of a request message.
16. Check the [Use] check-box of a standard message.
17. Specify [Format ID] of a standard message.
18. Click the [Browse] button of a standard message and specify a standard message format in [Message format].  
Note that when specifying a message format that references an external XML schema, you must specify a file that corresponds to the root schema. The external XML schema file referenced by the root schema is automatically taken.  
For details on the style of the message format that can be specified, see 2.6.5 *Scoping of XML schema* in the manual *Service Platform Basic Development Guide*.
19. Click the [Display] button of a standard message.  
The standard message format is displayed. Check the specified standard message format, if necessary.

20. Specify [Format ID] of service component message.
21. Click the [Display] button of service component message.  
The service component message format is displayed. Check the service component message format, if necessary.
22. Perform the following operations:
  - When you select the [Use] check box of a standard message**  
Perform steps from 23 to 25, and then proceed to step 26.
  - When you do not select the [Use] check box of a standard message**  
Proceed to step 26.
23. Enter the name of data transformation definition file.
24. Click the [Edit] button.  
The data transformation definition screen is displayed.  
Note that when defining the message format initially, the [Select root element] dialog box is displayed.  
Also, if you change the message format, a dialog box for confirming whether to reflect the change of message format is displayed. For details on the procedure to be followed in case of changed message format, see *6.3.2 Procedure for defining changed message formats* in the manual *Service Platform Basic Development Guide*.
25. Map the standard message and the contents of the service component message.
26. Similarly perform steps from 7. to 25. for the response message as well.
27. Click the Service adapter definition (details) tab.
28. Depending on whether the service component is on a Web server that requires user authentication, perform either of the following operations:
  - When user authentication is required**  
Perform steps from 29. to 31.
  - When user authentication is not required**  
Rest of the steps are not required.
29. Select the [Use] check box for the basic authentication.
30. In the user name and password columns, enter a user name and password required for the basic authentication.
31. In the Password (Re-enter) column, enter the same password as that entered in the password column.

## (6) Setting a client definition file

A client definition file controls the actions on the client (service requester) side. Set a client definition file using either of the methods, if necessary:

- Edit the template file
- Create a file, overwrite, and set it on the template file.

Setting of the client definition file is optional. When you do not set the client definition file, the contents of the template file are used as are.

### (a) Setup procedure (When editing a template file)

The setup procedure to be followed when editing a template file is as follows:

1. Click the [Edit] button in [Client definition file] of the service adapter settings screen (details).  
The editor used for editing a client definition file is started.
2. Edit the client definition file using the editor.
3. From Eclipse menu, select [File] - [Save] and save the definition contents.

### (b) Setup procedure (Create a file and overwrite the template file)

The setup procedure to be followed when you create a file and overwrite the template file:

1. Create a client definition file using a text editor and save it as "c4webcl.properties".

### 3. Defining Adapters

Enter "c4webcl.properties" in lower case.

- Click the [Browse] button in [Client definition file] of the service adapter settings screen (details), and specify the client definition file that is created.

#### (c) Setting contents

Setting contents of the client definition file differ depending on whether you use SOAP1.1 mode or SOAP1.1/1.2 combination mode. The respective setting contents are described here.

- When using SOAP1.1 mode

For details on the setting contents of the client definition file, see *10.3 Setting up the client definition file* in the manual *Application Server SOAP Application Development Guide*.

The following table describes the points to be considered for editing a client definition file when using SOAP1.1 mode:

Table 3–1: Precautions for creating the client definition file(SOAP1.1 mode)

Set item	Key name	Set value (Default value)	Precautions
Multi-reference	c4web.common.do_multirefs	false	The operation cannot be guaranteed if you specify a value other than the default value.
Data type definition	c4web.common.send_xsi_types	true	
Character reference option	c4web.common.character_reference	false	
HTTP session maintenance	c4web.application.app_maintainsession	false	
Name modification check option for SOAP header	c4web.common.enable_soapheader_check	true	This option is operated as false.
Prefix for trace file and application log	c4web.logger.log_file_prefix	Service-ID	The value is always changed to a service ID. You are not required to set up the value.

#### Note#

For details on how to inherit HTTP header and Cookie information using HTTP header inheritance function of the service adapter, see "Appendix F Inheriting HTTP header and Cookie information using service adapter" in the manual "Service Platform Basic Development Guide".

- When using SOAP1.1/1.2 combination mode

For details on the setting contents of the client definition file, see *Client definition file* in the manual *Service Platform Reference Guide*.

## 3.3.2 Defining Service Adapters (SessionBean)

This subsection describes how to define Session Bean adapters. You can define Session Bean adapters in the Service Adapter Settings window.

### (1) Message format for service components

For Session Bean, the message format definition file automatically created from the EAR file will be the message format definition file for the service component. Therefore, you do not have to set the message format for a service component.

### (2) Data transformation

If the message format of the service requester is different from the message format of the service component to be called by the service adapter, data transformation is required.

To perform data transformation of a request message, in the Service Adapter Settings window, set the message to be input from the service requester as the standard message, and in the Data-conversion definition screen, define the data transformation. To perform data transformation of a response message, set the standard message as the message returned from the service component, and in the Data-conversion definition screen, define the data transformation.

For details about data transformation, see *Chapter 6. Defining Data Transformation* in the manual *Service Platform Basic Development Guide*.

### (3) Specifying the service component name to be called, from among those registered in the JNDI namespace

In the Service Adapter Settings window, specify the service component name that is to be called, from among those registered in the JNDI namespace. Specify the address so that remote call is used. You cannot use local call for service components (Session Bean). The following gives examples of specifying a service component name.

- Correct example (myhost is a remote host name.)  
`corbaname::myhost:900#HITACHI_EJB/SERVERS/J2EESEServer/EJB/slsbsmpl/MyAdder`
- Incorrect examples  
`corbaname::localhost:900#HITACHI_EJB/SERVERS/J2EESEServer/EJB/slsbsmpl/MyAdder`  
`corbaname::127.0.0.1:900#HITACHI_EJB/SERVERS/J2EESEServer/EJB/slsbsmpl/MyAdder`  
`corbaname::HITACHI_EJB/SERVERS/J2EESEServer/EJB/slsbsmpl/MyAdder`

For the names registered in the JNDI namespace, see the following locations:

- See *Chapter 2. Naming Management* in the manual *Application Server Common Container Functionality Guide*.
- See *2.5 Registering a reference in the JNDI name space of the EJB container* in the manual *Application Server EJB Container Functionality Guide*.

### (4) Setting a user definition class (JAR file)

For Session Bean, you need to set a user definition class (JAR file) in order to use a stub to call a service component.

Note that, when setting a JAR file in the Service Adapter Settings window (Session Bean detailed window), you must set the file extension to `jar` by using lower-case letters.

Also, do not set the following file names for a JAR file:

- `csmsvcadpdef.jar`
- `cscmsg_adpejb.jar`

If the above file names are used, a warning message is displayed. If a warning message appears, delete the above file names from the user definition class. Then, rename the files and then retry.

### (5) Acquiring an RMI-IIOP stub and interface

Use the J2EE server's `cjgetstubsjar` command to acquire an RMI-IIOP stub and interface. For details about the `cjgetstubsjar` command, see *cjgetstubsjar(get RMI-IIOP stub and interface for application)* in the manual *Application Server Command Reference Guide*.

### (6) Operations in the Service Adapter Settings window

To define a Session Bean adapter:

1. Open the Service Adapter Settings window.  
 For details about how to open the Service Adapter Settings window, see *3.3.1(4) Displaying the service adapter settings screen*.
2. In **Service component control information**, edit **Service name**, **Service ID**, and **Maximum instances**, as necessary.
3. In **Service component control information**, specify the value for **Address**.

For **Address**, acquire the service component name that is to be called, from among those registered in the JNDI namespace, and then enter that name. For the names registered in the JNDI name space, see the following locations:

- See *Chapter 2. Naming Management* in the manual *Application Server Common Container Functionality Guide*.
- See *2.5 Registering a reference in the JNDI name space of the EJB container* in the manual *Application Server EJB Container Functionality Guide*.

4. From the **Operation** drop-down list of **Service component control information**, select the operation you want to edit.
5. Check the information displayed in **Operation information**.
6. If necessary, select the **Convert a system exception into a fault message** check box of **Service component control information**.
7. For request messages, perform steps 8 to 18.
8. Perform the following operations:
  - When specifying the standard message format  
Perform steps 9 to 12, and then go to step 13.
  - When not specifying the standard message format  
Go to step 13.
9. Select the **Use** check box for the standard message.
10. Specify the **Format ID** for the standard message.
11. Click the **Browse** button for the standard message, and specify the standard message format for **Message format**.  
Note that if you set a message format that references an external XML schema, be sure to set the file that corresponds to the root schema. The external XML schema file referenced by the root schema is automatically acquired.  
For details about the forms that can be specified for message formats, see *2.6.5 Scoping of XML schema* in the manual *Service Platform Basic Development Guide*.
12. Click the **Display** button for the standard message.  
The format of the standard message is displayed. If necessary, check the format of the specified standard message.
13. Specify **Format ID** for the service component message.
14. Click the **Display** button for the service component message.  
The format of the service component message is displayed. If necessary, check the format of the service component message.
15. Perform the following operations:
  - If you select the **Use** check box for the standard message  
Perform steps 16 to 18, and then go to step 19.
  - If you do not select the **Use** check box for the standard message  
Go to step 19.
16. Enter the file name for the data transformation definition.
17. Click the **Edit** button.  
The Data-conversion definition screen appears.  
Note that, for the first definition, the Select Root Element dialog box appears.  
If you changed the message format, a dialog box appears, confirming whether to apply the change. For details about the procedure when you have changed the message format, see *6.3.2 Procedure for defining changed message formats* in the manual *Service Platform Basic Development Guide*.
18. Map the contents of the standard message and the contents of the service component message.
19. For the response message, also perform steps 8 to 18.
20. Click the **Service-adapter definition (details)** tab.
21. Click the **Add** button to add the relevant user-defined class.

For the user-defined class, specify the JAR file for the stub and interface that you acquired in 3.3.2(5) *Acquiring an RMI-IIOP stub and interface*.

## (7) Setting the client definition file

In the same way as the Web service, set the client definition file that controls operations on the client (service requester) side.

Set the client definition file, as necessary, by using either of the following methods:

- Editing the template file
- Creating a new client definition file, and overwriting the definitions to the template file

Setting the client definition file is optional. If you do not set the client definition file, the contents of the template file are used as is.

### (a) Setting procedure (when editing the template file)

To edit the template file:

1. In the Service Adapter Settings (details) window, under **Client definition file**, click the **Edit** button.  
An editor for the client definition file opens.
2. In the editor, edit the client definition file.
3. From the Eclipse menu, select **File** and then **Save** to save the definitions.

### (b) Setting procedure (when creating a new client definition file and overwriting the template file)

To create a new client definition file and overwrite the template file:

1. Use a text editor or another program to create a client definition file with the name `c4webcl.properties`.  
Enter `c4webcl.properties` in lower-case letters.
2. In the Service Adapter Settings (details) window, under **Client definition file**, click the **Browse** button to specify the created client definition file.

### (c) Settings

For details about the settings in the client definition file, see 10.3 *Setting up the client definition file* in the manual *Application Server SOAP Application Development Guide*.

Note that, for a Session Bean adapter, some items can be set in the client definition file, but other items cannot. The following table lists and describes the items that can be set in the client definition file when a Session Bean adapter is used.

Table 3–2: Setting items in the client definition file

Key name	Description	Specifiable
<code>c4web.logger.log_level</code>	Importance of trace file output	Y
<code>c4web.logger.aplog_level</code>	Importance of application log output	Y
<code>c4web.logger.aplog_error_record</code>	Application log output when an abnormality occurs	Y
<code>c4web.logger.log_file_dir</code>	Output directory of the trace file and application log	Y
<code>c4web.logger.log_file_num</code>	Number of trace files	Y
<code>c4web.logger.log_file_size</code>	Size of trace files	Y
<code>c4web.logger.aplog_file_num</code>	Number of application log files	Y
<code>c4web.logger.aplog_file_size</code>	Size of application log files	Y
<code>c4web.logger.log_file_prefix</code>	Prefix of trace files and application logs	N <sup>#</sup>

### 3. Defining Adapters

Key name	Description	Specifiable
c4web.common.do_multirefs	Multi-reference (runtime option)	N
c4web.common.send_xsi_types	Data type definition (runtime option)	N
c4web.application.app_maintainsession	Retention of HTTP session (runtime option)	N
c4web.application.proxy_host	Host name of the proxy server (runtime option)	N
c4web.application.non_proxy_hosts	Group of host names that do not use the proxy server (runtime option)	N
c4web.application.proxy_port	Port number of the proxy server (runtime option)	N
c4web.application.proxy_user	Authentication user ID of the proxy server (runtime option)	N
c4web.application.proxy_password	Password for the authentication user ID of the proxy server (runtime option)	N
c4web.common.enable_soapheader_check	Option for checking the qualification of SOAP header names (runtime option)	N
c4web.application.socket_write_timeout	Timeout value for writing to the client socket (runtime option)	N
c4web.application.socket_read_timeout	Timeout value for reading from the client socket (runtime option)	N
c4web.application.socket_connect_timeout	Timeout value for connecting to the client socket (runtime option)	N
c4web.attachment.file_body_encoding	Body encoding type	N
c4web.attachment.attachment_temp_directory	Backup directory	N
c4web.attachment.attachment_file_delete_by_date	Number of days to retain unnecessary storage files	N
c4web.common.prf_trace_level	Option for trace output of performance analyses	Y

**Legend:**

Y: The item can be specified.

N: The item cannot be specified. If this item is specified, it is ignored.

#

The value of this item is always changed to the service ID, and therefore need not be specified.

### 3.3.3 Defining Service Adapters (MDB (WS-R))

This subsection explains how to define adapters for MDB (WS-R). You can define adapters for MDB (WS-R) in the Service Adapter Definition screen.

#### (1) Acquiring queue information and message type

When the service component is MDB (WS-R), you must acquire both the queue information and service type of the service component from the developer. You use the acquired information when defining a service adapter.

The following types of information must be acquired:

- Queue name
- Destination address (if network communication between an adapter and a service component is used)
- User name and password for the basic authentication (if network communication between an adapter and a service component is used, and the basic authentication is used)

- Message type

## (2) Service component message format

In the case of MDB (WS-R), you need to create and set up a message format of the service component.

For details about how to create a service component message format, see *4.3.4 Creating a Service Component Message (for MDB (WS-R or DB Queue))* in the manual *Service Platform Basic Development Guide*.

## (3) Data transformation

If the message format of service requester and message format of service component invoked by the service adapter is different, data transformation is required.

For performing data transformation, specify the message entered from the service requester in the standard message using the Service Adapter Definition screen, and then define data transformation in the Data Transformation Definition screen.

For details about data transformation, see *6. Defining Data Transformation* in the manual *Service Platform Basic Development Guide*.

## (4) Specifying invocation

You can specify the method to invoke a service component to be used. You can choose either remote or local invocation. If you choose remote invocation, specify the number of maximum messages. Also, set a basic authentication as required.

## (5) Specifying the maximum message count

If you specify remote invocation as a service component invocation method, you can specify the number of messages that can be processed at the same time, with a number changed from the initial value. You can set a value in a range from 1 to 65535.

## (6) Specifying the basic authentication

If you specify remote invocation as a service component invocation method, and when user authentication is required in MDB (WS-R) message transfer, the basic authentication (user name and password) settings will be needed.

## (7) Operations in the Service Adapter Definition screen

To define an adapter for MDB (WS-R):

1. Open the Service Adapter Definition screen.  
For details about how to open the Service Adapter Definition screen, see *3.3.1(4) Displaying the service adapter settings screen*.
2. Edit Service name, Service ID, and The number of the maximum instances in the HCSC component control information as and when required.
3. Specify an address for the HCSC component control information.  
For **Address**, specify the queue name acquired in *3.3.3(1) Acquiring queue information and message type*.
4. Check the contents displayed in the operation information.
5. Check the **Convert a system exception into a fault message** check box of service component control information as and when required.
6. Perform the following operations for request messages
  - When specifying a message format**  
Execute steps 7 to 10, and then proceed to step 11.
  - When not specifying a message format**  
Proceed to step 11.
7. Choose the Use check box for the message.

8. Specify Format ID for the message.
9. Click the **Browse** button for the message, and specify a message format in Message format.  
Note that for specifying a message format that references an external XML schema, make sure that you specify a file for the root schema. The external XML schema file referenced from the root schema is automatically imported.  
For details about message formats that can be specified, see *2.6.5 Application Scopes of XML Schema* in the manual *Service Platform Basic Development Guide*.
10. Click the Display button for the message.  
The message format is displayed. Check the specified message format as and when required.
11. Specify Format ID for the service component message.
12. Click the **Browse** button for the service component message, and specify a service component message format in Message format.
13. Click the Display button for the service component message.  
The service component message format is displayed. Check the specified service component message format as and when required.
14. Perform the following operations.  
**When the Use check box of the standard message is checked**  
Execute steps 15. to 17. and then proceed to step 18.  
**When the Use check box of the standard message is not checked**  
Proceed to step 18.
15. Enter the file name of the Data transformation definition.
16. Click the **Edit** button.  
The Data Transformation Definition screen is displayed.  
Note that when defining the message format initially, the **Choose root element** dialog box appears.  
If the message format is changed, a dialog box for confirming whether to reflect the change of message format is displayed. For details about the procedure of the changed message format, see *6.3.2 Procedure for defining changed message formats* in the manual *Service Platform Basic Development Guide*.
17. Map the contents of the standard message and the service component message.
18. Click the Service adapter definition (details) tab.
19. From the JMS Message Type drop-down list, choose a JMS message type.
20. Choose the service component invocation method in Invocation Setting.  
**When you choose the Remote Invocation radio button**  
Proceed to step 21.  
**When you choose the Local Invocation radio button**  
Proceed to step 29.
21. In The number of the maximum messages, specify the number of messages the service component can send.
22. In Destination URL, enter the destination address acquired in *3.3.3(1) Acquiring queue information and message type*.
23. Depending on whether the service component requires user authentication, perform either of the following operations:  
**When user authentication is required**  
Execute steps 24 to 26, and then proceed to step 27.  
**When user authentication is not required**  
Proceed to step 27.
24. Choose the **Use** check box for the basic authentication.
25. In the user name and password columns, enter a user name and password necessary for the basic authentication.
26. In the Password (Re-enter) column, enter the same password as that entered in the password column.
27. Enter the transmit queue creation destination RD area as required.

28. Choose the Sequence guarantee check box as and when required.
29. Check the definition contents, and, on the Eclipse File menu, choose Save to save the defined contents.

### 3.3.4 Defining Service Adapters (MDB (DB Queue))

This subsection explains how to define adapters for MDB (DB queue). You can define adapters for MDB (DB queue) in the Service Adapter Definition screen.

#### (1) Acquiring a queue name

When the service component is MDB (DB queue), you must acquire the queue name of the service component from the developer. You use this queue name when defining a service adapter.

#### (2) Service component message format

In the case of MDB (DB queue), you must create and set up a message format for the service component.

For details about how to create a service component message format, see *4.3.4 Creating a Service Component Message (for MDB (WS-R or Database Queue))* in the manual *Service Platform Basic Development Guide*.

#### (3) Data transformation

If the message format of service requester and message format of service component invoked by the service adapter is different, data transformation is required.

For performing data transformation, specify the message entered from the service requester in the standard message using the Service Adapter Definition screen, and then define data transformation in the Data Transformation Definition screen.

For details about data transformation, see *6. Defining Data Transformation* in the manual *Service Platform Basic Development Guide*.

#### (4) Operations in the Service Adapter Definition screen

To define an adapter for MDB (DB queue):

1. Open the Service Adapter Definition screen.  
For details about how to open the Service Adapter Definition screen, see *3.3.1(4) Displaying the service adapter settings screen*.
2. Edit Service name, Service ID, and The number of the maximum instances in the HCSC component control information as and when required.
3. Specify an address for the HCSC component control information.  
For **Address**, enter the registration-destination queue name for using a shared queue. For details about a registration-destination queue name for using a shared queue, see the manual *Cosminexus Application Server Cosminexus Reliable Messaging*.
4. From the Operation drop-down list of the HCSC component control information, choose the operation to be edited.
5. Check the contents displayed in the operation information.
6. Check the **Convert a system exception into a fault message** check box of service component control information as and when required.
7. Perform the following operations for request messages
  - When specifying a message format**  
Perform steps 8. to 11., and then proceed to step 12.
  - When not specifying a message format**  
Proceed to step 12.
8. Choose the Use check box for the message.

9. Specify Format ID for the message.
10. Click the **Browse** button for the message, and specify a message format in Message format.  
Note that for specifying a message format that references an external XML schema, make sure that you specify a file for the root schema. The external XML schema file referenced from the root schema is automatically imported.  
For details about message formats that can be specified, see *2.6.5 Scoping of XML schema* in the manual *Service Platform Basic Development Guide*.
11. Click the Display button for the message.  
The message format is displayed. Check the specified message format as and when required.
12. Specify Format ID for the service component message.
13. Click the **Browse** button for the service component message, and specify a service component message format in Message format.
14. Click the **Display** button for the service component message.  
The service component message format is displayed. Check the specified service component message format as and when required.
15. Perform the following operations.
  - When the Use check box of the standard message is checked**  
Execute steps 16. to 18. and then proceed to 19.
  - When the Use check box of the standard message is not checked**  
Proceed to step 19.
16. Enter the file name of the Data transformation definition.
17. Click the **Edit** button.  
The Data Transformation Definition screen is displayed.  
Note that when defining the message format initially, the **Choose root element** dialog box appears.  
If the message format is changed, a dialog box for confirming whether to reflect the change of message format is displayed. For details about the procedure of the changed message format, see *6.3.2 Procedure for defining changed message formats* in the manual *Service Platform Basic Development Guide*.
18. Map the contents of the standard message and the service component message.
19. Check the definition contents, and, on the Eclipse File menu, choose Save to save the defined contents.

## 3.3.5 Defining Database Adapters

This subsection describes how to define DB adapters.

### (1) Creating the SQL operation definition file

The SQL operation definition file defines information about the database accessed via the DB adapter in the execution environment and the SQL execution format. This file is used as the base when creating an XML format definition file for a DB adapter.

The following describes how to create the SQL operation definition file.

Create the SQL operation definition file in the XML document format by using a program such as a text editor. If you use any characters that cannot be used for XML documents (such as <, >, or &), you need to escape such characters or use the CDATA section.

Specify elements, attributes, and text data that conform to the W3C XML specifications.

You cannot use the XML name space.

The following shows details of the XML document to be created here.

- File name

Specify the name of the SQL operation definition file in the following format:

---

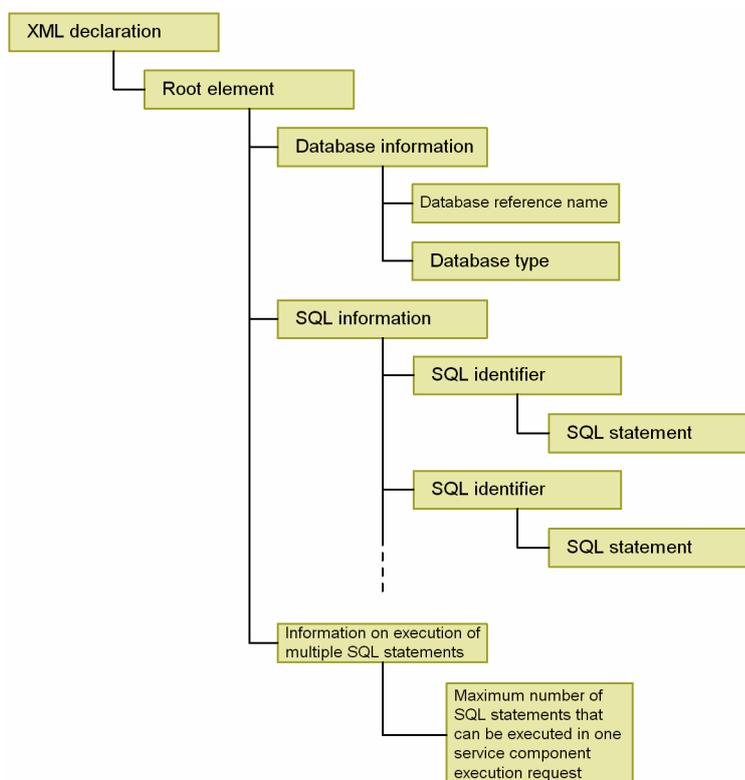
```
csa_sql_any-name.xml
```

---

- Structure

The following figure shows the structure of the SQL operation definition file.

Figure 3–2: Structure of the SQL operation definition file



- Format of the SQL operation definition file

The format of the SQL operation definition file is shown below. The italic characters indicate variable values.

---

```

<?xml version="XML-version" encoding="character-encoding"?>
<DBadapter_SQL_OPERATION dba_separate_transaction="transaction-separate-option">
  <DATABASE_DATA>
    <DB_NAME dynamic="connection-target-dynamic-change-option">database-reference-name</
DB_NAME>
    <DB_TYPE>database-type</DB_TYPE>
  </DATABASE_DATA>
  <SQL_DATA encoding="encoding-and-decoding-format-of-character-binary-data">
    <SQL-identifier out_maxOccurs="maximum-number-of-output-search-results">
      SQL-instruction <argument-element-name dba_inf="argument-type" data_type="data-
format" />
    </SQL-identifier>
    :
  </SQL_DATA>
  <MULTI_SQL_DATA>
    <MAX_SQL_NO>maximum-number-of-SQL-statements-that-are-executed-by-one-service-
component-request</MAX_SQL_NO>
  </MULTI_SQL_DATA>
</DBadapter_SQL_OPERATION>
  
```

---

The following shows the definitions in an SQL operation definition file.

#### XML declaration

Specify an XML version that conforms to the XML language specifications. Specify UTF-8 for the character encoding. You can omit specifying the character encoding. If you omit specifying the character encoding, UTF-8 or UTF-16 is used.

---

```

<?xml version="XML-version" encoding="character-encoding"?>
  
```

---

#### Root element (element name: DBadapter\_SQL\_OPERATION)

The root element of the SQL operation definition file.

---

```
<DBadapter_SQL_OPERATION dba_separate_transaction="Y|N">
  :
</DBadapter_SQL_OPERATION>
```

---

- `dba_separate_transaction`

Specify whether to start the transaction for the DB adapter separately from the business process. Specify `Y` if you want to separate transactions, or `N` if you do not want to separate transactions. If you omit this specification, the value of `dba-separate-transaction` in the HCSC server runtime definition file will be valid. The value specified for this attribute takes preference over the value of `dba-separate-transaction` in the HCSC server runtime definition file.

Before specifying `Y` for this attribute, change the value for the `trans-attribute` element in the HITACHI Application Integrated Property File of the DB adapter from `Required` to `RequiresNew`.

Database information (element name: `DATABASE_DATA`)

Specify the database information in the lower elements (`DB_NAME` and `DB_TYPE`).

---

```
<DATABASE_DATA>
  database-reference-name
  database-type
</DATABASE_DATA>
```

---

Database reference name (element name: `DB_NAME`)

Specify the database reference name of DB Connector.

Specify the same value as for the `res-ref-name` element in the HITACHI Application Integrated Property File in order to search for the resource reference in that file when the DB adapter is executed, based on the value specified here.

If `Y` is set for the `dynamic` attribute, you can omit specifying the value for this element, and the value already set for this element is ignored.

---

```
<DB_NAME dynamic="Y|N">database-reference-name</DB_NAME>
```

---

- `dynamic`

Specify whether to dynamically change the connection target.

Specify `Y` if you want to dynamically change the connection target, or `N` if you do not want to dynamically change the connection target. If you omit specifying the value, `N` is assumed.

Database type (element name: `DB_TYPE`)

Specify one of the following as the database type:

- `HIRDB`

Specify this option if you want to use HiRDB as the DBMS (if `DBConnector_DABJ_CP/XA.rar` is used as the DB Connector).

- `HIRDB-TYPE4`

Specify this option if you want to use HiRDB as the DBMS (if `DBConnector_HiRDB_Type4_CP/XA.rar` is used as the DB Connector).

- `ORACLE`

Specify this option if you want to use Oracle as the DBMS (if `DBConnector_DABJ_CP/XA.rar` is used as the DB Connector).

- `ORACLE-THIN`

Specify this option if you want to use Oracle as the DBMS (if `DBConnector_Oracle_CP/XA.rar` is used as the DB Connector).

---

```
<DB_TYPE>database-type</DB_TYPE>
```

---

SQL information (element name: `SQL_DATA`)

In the lower elements (`SQL-identifier` and `SQL-statement`), specify the SQL information to be executed.

---

```
<SQL_DATA encoding="hexBinary|base64Binary">
  SQL-identifier
  SQL-statement
</SQL_DATA>
```

---

- encoding

Specify the encoding and decoding format of binary data.

Specify `hexBinary` to encode or decode in hexBinary format, or `base64Binary` to encode or decode in base64Binary format. If you omit specifying this value, `hexBinary` is assumed.

This attribute is valid only when the following data formats are specified for the `data_type` attribute:

- For HiRDB: LONGVARBINARY
- For Oracle: VARBINARY, LONGVARBINARY, BLOB

#### ! Important note

When you perform data transformation for binary data, specify the same value for the `encoding` attribute in the SQL operation definition file and the encoding or decoding format in the binary format definition file. If different formats are specified, correct operations are not guaranteed.

#### Reference note

To change the encoding or decoding format to `base64binary` when the DB adapter is upgraded in an environment where a version earlier than 09-50 is used:

1. Re-create the binary format definition file in the development environment.
2. Modify the SQL operation definition file (set the encoding or decoding format).
3. Execute the `csamkxmls` command to create the XML schema by using the modified SQL operation definition file.
4. Set the XML schema and the binary format definition file to the DB adapter in the development environment.
5. Set the mapping definition again in the development environment.
6. Set the SQL operation definition file modified in the development environment to the DB adapter.
7. Deploy the DB adapter again.

SQL identifier (element name: any (1 to 256 characters))

This element identifies the SQL to be executed. If you omit specifying this element, the SQL statement is not executed. In the lower argument element, specify the SQL statement. You can specify `out_maxOccurs` as the attribute. You can also specify the SQL identifier multiple times (the number is equal to the number of SQL instructions to be executed).

You cannot specify `DBAdapter` or `DBA_MULTI_SQL` as the name of the SQL identifier.

```
<SQL-identifier out_maxOccurs="maximum-number-of-output-search-results">
  SQL-statement
</SQL-identifier>
:
```

- out\_maxOccurs

Specify the maximum number of output search results. If you omit specifying this value, 1000 is set. If you specify 0, 2147483647 is set.

SQL statement

Specify the SQL statement. The syntax of the SQL statement must conform to the specifications of the database to be used. Specify the variable parts (argument elements), such as the table and column names and data in the SQL statement.

You do not have to add a semicolon (;) to indicate the end of the SQL statement.

```
SQL-instruction <argument-element-name dba_inf="argument-type" data_type="data-format"/>
```

For the beginning of the SQL statement, specify `SELECT` **Δ** or `INSERT` **Δ** (**Δ** indicates a single-byte space). If an argument element follows immediately after `SELECT` **Δ** or `INSERT` **Δ**, you can omit **Δ** (single-byte space).

The following describes the specified information.

SQL instruction

You can specify `SELECT` or `INSERT` as the SQL instruction.

The following table lists and describes the data types that can be searched for by using `SELECT`.

Table 3–3: Data types that can be searched for by using SELECT (for HiRDB)

HiRDB data type	Searchable
	For HiRDB Type4 JDBC Driver
INT [EGER]	Y
SMALLINT	Y
[LARGE] DEC [IMAL] NUMERIC	Y
FLOAT DOUBLE PRECISION	Y
SMALLFLT REAL	Y
CHAR [ACTER]	Y
VARCHAR CHAR [ACTER] VARYING	Y
NCHAR NATIONAL CHAR	Y
NVARCHAR NATIONAL CHAR [ACTER] VARYING NCHAR VARYING	Y
MCHAR	Y
MVARCHAR	Y
DATE	Y
TIME	Y
TIMESTAMP	Y
INTERVAL YEAR TO DAY	--
INTERVAL HOUR TO SECOND	--
BLOB BINARY LARGE OBJECT	Y
BINARY	Y
BOOLEAN	--

Legend:

Y: Can be searched for by using SELECT.

--: Cannot be searched for by using SELECT.

#

You can omit the parts enclosed in square brackets ([ ]).

Table 3–4: Data types that can be searched for by using SELECT (for Oracle)

Oracle data type	Searchable
	For Oracle JDBC Thin Driver
VARCHAR2	Y
NVARCHAR2	Y

Oracle data type	Searchable
	For Oracle JDBC Thin Driver
NUMBER	Y
LONG	Y
DATE	Y#1
BINARY_FLOAT#2	--
BINARY_DOUBLE#2	--
TIMESTAMP	--
TIMESTAMP WITH TIME ZONE	--
TIMESTAMP WITH LOCAL TIME ZONE	--
INTERVAL YEAR TO MONTH	--
INTERVAL DAY TO SECOND	--
RAW	Y
LONG RAW	Y
ROWID	Y
UROWID	--
CHAR	Y
NCHAR	Y
CLOB	Y
NCLOB	Y
BLOB	Y
BFILE	--
User-defined type (object type)	--
User-defined type (REF data type)	--
User-defined type (VARRAY)	--
User-defined type (nested table)	--

## Legend:

Y: Can be searched for by using SELECT.

--: Cannot be searched for by using SELECT.

## #1

When you use Oracle JDBC Thin Driver of Oracle 11g or Oracle12c, you need to set the following value as the JVM startup parameter in the definition of the logical J2EE server:

```
-Doracle.jdbc.mapDateToTimestamp=false
```

For details about how to set the startup parameter, see the *Application Server Management Portal Operation Guide*.

## #2

This data type can be used only when the connection-target database is Oracle 11g.

- Point

You can use a function provided by the database to convert an unsearchable data type to one that can be searched. In the following example, the `TO_CHAR` function is used to convert the `TIMESTAMP` type (column name: `c_ts`) of Oracle to the `VARCHAR2` type for searching.

```
SELECT TO_CHAR(c_ts) FROM table-name
```

## Argument element name

Specify an argument element name. If the same name is specified for multiple argument elements, the arguments are treated as the same argument. If two or more argument elements have the same name, but have different values for the `dba_inf` and `data_type` attributes, an error occurs.

`dba_inf="argument-type"`

Specify the argument type.

The following describes what can be specified for the argument types.

- `table`  
Specify this when the argument is the name of a table that exists in the database. String-type data is used for the argument.
- `column`  
Specify this when the argument is the name of a column that exists in the database. String-type data is used for the argument.
- `data`  
Specify this when the argument is data other than the name of a table or column. If you specify this value, specify for the `data_type` attribute the type of data to be used for the argument. Also, convert the data to be used for the argument to the Java data type or class shown in Table 3-3 and Table 3-4, and then add it to the SQL statement. If the data to be used for the argument cannot be converted, an error occurs.  
Note that, when you specify this value, the argument element is treated as the `IN` parameter (`?` parameter or bind variable). In the SQL statement, you can specify the `IN` parameter at the location defined in the specifications of the database to be used. If you define the `IN` parameter at a location that is not allowed by the specifications, an error occurs. The argument element is replaced with `?` and then executed when SQL is executed.
- `preset`  
Specify this when the argument is data other than the name of a table or column. When you specify this value, insert the data to be used for the argument into the SQL statement as is. String-type data is used for the argument.

`data_type="data-type"`

When you specify `data` for the `dba_inf` attribute, specify the type of data to be used for the argument. The following table shows the correspondence between the data types of the database and the data types that can be specified for `data_type`.

Table 3–5: Data types that can be specified for `data_type` (for HiRDB)

HiRDB data type	Data type that can be specified for <code>data_type</code>	Data type and class that the data to be used for the argument is converted to
INT [EGER]	INTEGER	int
SMALLINT	SMALLINT	short
[LARGE] DEC [IMAL] NUMERIC	DECIMAL	java.math.BigDecimal
FLOAT DOUBLE PRECISION	FLOAT	double
SMALLFLT REAL	REAL	float
CHAR [ACTER] NCHAR NATIONAL CHAR MCHAR	CHAR	java.lang.String
VARCHAR CHAR [ACTER] VARYING	VARCHAR	java.lang.String

HiRDB data type	Data type that can be specified for data_type	Data type and class that the data to be used for the argument is converted to
NVARCHAR NATIONAL CHAR [ACTER] VARYING NCHAR VARYING	VARCHAR	java.lang.String
MVARCHAR		
DATE	DATE	java.sql.Date
TIME	TIME	java.sql.Time
TIMESTAMP	TIMESTAMP	java.sql.Timestamp
BLOB BINARY LARGE OBJECT	LONGVARBINARY	byte[]
BINARY	LONGVARBINARY	byte[]

Note:

You can omit the parts enclosed in square brackets ([ ]).

Table 3–6: Data types that can be specified for data\_type (for Oracle)

Oracle data type	Data type that can be specified for data_type	Data type and class that the data to be used for the argument is converted to
LONG	LONGVARCHAR	java.io.Reader
NUMBER	NUMERIC	java.math.BigDecimal
VARCHAR2	VARCHAR	java.lang.String
NVARCHAR2	--	
ROWID	CHAR	
CHAR	CHAR	
NCHAR	--	
CLOB	CLOB <sup>#1</sup>	
NCLOB	--	
DATE	DATE <sup>#2</sup>	java.sql.Date
	TIME <sup>#3</sup>	java.sql.Time
	TIMESTAMP <sup>#4</sup>	java.sql.Timestamp
RAW	VARBINARY	byte[]
LONG RAW	LONGVARBINARY	byte[]
BLOB	BLOB	byte[]

Legend:

--: This data type cannot be specified.

#1

This data type can be specified only when the JDBC driver is version 10.2. If a JDBC driver whose version is other than 10.2 is used, the data might be corrupted.

#2

Specify this data type when the format of the setting value is *yyyy-mm-dd*.

#3

Specify this data type when the format of the setting value is *hh.mm.ss*.

### 3. Defining Adapters

#4

Specify this data type when the format of the setting value is *yyyy-mm-dd hh.mm.ss*.

Execution information of multiple SQL statements (element name: MULTI\_SQL\_DATA)

When executing multiple SQL statements in response to one service component execution request from a service requester, specify the maximum number of SQL statements to be executed in the lower element (MAX\_SQL\_NO). You can omit specifying this element if you are executing a single SQL statement.

---

```
<MULTI_SQL_DATA>
  maximum-number-of-SQL-statements-to-be-executed
</MULTI_SQL_DATA>
```

---

Maximum number of SQL statements to be executed (element name: MAX\_SQL\_NO)

Specify the maximum number of SQL statements to be executed (maximum number of DBA\_MULTI\_SQL elements in the request message from the service requester) in the range 1 to 2,147,483,647. If you omit specifying the value, 1024 is set.

---

```
<MAX_SQL_NO>maximum-number-of-SQL-statements-that-are-executed-by-one-service-component-
execution-request</MAX_SQL_NO>
```

---

- Example of creating the SQL operation definition file

The following is an example of creating the SQL operation definition file.

Example conditions

Database reference name: DB\_SERVER1

Schema name: DBA

Table name: ORDER\_TABLE

Table structure:

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

SQL format to be executed:

---

```
OPERATION1:SELECT * FROM DBA.ORDER TABLE WHERE <val1> <val2> <val3>
OPERATION2:SELECT <val1> SUM(<val2>) FROM DBA.ORDER_TABLE GROUP BY <val3>
OPERATION3:INSERT INTO DBA.ORDER_TABLE VALUES(<val1>,<val2>,<val3>,<val4>)
```

---

Example of creating the SQL operation definition file (when a single SQL statement is executed)

---

```
<?xml version="1.0" encoding="UTF-8" ?>
<DBadapter SQL OPERATION>
  <DATABASE_DATA>
    <DB_NAME>DB_SERVER1</DB_NAME>
    <DB_TYPE>HIERDB</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>
    <OPERATION2>
      SELECT <val1 dba_inf="column"/> SUM(<val2 dba_inf="column"/>)
        FROM DBA.ORDER TABLE GROUP BY <val3 dba_inf="column"/>
    </OPERATION2>
    <OPERATION3>
      INSERT INTO DBA.ORDER TABLE
        VALUES(<val1 dba_inf="data" data_type="INTEGER" />,
          <val2 dba_inf="data" data_type="CHAR"/>,
          <val3 dba_inf="data" data_type="CHAR"/>,
```

---

---

```

        <val4 dba_inf="data" data_type="INTEGER"/>
    </OPERATION3>
</SQL_DATA>
</DBAdapter_SQL_OPERATION>

```

---

Example of creating the SQL operation definition file (when multiple SQL statements are executed)

In the following example, the maximum number of SQL statements to be executed in response to one service component execution request is set to 10.

---

```

<?xml version="1.0" encoding="UTF-8" ?>
<DBAdapter_SQL_OPERATION>
  <DATABASE_DATA>
    <DB_NAME>DB_SERVER1</DB_NAME>
    <DB_TYPE>HIERDB</DB_TYPE>
  </DATABASE_DATA>
  <SQL_DATA>
    <OPERATION1>
      SELECT * FROM DBA.ORDER_TABLE WHERE <val1 dba_inf="column"/>
        <val2 dba_inf="presët"/>
        <val3 dba_inf="data" data_type="CHAR"/>
    </OPERATION1>
    <OPERATION2>
      SELECT <val1 dba_inf="column"/> SUM(<val2 dba_inf="column"/>)
        FROM DBA.ORDER_TABLE GROUP BY <val3 dba_inf="column"/>
    </OPERATION2>
    <OPERATION3>
      INSERT INTO DBA.ORDER_TABLE
        VALUES(<val1 dba_inf="data" data_type="INTEGER" />,
          <val2 dba_inf="data" data_type="CHAR"/>,
          <val3 dba_inf="data" data_type="CHAR"/>,
          <val4 dba_inf="data" data_type="INTEGER"/>)
    </OPERATION3>
  </SQL_DATA>
  <MULTI_SQL_DATA>
    <MAX_SQL_NO>10</MAX_SQL_NO>
  </MULTI_SQL_DATA>
</DBAdapter_SQL_OPERATION>

```

---

## (2) Creating a message format

WSDL is not generated for a DB adapter, so the message format of a service component is also not generated automatically. Therefore, you need to create the message format of the service component (XML format definition file for the DB adapter), and then specify the settings.

### (a) Creating the XML format definition file for a DB adapter

To create the XML format definition file for a DB adapter, specify the name of the SQL operation definition file created in (1) *Creating the SQL operation definition file* for the `-o` option, and then execute the `csamkxmls` command.

- Example of executing the command

The following shows the execution format of the `csamkxmls` command:

---

```

installation-directory-of-the-service-platform\CSC\bin\csamkxmls -o name-of-the-SQL-
operation-definition-file -x name-of-the-XML-format-definition-file-for-DB-adapter

```

---

For details about the `csamkxmls` command, see *csamkxmls(Creating XML Format Definition File for Database Adapter)* in the manual *Service Platform Reference Guide*.

#### ! Important note

When the `csamkxmls` command is executed, if the SQL operation definition file is too large, the `java.lang.OutOfMemoryError` message might be displayed.

If this error is displayed, for the `HCSCDBA_XMX` environment variable, set a value larger than the size currently allocated to the JVM memory, and then re-execute the command.

For the `HCSCDBA_XMX` environment variable, specify the value (unit: Mbyte) set for the JVM heap size (`-Xmx`). If you omit specifying this value, the JVM heap size depends on the environment in which the Java command is executed.

Setting example

```
set HCSCDBA_XMX=64
```

---

**! Important note**

The XML format definition file for DB adapter contains as many formats as the number of SQL identifiers specified in the SQL operation definition file. If you execute a single SQL statement, you need to change the operation defined for each SQL identifier when defining data transformation (by adding operations in the Service Adapter Settings window). If you execute multiple SQL statements, you do not have to change the operation for each SQL identifier when defining data transformation.

For details about defining data transformation, see *Chapter 6. Defining Data Transformation* in the manual *Service Platform Basic Development Guide*. For details about the Service Adapter Settings window, see *1.2.2 Service Adapter Definition Window* in the manual *Service Platform Reference Guide*.

(b) Example of creating an XML format definition file for a DB adapter

The following is an example of creating an XML format definition file for a DB adapter.

Figure 3–3: (Example 1) When using the SELECT statement to use a value in the WHERE clause as an argument

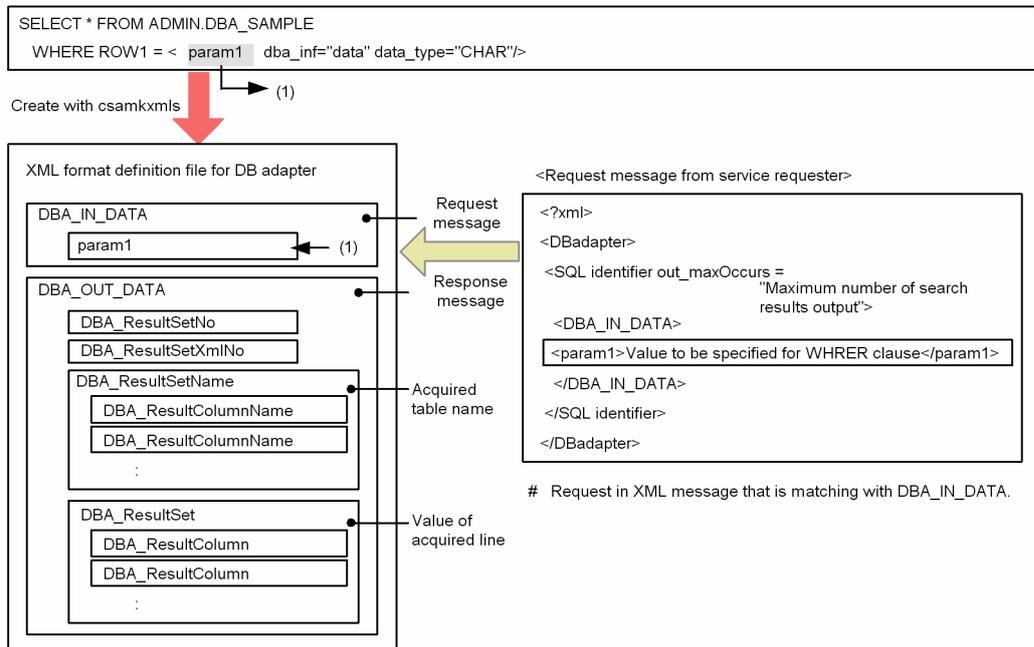


Figure 3-4: (Example 2) When using the SELECT statement to use the column names to be acquired, as arguments

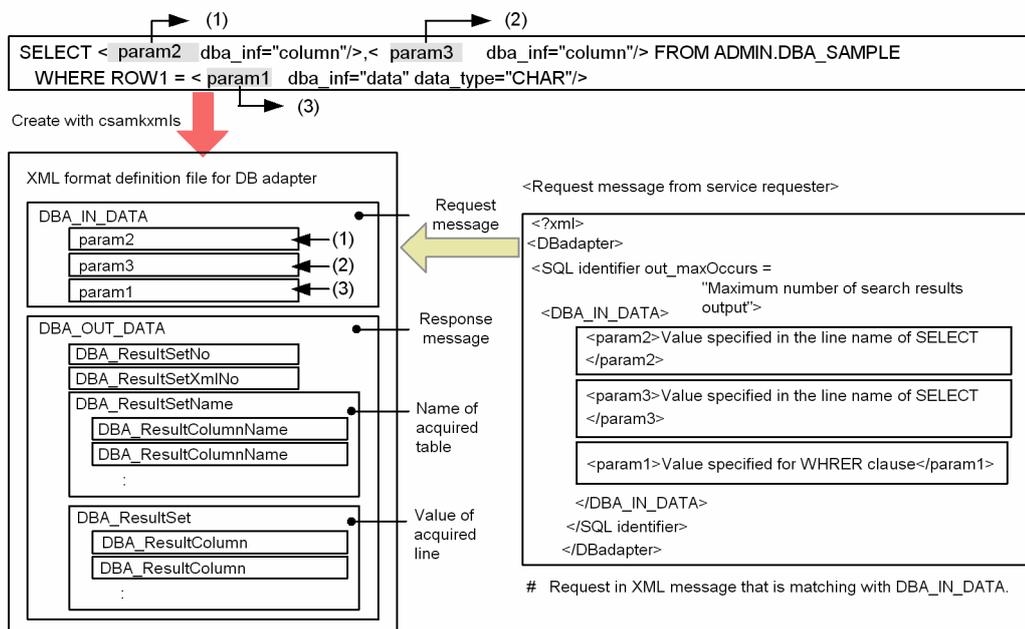
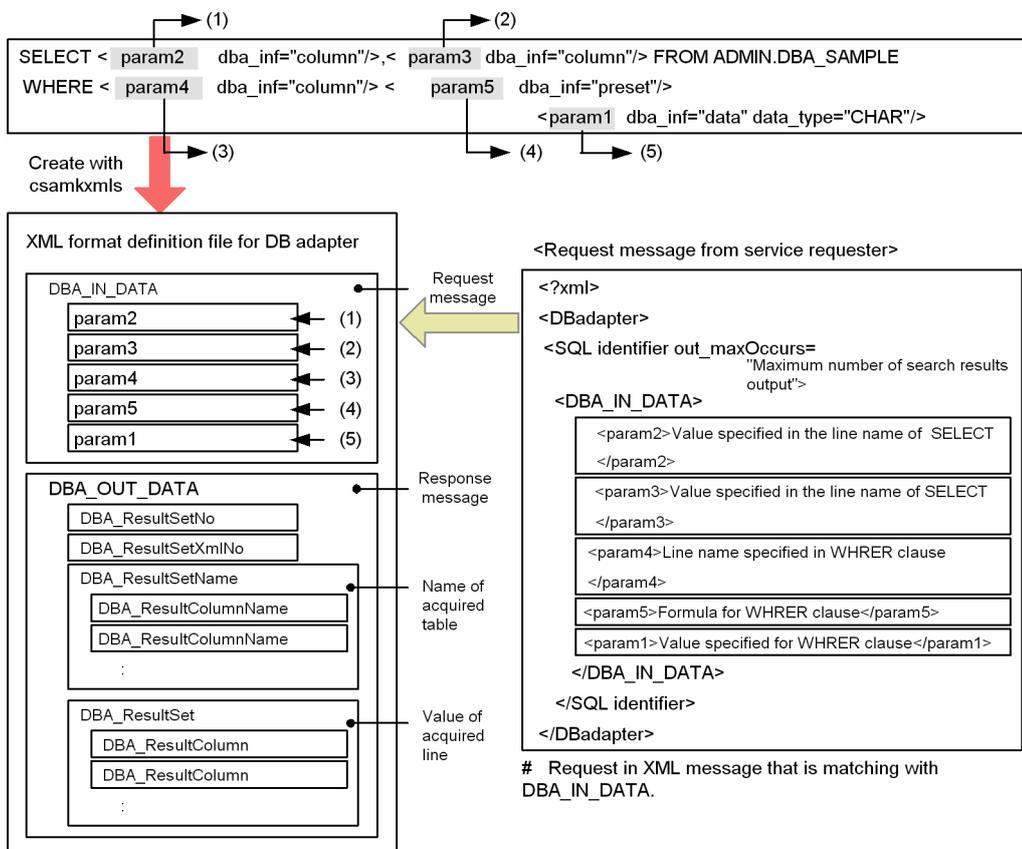


Figure 3-5: (Example 3) When using the SELECT statement to use the column names and expressions in the WHERE clause as arguments



### (3) Operations in the Service Adapter Settings window

To define a DB adapter:

1. Open the Service Adapter Settings window.  
For details about how to open the Service Adapter Settings window, see 3.3.1(4) *Displaying the service adapter settings screen*.
2. In **Service component control information**, edit **Service name** and **Service ID**, as necessary.  
Enter the value for **Service name** when adding the DB adapter if you want to change the entered name.  
The service ID is the identification information that is required when the administrator performs an operation on the HCSC server. We recommend that you specify a value that is related to the service name.
3. In **Service component control information**, click the **Add** button to add an operation.  
The operation name is the identification information that is required when a service requester calls the HCSC server.
4. If necessary, select the **Convert a system exception into a fault message** check box in **Service component control information**.
5. From the **Communication model** drop-down list of the operation information, select **Sync** or **Async**.  
If you need a response, select **Sync**.
6. For the request message, perform steps 7 to 18.
7. Perform the following operations:  
When specifying the standard message format  
Perform steps 8 to 11, and then go to step 12.  
When not specifying the standard message format  
Go to step 12.
8. Select the **Use** check box for the standard message.
9. Specify the **Format ID** for the standard message.  
Specify any value for **Format ID**.
10. Click the **Browse** button for the standard message, and specify the standard message format for **Message format**.  
For details about the forms that can be specified for message formats, see 2.6.5 *Scoping of XML schema* in the manual *Service Platform Basic Development Guide*. Note that you cannot specify a message format that refers to an external XML schema.
11. Click the **Display** button for the standard message.  
The format of the standard message is displayed. If necessary, check the format of the specified standard message.
12. Specify the **Format ID** for the service component message.  
Specify any value for **Format ID**.
13. Click the **Browse** button for the service component message, and specify the service component message format for **Message format**.  
Here, specify the XML format definition file you created for the DB adapter.  
You can specify only the XML message for **Message format**.
14. Click the **Display** button for the service component message.  
The format of the service component message is displayed. If necessary, check the format of the specified service component message.
15. If necessary, define a data transformation.  
Perform the following operations:  
When selecting the **Use** check box for the standard message  
Perform steps 16 to 18, and then go to step 19.  
When not selecting the **Use** check box for the standard message  
Go to step 19.
16. Enter the file name for the data transformation definition.
17. Click the **Edit** button.

The Data-conversion definition screen appears.

Note that, for the first definition, the Select Root Element dialog box appears.

If you changed the message format, a dialog box appears confirming whether to apply the change on the message format. For details about the procedure when you have changed the message format, see *6.3.2 Procedure for defining changed message formats* in the manual *Service Platform Basic Development Guide*.

18. Map the contents of the standard message and the contents of the service component message.

19. If, you selected **Sync** for **Communication model** in step 5, also perform steps 7 to 18 for the response message.

20. Click the **Service-adapter definition (details)** tab.

21. Check the name of the service adapter (EJB-JAR file) and self-defined file.

Ensure that the name of the service adapter (EJB-JAR file) is `cscdba_ejb.jar`, and the name of the self-defined file is `cscadapter_property.xml`.

You do not have to set the utility class (JAR file).

22. Click the **Add** button for the self-defined file, and add the following file:

- SQL operation definition file  
[`csa_sql_any-name.xml`]

#### (4) Editing the HITACHI Application Integrated Property File

Use the template file provided by the service platform to edit the HITACHI Application Integrated Property File.

To edit the HITACHI Application Integrated Property File:

1. In the Service Adapter Settings (details) window, set `cscadapter_property.xml` for **Self-defined file**, and then click the **Edit** button.

An editor for the HITACHI Application Integrated Property File opens.

2. In the editor, edit the HITACHI Application Integrated Property File.

For details about settings properties by using the HITACHI Application Integrated Property File, see *9.2 Property settings using the HITACHI Application Integrated Property File* in the manual *Application Server Application Setup Guide*. For details about the HITACHI Application Integrated Property File, see *3.1 HITACHI Application Integrated Property file* in the manual *Application Server Application and Resource Definition Reference Guide*.

The following table lists and describes the settings in the HITACHI Application Integrated Property File.

Table 3–7: Settings in the HITACHI Application Integrated Property File

Tag	Description	Setting	Can be changed	Setting example
<code>&lt;hitachi-application-all-property&gt;</code>	Root tag	--	No	--
<code>&lt;ejb-jar&gt;</code>	Opening tag for the definition of information regarding EJB	--	No	--
<code>&lt;hitachi-session-bean-property&gt;</code>	Opening tag for the definition of the Session Bean attribute	--	No	--
<code>&lt;resource-ref&gt;</code>	Opening tag for the resource reference <sup>#1</sup> definition	--	No	--
<code>&lt;res-ref-name&gt;</code>	Resource reference name	Database reference name specified for the <code>&lt;DB_NAME&gt;</code> tag in the SQL operation definition file	Yes	DB_SERVER1

### 3. Defining Adapters

Tag	Description	Setting	Can be changed	Setting example
<res-sharing-scope>	Specification of whether to share the referenced resource	Specification of whether to share the DB Connector referenced by the DB adapter	Yes	Shareable
<linked-to>	Data source display name	Name of the resource adapter specified for the property definition ( <i>display-name</i> ) of the DB Connector	Yes	For example, DB_Connector_for_HiRDB_Type4_XA or DB_Connector_for_Oracle_XA#2.  If you changed <i>display-name</i> , set the new name.
</resource-ref>	Closing tag for the resource reference definition	--	No	--
<container-transaction>	Opening tag for the definition regarding container transaction	--	No	--
<trans-attribute>	Transaction attribute assigned to the method	Required (inherit the transaction)	Yes	Required <sup>#3</sup>
</container-transaction>	Closing tag for the definition regarding container transaction	--	No	--
<session-runtime>	Opening tag for the definition of runtime	--	No	--
<stateless>	Opening tag for the definition of stateless	--	No	--
<pooled-instance>	Opening tag for the definition regarding instances in the pool	--	No	--
<minimum>	Minimum number of instances in the pool	Minimum number of instances in the DB adapter	Yes	$0 \text{ or } 1 \leq \text{minimum} \leq \text{maximum}$
<maximum>	Maximum number of instances in the pool	Maximum number of instances in the DB adapter	Yes	$0 \text{ (infinite) or } 1 \leq \text{maximum} \leq \text{maximum-sessions}$
</pooled-instance>	Closing tag for the definition regarding instances in the pool	--	No	--
</stateless>	Closing tag for the definition of stateless	--	No	--
</session-runtime>	Closing tag for the definition of runtime	--	No	--

Tag	Description	Setting	Can be changed	Setting example
</hitachi-session-bean-property>	Closing tag for the definition of the Session Bean attribute	--	No	--
</ejb-jar>	Closing tag for the definition of information regarding EJB	--	No	--
</hitachi-application-all-property>	Root tag	--	No	--

## Legend:

--: The setting for this tag cannot be modified.

## #1

If Y is set for the `dynamic` attribute in the SQL operation definition file, omit the definitions of this tag and its child elements.

## #2

If you changed `display-name`, specify the new name.

## #3

To start a transaction for the DB adapter separately from the business process, set `true` for the `dba-separate-transaction` property in the HCSC server runtime definition file, or Y for the `dba_separate_transaction` attribute in the SQL operation definition file. In addition, set `RequiresNew` for the `<trans-attribute>` tag in the HITACHI Application Integrated Property File.

 Important note

Do not set or change items other than those that can be set or changed, as shown in Table 3-7.

3. From the Eclipse menu, select **File** and then **Save** to save the definitions.

## (5) Data transformation

To perform data transformation of a request message, in the Service Adapter Settings window, set the message to be input from the service requester as the standard message, and in the Data-conversion definition screen, define data transformation. To perform data transformation of a response message, set the standard message for the message returned from the service component, and in the Data-conversion definition screen, define data transformation.

### (a) Cases when data transformation is required

If the message format of the service requester is different from the message format of the DB adapter message, data transformation is required.

In the following cases, specify the standard message format to define data transformation:

- When you want to call the DB adapter by using a message format different from the message format specified for the DB adapter (XML format definition file).
- When you want to receive a response from the DB adapter by using a message format different from the message format specified for the DB adapter (XML format definition file).

### (b) Defining data transformation

- When performing data transformation for the request message

In the Service Adapter Settings window, set the message input from the service requester as the standard message, and then in the Data-conversion definition screen, define data transformation.

Map the request message to the `DBA_IN_DATA` that has the message format specified for the DB adapter (XML format definition file).

- When performing data transformation for the response message  
Set the standard message for the message returned from the service component, and then in the Data-conversion definition screen, define data transformation.  
Map the response message to the DBA\_OUT\_DATA that has the message format specified for the DB adapter (XML format definition file).

For details about data transformation, see *Chapter 6. Defining Data Transformation* in the manual *Service Platform Basic Development Guide*.

## (6) Replacing the EAR file

Set the EAR file when adding a DB adapter. If you need to modify the EAR file after adding a DB adapter, in the Service Adapter Settings window of the DB adapter, replace the EAR file.

To replace the EAR file in the Service Adapter Settings window of the DB adapter:

1. On the Service Definition List in the tree view, double-click the DB adapter for which you replace the EAR file.  
The Service Adapter Settings window appears.
2. Click the **Service-adapter definition (details)** tab at the bottom of the window.  
The Service Adapter Settings (details) window appears.
3. In the Service Adapter Definition (details) window, click the **Browse** button.
4. Specify the EAR file to be replaced.

## (7) Creating a service requester

How to create a service requester (that issues requests to the DB adapter) depends on the type of standard reception that receives request messages. Create a service requester with reference to the creation method for each type of standard reception. The following table describes the reference destinations.

Table 3–8: How to create a service requester (which issues requests to the DB adapter)

Standard reception type		See:
Synchronous reception	Web Service (SOAP1.1)	See 8.2 <i>Service Requester That Sends Requests to a Standard Synchronous Reception (Web Services) (SOAP communication infrastructure)</i> in the manual <i>Service Platform Basic Development Guide</i> .
	Web Service (SOAP1.2)	See 8.3 <i>Creating a service requester using standard synchronous reception (Web Services) (JAX-WS engine)</i> in the manual <i>Service Platform Basic Development Guide</i> .
	Session Bean	See 8.4 <i>Service Requester That Sends Requests to a Standard Synchronous Reception (SessionBean)(JAX-WS engine)</i> in the manual <i>Service Platform Basic Development Guide</i> .
Asynchronous reception	MDB (WS-R)	See 8.5 <i>Service Requester That Sends Requests to a Standard Asynchronous Reception (MDB (WS-R))</i> in the manual <i>Service Platform Basic Development Guide</i> .
	MDB (DB queue)	See 8.6 <i>Service Requester That Sends Requests to a Standard Asynchronous Reception (MDB (database queue))</i> in the manual <i>Service Platform Basic Development Guide</i> .

The following subsections describe the DB adapter message formats.

If the message format of a service requester for a DB adapter is different from the message format of a DB adapter message, data transformation is required. For details about data transformation, see *Chapter 6. Defining Data Transformation* in the manual *Service Platform Basic Development Guide*.

### (a) Request message format

Create a request message in the XML document format.

Specify the elements, attributes, and text data, by conforming to W3C XML specifications. If you use a character that cannot be used for XML documents (such as <, >, or &), you need to escape such characters or use the CDATA section.

The request message format for when requesting execution of a single SQL statement is different from the request message format for when requesting execution of multiple SQL statements.

The request message format is shown below. *Italic characters indicate variable values.*

When requesting execution of a single SQL statement

Enter the *SQL-identifier* element as a lower element of the root element (DBadapter element). You can enter only one *SQL-identifier* element.

---

```
<DBadapter>
  <SQL-identifier out_maxOccurs="maximum-number-of-output-search-results">
    <DBA_IN_DATA>
      <argument-element nulldata="treatment-of-an-empty-element">argument-element-data</
argument-element>
      :
    </DBA_IN_DATA>
  </SQL-identifier>
</DBadapter>
```

---

When requesting execution of multiple SQL statements

Enter the DBA\_MULTI\_SQL element as a lower element of the root element (DBadapter element). You can enter multiple DBA\_MULTI\_SQL elements. The maximum number of DBA\_MULTI\_SQL elements that can be entered is the value set for the MAX\_SQL\_NO element in the SQL operation definition file.

---

```
<DBadapter>
  <DBA_MULTI_SQL>
    <SQL-identifier out_maxOccurs="maximum-number-of-output-search-results">
      <DBA_IN_DATA>
        <argument-element nulldata="treatment-of-an-empty-element">argument-element-data</
argument-element>
        :
      </DBA_IN_DATA>
    </SQL-identifier>
  </DBA_MULTI_SQL>
  :
</DBadapter>
```

---

#### (b) Elements and attributes of the request message

The table below lists and describes the elements and attributes of the request message. Note that the specification of the elements and attributes differs depending on the setting values in the SQL operation definition file. For details about the SQL operation definition file, see (1) *Creating the SQL operation definition file*.

Table 3–9: Elements and attributes of the request message

Element name or attribute name	Type	Description and setting value	Type of setting value	Can be omit
DBadapter	Element	This element is the root element, and has the following lower elements:  When requesting execution of a single SQL statement: The lower element is the <i>SQL-identifier</i> element.  When requesting execution of multiple SQL statements: The lower element is the DBA_MULTI_SQL element.	--	N
DBA_MULTI_SQL	Element	This element indicates that the request from a service requester requests execution of multiple SQL statements. Set the input information for the lower elements after <i>SQL-identifier</i> .  The maximum number of DBA_MULTI_SQL elements that can be entered is the value set for the MAX_SQL_NO element in the SQL operation definition file.	--	Y

### 3. Defining Adapters

Element name or attribute name	Type	Description and setting value	Type of setting value	Can be omit
DBA_MULTI_SQL	Element	Note: You cannot enter this element when requesting execution of a single SQL statement.	--	Y
<i>SQL-identifier</i>	Element	For <i>SQL-identifier</i> , specify the name of the SQL identifier specified in the SQL operation definition file. Set the input information for the lower elements after DBA_IN_DATA. The attribute includes <i>out_maxOccurs</i> .	--	Y
<i>out_maxOccurs</i>	Attribute	Specify the maximum number of output search results. Specify 0 or a larger value.  When the value is not specified: The value of <i>out_maxOccurs</i> specified in the SQL operation definition file is set.  When 0 is set for the value: The maximum number is set to 2,147,483,647.	xsd:int	Y
DBA_IN_DATA	Element	Set the data for the argument elements set in the SQL operation definition file for the lower elements after <i>argument-element</i> .	--	N
<i>argument-element</i>	Element	For <i>argument-element</i> , specify the name of the argument element specified in the SQL operation definition file. The data for the argument element specified in the SQL operation definition file is stored in this element. The attribute includes <i>nulldata</i> .  Note: If the relevant SQL identifier has no argument element in the settings in the SQL operation definition file, you cannot set this element.	xsd:string or the setting value for <i>data_type#</i>	N
<i>nulldata</i>	Attribute	Specify whether the data for the empty element is treated as <i>null</i> when <i>dba_inf</i> (argument type) specified in the SQL operation definition file is <i>data</i> . (You cannot specify this attribute if <i>dba_inf</i> is not <i>data</i> .)  If the data for the empty element is treated as <i>null</i> : Specify Y.  If the data for the empty element is not treated as <i>null</i> : Specify N.  Note that, if specification of this value is omitted, N is set for this attribute.	xsd:string {Y N}	Y

**Legend:**

--: This element or attribute has no setting value.

Y: This element or attribute can be omitted.

N: This element or attribute cannot be omitted.

#

The type of argument element to be stored depends on the value for *dba\_inf* (argument type) specified for the relevant argument element in the SQL operation definition file. If *dba\_inf* is *table*, *column*, or *preset*, the argument type is the *xsd:string* type. If *dba\_inf* is *data*, follow the setting value of *data\_type* (data type). For the correspondence between the setting values of *data\_type* and the argument element types, see Table 3-10 and Table 3-11.

If *dba\_inf* is *data*, you do not have to enclose the data to be stored in the argument element in single quotation marks (').

Table 3–10: Correspondence between the setting values for data\_type and the argument element types (for HiRDB)

Setting value for data_type	Argument element type
INTEGER	xsd:int <sup>#</sup>
SMALLINT	xsd:short <sup>#</sup>
DECIMAL	xsd:string
FLOAT	xsd:double <sup>#</sup>
REAL	xsd:float <sup>#</sup>
CHAR	xsd:string
VARCHAR	xsd:string
DATE	xsd:date <sup>#</sup>
TIME	xsd:string
TIMESTAMP	xsd:string
LONGVARIABLE	xsd:hexBinary
	xsd:base64Binary

#

You are also allowed to set an empty element.

Table 3–11: Correspondence between the setting values for data\_type and the argument element types (for Oracle)

Setting value for data_type	Argument element type
NUMERIC	xsd:string
CHAR	xsd:string
VARCHAR	xsd:string
DATE	xsd:date <sup>#</sup>
TIME	xsd:string
TIMESTAMP	xsd:string
LONGVARCHAR	xsd:string
CLOB	xsd:string
VARBINARY	xsd:hexBinary
	xsd:base64Binary
LONGVARIABLE	xsd:hexBinary
	xsd:base64Binary
BLOB	xsd:hexBinary
	xsd:base64Binary

#

You are also allowed to set an empty element.

Note that the values of argument elements set in the request message sent to the DB adapter are treated as shown in the following table.

Table 3–12: Treatment of argument elements in the request message sent to the DB adapter

Value of the argument element in the request message	Setting value in the SQL operation definition file		Value of the nulldata attribute in the request message	Handling of the argument element
	Attribute value dba_inf	Attribute value for data_type		
Empty element	data	--	Y or $\Delta$ Y $\Delta$ #	null
			Value other than Y or $\Delta$ Y $\Delta$ (including when specification of the value is omitted)	Empty character string (" ")
	Other than data	--	--	Empty character string (" ")
Other elements	data	CHAR, VARCHAR, LONGVARCHAR, CLOB	--	The space characters before and after the value are not deleted. <i>&lt;argument-element&gt;</i> $\Delta$ ABC $\Delta$ <i>&lt;/argument-element&gt;</i> is treated as $\Delta$ ABC $\Delta$ , where the space characters are not deleted.
		Other than CHAR, VARCHAR, LONGVARCHAR, and CLOB	--	The space characters before and after the value are deleted. <i>&lt;argument-element&gt;</i> $\Delta$ 123 $\Delta$ <i>&lt;/argument-element&gt;</i> is treated as 123, where the space characters are deleted.
	Other than data	--	--	The space characters before and after the value are not deleted. <i>&lt;argument-element&gt;</i> $\Delta$ ABC $\Delta$ <i>&lt;/argument-element&gt;</i> is treated as $\Delta$ ABC $\Delta$ , where the space characters are not deleted.

## Legend:

--: This value is not used for judgment.

 $\Delta$  : One or more space characters (single-byte space character, line feed character, tab character)

#

Space characters before and after the attribute value of nulldata are ignored.

## (c) Response message format

A response message is output in the XML document format.

If the search result (e.g., the column name) contains a character that cannot be used for XML documents, such a character is escaped and set in the response message. For example, if the search result contains < or &, these are escaped to &lt; or &amp;, respectively. Therefore, you do not have to escape the data stored in the database.

The response message format is different when responding to an execution request for a single SQL statement is different, from when responding to an execution request for multiple SQL statements.

The response message format is shown below. Italic characters indicate variable values.

When responding to an execution request for a single SQL statement

The *SQL-identifier* element is output as a lower element of the root element (DBadapter element).

```

<DBadapter>
  <SQL-identifier out_maxOccurs="maximum-number-of-output-search-results">
    <DBA_IN_DATA>
      <argument-element nulldata="treatment-of-an-empty-element">argument-element-data</
argument-element>
      :
    </DBA_IN_DATA>
    <DBA_OUT_DATA>
      <DBA_ResultSetNo>SQL-return-value</DBA_ResultSetNo>
      <DBA_ResultSetXmlNo>number-of-search-results</DBA_ResultSetXmlNo>
      <DBA_ResultSetName>
        <DBA_ResultSetColumnName cid="column-number">column-name</DBA_ResultSetColumnName>

```

---

```

      :
      </DBA_ResultSetName>
      <DBA_ResultSet lid="line-number">
        <DBA_ResultSetColumn cid="column-number">search-result-of-the-relevant-n-th-column</
DBA_ResultSetColumn>
        :
      </DBA_ResultSet>
      :
    </DBA_OUT_DATA>
  </SQL-identifier>
</DBadapter>

```

---

#### When responding to an execution request for multiple SQL statements

The DBA\_MULTI\_SQL element is output as a lower element of the root element (DBadapter element). As many DBA\_MULTI\_SQL elements as DBA\_MULTI\_SQL elements (written in the request message) are output.

---

```

<DBadapter>
  <DBA_MULTI_SQL>
    <SQL-identifier out_maxOccurs="maximum-number-of-output-search-results">
      <DBA_IN_DATA>
        <argument-element nulldata="treatment-of-an-empty-element">argument-element-data</
argument-element>
        :
      </DBA_IN_DATA>
      <DBA_OUT_DATA>
        <DBA_ResultSetNo>SQL-return-value</DBA_ResultSetNo>
        <DBA_ResultSetXmlNo>number-of-search-results</DBA_ResultSetXmlNo>
        <DBA_ResultSetName>
          <DBA_ResultSetColumnName cid="column-number">column-name</DBA_ResultSetColumnName>
          :
        </DBA_ResultSetName>
        <DBA_ResultSet lid="line-number">
          <DBA_ResultSetColumn cid="column-number">search-result-of-the-relevant-n-th-column</
DBA_ResultSetColumn>
          :
        </DBA_ResultSet>
        :
      </DBA_OUT_DATA>
    </SQL-identifier>
  </DBA_MULTI_SQL>
  :
</DBadapter>

```

---

Note that, if the input information in the request message is 0, the response message is output in the following format:

### 3. Defining Adapters

Request message when input information is 0

```

<DBadapter>
  <DBA_MULTI_SQL>
    <SQLidentifier out_maxOccurs="Maximum number of search result ouput cases">
      <DBA_IN_DATA>
        <Argument element nulldata="Handling empty element">Data of argument element</Argument
        element>
          :
        </DBA_IN_DATA>
      <DBA_OUT_DATA>
        <DBA_ResultSetNo>SQL return value</DBA_ResultSetNo>
        <DBA_ResultSetXmlNo>Number of search result cases</DBA_ResultSetXmlNo>
        <DBA_ResultSetName>
          <DBA_ResultSetColumnName cid="column number">Column name</DBA_ResultSetColumnName>
          :
        </DBA_ResultSetName>
        <DBA_ResultSet lid="Line number">
          <DBA_ResultSetColumn cid="Column number">Search result of n column of corresponding line</
          DBA_ResultSetColumn>
          :
        </DBA_ResultSet>
        :
      </DBA_OUT_DATA>
    </SQL identifier>
  </DBA_MULTI_SQL>
  :
</DBadapter>

```

When there is element message indicating that the data in this part is omitted, the input information is 0.



Reply message when input information is 0

```

<DBadapter>
</DBadapter>

```

#### (d) Elements and attributes of the response message

The following table lists and describes the elements and attributes of the response message.

Table 3–13: Elements and attributes of the response message (service requester for a DB adapter)

Element name or attribute name	Type	Description and setting value	Type of stored value	Ca n be om itte d
DBadapter	Element	This is the root element, and has the following lower elements:  When responding to an execution request for a single SQL statement: The <i>SQL-identifier</i> element is output as the lower element.	--	N

Element name or attribute name	Type	Description and setting value	Type of stored value	Can be omitted
DBadapter	Element	When responding to an execution request for multiple SQL statements: The DBA_MULTI_SQL element is output as the lower element.	--	N
DBA_MULTI_SQL	Element	This element indicates a response to an execution request for multiple SQL statements from the service requester. The input information (information set in the request message) and output information (SQL execution result) are output to the lower elements after <i>SQL-identifier</i> . As many DBA_MULTI_SQL elements as DBA_MULTI_SQL elements (written in the request message) are output.	--	N#1
<i>SQL-identifier</i>	Element	The input information (information set in the request message) and output information (SQL execution result) are stored in the lower elements DBA_IN_DATA and DBA_OUT_DATA, respectively. <i>SQL-identifier</i> is the SQL identifier set in the request message. Note that the <code>out_maxOccurs</code> attribute (maximum number of output search results) is stored only when it is set in the request message.	--	N
DBA_IN_DATA	Element	Input information set in the request message is stored in the lower element of this element.	--	N
DBA_OUT_DATA	Element	Information output when SQL is executed is stored in the lower element. The output information to be stored depends on the executed SQL statement. When INSERT is executed: DBA_ResultSetNo is stored as the lower element. When SELECT is executed: The following elements are stored as the lower elements: <ul style="list-style-type: none"> <li>DBA_ResultSetNo</li> <li>DBA_ResultSetXmlNo</li> <li>DBA_ResultSetName</li> <li>DBA_ResultSet</li> </ul>	--	N
DBA_ResultSetNo	Element	The output information to be stored depends on the executed SQL statement. When INSERT is executed: The number of updated lines (the value acquired by <code>executeUpdate</code> of the <code>PreparedStatement</code> class) is stored. The value to be stored when the element value exceeds 2, 147, 483, 647 depends on the specifications of <code>executeUpdate</code> of the <code>PreparedStatement</code> class of the JDBC driver used by the DB Connector. When SELECT is executed: The number of search results is stored. If the element value to be stored exceeds 2, 147, 483, 647, 2, 147, 483, 647 is stored.	xsd:int	N

### 3. Defining Adapters

Element name or attribute name	Type	Description and setting value	Type of stored value	Can be omitted
DBA_ResultSetXmlNo	Element	<p>If the executed SQL instruction is <code>SELECT</code>, among the actual number of search results (<code>DBA_ResultSetNo</code> value), only the number of search results that are stored in this response message is stored.</p> <p>If the <code>DBA_ResultSetNo</code> value is larger than the value specified for <code>out_maxOccurs</code> in the SQL operation definition file or in the request message, the value specified for <code>out_maxOccurs</code> is stored.</p> <p>If the element value to be stored exceeds <code>2,147,483,647, 2,147,483,647</code> is stored.</p>	xsd:int	Y1
DBA_ResultSetName	Element	<p>If the executed SQL instruction is <code>SELECT</code>, the column name of the search result is stored in the lower element <code>DBA_ResultColumnName</code>.</p> <p>Note that, if the search result is 0, this element will not be stored.</p>	--	Y2
DBA_ResultColumnName	Element	<p>If the executed SQL instruction is <code>SELECT</code>, the name of the relevant column is stored.</p> <p>Note that, if the resulted column is a column of the database, it is set in <i>column-name</i> format. If the resulted column name has not been acquired, a name that conforms with the specifications of the database to which the SQL statement was issued will be returned.</p> <p>The attribute includes <code>cid</code>.</p>	xsd:string	Y2
cid	Attribute	Indicates the column number of the value to be stored in <code>DBA_ResultColumnName</code> .	xsd:int	
DBA_ResultSet	Element	<p>If the executed SQL instruction is <code>SELECT</code>, the result for each line is stored in the lower element <code>DBA_ResultColumn</code>. Note that, if the search result is 0, this element will not be stored.</p> <p>The attribute includes <code>lid</code>.</p>	--	Y2
lid	Attribute	<p>Indicates the line number of the value to be stored in <code>DBA_ResultSet</code>.</p> <p>If the element value to be stored exceeds <code>2,147,483,647, 2,147,483,647</code> is stored.</p>	xsd:int	
DBA_ResultColumn	Element	<p>The search result for the <i>n</i>-th column of the relevant line (the line indicated by <code>DBA_ResultSet</code>) will be stored.<sup>#2</sup></p> <p>The attribute includes <code>cid</code> and <code>nulldata</code>.</p>	xsd:string	Y2
cid	Attribute	Indicates the column number of the value to be stored in <code>DBA_ResultColumn</code> .	xsd:int	
nulldata	Attribute	<p>Indicates whether the data stored in <code>DBA_ResultColumn</code> is null.</p> <p>If the data is null, <code>Y</code> is set. If the data is not null, this attribute is not set.</p>	xsd:string	

Legend:

--: The value is not stored.

Y1: Output only when the SQL instruction is `SELECT`.

Y2: Output when the SQL instruction is `SELECT` and there are one or more execution results.

N: Always output. This element is not omitted.

#1

This element is always output when responding to an execution request for multiple SQL statements. This element is not output when responding to an execution request for a single SQL statement.

#2

The value acquired by `getString` of the `ResultSet` class is stored. Therefore, the format of the search result conforms to the specifications of `getString` of the `ResultSet` class of JDBC driver used by the DB Connector. If you want to change the format of the search result, use the function provided by the database for searching.

The following is an example of using the `VARCHAR_FORMAT` function to change the format of the `TIMESTAMP` type (column name: `c_ts`) of HiRDB.

---

```
SELECT VARCHAR_FORMAT(c_ts, 'YYYY-MONTH-DD HH:MI') FROM table-name
```

---

### 3.3.6 Defining a TP1 adapter

INTENTIONALLY DELETED

### 3.3.7 Defining file adapters

This subsection describes how to define file adapters.

#### (1) Creating a message format

Contents of the message format to be created in file adapters are described here.

##### (a) Request message format of file adapters

Send the following information in the request message used for requesting execution of file operations:

- File operation information
- Data for writing a file

- **Information of file operations**

File operation information is required to access a file. The following table describes the contents of file operation information:

Table 3–14: File operation information

Information	Description
File name	<p>Specifies the name of the file that is to be read or written. Specify the file name in the absolute path. Length of the file name can be within the range of 1 to 200 bytes when it is converted to UTF-8. Spaces present before and after the file name are deleted. For Windows, the path is case-insensitive.</p> <p>You cannot specify a file in the following manner:</p> <ul style="list-style-type: none"> <li>• A file name containing relative path, expression of a parent directory ("<code>..</code>", "<code>../</code>"), and current directory expression ("<code>..</code>", "<code>./</code>").</li> <li>• For UNIX, file name containing symbolic link.</li> <li>• For Windows, a file name containing the UNC format, stream name of NTFS or reserved device name.</li> </ul>
Reading mode	<p>Specifies the mode for reading a file. The mode name is as follows:</p> <ul style="list-style-type: none"> <li>• Batch</li> </ul> <p>Spaces present before and after the specified value are deleted.</p>
Writing mode	<p>Specifies the mode for writing a file. Mode names are as follows:</p> <ul style="list-style-type: none"> <li>• New</li> <li>• Add</li> </ul> <p>Spaces present before and after the specified value are deleted.</p>

- **Data for writing a file**

Specifies the user data to be specified in a request message, when sending a writing request for an XML file or a binary file.

The contents of a request message differ as per the format or type of the file to be read or written. Therefore, describe only the required information in a message from the file operation information and data for writing a file at the time of message creation.

The following table describes the information required for a request message, when reading an XML file:

Table 3–15: Information of a request message when reading an XML file (XML message)

Information	Tag name within a message	Specified value	Default value at the time of omission	Specification
Root element	ReadRequest	-	-	Y
File header element	fileheader	-	-	Y
File name element	filename	File name	-	Y
File mode element	filemode	all	all	N

Legend:

Y: Must be specified.

N: Can be omitted.

-: Not applicable.

The following table describes the information required for a request message, when writing an XML file:

Table 3–16: Information of a request message when writing an XML file (XML message)

Information	Tag name within a message	Specified value	Default value at the time of omission	Specification
Root element	WriteRequest	-	-	Y
File header element	fileheader	-	-	Y
File name element	filename	File name	-	N
File mode element	filemode	new	new	N
Data for writing a file	userdata	Optional to user	-	Y

Legend:

Y: Must be specified.

N: Can be omitted.

-: Not applicable.

The following table describes the information required for a request message, when reading a binary file:

Table 3–17: Information of a request message when reading a binary file (XML message)

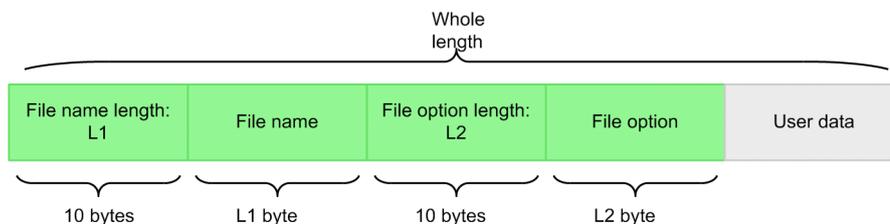
Information	Tag name within a message	Specified value	Default value at the time of omission	Specification
Root element	ReadRequest	-	-	Y
File header element	fileheader	-	-	Y
File name element	filename	File name	-	Y
File mode element	filemode	all	all	N

Legend:

- Y: Must be specified.
- N: Can be omitted.
- : Not applicable.

When writing a binary file, the request message will have binary format. The following figure shows the structure of a request message in binary format:

Figure 3–6: Structure of a request message in binary format



The overall length of a message must consist of the following relational expression:

$$\text{Overall length} \geq 10 + L1 + 10 + L2$$

The following table describes the information required for a request message, when writing a binary file:

Table 3–18: Information of a request message when writing a binary file (binary message)

Information	Specified value	Default value at the time of omission	Specification
Length of a file name	Specify byte length of a file name with decimal numeric string.	-	Y
File name	File name	-	Y
Length of a file option	Specify byte length of a file option with decimal numeric string.	-	Y
File option	New writing: mode=new Additional writing: mode=append	mode=new	N
User data	Optional to user	-	Y#

Legend:

- Y: Must be specified.
- N: Can be omitted.
- : Not applicable.

Note#

- You can specify 0 byte.

For details on the sample of each request message, see (f) *Message examples*.

#### (b) Response message format of file adapters

Send the following information in the response message that returns the information on reading a file adapter to the invoker of a file adapter:

- Data for reading a file
- File writing process result
- **Data for reading a file**
  - Specifies the user data to be stored and to be returned to a response message, when reading an XML file or a binary file.

- **Result of file writing process**

The result of writing process is stored in a response message, when writing an XML file or a binary file. If the writing process fails, the file adapter notifies an exception to HCSC server and does not return a response message. The result of writing process is stored to a response message only when the writing process ends successfully. Therefore, return value of the response message must be "OK" at the time of writing.

The contents of a response message differ as per the format and type of the file to be read or written. The following table describes the information of a response message that is returned when a file is to be read and written:

Table 3–19: Information of response message

File format and type	Information	Tag name within a message	Return value
Reading an XML file	Data for reading a file	Optional to user	Optional to user
Writing an XML file	Result of file writing process	status	OK
Reading a binary file	Data for reading a file	-	Optional to user
Writing a binary file	Result of file writing process	status	OK

Legend:

-: Not applicable.

(c) Types of a message format

If message formats of a file adapter invoker (standard message) and a file adapter (service component message) are different, create message format definition files of a file adapter invoker (standard message) and a file adapter (service component message). Map the respective message formats from the data transformation (mapping) definition screen.

The following table describes the message format definition file to be created when performing data transformation:

Table 3–20: Message format definition file to be created

File format and type	Request message		Response message	
	Standard message (invoker)	Service component message (file adapter)	Standard message (invoker)	Service component message (file adapter)
Reading an XML file	XML <sup>#</sup>	XML	XML <sup>#</sup>	XML
Writing an XML file	XML <sup>#</sup>	XML	XML <sup>#</sup>	XML
Reading a binary file	XML <sup>#</sup>	XML	XML	Binary
Writing a binary file	XML	Binary	XML <sup>#</sup>	XML

Legend:

XML: XML format definition file

Binary: Binary format definition file

Note<sup>#</sup>

When the message format of file adapter invoker (standard message) and file adapter (service component message) is same, you need not create and set the message format definition file of the file adapter invoker (standard message).

The message format definition file to be created differs depending on whether to handle data in XML format or in binary format (other than XML format) in a message to be used in a file adapter.

- When handling data in XML format

When handling data in XML format, create an XML format definition file. Create the XML format definition file as XML schema file (extension: .xsd). For details on how to create XML format definition file, see (d) *Creating an XML format definition file*.

- When handling data in binary format

When handling data in binary format (other than XML format), create a binary format definition file. The binary format definition file is created, when you define the storage format of a value within the data in binary format (such as length of a value, its relation with another value) and add the defined information in XML schema file (extension: .fdx). The binary format definition file is created as per XML specifications. Hence you can use this file as the transformation source and destination, when defining data transformation in same way as the XML format definition file. For details on how to create a binary format definition file, see (e) *Creating a binary format definition file*.

#### (d) Creating an XML format definition file

Method of creating a standard message and file adapter (service component message) is as follows:

- **Creating an XML format definition file of a standard message**

Create an XML format definition file of a standard message using WST (Web Standard Tools) provided by Eclipse.

The XML format definition file to be created must comply with the scoping of the XML schema that can be used on the HCSC server. For details on the conditions, see 2.6.5 *Scoping of XML schema* in the manual *Service Platform Basic Development Guide*.

If an error occurs, when you select the message format file to be used for defining a file adapter and a business process (error message having message ID starting with KECX), an error exists in XML schema file. Here, see the message *11.KECX (message output by XML Processor)* in the manual *Application Server Messages*, and take actions.

- **Creating an XML format definition file of file adapter (service component message)**

The following table describes the definition file used by a user and the need for editing this file, when creating the XML format definition file of a file adapter (service component message):

Table 3–21: Definition file used by the user and the need for editing this file

File operation mode	Message type	Definition file that is used	Editing by user
Reading	Request message	adpff_read.xsd	N
	Response message	Created by user	Y
Writing	Request message	adpff_header.xsd	N
		adpff_write.xsd	Y
	Response message	adpff_result.xsd	N

Legend:

Y: Must be edited.

N: Need not be edited.

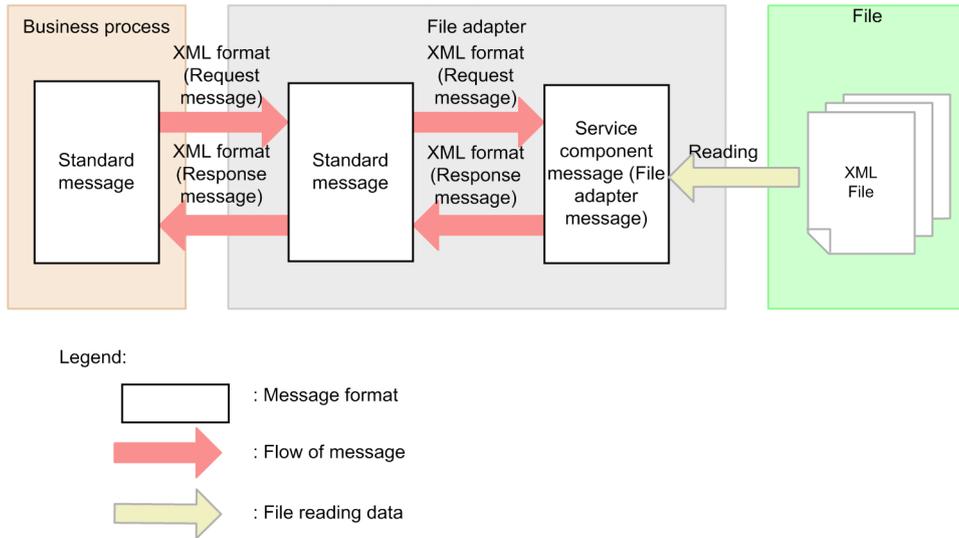
The method of creating the XML format definition file is described here for each file operation mode:

#### **For reading**

When reading an XML file using a file adapter, create the following message format definition file:

- Request message: XML format definition file
- Response message: XML format definition file

Figure 3–7: Message flow at the time of reading an XML file



- Request message

For the XML format definition file for a request message, use the following files:

Format definition file for a request message (for reading XML and binary format)

*Installation directory of the service platform*\CSC\custom-adapter\File\schema\adpff\_read.xsd

- Response message

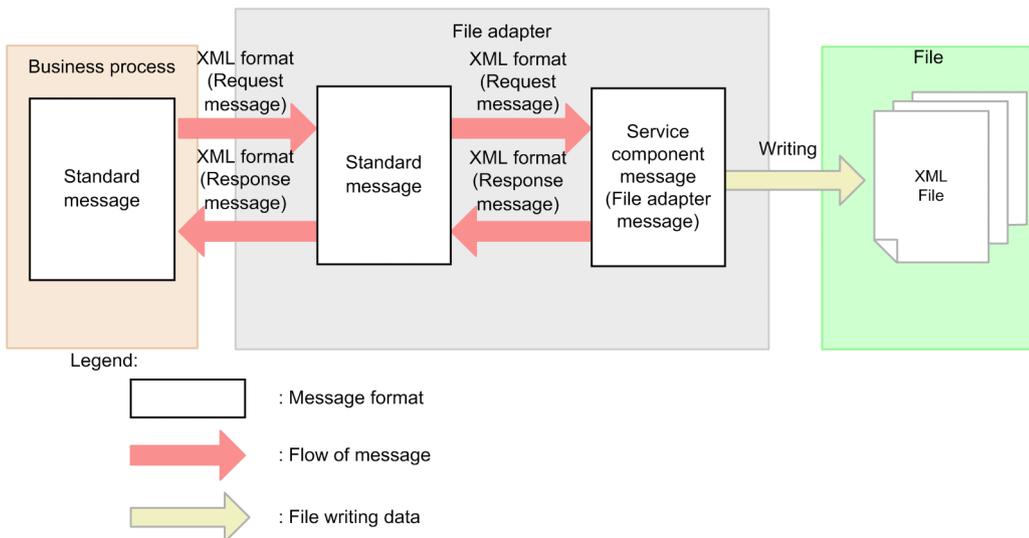
The user creates the XML format definition file for the response message on his own.

**For writing**

When writing an XML file using a file adapter, create the following message format definition file:

- Request message: XML format definition file
- Response message: XML format definition file

Figure 3–8: Message flow at the time of writing an XML file



- Request message

Use the following file and the template file to create the XML format definition file for a request message:

File operation information file for a request message (for XML format writing)

*Installation directory of the service platform*\CSC\custom-adapter\File\schema\adpff\_header.xsd

Format definition template file for a request message (for XML format writing)

Installation directory of the service platform\CSC\custom-adapter\File\schema\templates\adpff\_write.xsd

The user must edit the format definition template file for a request message. The following figure shows the locations to be edited in the format definition template file for a request message:

Figure 3–9: Format definition template file for a request message

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <!-- All Rights Reserved. Copyright (C) 2007, Hitachi, Ltd. -->
3 <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
4     xmlns:adpff="http://FlatFiles.cstmadv.csc.soft.Hitachi.co.jp/WriteRequest"
5     xmlns: <!-- prefix -->=<!-- Define targetNamespace. -->
6     targetNamespace=<!-- Define targetNamespace. -->
7
8
9 <xsd:import namespace="http://FlatFiles.cstmadv.csc.soft.Hitachi.co.jp/WriteRequest"
10     schemaLocation="adpff_header.xsd"/>
11
12
13
14 <xsd:element name="WriteRequest" type="<!-- prefix -->:format"/>
15 <xsd:complexType name="format">
16 <xsd:sequence>
17 <xsd:element name="fileheader" type="adpff:fileheader"/>
18 <xsd:element name="userdata" type="<!-- prefix -->:userdata"/>
19 </xsd:sequence>
20 </xsd:complexType>
21
22 <xsd:complexType name="userdata">
23 <!-- Define XML schema of the user data. -->
24 </xsd:complexType >
25 </xsd:schema>
26

```

Legend:

: Locations to be edited by user

The following table describes the format definition template file for a request message according to the line number given in the figure. Note that line numbers given in the figure are not included in the actual template.

Table 3–22: Description of the format definition template file for a request message

Line number	Description
4 <sup>th</sup> line	Specifies name space declaration of the file operation information file (for XML format writing: adpff_header.xsd) for a request message.
5 <sup>th</sup> line	Specifies name space declaration of this template file. The user declares any prefix in the value of targetName space attribute specified in the 7 <sup>th</sup> line.
6 <sup>th</sup> line	When importing another schema file, specify name space declaration of that schema. <sup>#1, #2</sup>
7 <sup>th</sup> line	Specifies the target name space of this template file. The user specifies a unique target name space in the value of targetName space attribute.
9 <sup>th</sup> ~10 <sup>th</sup> line	Specifies import declaration of the file operation information file for a request message (for XML format writing: adpff_header.xsd). For this template file, import the file operation information file for a request message (for XML format writing). The user modifies the value of schemaLocation attribute in the relative path on which the file operation information file for a request message exists (for XML format writing). Same directory is mentioned by default.
14 <sup>th</sup> ~20 <sup>th</sup> line	Specifies WriteRequest element declaration. It is the root element of a request message. For a file adapter, acquire this element name and determine file I/O type (WriteRequest: Writing, ReadRequest: Reading). Wait for fileheader element and userdata element. The user specifies the prefix declared in the 5 <sup>th</sup> line as a prefix of userdata element.

### 3. Defining Adapters

Line number	Description
22 <sup>nd</sup> line	Specifies userdata element declaration.
23 <sup>rd</sup> line	Specifies the XML schema information used for user data. <sup>#1, #2</sup>

**Note#1**

There are 2 types of specification methods. One is to specify directly in this file. Another is to create a file separately to specify the XML schema information used for user data and import it from this file. When importing, specify the target name space of the schema to be imported in the 6th line and import statement of the schema to be imported in 12th line. For details on the specification example, see *(f) Message examples*.

**Note#2**

When specifying the XML schema information used for user data, you cannot declare multiple elements in the child element of the userdata element. Declare only 1 child element of userdata element, since it becomes the root element when writing an XML file.

- Response message

Use the following file for the XML format definition file for a response message:

Format definition file for a response message (for XML and binary format writing)

*Installation directory of the service platform*\CSC\custom-adapter\File\schem  
 \adpff\_result.xsd

**(e) Creating a binary format definition file**

The method to create a binary format definition file is described here.

The following table describes the definition file used by the user and the need for editing this file, when creating a binary format definition file:

Table 3–23: Definition file used by the user and the need for editing this file

File operation mode	Message type	Definition file that is used	Editing by user
Reading	Request message	adpff_read.xsd	N
	Response message	Created by user	Y
Writing	Request message	adpff_write.fdx	Y
	Response message	adpff_result.xsd	N

**Legend:**

Y: Must be edited.

N: Need not be edited.

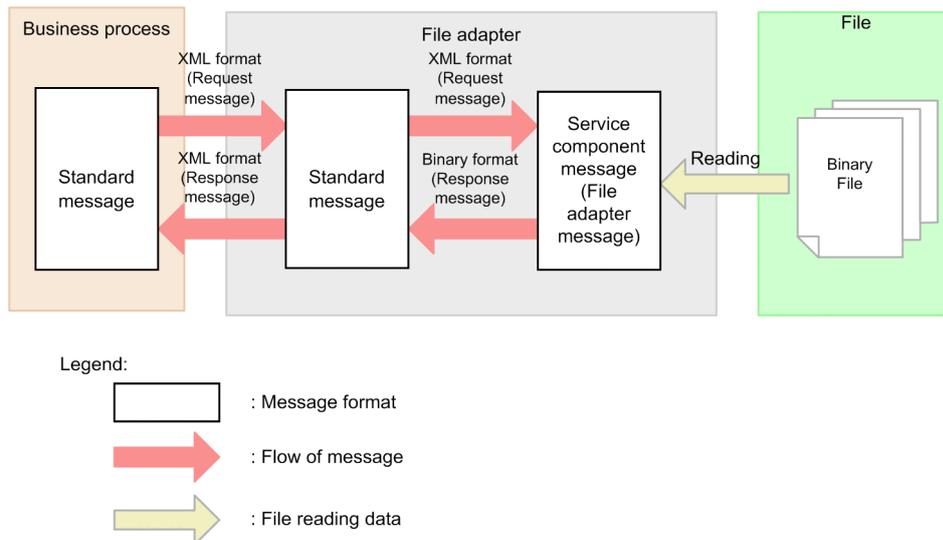
The method to create a binary format definition file is described here.

**• For reading**

When reading a binary file using a file adapter, create the following message format definition file:

- Request message: XML format definition file
- Response message: Binary format definition file

Figure 3–10: Message flow at the time of reading a binary file



**Request message**

For the XML request message format definition file, use the following file:

Format definition file for a request message (for XML and binary format reading)

*Installation directory of the service platform*\CSC\custom-adapter\File\schema\adpff\_read.xsd

**Response message**

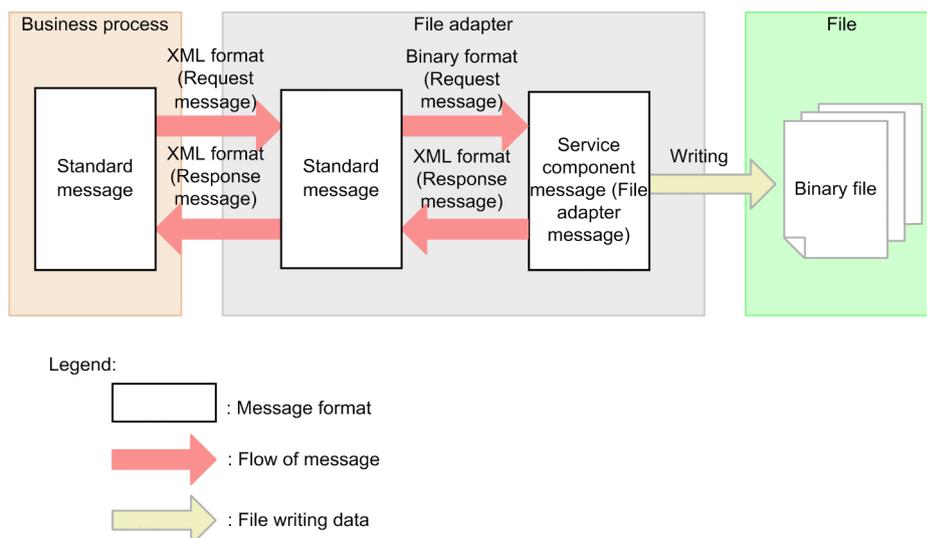
The user creates the binary format definition file for a response message on his own.

• **For writing**

When writing a binary file using a file adapter, create the following message format definition file:

- Request message: Binary format definition file
- Response message: XML format definition file

Figure 3–11: Message flow at the time of writing a binary file



**Request message**

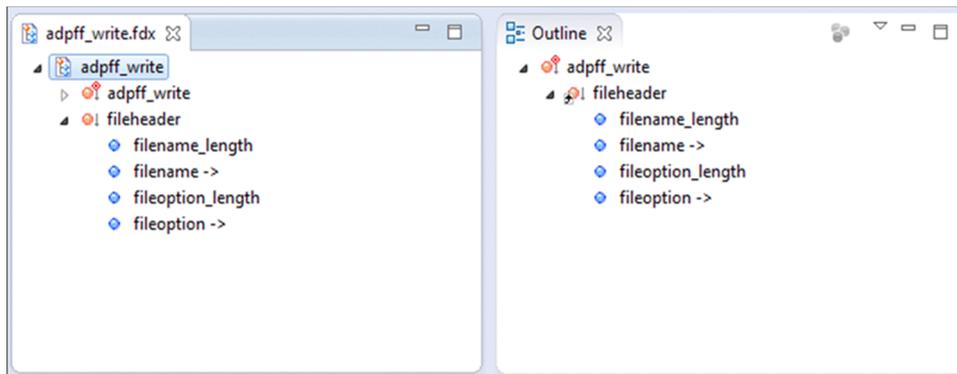
The user creates the binary format definition file for a request message by editing the following template file in the binary format definition screen:

Format definition template file for a request message (for binary format writing)

Installation directory of the service platform\CSC\custom-adapter\File\schema\templates\adpff\_write.fdx

The following figure shows the format definition template file for a request message:

Figure 3–12: Format definition template file for a request message screen



The following table describes the setup value of the format definition template file for a request message:

Table 3–24: Setup value of the format definition template file for a request message

Element name	Data type	Size (byte)	Occurrence count	Description
adpff_write	-	-	-	Root element
fileheader	-	-	1 time	File operation information element
filename_length	Integer	10	1 time	File name length element
filename	String	Variable	1 time	File name element
fileoption_length	Integer	10	1 time	File option length element
fileoption	String	Variable	1 time	File option element

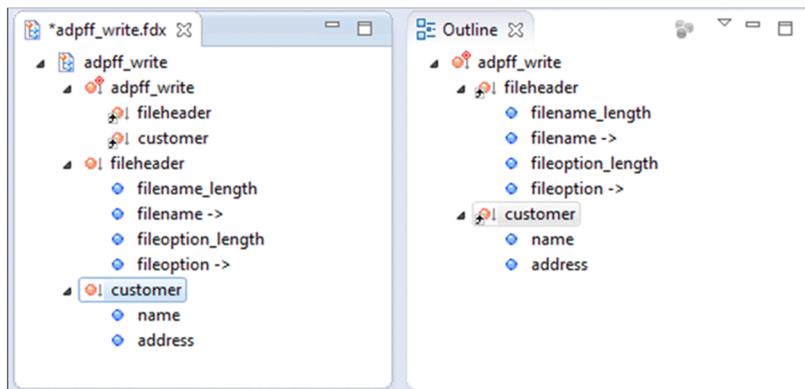
Legend:

-: Not applicable.

- **Creation method**

The user creates a user data element of any element name as a child element of the root element (adpff\_write element). However, the user must create this element such that fileheader element becomes first. Operation is not guaranteed, if the fileheader element is not first. Once the user creates the binary format definition file, he saves it with any name, and specifies it in the service adapter settings screen. The following figure shows the sample screen of the binary format definition file after it is created. The user declares a customer element as the user-data element in the sample screen and creates the name element and the address element as its child element.

Figure 3–13: Sample screen after the binary format definition file is created



### Response message

For the XML format definition file for a response message, use the following file:

Format definition file for a response message (for XML and binary format writing)

*Installation directory of the service platform*\CSC\custom-adapter\File\schemata\adpff\_result.xsd

### (f) Message examples

Examples of a request message when performing each file operation are as follows:

To describe message examples, the following table lists the format definition files for a request message used for defining the request message format:

Table 3–25: Format definition file list for a request message that is provided

File name	Contents	Editing
adpff_read.xsd	Format definition file for a request message (for XML and binary format reading)	N
adpff_write.xsd	Format definition template file for a request message (for XML format writing)	Y
adpff_write.fdx	Format definition template file for a request message (for binary format writing)	Y

Legend:

Y: This file is to be used after editing it.

N: This file is to be used without editing it.

#### • Request message at the time of reading a file

Use the request message format definition file (for XML and binary format reading) for a request message format at the time of reading a file. Use the contents of the request message format definition file as are without changing them.

An example of describing a request message at the time of reading a file is described here.

```
<?xml version="1.0" encoding="UTF-8"?>
<adpff:ReadRequest xmlns:adpff="http://
FlatFiles.cstmadv.csc.soft.Hitachi.co.jp/ReadRequest">
  <fileheader>
    <filename>C:\WORK\ReadFile.xml</filename>
    <filemode>all</filemode>
  </fileheader>
</adpff:ReadRequest>
```

#### • Request message at the time of writing a file (XML format)

Use the format definition template file (for XML format writing) for the request message format at the time of writing a file in XML format. Append the XML schema information of user data to the request message format definition file and create the request message format definition file.

Following are the 2 methods for appending the XML schema information of user data:

- Specify the XML schema information directly in the request message format definition file

- Import the XML schema information in the request message format definition file

#### When specifying XML schema information of user data directly

An example of specifying the XML schema information for the user data directly in the request message format definition file is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- All Rights Reserved. Copyright (C) 2007, Hitachi, Ltd. -->
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            xmlns:adpff="http://FlatFiles.cstmadp.csc.soft.Hitachi.co.jp/
WriteRequest"
            xmlns:wr="http://WriteRequest"
            targetName space="http://WriteRequest">
  <xsd:import name space="http://
FlatFiles.cstmadp.csc.soft.Hitachi.co.jp/WriteRequest"
            schemaLocation="adpff_header.xsd"/>
  <xsd:element name="WriteRequest" type="wr:format"/>

  <xsd:complexType name="format">
    <xsd:sequence>
      <xsd:element name="fileheader" type="adpff:fileheader"/>
      <xsd:element name="userdata" type="wr:userdata"/>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="userdata">
    <xsd:sequence>
      <xsd:element name="customer" type="wr:customer"/>
    </xsd:sequence>
  </xsd:complexType>

  <xsd:complexType name="customer">
    <xsd:sequence>
      <xsd:element name="name" type="xsd:string" />
      <xsd:element name="address" type="xsd:string" />
    </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```

An example of describing a request message that complies with above-mentioned XML schema is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<wr:WriteRequest xmlns:wr="http://WriteRequest">
  <fileheader>
    <filename>C:\WORK\writeXML.xml</filename>
    <filemode>new</filemode>
  </fileheader>
  <userdata>
    <customer>
      <name>Hitachi Taro</name>
      <address>Yokohama Totsuka</address>
    </customer>
  </userdata>
</wr:WriteRequest>
```

#### Importing the XML schema information of the user data

An example of importing the XML schema information of the user data in the request message format definition file and specifying that information is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- All Rights Reserved. Copyright (C) 2007, Hitachi, Ltd. -->
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
            xmlns:adpff="http://FlatFiles.cstmadp.csc.soft.Hitachi.co.jp/
WriteRequest"
            xmlns:wr="http://WriteRequest"
            targetName space="http://WriteRequest"
            xmlns:usr="http://UserSchema">

  <xsd:import name space="http://
FlatFiles.cstmadp.csc.soft.Hitachi.co.jp/WriteRequest"
            schemaLocation="adpff_header.xsd"/>
```

```

<xsd:import name space="http://UserSchema" schemaLocation="User.xsd"/>

<xsd:element name="WriteRequest" type="wr:format"/>
<xsd:complexType name="format">
  <xsd:sequence>
    <xsd:element name="fileheader" type="adpff:fileheader"/>
    <xsd:element name="userdata" type="wr:userdata"/>
  </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="userdata">
  <xsd:sequence>
    <xsd:element ref="usr:customer"/>
  </xsd:sequence>
</xsd:complexType>
</xsd:schema>

```

The contents of the imported User.xsd are as follows:

```

<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:usr="http://UserSchema"
  targetName space="http://UserSchema">
  <xsd:element name="customer">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="name" type="xsd:string"/>
        <xsd:element name="address" type="xsd:string"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>

```

An example of specifying a request message that complies with the above-mentioned XML schema is as follows:

```

<?xml version="1.0" encoding="UTF-8"?>
<wr:WriteRequest xmlns:wr="http://WriteRequest">
  <fileheader>
    <filename>C:\WORK\writeXML.xml</filename>
    <filemode>new</filemode>
  </fileheader>
  <userdata>
    <usr:customer xmlns:usr="http://UserSchema">
      <name>Hitachi Tarou</name>
      <address>Yokohama Totsuka</address>
    </usr:customer>
  </userdata>
</wr:WriteRequest>

```

- **Request message at the time of writing a file (binary format)**

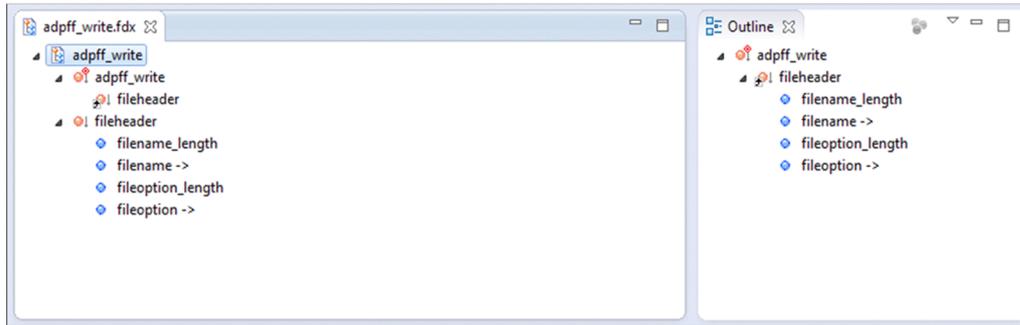
Use the format definition template file for a request message (for writing in binary format ) in the request message format at the time of writing a file in binary format. Import the request message format definition file in the project for creating the binary format definition file of Eclipse and add the definition for the user data.

A request message is handled in the XML format on a business process. Hence convert this message into a request message in binary format of a file adapter by transforming the file from the XML format to binary format using data transformation functionality.

An example of defining the request message format definition file and an example of defining data transformation of a request message are as follows:

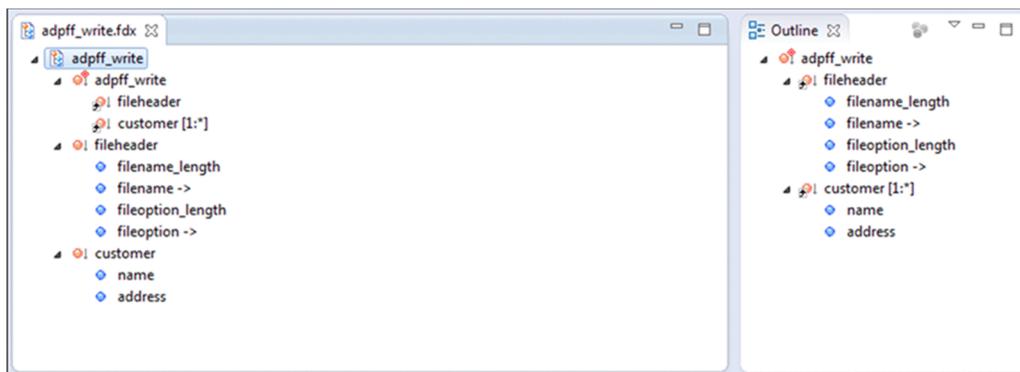
If you import the request message format definition file that is provided, the definition of file operation information of a file adapter is displayed.

Figure 3–14: Definition screen in which the provided file is imported



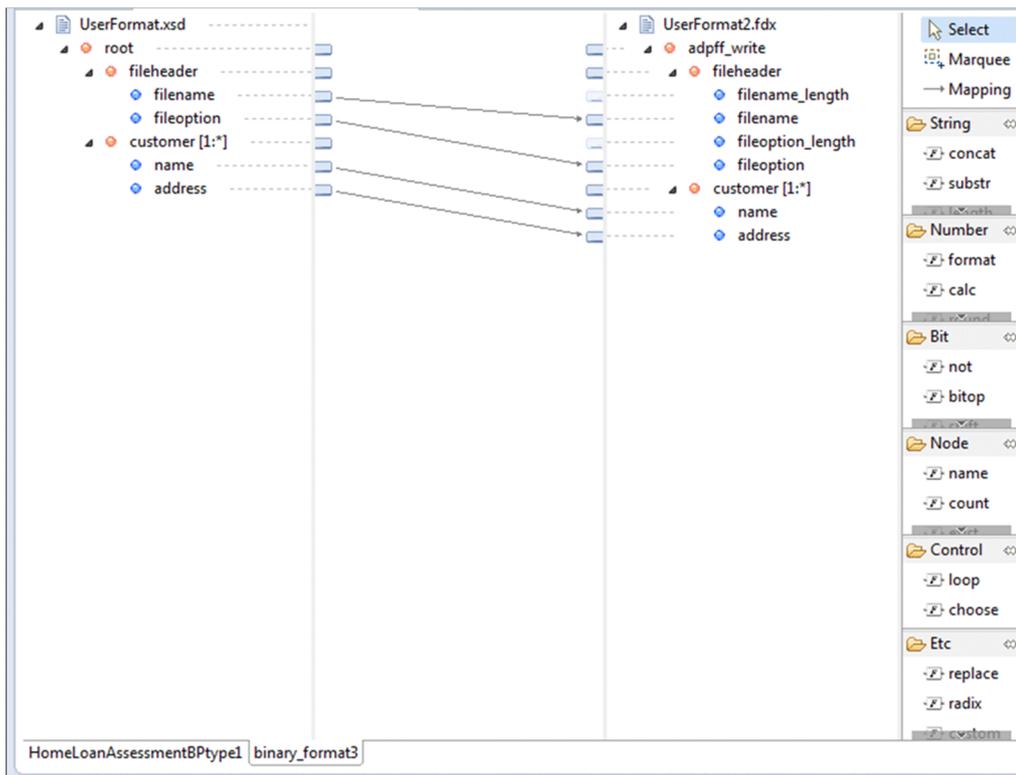
Add the definition of the user data in this definition screen. Elements of name and address are defined as customer data in the following example:

Figure 3–15: Definition screen in which definition of the user data is added



Transform the data from the calling request message format of a file adapter into the binary format definition file that is created. Data length of filename and fileoption elements is automatically set in filename\_length and fileoption\_length elements using the data transformation function.

Figure 3–16: Data transformation definition screen



## (2) Defining data transformation

Set the message format definition file of transformation source and the message format definition file of transformation destination on the data transformation (mapping) definition screen and define the transformation.

For details on how to define data transformation, see 6. *Defining Data Transformation* in the manual *Service Platform Basic Development Guide*.

## (3) Operations to be performed on service adapter settings screen

The procedure for defining a file adapter is as follows:

1. Display service adapter settings screen.  
For details on how to display the service adapter settings screen, see 3.3.1(4) *Displaying the service adapter settings screen*.
2. Edit [Service name] and [Service ID] of the service component control information, if necessary.
3. Click the [Add] button of the service component control information and specify the definition pattern and the operation name.
4. Select [Synchronous] from the drop-down list of [Communication model] of the operation information.
5. Perform steps from 6. to 17. for a request message.
6. Perform the following operation:
  - When you specify a standard message format**  
Perform steps from 7. to 10. and proceed to step 11.
  - When you do not specify a standard message format**  
Proceed to step 11.
7. Check the [Use] check box of a standard message.
8. Specify [Format ID] of a standard message.

9. Click the [Browse] button of a standard message and specify a standard message format in [Message format].  
For details on the message format that can be specified, see 2.6.5 *Scoping of XML schema* in the manual *Service Platform Basic Development Guide*.
10. Click the [Display] button of a standard message.  
If the message format is XML, a standard message format is displayed. Confirm the specified standard message format, if necessary.
11. Specify [Format ID] of service component message.
12. Confirm that the following message format is set as per the definition pattern specified in step. 3.  
**If the definition pattern is "XML reading" or "Binary reading"**  
Service component of request message: adpff\_read.xsd  
Service component of response message: None  
**If the definition pattern is "XML writing" or "Binary writing"**  
Service component of request message: None  
Service component of response message: adpff\_result.xsd
13. Click the [Display] button of service component message.  
If the message format is XML, the service component message format is displayed. Confirm the specified service component message format, if necessary.
14. Perform the following operation:  
If you check the [Use] check box of a standard message  
Perform steps from 15. to 17. And proceed to step 18.  
If you do not check the [Use] check box of a standard message  
Proceed to step 18.
15. Enter the file name of the data transformation definition.
16. Click the [Edit] button.  
The data transformation definition screen is displayed. When defining for the first time, the [Select root element] dialog box is displayed.
17. Map the contents of a standard message with that of a service message.
18. Perform steps from 6. to 17. even for a response message.
19. Click the service adapter definition (details) tab.  
The service adapter settings screen (details) is displayed.
20. Confirm the name of the service adapter (EJB-JAR file).
21. Confirm that the following utility class is added to the utility class (JAR file):
  - adpffpc.jar (Protocol converter archive file)
22. Confirm the definition contents, select [File] - [Save] from Eclipse menu and save the definition contents.

#### (4) Creating the file adapter execution environment property file

Create the file adapter execution environment property file using a sample file. Storage location of the sample file is as follows:

```
Installation directory of the service platform\CSC\custom-adapter\File\config\templates\adpff_exe.properties
```

The procedure for creating the file adapter execution environment property file is as follows:

1. Copy the sample file and store it in the following directory:  

```
Installation directory of the service platform\CSC\custom-adapter\File\config
```
2. Change the name of the sample file to <Service ID>.properties.
3. Edit and save the definition contents.

The file adapter execution environment property file is reflected in the execution environment, when you start a file adapter. When changing the contents of the file adapter execution environment property file, stop the file adapter, and then perform the operation. If you restart the file adapter, the changed contents are reflected in the execution environment.

For details on the file adapter execution environment property file, see *File adapter execution environment property file* in the manual *Service Platform Reference Guide*.

### 3.3.8 Defining an Object Access adapter

INTENTIONALLY DELETED

### 3.3.9 Defining Message Queue adapters

The method of defining Message Queue adapters is described here.

#### (1) Creating the message format definition file

There are 2 types of messages in Message Queue adapters, namely request message and response message. The contents of each of these messages are described here.

For details on the procedure for creating a message format, see *4. Creating a Message Format* in the manual *Service Platform Basic Development Guide*.

You can specify the XML format message or the binary format message in request and response messages of Message Queue adapters.

When sending a message in the XML format, you can specify only an XML format message. When sending a binary message, you can specify only a binary format message. Also, when receiving an XML format message, you can specify only an XML format message. When receiving a binary message, you can specify only a binary format message.

#### ! Important note

For the user data of the sent message or the received message, specify more than 1 byte. If length of the user data is 0 byte, an error occurs.

#### (a) Request message

A message used to request sending of a message from a calling Message Queue adapter to the transmission queue and receiving of a message from the reception queue, is called as a request message.

The information to be sent as a request message is described here.

#### Queue operation information

The queue operation information is the information required to access a queue. The following table describes the contents of the queue operation information:

Table 3–26: Queue operation information

Information	Description
Transmission queue-specific information, reception queue-specific information	Specify either the transmission queue or the reception queue to be accessed. Specify the name of resource-env-ref mentioned in the <resource-env-ref-name> tag, which is present in the <resource-env-ref> tag of HITACHI Application Integrated Property File.
Message correlation identifier	Specify this identifier, when receiving a message matching with the message correlation identifier, in case of a request to receive a message or a request to browse and receive a message. Specify the message correlation identifier, which is set at the time of the message transmission response within 24 bytes as a value. For details on the format of the message correlation identifier, see the description related to the method of generating message correlation identifier in <i>2.10.2 Access to message queue using Message Queue adapter</i> in the manual <i>Service Platform Overview</i> .

**Sent message**

A sent message is the user data to be specified in a request message, when requesting to send the XML format or binary format messages.

**(b) Contents of a request message**

Mention only the required information in a request message, from among the information regarding the existence of the transmission queue-specific information and the reception queue-specific information, existence of the specifying message correlation identifier, and the sent message format (binary format or XML format).

- **For a request to send a message**

The following table describes the request message format when specifying the transmission queue-specific information in the XML format:

Table 3–27: Format of the transmission request message in the XML format (when the destination queue-specific information is specified)

Information	Tag name in a message	Specified value	Mandatory
Root element	SendRequest	-	Y
Header information	mqheader	-	Y
Transmission queue-specific information	sndresname	Transmission queue-specific information <sup>#</sup>	Y
Sent message	userdata	Optional to user	Y

Legend:

Y: Must be specified.

-: There is no specified value.

Note#

If you omit the `<res_name>` tag of Message Queue adapter communication-configuration definition, you must specify this transmission queue-specific information. If you specify the `<res_name>` tag of Message Queue adapter communication-configuration definition, this specification is ignored and the specified value of Message Queue adapter communication-configuration definition is used.

The following table describes the request message format, when you do not specify the transmission queue-specific information in the XML format:

Table 3–28: Format of the transmission request message in the XML format (when the destination queue-specific information is not specified)

Information	Tag name in a message	Specified value	Mandatory
Root element	SendRequest	-	Y
Header information	mqheader	-	Y
Sent message	userdata	Optional to user	Y

Legend:

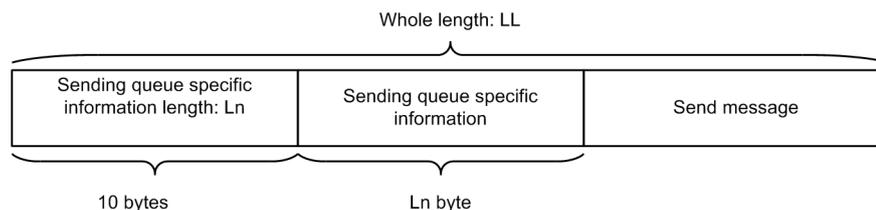
Y: Must be specified.

-: There is no specified value.

The case of specifying the transmission queue-specific information in the binary format is described here.

When sending a binary message, the request message format will be the binary format. The following figure shows the structure of a request message in the binary format:

Figure 3–17: Structure of the transmission request message in the binary format



The entire length of a message must satisfy the following relation:

$$LL \geq 10 + Ln$$

The following table describes the information required for a request message, when sending a binary message:

Table 3–29: Format of the transmission request message in the binary format (when the destination queue-specific information is specified)

Information	Tag name in a message	Specified value	Mandatory
Length of transmission queue-specific information	Specify the byte length in the transmission queue-specific information by using a decimal number and a string	10 bytes	Y
Transmission queue-specific information	Transmission queue-specific information (string)	Optional	Y
Sent message	Optional to user	Optional	Y

Legend:

Y: Must be specified.

The following table describes the request message format, when you do not specify the transmission queue-specific information in the binary format:

Table 3–30: Format of the transmission request message in the binary format (when the destination queue-specific information is not specified)

Information	Tag name in a message	Specified value	Mandatory
Sent message	Optional to user	Optional	Y

Legend:

Y: Must be specified.

- **For a request to receive a message**

You cannot specify a request to receive a message in the binary format.

The following table describes the format of a request to receive a message in the XML format:

Table 3–31: Format of a request to receive a message in the XML format

Information	Tag name in a message	Specified value	Mandatory
Root element	RecvRequest	-	Y
Header information	mqheader	-	Y
Reception queue-specific information	rcvresname	Reception queue-specific information <sup>#1</sup>	N
Message correlation identifier	correlationid	Correlation identifier of a received message <sup>#2</sup>	N

Legend:

Y: Must be specified.

### 3. Defining Adapters

N: Can be omitted.

-: There is no specified value.

**Note#1**

If you omit the <res\_name> tag of Message Queue adapter communication-configuration definition, you must specify this reception queue-specific information. If you specify the <res\_name> tag of Message Queue adapter communication-configuration definition, this specification is ignored, and the specified value of Message Queue adapter communication-configuration definition is used.

**Note#2**

If the specified value of the <id\_type> tag of Message Queue adapter communication-configuration definition is Type2, you must specify this message correlation identifier. If you omit the <id\_type> tag of Message Queue adapter communication-configuration definition or if you specify Type1, this specification is ignored.

**• For a request to browse and receive a message**

You cannot specify a request to browse and receive a message in the binary format.

The following table describes the format of a request message used to browse and receive a message in the XML format:

**Table 3–32: Format of a request message used to browse and receive a message in the XML format**

Information	Tag name in a message	Specified value	Mandatory
Root element	BrowseRequest	-	Y
Header information	mqheader	-	Y
Reception queue-specific information	rcvresname	Reception queue-specific information <sup>#1</sup>	N
Message correlation identifier	correlationid	Correlation identifier of the received message <sup>#2</sup>	N

**Legend:**

Y: Must be specified.

N: You can omit this information.

-: There is no specified value.

**Note#1**

If you omit the <res\_name> tag of Message Queue adapter communication-configuration definition, you must specify this reception queue-specific information. If you specify the <res\_name> tag of Message Queue adapter communication-configuration definition, this specification is ignored, and the specified value of Message Queue adapter communication-configuration definition is used.

**Note#2**

If the specified value of the <id\_type> tag of Message Queue adapter communication-configuration definition is Type2, you must specify this message correlation identifier. If you omit the <id\_type> tag of Message Queue adapter communication-configuration definition or specify Type1, this specification is ignored.

**• For a request to send and receive messages**

The following table describes the request message format used when specifying the transmission queue-specific information and the reception queue-specific information in the XML format:

**Table 3–33: Format of the request to send and receive messages in the XML format (when the transmission and reception queue-specific information is specified)**

Information	Tag name in a message	Specified value	Mandatory
Root element	SndRcvRequest	-	Y
Header information	mqheader	-	Y
Transmission queue-specific information	sndresname	Transmission queue-specific information <sup>#</sup>	Y
Reception queue-specific information	rcvresname	Reception queue-specific information <sup>#</sup>	Y
Sent message	userdata	Optional	Y

Legend:

- Y: Must be specified.
- : There is no specified value.

Note#

If you omit the `<res_inf>` tag of Message Queue adapter communication-configuration definition, you must specify this transmission queue-specific information and reception queue-specific information. If you specify the `<res_inf>` tag of Message Queue adapter communication-configuration definition, this specification is ignored, and the specified value of Message Queue adapter communication-configuration definition is used.

The following table describes the request message format used when you do not specify the transmission queue-specific information and the reception queue-specific information in the XML format:

Table 3–34: Format of the request to send and receive messages in the XML format (when the transmission and reception queue-specific information is not specified)

Information	Tag name in a message	Specified value	Mandatory
Root element	SndRcvRequest	-	Y
Header information	mqheader	-	Y
Sent message	userdata	Optional	Y

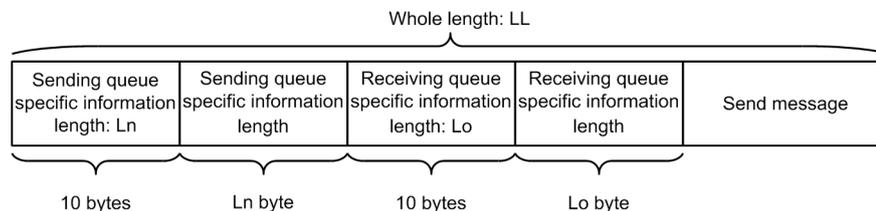
Legend:

- Y: Must be specified.
- : There is no specified value.

The case of specifying transmission queue and reception queue-specific information in the binary format is described here.

When sending and receiving a binary message, the request message will be in the binary format. The following figure shows the structure of a request message in the binary format:

Figure 3–18: Structure of the message transmission and reception request in the binary format



The entire length of a message must satisfy the following relation:

$$LL \geq 10 + Ln + 10 + Lo$$

The following table describes the information required for a request message when sending and receiving a binary message:

Table 3–35: Format of the transmission request message in the binary format (when the destination queue and the reception queue-specific information is specified)

	Tag name in a message	Specified value	Mandatory
Length of transmission queue-specific information	Specify the byte length of the transmission queue-specific information using the decimal number and string	10 bytes	Y
Transmission queue-specific information	Transmission queue-specific information (string)	Optional	Y
Length of receive-queue specific information	Specify the byte length of the reception queue-specific information using the decimal number and string	10 bytes	Y

	Tag name in a message	Specified value	Mandatory
Transmission queue-specific information	Reception queue-specific information (string)	Optional	Y
Sent message	Optional to user	Optional	Y

Legend:

Y: Must be specified.

The following table describes the request message format used when you do not specify the transmission queue and the reception queue-specific information in the binary format:

Table 3–36: Format of the transmission request message in the binary format (when the destination queue and the reception queue-specific information is not specified)

Information	Tag name in a message	Specified value	Mandatory
Sent message	Optional to user	Optional	Y

Legend:

Y: Must be specified.

(c) Response message

A message that is returned to the invoker of Message Queue adapter is called as response message. The information returned as a response message is described here.

**Received message**

Specifies a user data that is stored in a response message, when receiving a message in the XML format or the binary format.

**Result of message transmission**

The result of the transmission process is stored in a response message, when sending a message in the XML format or the binary format. If the sending process fails, Message Queue adapter notifies an exception to the custom adapter development framework and does not return a response message. When a message is sent successfully, the result of the message transmission is stored in a response message. Therefore, return value of a response message will surely be "OK" at the time of sending a message.

**Message correlation identifier**

Specifies a message correlation identifier set in JMSCorrelationID header field, when sending a message in XML format or the binary format.

(d) Contents of a response message

The contents of a response message differ according to the format of a received message (binary format or XML format).

Note that you cannot specify a response for message transmission in the binary format.

The format of the transmission response message in the XML format is as follows:

• **In case of a response message for a request to send a message**

The following table describes the format of the transmission response message in the XML format:

Table 3–37: Format of the transmission response message in the XML format

Information	Tag name in a message	Return value
Root element	SendResponse	-
Result of message transmission	status	OK
Message correlation identifier	correlationid	Message correlation identifier <sup>#</sup>

Legend:

-: There is no return value.

Note#

The message correlation identifier is returned, when you specify Type2 in the `<id_type>` of Message Queue adapter communication-configuration definition.

- **In case of a response message for a request to receive a message**

The following table describes the format of a response message for receiving messages in the XML format:

Table 3–38: Format of a response message for receiving messages in the XML format

Information	Tag name in a message	Return value
Received message	Optional to user	Optional to user

The following table describes the format of a response message for receiving messages in the binary format:

Table 3–39: Format of a response message for receiving messages reception in the binary format

Information	Tag name in a message
Received message	Optional to user

- **In case of a response message for a request to browse and receive a message**

The following table describes the format of a response message for a request to browse and receive a message in XML format:

Table 3–40: Format of a response message for a request to browse and receive a message in XML format

Information	Tag name in a message	Return value
Received message	Optional to user	Optional to user

The following table describes the response message for a request to browse and receive a message in the binary format:

Table 3–41: Format of a response message for a request to browse and receive a message in the binary format

Information	Tag name in a message
Received message	Optional to user

- **Response message for a request to send and receive a message**

The following table describes the format of a response message for a request to send and receive a message in the XML format:

Table 3–42: Format of a response message for a request to send and receive a message in the XML format

Information	Tag name in a message	Return value
Received message	Optional to user	Optional to user

The following table describes the format of a response message for a request to send and receive a message in the binary format:

Table 3–43: Format of a response message for a request to send and receive a message in the binary format

Information	Tag name in a message
Received message	Optional to user

## (e) Creating a format definition file

From among the various message formats, the message format set up in a service component is the message format of a Message Queue adapter. The method of creating the message format of a Message Queue adapter is described here.

**Request message in XML format**

To create an XML format definition file for a request message, the user uses either the template file or the file provided with the product. In other cases, the user creates this definition file on his own.

**When using XML format definition template file**

- Specify the transmission queue in a request to send a message.
- Specify the transmission queue and the reception queue in a request to send and receive a message.

**When using the file provided with the product**

- Specify the reception queue and the message correlation identifier in a request to receive a message.
- Specify the reception queue and the message correlation identifier in a request to browse and receive a message.

**When the user creates the definition file on his own**

When the case is other than the above-mentioned case of using the XML format definition template file and the case of using the file provided with the product, the user must create the definition file on his own.

The following figure gives a description on the XML format definition template file and customization:

Figure 3–19: Template file for a request to send a message (adpmq\_snd\_request.xsd)

```

1  <?xml version="1.0" encoding="UTF-8"?>
2  <!-- All Rights Reserved. Copyright (C) 2008, Hitachi, Ltd. -->
3  <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
4      xmlns:prefix -->= <!-- Define targetNamespace. -->
5      targetNamespace= <!-- Define targetNamespace. -->
6
7
8
9
10 <xsd:element name="SendRequest" type="prefix:format"/>
11 <xsd:complexType name="format">
12 <xsd:sequence>
13 <xsd:element name="mqheader" type="prefix:mqheader"/>
14 <xsd:element name="userdata" type="prefix:userdata"/>
15 </xsd:sequence>
16 </xsd:complexType>
17
18 <xsd:complexType name="mqheader">
19 <xsd:sequence>
20 <xsd:element name="sndresname" type="xsd:string" minOccurs="0"/>
21 </xsd:sequence>
22 </xsd:complexType>
23
24 <xsd:complexType name="userdata">
25 <!-- Define XML schema of the user data. -->
26 </xsd:complexType>
27 </xsd:schema>

```

Legend:

: Locations to be edited by user

Table 3–44: Contents of the template file for a request to send a message

Line number	Description
4 <sup>th</sup> line	Indicates name space declaration of this template file. The user declares any prefix in the value of targetName space attribute specified in the 6 <sup>th</sup> line.

Line number	Description
6 <sup>th</sup> line	Indicates the target name space of this template file. The user specifies a unique target name space in the value of targetName space attribute.
8 <sup>th</sup> line	When importing the XML schema information for the user data on the 25 <sup>th</sup> line, mention the target name space of the schema to be imported in the 5 <sup>th</sup> line and mention import statement of the schema to be imported in the 8 <sup>th</sup> line.
10 <sup>th</sup> line~16 <sup>th</sup> line	Indicates declaration of SendRequest element. These lines contain mqheader element and userdata element which becomes the root element of a request message. The user describes the prefix declared in the 4 <sup>th</sup> line as the prefix of mqheader element and userdata element.
18 <sup>th</sup> ~22 <sup>nd</sup> line	Indicates declaration of mqheader element.
24 <sup>th</sup> line	Indicates declaration of userdata element.
25 <sup>th</sup> line	<p>Describes the XML schema information for the user data.</p> <p>Description method</p> <p>There are 2 types of description methods. One is to describe directly in this file and another is to create a separate file for describing the XML schema information for the user data and to import the description from this file. When importing the description, mention the target name space of the schema to be imported in the 5<sup>th</sup> line and the import statement of the schema to be imported in the 8<sup>th</sup> line.</p> <p>Notes</p> <p>When specifying the XML schema information for the user data, you should not declare multiple elements in the child element of userdata element. You must declare only 1 child element of the userdata element, since the child element of userdata element becomes the root element, when an XML file is to be output.</p>

Figure 3–20: Format definition template file for a request to send and receive a message (adpmq\_sndrcv\_request.xsd)

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <!-- All Rights Reserved. Copyright (C) 2008, Hitachi, Ltd. -->
3 <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
4     xmlns:prefix -->= <!-- Define targetNamespace. -->
5     targetNamespace= <!-- Define targetNamespace. -->
6
7
8
9
10 <xsd:element name="SndRcvRequest" type="prefix:format"/>
11 <xsd:complexType name="format">
12     <xsd:sequence>
13         <xsd:element name="mqheader" type="prefix:mqheader"/>
14         <xsd:element name="userdata" type="prefix:userdata"/>
15     </xsd:sequence>
16 </xsd:complexType>
17
18 <xsd:complexType name="mqheader">
19     <xsd:sequence>
20         <xsd:element name="sndresname" type="xsd:string" minOccurs="0"/>
21         <xsd:element name="rcvresname" type="xsd:string" minOccurs="0"/>
22     </xsd:sequence>
23 </xsd:complexType>
24
25 <xsd:complexType name="userdata">
26     <!-- Define XML schema of the user data. -->
27 </xsd:complexType>
28 </xsd:schema>

```

Legend:

: Locations to be edited by user

Table 3–45: Contents of the template file used for a request to send and receive a message

Line number	Description
4 <sup>th</sup> line	Indicates name space declaration of this template file. The user declares any prefix in the value of targetName space attribute specified in the 6 <sup>th</sup> line.
6 <sup>th</sup> line	Indicates the target name space of this template file. The user specifies a unique target name space in the value of targetName space attribute.
8 <sup>th</sup> line	When importing the XML schema information for the user data on the 26 <sup>th</sup> line, mention the target name space of the schema to be imported in the 5 <sup>th</sup> line and mention import statement of the schema to be imported in the 8 <sup>th</sup> line.
10 <sup>th</sup> line~16 <sup>th</sup> line	Indicates declaration of SndRcvRequest element. These lines contain mqheader element and userdata element which becomes the root element of a request message. The user describes the prefix declared in the 4 <sup>th</sup> line as the prefix of mqheader element and userdata element.
18 <sup>th</sup> line~23 <sup>rd</sup> line	Indicates declaration of mqheader element.
21 <sup>st</sup> line	Indicates the specified element of reception queue-specific information. This specification location differs from the template file for a request to send a message.
25 <sup>th</sup> line	Indicates declaration of userdata element.
26 <sup>th</sup> line	Describes the XML schema information for the user data.  Description method There are 2 types of description methods. One is to describe directly in this file and another is to create a separate file for describing the XML schema information for the user data and to import the description from this file. When importing the description, mention the target name space of the schema to be imported in the 5 <sup>th</sup> line and the import statement of the schema to be imported in the 8 <sup>th</sup> line.  Notes When specifying the XML schema information for the user data, you should not declare multiple elements in the child element of userdata element. You can declare only 1 child element of userdata element, since the child element of userdata element becomes the root element, when an XML file is to be output.

**Response message in the XML format**

To create an XML format definition file for a response message, the user either uses the format definition file provided with the product or creates this definition file on his own.

**When using the file provided with the product**

Use the file provided with the product when the following conditions are true:

- When you send a response for message transmission.
- When you receive the fault response information message in response to receiving a message reception, response to browsing and receiving a message, or response for sending and receiving messages.

**When the user creates the XML format definition file for a response message on his own**

Apart from the above-mentioned case of using the file provided with the product, in other cases, the user should create the XML format definition file for a response message on his own.

**Request message in the binary format**

The user should either edit the binary format definition file for a request message on the binary format definition screen using the template file provided with the product or he should create this file on his own.

**When using the binary format definition template file**

- When you specify the transmission queue-specific information in a request to send a message
- When you specify the transmission queue-specific information and the reception queue-specific information in a request to send and receive a message.

**When the user creates a binary format definition file for a request message on his own**

When the case is other than the above-mentioned case of using the file provided with the product, the user should create the binary format definition file for a request message on his own.

**Response message in the binary format**

The following are the cases with respect to binary format definition file for a response message. However, the user should create the binary format definition file on his own.

- A binary message is to be received in the response for receiving messages.
- A binary message is to be received in the response for browsing and receiving a message.
- A binary message is to be received in the response for sending and receiving a message.

## (2) Defining data transformation

Set the message format definition file of the transformation source and the message format definition file of the transformation destination on the data transformation (mapping) definition screen and define the data transformation.

For details on how to define data transformation, see 6. *Defining Data Transformation* in the manual *Service Platform Basic Development Guide*.

## (3) Operations to be performed on the service adapter settings screen

The procedure for defining a Message Queue adapter is described here. For details on the setting contents, see 3.3.15(7) *For Message Queue adapters*.

1. Display the service adapter settings screen.

For details on how to display the service adapter settings screen, see 3.3.1(4) *Displaying the service adapter settings screen*.

2. Edit [Service name] and [Service ID] of the service component control information, if necessary.

3. Click the [Add] button of the service component control information and specify the definition pattern, operation name, and field name#.

The operation name must match with the operation name of Message Queue adapter communication-configuration definition (op\_name).

Note# Specify these details when the definition pattern is not "Send a message".

4. Select "Synchronous" from the drop-down list of "Communication model" of the operation information.

5. For a request message, perform steps from 6. to 17.

6. Perform the following operations:

**When you specify the standard message format**

Perform steps from 7. to 10. And proceed to step 11.

**When you do not specify the standard message format**

Proceed to step 11.

7. Check the [Use] check box for the standard message.

8. Specify [Format ID] of the standard message.

9. Click the [Browse] button of the standard message, and then specify the standard message format in [Message format].

For details on the message format that can be specified, see 2.6.5 *Scoping of XML schema* in the manual *Service Platform Basic Development Guide*.

10. Click the [Display] button of the standard message.

If the message format is XML, the standard message format is displayed. Confirm the specified standard message format, if necessary.

11. Specify [Format ID] of the service component message.

12. Confirm that the following message format is set as per the definition pattern specified in step 3:

**If the definition pattern is "Send a message"**

Service component of a request message: None

Service component of a response message: adpmq\_snd\_response.xsd

Fault message: None

**If the definition pattern is "Receive a message"**

Service component of a request message: adpmq\_rcv\_request.xsd

Service component of a response message: None

Fault message: adpmq\_fault\_response.xsd

**If the definition pattern is "Browse and receive a message"**

Service component of a request message: adpmq\_browse\_request.xsd

Service component of a response message: None

Fault message: adpmq\_fault\_response.xsd

**If the definition pattern is "Send and receive a message"**

Service component of a request message: None

Service component of a response message: None

Fault message: adpmq\_fault\_response.xsd

13. Click the [Display] button of a service component message.

The service component message format is displayed. Confirm the service component message format, if necessary.

14. Perform the following operations:

**If you check the [Use] check box for the standard message**

Perform the steps from 15. to 17. and proceed to step 18.

**If you do not check the [Use] check box of the standard message**

Proceed to step 18.

15. Enter the file name of the data transformation definition.

16. Click the [Edit] button.

The data transformation definition screen is displayed. When defining data transformation for the first time, the [Select root element] dialog box is displayed.

17. Map the contents of a standard message with the contents of a service message.

18. Perform steps from 6. to 17. even for a response message.

19. Click the service adapter settings (details) tab.

The service adapter settings screen (details) is displayed.

20. Confirm the names of the service adapter (EJB-JAR file) and the utility class (JAR file).

Confirm that the name of the service adapter (EJB-JAR file) is "cscmsg\_adpejb.jar" and the name of the utility class (JAR file) is "adpmqpc.jar".

21. Confirm that the following files are added in the self-defined file:

- customadapter\_properties.xml (Message Queue adapter environment-definition file)
- adpmq\_communication (Message Queue adapter communication-configuration definition file)
- cscadapter\_property.xml (HITACHI Application Integrated Property File)

#### (4) Editing the Message Queue adapter environment-definition file

Define the information required for the environment definition of a Message Queue adapter in the Message Queue adapter environment-definition file in XML format.

The procedure for editing the Message Queue adapter environment-definition file is as follows:

1. Select "customadapter\_properties.xml" in "Self-defined file" of the service adapter settings screen, and then click the [Edit] button.

The editor required for editing the Message Queue adapter environment-definition file is started.

2. Edit the Message Queue adapter environment-definition file using the editor.

For details on the Message Queue adapter environment definition file, see *Message Queue adapter environment definition file* in the manual *Service Platform Reference Guide*.

3. From the Eclipse menu select [File] - [Save], and then save the definition contents.

## (5) Editing the Message Queue adapter communication-configuration definition file

Define the information required for the communication-configuration definition of a Message Queue adapter in the Message Queue adapter communication-configuration definition file in the XML format.

The procedure to edit the Message Queue adapter communication-configuration definition file is as follows:

1. Select "adpmq\_communication.xml" in "Self-defined file" of the service adapter settings screen (details), and then click the [Edit] button.  
The editor required for editing the Message Queue adapter communication-configuration definition file is started.
2. Edit the Message Queue adapter communication-configuration definition file using the editor.  
For details on the Message Queue adapter communication-configuration definition file, see *Message Queue adapter communication-configuration definition file* in the manual *Service Platform Reference Guide*.
3. From the Eclipse menu Select [File] - [Save] , and then save the definition contents.

## (6) Editing HITACHI Application Integrated Property File

The method of editing the HITACHI Application Integrated Property File is described here.

### (a) Editing procedure

The procedure for editing the HITACHI Application Integrated Property File is as follows:

1. Select "cscadapter\_property.xml" in "Self-defined file" in the Service adapter settings screen (details), and click the "Edit" button.  
Start the editor used for editing HITACHI Application Integrated Property File.
2. Edit the HITACHI Application Integrated Property File using the editor.  
For details on the setup contents, see (b) *Definition details*.
3. From the Eclipse menu, select [File] - [Save], and then save the definition contents.

### (b) Definition details

The contents of the HITACHI Application Integrated Property File are as follows:

---

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hitachi-application-all-property PUBLIC
'-//Hitachi, Ltd.//DTD Application All Property 7.6//EN' 'http://localhost/hitachi-
application-all-property_7_6.dtd'>
<!-- All Rights Reserved. Copyright (C) 2008, Hitachi, Ltd. -->

<hitachi-application-all-property>
  <hitachi-application-property>
    <lookup-name>CSCMQ</lookup-name>
    <security-prop>
      <security-method>no_security_for_methods_without_roles</security-method>
    </security-prop>
    <managed-by-ctm>>false</managed-by-ctm>
  </hitachi-application-property>
  <ejb-jar>
    <hitachi-ear-jar-property>
      <display-name>CSCMsgServiceAdapter</display-name>
    </hitachi-ear-jar-property>
    <hitachi-session-bean-property>
      <display-name>CSCMsgServiceAdapterEJB</display-name>
      <session-type>Stateless</session-type>
      <transaction-type>Container</transaction-type>
      <resource-ref>
        <res-ref-name>jms/resource_adapter_specific_information</res-ref-name>
        <res-type>javax.jms.QueueConnectionFactory</res-type>
        <res-auth>Container</res-auth>
        <res-sharing-scope>Unshareable</res-sharing-scope>
        <linked-to>name displayed for resource adapter</linked-to>
      </resource-ref>
      <resource-env-ref>
        <resource-env-ref-name>transmission_queue_specific_information</resource-env-ref-name>
        <resource-env-ref-type>javax.jms.Queue</resource-env-ref-type>
        <linked-adminobject>
          <resourceadapter-name> name displayed for resource adapter</resourceadapter-name>
          <adminobject-name>name of the admin object</adminobject-name>
        </linked-adminobject>
      </resource-env-ref>
    </hitachi-session-bean-property>
  </ejb-jar>
</hitachi-application-all-property>
```

---

---

```

    </linked-adminobject>
  </resource-env-ref>
  <container-transaction>
    <description></description>
    <method>
      <description></description>
      <method-intf></method-intf>
      <method-name>*</method-name>
    </method>
    <trans-attribute>Required</trans-attribute>
  </container-transaction>
  <session-runtime>
    <lookup-name>CSCMsgServiceAdapterEJB</lookup-name>
    <optional-name></optional-name>
    <maximum-sessions>0</maximum-sessions>
    <stateless>
      <pooled-instance>
        <minimum>minimum number of instances</minimum>
        <maximum>maximum number of instances</maximum>
      </pooled-instance>
    </stateless>
  </session-runtime>
</hitachi-session-bean-property>
</ejb-jar>
</hitachi-application-all-property>

```

---

Setup value of the definition is described here.

- **Application attribute <hitachi-application-property>**

<lookup-name>

Specify "CSCMQ" as the name to be used when looking up EJB from the client.

<security-method>

Specify "no\_security\_for\_methods\_without\_roles" as the method for managing security.

<managed-by-ctm>

Specify "false (Do not link to CTM)" for linkage to CTM.

- **EJB-JAR attribute <hitachi-ejb-jar-property>**

<display-name>

Specify "CSCMsgServiceAdapter" as the name to be displayed for EJB-JAR.

- **Session Bean attribute <hitachi-session-bean-property>**

<display-name>

Specify "CSCMsgServiceAdapterEJB" as the name to be displayed for Session Bean.

<session-type>

Specify "Stateless" as the Session Bean type.

<transaction-type>

Specify "Container" as Transaction Management Type.

<res-ref-name>

Specify resource adapter-specific information. Specify any name to identify a resource adapter for this specified value. Mention "jms/" in the beginning when specifying the name.

Set a value same as this element in the <res\_ref\_name> element of Message Queue adapter communication-configuration definition.

<res-type>

Specify a resource type. Specify "javax.QueueConnectionFactory" as the connection definition identifier (setup value of <connectionfactory-interface> tag) of the destination resource adapter.

<res-auth>

Specify "Container" as the authentication method.

<res-sharing-scope>

Specify "Unshareable" for the option to indicate whether the referenced resource adapter can be shared.

<linked-to>

Specify the name to be displayed for the resource adapter to be referenced.

Specify `<name to displayed for resource adapter>!<connection definition identifier>` for Outbound resource adapter.

`<Connection definition identifier >` is a value specified in the `<connectionfactory-interface>` tag of the deploy descriptor of a resource adapter (`ra.xml`).

`<res-env-ref-name>`

Specify the transmission queue-specific information or the reception queue-specific information. Specify any name to identify the destination queue for this specified value. Mention "jms/" in the beginning when specifying the name.

Set a value same as this element in `<res_name>`, `<snd_res_name>` or `<rcv_res_name>` elements of Message Queue adapter communication-configuration definition.

`<res-env-ref-type>`

Specify "javax.jms.Queue" as the class type of the value of a resource environment variable.

`<resourceadapter-name>`

Specify the name to be displayed for a resource adapter of a resource environment variable.

`<adminobjectname>`

Specify the name of an admin object. Specify a value same as `<adminobject-name>` of deploy descriptor (`ra.xml`) of a resource adapter.

`<trans-attribute>`

Specify "Required" as the transaction attribute assigned to a method.

`<lookup-name>`

Specify "CSCMsgServiceAdapterEJB" as the name to be used at the location where you look up EJB from a client.

`<maximum-sessions>`

Specify "0 (unlimited)" as maximum number of sessions.

`<minimum>`

Specify minimum number of instances in a pool.

`<maximum>`

Specify maximum number of instances in a pool.

For details on the HITACHI Application Integrated Property File, see the following sections:

- *9.2 Property settings as per HITACHI Application Integrated Property File* in the manual *Application Server Application Setup Guide*.
- *3.1 HITACHI Application Integrated Property File* in the manual *Application Server Application and Resource Definition Reference Guide*.

## (7) Creating Message Queue adapter runtime-environment property file

You must create Message Queue adapter runtime-environment property file for Message Queue adapter.

For details on the Message Queue adapter runtime-environment property file, see *Message Queue adapter runtime-environment property file* in the manual *Service Platform Reference Guide*.

## 3.3.10 Defining FTP adapters

This subsection describes how to define FTP adapters.

### (1) Creating a message format

For the message format definition file of FTP adapters, use the schema provided by the service platform. Therefore, you need not create the message format definition file.

The contents of the message format used by FTP adapters are described here.

The storage location of the file is "Installation directory of the service platform\CSC\custom-adapter\FTP\schema".

(a) Request message format for FTP adapters

The operation-wise request message format for FTP adapters is described here.

● **PUT operation**

The following table describes the request message format of PUT operation to be passed on to an FTP adapter from a business process. The file name is "ftpadp\_put\_request.xsd". The name of the name space is "http://www.hitachi.co.jp/soft/xml/cosminexus/csaftp/put\_request".

Table 3–46: Request message format (PUT operation)

Tag name	Occurrence count#1	Description
<request>	1 time	-
<host-ipaddr>	0 or 1 time	<p>Specifies the IP address or the host name of the FTP server to be connected.</p> <ul style="list-style-type: none"> <li>If the tag and value both are present connect is issued in the specified string.</li> <li>If the tag is present and the value is not present A system exception is to be returned.</li> <li>If the tag is not present The value given in FTP adapter runtime-environment property file becomes valid. If a value is not specified in FTP adapter runtime-environment property file, a system exception is to be returned.</li> </ul> <p>When using an OS from Windows Vista and Windows Server 2008 onward, if you specify loopback IP address (127.0.0.1 or localhost), a failure might occur in FTP connection. Specify the real IP address or the host name.</p>
<host-con-port>	0 or 1 time	<p>Specifies the port number for the control connection of FTP server.</p> <ul style="list-style-type: none"> <li>If the tag and value both are present Specify a port number within the range from 1 to 65535. If you specify any value other than this, a system exception is to be returned.</li> <li>If the tag is present and the value is not present, 21 is specified by default.</li> <li>If the tag is not present The value given in FTP adapter runtime-environment property file becomes valid.</li> </ul>
<file-name-charset>	0 or 1 time	<p>Specifies a tag that indicates the name of the character set to be used when sending and receiving information such as file names using the control connection between FTP servers. The specified value is either of the following values:</p> <p>UTF-8: Use "UTF-8" in the character set. MS932: Use "MS932" in the character set.</p> <ul style="list-style-type: none"> <li>In case of character set for which the tag is present and the value can be used It will be applied as the character set to be used when sending and receiving information such as file names.</li> <li>In case of the character set name for which the tag is present and the value cannot be used A system exception is to be returned.</li> <li>If the tag is present and the value is not present By default "UTF-8" is used.</li> <li>If the tag is not present The value given in FTP adapter runtime-environment property file becomes valid.</li> </ul>

Tag name	Occurrence count <sup>#1</sup>	Description
<ftps>	0 or 1 time	Specifies a tag indicating the settings related to FTPS. If the tag is not present, the value given in FTP adapter runtime-environment property file becomes valid for the entire setting related to FTPS.
<ftps-enable>	0 or 1 time	<p>Specifies a tag that indicates whether to use FTPS to connect to an FTP server. The specified value is either of the following values:</p> <p><code>true</code>: Use FTPS to connect to an FTP server.</p> <p><code>false</code>: Use normal FTP to connect to an FTP server.<sup>#2</sup></p> <p>When you use FTPS, the communication of control connection and data connection is encrypted. However, the destination FTP server must support the connection using FTPS. Even if you specify "true", when encryption of the communication of data connection is not set, the communication of data connection is also not encrypted.</p> <ul style="list-style-type: none"> <li>• If the tag is present and the value is "true" or "false" The specified value is applied.</li> <li>• If the tag is present and the value is other than "true" or "false" A system exception is to be returned.</li> <li>• If the tag is present and the value is not present By default "false" is used.</li> <li>• If the tag is not present The value given in FTP adapter runtime-environment property file becomes valid.</li> </ul>
<ftps-protocol-name>	0 or 1 time	<p>Specifies a tag that indicates the protocol for security communication to be used, when using FTPS to connect to an FTP server. The specified value is either of the following values:</p> <p><code>TLS</code>: Use TLS in the protocol for security communication.</p> <p><code>SSL</code>: Use SSL in the protocol for security communication.</p> <ul style="list-style-type: none"> <li>• If the tag is present and the value is "TLS" or "SSL" The specified value is applied.</li> <li>• If the tag is present and the value is other than "TLS" or "SSL" A system exception is to be returned.</li> <li>• If the tag is present and the value is not present "TLS" is used by default.</li> <li>• If the tag is not present The value given in FTP adapter runtime-environment property file becomes valid.</li> </ul> <p>This setting becomes valid only when using FTPS to connect to an FTP server.</p>
<ftps-implicit-mode>	0 or 1 time	<p>Specifies a tag that indicates whether to use Implicit mode of FTPS, when using FTPS to connect to an FTP server. The specified value is either of the following values:</p> <p><code>true</code>: Use Implicit mode.</p> <p><code>false</code>: Use Explicit mode.<sup>#2</sup></p> <ul style="list-style-type: none"> <li>• If the tag is present and the value is "true" or "false" The specified value is applied.</li> <li>• If the tag is present and the value is other than "true" or "false" A system exception is to be returned.</li> <li>• If the tag is present and the value is not present "false" is used by default.</li> <li>• If the tag is not present The value given in FTP adapter runtime-environment property file becomes valid.</li> </ul>

### 3. Defining Adapters

Tag name	Occurrence count <sup>#1</sup>	Description
<ftp-implicit-mode>	0 or 1 time	This setting becomes valid only when using FTPS to connect to an FTP server.
<ftp-data-con-secure>	0 or 1 time	<p>Specifies a tag that indicates whether to encrypt the communication of data connection, when using FTPS to connect to an FTP server. The specified value is either of the following values:</p> <p><code>true</code>: Encrypt the communication of data connection.</p> <p><code>false</code>: Do not encrypt the communication of data connection.</p> <ul style="list-style-type: none"> <li>• If the tag is present and the value is "true" or "false" The specified value is applied.</li> <li>• If the tag is present and the value is other than "true" or "false" A system exception is to be returned.</li> <li>• If the tag is present and the value is not present By default "true" is used.</li> <li>• If the tag is not present The value given in FTP adapter runtime-environment property file becomes valid.</li> </ul> <p>This setting becomes valid only when using FTPS to connect to an FTP server.</p>
<ftp-server-authentication>	0 or 1 time	<p>Specifies a tag that indicates whether to perform server authentication, when using FTPS to connect to an FTP server. The specified value is either of the following values:</p> <p><code>true</code>: Perform server authentication.</p> <p><code>false</code>: Do not perform server authentication.</p> <ul style="list-style-type: none"> <li>• If the tag is present and the value is "true" or "false" The specified value is applied.</li> <li>• If the tag is present and the value is other than "true" or "false" A system exception is to be returned.</li> <li>• If the tag is present and the value is not present "false" is used by default.</li> <li>• If the tag is not present The value given in FTP adapter runtime-environment property file becomes valid.</li> </ul> <p>This setting becomes valid only when using FTPS to connect to an FTP server. Note that when you specify "true", settings related to keystore are required. For details on how to set keystore, see <i>Appendix H.1 Secure connection using FTPS (FTP adapter)</i>.</p>
<ftp-user> <sup>#3</sup>	0 or 1 time	<p>Specifies a tag that indicates the login user name of the FTP server to be connected. Use it as an argument for <code>USER</code> command. The login user specified in this tag must have the permission to execute file operations.</p> <p>If this tag is set repetitively in the FTP adapter runtime-environment property file, the value of a request message becomes valid.</p> <ul style="list-style-type: none"> <li>• If the tag is present and a normal value is present "USER "Value"" is issued. If the value contains linefeed, a system exception is to be returned.</li> <li>• If the tag is present and the value is not present or if an invalid character is included in the value a system exception is to be returned.</li> <li>• If the tag is not present The value given in FTP adapter runtime-environment property file becomes valid. If a value is not specified in FTP adapter runtime-environment property file, a system exception is to be returned.</li> </ul>

Tag name	Occurrence count <sup>#1</sup>	Description
<ftp-acct> <sup>#3</sup>	0 or 1 time	<p>Specifies a tag that indicates the billing information of the FTP server to be connected.</p> <p>If this tag is set repetitively in FTP adapter runtime-environment property file, the value of a request message becomes valid.</p> <p>User for whom the billing information is required The following value is used as per the condition:</p> <ul style="list-style-type: none"> <li>• If the tag is present and the value is normal The contents of the tag are used in ACCT command. If a linefeed is included in the value, a system exception is to be returned.</li> <li>• If the tag is present and an invalid character is included in the value A system exception is to be returned.</li> <li>• If the tag is present and the value is not present "ACCT"" is issued.</li> <li>• If the tag is not present The value of ACCT specified in FTP adapter account definition file is used. If ACCT is not present in FTP adapter account definition file, "ACCT "" is issued.</li> </ul> <p>User for whom the billing information is not required Specification is ignored.</p>
<ftp-type>	0 or 1 time	<p>Specifies a tag that indicates the data type to be transferred (transmission mode). Use this tag as an argument in TYPE command. If this tag is set repetitively in FTP adapter runtime-environment property file, the value given in the request message becomes valid.</p> <p>The specified value is either of the following values: A: Transfer the data in ASCII format. I: Transfer the data in binary format.</p> <ul style="list-style-type: none"> <li>• If the tag is present and the value is either "A" or "I" The tag is used as an argument of TYPE command.</li> <li>• If the tag is present and the value is neither "A" nor "I" A system exception is to be returned.</li> <li>• If the tag is present and the value is not present TYPE command is not executed.<sup>#4</sup></li> <li>• If the tag is not present The value given in FTP adapter runtime-environment property file becomes valid. If a value is not specified in FTP adapter runtime-environment property file, TYPE command is not executed.<sup>#4</sup></li> </ul>
<ftp-mode>	0 or 1 time	<p>Specifies a tag that indicates transfer mode (compression mode). Use this tag as an argument in MODE command. If this tag is set repetitively in FTP adapter runtime-environment property file, the value given in the request message becomes valid.</p> <p>The specified value is either of the following values: S: Transfer the file without compressing it. C: Compress the file and transfer it.</p> <ul style="list-style-type: none"> <li>• If the tag is present and the value is either "S" or "C" The tag is used as an argument of MODE command.</li> <li>• If the tag is present and the value is neither "S" nor "C" A system exception is to be returned.</li> <li>• If the tag is present and the value is not present MODE command is not executed.</li> <li>• If the tag is not present</li> </ul>

### 3. Defining Adapters

Tag name	Occurrence count <sup>#1</sup>	Description
<ftp-mode>	0 or 1 time	The value given in FTP adapter runtime-environment property file becomes valid. If a value is not specified even in FTP adapter runtime-environment property file, MODE command is executed using default "S".
<transfer-type>	1 time	Specifies a tag that indicates the transfer type. Set either of the following values: STOR: Overwriting a file APPE: Adding information to a file <ul style="list-style-type: none"> <li>If the tag is present and the value is either "STOR" or "APPE" The tag is used as transfer type.</li> <li>If the tag is not present or the tag is present and the value is neither "STOR" nor "APPE" A system exception is to be returned.</li> </ul>
<request-id>	0 or 1 time	Specifies a tag that indicates the request ID created in FTP reception. <ul style="list-style-type: none"> <li>If the tag and the value both are present The tag is used as request ID.</li> <li>If the tag is not present or if the tag is present and the value is not present When using a working folder, a system exception is to be returned. This exception is ignored, when using a common folder.</li> </ul>
<local-folder>	1 time	Specifies a tag that indicates a local folder. <ul style="list-style-type: none"> <li>When common="true" is specified in tag attribute Use a common folder as a local folder and specify "Common folder definition name" in this tag. When you cannot acquire a common folder from the specified common folder definition name, a system exception is to be returned.</li> <li>When common="false" is specified in tag attribute Use a working folder as a local folder.</li> <li>If the common attribute is not present or the value is neither "true" nor "false" A system exception is to be returned.</li> </ul>
<local-file-name>	1 time	Specifies a tag that indicates the name of a local file. Specify a file, which is present just under the working folder or common folder. You can use a delimiting character such as slash (/) or back slash (\) only at the beginning of a file name. If you use a slash (/) at the beginning of a file name, slash (/) is ignored. Also, specify a file not having a symbolic link. <ul style="list-style-type: none"> <li>If the tag is present and the value follows the above-mentioned limitation The tag is used as file name.</li> <li>If the tag is present and the value does not follow the above-mentioned limitation A system exception is to be returned.</li> <li>If the tag is not present or if the tag is present and the value is not present A system exception is to be returned.</li> </ul>
<remote-path> <sup>#5</sup>	1 time	Specifies a tag that indicates the remote path name (name of the destination path at the time of transferring to FTP server). <ul style="list-style-type: none"> <li>If the tag and the value both are present The tag is used as remote path name.</li> <li>If the tag is not present or if the tag is present and the value is not present A system exception is to be returned.</li> </ul>
<ftp-commands-before>	0 or 1 time	Specifies a tag that indicates the FTP command and its argument to be executed prior to file transfer. <sup>#6</sup>

Tag name	Occurrence count <sup>#1</sup>	Description
<ftp-commands-before>	0 or 1 time	<p>When executing multiple commands, separate the commands in the sequence in which they will be executed using a semicolon (;), and then and set. (Example: MKD transdir;CWD transdir).</p> <ul style="list-style-type: none"> <li>If the tag and the value both are present The tag is used as the command to be executed prior to file transfer.</li> <li>If the tag is not present or the tag is present and the value is not present There is no command to be executed prior to file transfer.</li> </ul>
<ftp-commands-after>	0 or 1 time	<p>Specifies a tag that indicates the FTP command and its argument to be executed after file transfer.<sup>#6</sup></p> <p>When executing multiple commands, separate the commands in the sequence in which they will be executed using a semicolon (;) and set them (Example: RNFR oldfile.txt;RNTO newfile.txt).</p> <ul style="list-style-type: none"> <li>If the tag and the value both are present The tag is used as the command to be executed after file transfer.</li> <li>If the tag is not present or the tag is present and the value is not present There is no command to be executed after file transfer.</li> </ul>

## Legend:

-: There is no corresponding item.

## Note#1

If the specified occurrence count is exceeded, the operation is not guaranteed.

## Note#2

If you specify false in the value of the <ftps-enable> tag or the <ftps-implicit-mode> tag, when the connection destination FTP server is running in Implicit mode of FTPS, waiting status might appear after connecting to the FTP server till the timeout occurs.

The smallest value from among the following values is the duration till the timeout occurs:

- Value specified in ftpadp.control-con.connect.timeout property of FTP adapter runtime-environment property file
- Time-out value set in connection destination FTP server

If the time-out occurs as per the value specified in ftpadp.control-con.connect.timeout property of FTP adapter runtime-environment property file, after you perform retry operation for the count specified in ftpadp.control-con.retry.count property of FTP adapter runtime-environment property file, error of KDEK30407-E occurs.

If the time-out occurs as per the value set in the connection destination FTP server, error of KDEK30428-E occurs.

## Note#3

You can use only the following characters:

one-byte alphanumeric characters, one-byte space, exclamation mark (!), double-quotation mark ("), number sign (#), dollar sign (\$), percentage sign (%), ampersand (&), apostrophe ('), left parenthesis ( ( ), right parenthesis ( ) ), asterisk (\*), positive sign (+), comma (,), hyphen (-), period (.), slash (/), colon (:), semi-colon (;), left-angle bracket (<), right-angle bracket (>), equal sign (=), question mark (?), at sign (@), left-square bracket ( [ ), right-square bracket ( ] ), yen mark (\), circumflex accent (^), underscore (\_), accent grave (`), left curly bracket ( { ), right curly bracket ( } ), pipeline ( | ), wave dash (~)

## Note#4

The file is transferred in ASCII format by default as per the specifications of FTP protocol (RFC959).

## Note#5

If you specify a value same as the <transfer-path> tag of a request message of the FTP reception in the <remote-path> tag of FTP adapter, the path starts with a slash (/).

When you specify a relative path or when you want to specify a path in the absolute path format of Windows, if there is slash (/) at the beginning of the path, a problem occurs. In such cases, extract the string from 2nd character onward from the value of the <transfer-path> tag of the request message of the FTP reception using substring acquisition function (substr) in the data transformation activity of a business process and set that string in the <remote-path> tag of FTP adapter.

## Note#6

The command specified in the <ftp-commands-before> tag and the <ftp-commands-after> tag the <ftp-commands-after> tag is sent as-is to the FTP server. The FTP adapter is not affected by the impact of the execution of the command and the validity of the command option.

### 3. Defining Adapters

Therefore, if you specify the `OPTS UTF {ON|OFF}` command in the `<ftp-commands-before>` tag and the `<ftp-commands-after>` tag, you must specify the character set that is applied to the communication with FTP server in either of the following:

- `<file-name-charset>` tag of a request message
- `ftpdp.file-name.charset` property of FTP adapter runtime-environment property file

#### ● GET operation

The following table describes the request message format of GET operation to be passed to FTP adapter by a business process. File name is `"ftpdp_get_request.xsd"`. The name of the name space is `"http://www.hitachi.co.jp/soft/xml/cosminexus/csafp/get_request"`.

Table 3-47: Request message format (GET operation)

Tag name	Occurrence count#1	Description
<code>&lt;request&gt;</code>	1 time	-
<code>&lt;host-ipaddr&gt;</code>	0 or 1 time	<p>Specifies the IP address or the host name of the FTP server to be connected.</p> <ul style="list-style-type: none"> <li>• If the tag and the value both are present connect is issued using the specified string.</li> <li>• If the tag is present and the value is not present a system exception is to be returned.</li> <li>• If the tag is not present The value given in FTP adapter runtime-environment property file becomes valid. If a value is not specified in FTP adapter runtime-environment property file, a system exception is to be returned.</li> </ul> <p>When using an OS from Windows Vista and Windows Server 2008 onward, if you specify a loopback IP address (127.0.0.1 or localhost), a failure might occur in FTP connection. Specify the real IP address or the host name.</p>
<code>&lt;host-con-port&gt;</code>	0 or 1 time	<p>Specifies the port number for control connection of the FTP server.</p> <ul style="list-style-type: none"> <li>• If the tag and the value both are present Specify a port number within the range from 1 to 65535. If you specify any other number, a system exception is to be returned.</li> <li>• If the tag is present and the value is not present 21 is specified by default.</li> <li>• If the tag is not present The value given in FTP adapter runtime-environment property file becomes valid.</li> </ul>
<code>&lt;file-name-charset&gt;</code>	0 or 1 time	<p>Specifies a tag that indicates the name of the character set to be used when sending and receiving information such as file name using control connection with the FTP server. The specified value is either of the following values:</p> <p>UTF-8: Use "UTF-8" in the character set. MS932: Use "MS932" in the character set.</p> <ul style="list-style-type: none"> <li>• In case of a character set for which the tag is present and the value can be used It will be applied as the character set to be used when sending and receiving information such as file names.</li> <li>• In case of a character set name for which the tag is present and the value cannot be used A system exception is to be returned.</li> <li>• If the tag is present and the value is not present By default "UTF-8" is used.</li> <li>• If the tag is not present The value given in FTP adapter runtime-environment property file becomes valid.</li> </ul>

Tag name	Occurrence count <sup>#1</sup>	Description
<ftps>	0 or 1 time	Specifies a tag that indicates the settings related to FTPS. If the tag is not present, the value given in FTP adapter runtime-environment property file becomes valid for all settings related to FTPS.
<ftps-enable>	0 or 1 time	<p>Specifies a tag that indicates whether to use FTPS to connect to an FTP server. The specified value is either of the following values:</p> <p><code>true</code>: Use FTPS to connect to an FTP server.</p> <p><code>false</code>: Use a normal FTP to connect to an FTP server.<sup>#2</sup></p> <p>When using FTPS, the communication of control connection and data connection is encrypted. However, the destination FTP server must support the connection done by FTPS. Even if you specify <code>true</code>, when the encryption of communication of data connection is not set, even the communication of data connection is not encrypted.</p> <ul style="list-style-type: none"> <li>• If the tag is present and the value is <code>true</code> or <code>false</code> The specified value is applied.</li> <li>• If the tag is present and the value is other than <code>true</code> or <code>false</code> A system exception is to be returned.</li> <li>• If the tag is present and the value is not present <code>false</code> is used by default.</li> <li>• If the tag is not present The value given in FTP adapter runtime-environment property file becomes valid.</li> </ul>
<ftps-protocol-name>	0 or 1 time	<p>Specifies a tag that indicates the protocol for security communication to be used, when using FTPS to connect to an FTP server. The specified value is either of the following values:</p> <p><code>TLS</code>: Use TLS in the protocol for security communication.</p> <p><code>SSL</code>: Use SSL in the protocol for security communication.</p> <ul style="list-style-type: none"> <li>• If the tag is present and the value is <code>TLS</code> or <code>SSL</code> The specified value is applied.</li> <li>• If the tag is present and the value is other than <code>TLS</code> or <code>SSL</code> a system exception is to be returned.</li> <li>• If the tag is present and the value is not present <code>TLS</code> is used by default.</li> <li>• If the tag is not present The value given in FTP adapter runtime-environment property file becomes valid.</li> </ul> <p>This setting becomes valid only when using FTPS to connect to an FTP server.</p>
<ftps-implicit-mode>	0 or 1 time	<p>Specifies a tag that indicates whether to use Implicit mode of FTPS, when using FTPS to connect to an FTP server. The specified value is either of the following values:</p> <p><code>true</code>: Use Implicit mode.</p> <p><code>false</code>: Use Explicit mode.<sup>#2</sup></p> <ul style="list-style-type: none"> <li>• If the tag is present and the value is <code>true</code> or <code>false</code> The specified value is applied.</li> <li>• If the tag is present and the value is other than <code>true</code> or <code>false</code> A system exception is to be returned.</li> <li>• If the tag is present and the value is not present <code>false</code> is used by default.</li> <li>• If the tag is not present</li> </ul>

### 3. Defining Adapters

Tag name	Occurrence count <sup>#1</sup>	Description
<ftps-implicit-mode>	0 or 1 time	<p>The value given in FTP adapter runtime-environment property file becomes valid.</p> <p>This setting becomes valid only when using FTPS to connect to an FTP server.</p>
<ftps-data-con-secure>	0 or 1 time	<p>Specifies a tag that indicates whether to encrypt the communication of data connection, when using FTPS to connect to an FTP server. The specified value is either of the following values:</p> <p><code>true</code>: Encrypt the communication of data connection.</p> <p><code>false</code>: Do not encrypt the communication of data connection.</p> <ul style="list-style-type: none"> <li>• If the tag is present and the value is "true" or "false" The specified value is applied.</li> <li>• If the tag is present and the value is other than "true" or "false" A system exception is to be returned.</li> <li>• If the tag is present and the value is not present "true" is used by default.</li> <li>• If the tag is not present The value given in FTP adapter runtime-environment property file becomes valid.</li> </ul> <p>This setting becomes valid only when using FTPS to connect to an FTP server.</p>
<ftps-server-authentication>	0 or 1 time	<p>Specifies a tag that indicates whether to perform server authentication, when using FTPS to connect to an FTP server. The specified value is either of the following values:</p> <p><code>true</code>: Perform server authentication.</p> <p><code>false</code>: Do not perform server authentication.</p> <ul style="list-style-type: none"> <li>• If the tag is present and the value is "true" or "false" The specified value is applied.</li> <li>• If the tag is present and the value is other than "true" or "false" A system exception is to be returned.</li> <li>• If the tag is present and the value is not present "false" is used by default.</li> <li>• If the tag is not present The value given in FTP adapter runtime-environment property file becomes valid.</li> </ul> <p>This setting becomes valid only when using FTPS to connect to an FTP server. Note that you must perform settings related to keystore to specify "true". For details on how to set keystore, see <i>Appendix H.1 Secure connection using FTPS (FTP adapter)</i>.</p>
<ftp-user> <sup>#3</sup>	0 or 1 time	<p>Specifies a tag that indicates the login user name of FTP server to be connected. Use it as an argument of <code>USER</code> command. The login user specified in this tag must have permission to execute file operations.</p> <p>If this tag is set repetitively in FTP adapter runtime-environment property file, the value of a request message becomes valid.</p> <ul style="list-style-type: none"> <li>• If the tag is present and a normal value is present "USER "Value"" is issued. If the value contains linefeed, a system exception is to be returned.</li> <li>• If the tag is present and the value is not present, or if an invalid character is included in the value A system exception is to be returned.</li> <li>• If the tag is not present</li> </ul>

Tag name	Occurrence count <sup>#1</sup>	Description
<ftp-user> <sup>#3</sup>	0 or 1 time	The value given in FTP adapter runtime-environment property file becomes valid. If a value is not specified in FTP adapter runtime-environment property file, a system exception is to be returned.
<ftp-acct> <sup>#3</sup>	0 or 1 time	<p>Specifies a tag that indicates the billing information of the FTP server to be connected. If this tag is set repetitively in FTP adapter runtime-environment property file, the value of a request message becomes valid.</p> <p>User for whom the billing information is required The following value is used as per the condition:</p> <ul style="list-style-type: none"> <li>• If the tag is present and the value is normal The contents of the tag are used in ACCT command. If a linefeed is included in the value, a system exception is to be returned.</li> <li>• If the tag is present and an invalid character is included in the value A system exception is to be returned.</li> <li>• If the tag is present and the value is not present "ACCT "" " is issued.</li> <li>• If the tag is not present The value of ACCT specified in FTP adapter account definition file is used. If ACCT is not present in FTP adapter account definition file, "ACCT "" " is issued.</li> </ul> <p>User for whom the billing information is not required Specification is ignored.</p>
<ftp-type>	0 or 1 time	<p>Specifies a tag that indicates the data type to be transferred (transmission mode). Use this tag as an argument in TYPE command. If this tag is set repetitively in FTP adapter runtime-environment property file, the value given in the request message becomes valid.</p> <p>The specified value is either of the following values: A: Transfer the data in ASCII format. I: Transfer the data in binary format.</p> <ul style="list-style-type: none"> <li>• If the tag is present and the value is either "A" or "I" The tag is used as an argument of TYPE command.</li> <li>• If the tag is present and the value is neither "A" nor "I" a system exception is to be returned.</li> <li>• If the tag is present and the value is not present TYPE command is not executed.<sup>#4</sup></li> <li>• If the tag is not present The value given in FTP adapter runtime-environment property file becomes valid. If a value is not specified in FTP adapter runtime-environment property file, TYPE command is not executed.<sup>#4</sup></li> </ul>
<ftp-mode>	0 or 1 time	<p>Specifies a tag that indicates the transfer mode (compression mode). Use this tag as an argument in the MODE command. If this tag is set repetitively in FTP adapter runtime-environment property file, the value given in the request message becomes valid.</p> <p>The specified value is either of the following values: S: Transfer the file without compressing it. C: Compress the file and transfer it.</p> <ul style="list-style-type: none"> <li>• If the tag is present and the value is either "S" or "C" The tag is used as an argument of the MODE command.</li> <li>• If the tag is present and the value is neither "S" nor "C" A system exception is to be returned.</li> </ul>

### 3. Defining Adapters

Tag name	Occurrence count <sup>#1</sup>	Description
<ftp-mode>	0 or 1 time	<ul style="list-style-type: none"> <li>If the tag is present and the value is not present The <code>MODE</code> command is not executed.</li> <li>If the tag is not present The value given in FTP adapter runtime-environment property file becomes valid. If a value is not specified even in FTP adapter runtime-environment property file, the <code>MODE</code> command is executed using default "S".</li> </ul>
<request-id>	0 or 1 time	<p>Specifies a tag that indicates the request ID created in FTP reception.</p> <ul style="list-style-type: none"> <li>If the tag and the value both are present The tag is used as request ID.</li> <li>If the tag is not present or if the tag is present and the value is not present When using a working folder, a system exception is to be returned. This exception is ignored, when using a common folder.</li> </ul>
<local-folder>	1 time	<p>Specifies a tag that indicates a local folder.</p> <ul style="list-style-type: none"> <li>When <code>common="true"</code> is specified in tag attribute Use a common folder as a local folder and specify "Common folder definition name" in this tag. When you cannot acquire a common folder from the specified common folder definition name, a system exception is to be returned.</li> <li>When <code>common="false"</code> is specified in tag attribute Use a working folder as a local folder.</li> <li>If the common attribute is not present or the value is neither "true" nor "false" A system exception is to be returned.</li> </ul>
<local-file-name>	1 time	<p>Specifies a tag that indicates name of a local file.</p> <p>When using a working folder</p> <ul style="list-style-type: none"> <li>If the tag and the value both are present This setting is ignored and the process continues.</li> <li>If the tag is not present or if the tag is present and the value is not present The process continues.</li> </ul> <p>When using a common folder</p> <p>Specify a file, which is present just under a common folder. You can use a delimiting character such as slash (/) or yen mark (\) only at the beginning of a file name. If you use a slash (/) at the beginning of a file name, slash (/) is ignored. Also, specify a file not having a symbolic link.</p> <ul style="list-style-type: none"> <li>If the tag is present and the value follows the above-mentioned limitation The tag is used as file name. If the tag is present and the value does not follow the above-mentioned limitation A system exception is to be returned.</li> <li>If the tag is not present or if the tag is present and the value is not present A system exception is to be returned.</li> </ul>
<remote-path> <sup>#5</sup>	1 time	<p>Specifies a tag that indicates the remote path name (name of the destination path at the time of transferring to the FTP server).</p> <ul style="list-style-type: none"> <li>If the tag and the value both are present The tag is used as remote path name.</li> <li>If the tag is not present or if the tag is present and the value is not present A system exception is to be returned.</li> </ul>
<ftp-commands-before> <sup>#6</sup>	0 or 1 time	<p>Specifies a tag that indicates the FTP command and its argument to be executed prior to file transfer.</p>

Tag name	Occurrence count#1	Description
<ftp-commands-before>#6	0 or 1 time	<p>When executing multiple commands, separate the commands in the sequence in which they will be executed using a semicolon (;) and set them. (Example: MKD transdir;CWD transdir)</p> <ul style="list-style-type: none"> <li>If the tag and the value both are present The tag is used as the command to be executed prior to file transfer.</li> <li>If the tag is not present or the tag is present and the value is not present There is no command to be executed prior to file transfer.</li> </ul>
<ftp-commands-after>#6	0 or 1 time	<p>Specifies a tag that indicates the FTP command and its argument to be executed after file transfer.</p> <p>When executing multiple commands, separate the commands in the sequence in which they will be executed using a semicolon (;) and set them. (Example: RNFR oldfile.txt;RNTO newfile.txt)</p> <ul style="list-style-type: none"> <li>If the tag and the value both are present The tag is used as the command to be executed after file transfer.</li> <li>If the tag is not present or the tag is present and the value is not present There is no command to be executed after file transfer.</li> </ul>

## Legend:

-: There is no corresponding item.

## Note#1

If the specified occurrence count is exceeded, the operation is not guaranteed.

## Note#2

When the destination FTP server is running in Implicit mode of FTPS, if you specify false in the value of the <ftps-enable> tag or the <ftps-implicit-mode> tag, waiting status might appear after you connect to the FTP server till the timeout occurs.

The smallest value from among the following values is the duration till the timeout occurs:

- Value specified in ftpadp.control-con.connect.timeout property of FTP adapter runtime-environment property file
- Time-out value set in the connection destination FTP server

If the time-out occurs as per the value specified in ftpadp.control-con.connect.timeout property of FTP adapter runtime-environment property file, after you perform retry operation for the count specified in ftpadp.control-con.retry.count property of FTP adapter runtime-environment property file, error of KDEK30407-E occurs.

If the time-out occurs as per the value set in destination FTP server, error of KDEK30428-E occurs.

## Note#3

You can use only the following characters:

one-byte alphanumeric characters, one-byte space, exclamation mark (!), double-quotation mark ("), number sign (#), dollar sign (\$), percentage sign (%), ampersand (&), apostrophe ('), left parenthesis ((), right parenthesis ()), asterisk (\*), positive sign (+), comma (,), hyphen (-), period (.), slash (/), colon (:), semi-colon (;), left-angle bracket (<), right-angle bracket (>), equal sign (=), question mark (?), at sign (@), left-square bracket ([), right-square bracket (]), yen mark (\), circumflex accent (^), underscore (\_), accent grave (`), left curly bracket ({), right curly bracket (}), pipeline (|), wave dash (~)

## Note#4

The file is transferred in ASCII format by default as per the specifications of FTP protocol (RFC959).

## Note#5

If you specify a value same as the <transfer-path> tag of a request message of the FTP reception in the <remote-path> tag of the FTP adapter, the path starts with a slash (/).

When you specify a relative path or when you want to specify a path in the absolute path format of Windows, if there is slash (/) at the beginning of the path, a problem occurs. In such cases, extract the string from 2nd character onward from the value of the <transfer-path> tag of the request message of the FTP reception using substring acquisition function (substr) in the data transformation activity of a business process and set that string in the <remote-path> tag of the FTP adapter.

## Note#6

The command specified in the <ftp-commands-before> tag and the <ftp-commands-after> tag the <ftp-commands-after> tag is sent as is to FTP server. The FTP adapter is not affected by the impact of the execution of the command and the validity of the command option.

Therefore, if you specify `OPTS UTF {ON|OFF}` command in the `<ftp-commands-before>` tag and the `<ftp-commands-after>` tag, you must specify the character set that is applied to the communication with the FTP server in either of the following:

- `<file-name-charset>` tag of a request message
- `ftpdp.file-name.charset` property of FTP adapter runtime-environment property file

#### ● GETINFO operation

The following table describes the request message format of GETINFO operation, which a business process passes on to an FTP adapter. The file name is `"ftpdp_getinfo_request.xsd"`. The name of the name space is `"http://www.hitachi.co.jp/soft/xml/cosminexus/csaftp/getinfo_request"`.

Table 3-48: Request message format (GETINFO operation)

Tag name	Occurrence count #1	Description
<code>&lt;request&gt;</code>	1 time	-
<code>&lt;host-ipaddr&gt;</code>	0 or 1 time	<p>Specifies the IP address or the host name of the FTP server to be connected.</p> <ul style="list-style-type: none"> <li>• If the tag and the value both are present connect is issued using the specified string.</li> <li>• If the tag is present and the value is not present a system exception is to be returned.</li> <li>• If the tag is not present The value given in FTP adapter runtime-environment property file becomes valid. If a value is not specified in FTP adapter runtime-environment property file, a system exception is to be returned.</li> </ul> <p>When using an OS from Windows Vista and Windows Server 2008 onward, if you specify loopback IP address (127.0.0.1 or localhost), a failure might occur in FTP connection. Specify the real IP address or the host name.</p>
<code>&lt;host-con-port&gt;</code>	0 or 1 time	<p>Specifies the port number for the control connection of FTP server.</p> <ul style="list-style-type: none"> <li>• If the tag and the value both are present Specify a port number within the range from 1 to 65535. If you specify any value other than this, a system exception is to be returned.</li> <li>• If the tag is present and the value is not present 21 is specified by default.</li> <li>• If the tag is not present The value given in FTP adapter runtime-environment property file becomes valid.</li> </ul>
<code>&lt;file-name-charset&gt;</code>	0 or 1 time	<p>Specifies a tag that indicates the name of the character set to be used when sending and receiving the following information using the control connection between FTP servers.</p> <ul style="list-style-type: none"> <li>• Information to be sent and received using control connection (such as file name)</li> <li>• Information received from an FTP server through a data connection at the time of executing <code>LIST</code> command or <code>NLST</code> command (information of file list)</li> </ul> <p>The specified value is either of the following values:  <code>UTF-8</code>: Use "UTF-8" in the character set.  <code>MS932</code>: Use "MS932" in the character set.</p> <ul style="list-style-type: none"> <li>• In case of character set for which the tag is present and the value can be used It will be applied as the character set to be used when sending and receiving information such as file names.</li> <li>• In case of character set name for which the tag is present and the value cannot be used A system exception is to be returned.</li> </ul>

Tag name	Occurrence count #1	Description
<file-name-charset>	0 or 1 time	<ul style="list-style-type: none"> <li>If the tag is present and the value is not present "UTF-8" is used by default.</li> <li>If the tag is not present The value given in FTP adapter runtime-environment property file becomes valid.</li> </ul>
<ftps>	0 or 1 time	Specifies a tag that indicates the settings related to FTPS. If the tag is not present, the value given in FTP adapter runtime-environment property file becomes valid for all the settings related to FTPS.
<ftps-enable>	0 or 1 time	<p>Specifies a tag that indicates whether to use FTPS to connect to an FTP server. The specified value is either of the following values:</p> <p>true: Use FTPS to connect to an FTP server.</p> <p>false: Use normal FTP to connect to an FTP server.<sup>#2</sup></p> <p>When you use FTPS, the communication of control connection and data connection is encrypted. However, the connection destination FTP server must support the connection using FTPS. Even if you specify "true", when encryption of communication of data connection is not set, communication of data connection is also not encrypted.</p> <ul style="list-style-type: none"> <li>If the tag is present and the value is "true" or "false" The specified value is applied.</li> <li>If the tag is present and the value is other than "true" or "false" a system exception is to be returned.</li> <li>If the tag is present and the value is not present "false" is used by default.</li> <li>If the tag is not present The value given in FTP adapter runtime-environment property file becomes valid.</li> </ul>
<ftps-protocol-name>	0 or 1 time	<p>Specifies a tag that indicates the protocol for security communication to be used, when using FTPS to connect to an FTP server. The specified value is either of the following values:</p> <p>TLS: Use TLS in the protocol for security communication.</p> <p>SSL: Use SSL in the protocol for security communication.</p> <ul style="list-style-type: none"> <li>If the tag is present and the value is "TLS" or "SSL" The specified value is applied.</li> <li>If the tag is present and the value is other than "TLS" or "SSL" A system exception is to be returned.</li> <li>If the tag is present and the value is not present "TLS" is used by default.</li> <li>If the tag is not present The value given in FTP adapter runtime-environment property file becomes valid.</li> </ul> <p>This setting becomes valid only when using FTPS to connect to an FTP server.</p>
<ftps-implicit-mode>	0 or 1 time	<p>Specifies a tag that indicates whether to use Implicit mode of FTPS, when using FTPS to connect to an FTP server. The specified value is either of the following values:</p> <p>true: Use Implicit mode.</p> <p>false: Use Explicit mode.<sup>#2</sup></p> <ul style="list-style-type: none"> <li>If the tag is present and the value is "true" or "false" The specified value is applied.</li> </ul>

### 3. Defining Adapters

Tag name	Occurrence count #1	Description
<ftps-implicit-mode>	0 or 1 time	<ul style="list-style-type: none"> <li>• If the tag is present and the value is other than "true" or "false" a system exception is to be returned.</li> <li>• If the tag is present and the value is not present "false" is used by default.</li> <li>• If the tag is not present The value given in FTP adapter runtime-environment property file becomes valid.</li> </ul> <p>This setting becomes valid only when using FTPS to connect to an FTP server.</p>
<ftps-data-con-secure>	0 or 1 time	<p>Specifies a tag that indicates whether to encrypt the communication of data connection, when using FTPS to connect to an FTP server. The specified value is either of the following values:</p> <p>true: Encrypt the communication of data connection. false: Do not encrypt the communication of data connection.</p> <ul style="list-style-type: none"> <li>• If the tag is present and the value is "true" or "false" The specified value is applied.</li> <li>• If the tag is present and the value is other than "true" or "false" A system exception is to be returned.</li> <li>• If the tag is present and the value is not present "true" is used by default.</li> <li>• If the tag is not present The value given in FTP adapter runtime-environment property file becomes valid.</li> </ul> <p>This setting becomes valid only when using FTPS to connect to an FTP server.</p>
<ftps-server-authentication>	0 or 1 time	<p>Specifies a tag that indicates whether to perform server authentication, when using FTPS to connect to an FTP server. The specified value is either of the following values:</p> <p>true: Perform server authentication. false: Do not perform server authentication.</p> <ul style="list-style-type: none"> <li>• If the tag is present and the value is "true" or "false" The specified value is applied.</li> <li>• If the tag is present and the value is other than "true" or "false" A system exception is to be returned.</li> <li>• If the tag is present and the value is not present "false" is used by default.</li> <li>• If the tag is not present The value given in FTP adapter runtime-environment property file becomes valid.</li> </ul> <p>This setting becomes valid only when using FTPS to connect to an FTP server. Note that when you specify "true", settings related to keystore are required. For details on how to set a keystore, see, <i>Appendix H.1 Secure connection using FTPS (FTP adapter)</i>.</p>
<ftp-user>#3	0 or 1 time	<p>Specifies a tag that indicates the login user name of the FTP server to be connected. Use it as an argument for USER command. The login user specified in this tag must have permission to execute file operations.</p> <p>If this tag is set repetitively in FTP adapter runtime-environment property file, the value of a request message becomes valid.</p> <ul style="list-style-type: none"> <li>• If the tag is present and a normal value is present</li> </ul>

Tag name	Occurrence count #1	Description
<ftp-user>#3	0 or 1 time	<p>"USER "Value"" is issued. If the value contains linefeed, a system exception is to be returned.</p> <ul style="list-style-type: none"> <li>If the tag is present and the value is not present or if an invalid character is included in the value A system exception is to be returned.</li> <li>If the tag is not present The value given in FTP adapter runtime-environment property file becomes valid. If a value is not specified in FTP adapter runtime-environment property file, a system exception is to be returned.</li> </ul>
<ftp-acct>#3	0 or 1 time	<p>Specifies a tag that indicates the billing information of the FTP server to be connected.</p> <p>If this tag is set repetitively in FTP adapter runtime-environment property file, the value of a request message becomes valid.</p> <p>User for whom the billing information is required The following value is used as per the condition:</p> <ul style="list-style-type: none"> <li>If the tag is present and the value is normal The contents of the tag are used in ACCT command. If a linefeed is included in the value, a system exception is to be returned.</li> <li>If the tag is present and an invalid character is included in the value A system exception is to be returned.</li> <li>If the tag is present and the value is not present "ACCT"" is issued.</li> <li>If the tag is not present The value of ACCT specified in FTP adapter account definition file is used. If ACCT is not present in FTP adapter account definition file, "ACCT "" is issued.</li> </ul> <p>User for whom the billing information is not required Specification is ignored.</p>
<getinfo-type>	1 time	<p>Specifies a tag that indicates the acquisition type of the information acquired from the FTP server to be connected.</p> <p>Either of the following values is the setup value: LIST: Execute LIST command and acquire the list of file information. NLST: Execute NLST command and acquire the list of file names.</p> <ul style="list-style-type: none"> <li>If the tag is present and the value is either "LIST" or "NLST" Use the specified value as the acquisition type.</li> <li>If the tag is present and the value is other than "LIST" or "NLST" A system exception is to be returned.</li> <li>If the tag is not present or if the tag is present and the value is not present A system exception is to be returned.</li> </ul>
<getinfo-option>	0 or 1 time	<p>Specifies a tag that indicates the acquisition option for the information acquired from the FTP server to be connected. This option is to be specified when executing the command specified in &lt;getinfo-type&gt;.</p> <ul style="list-style-type: none"> <li>If the tag and the value both are present Mention the specified value as an option and execute the command specified in &lt;getinfo-type&gt;.</li> <li>The tag is not present or if the tag is present and the value is not present Execute the command specified in &lt; getinfo-type &gt; without specifying an option.</li> </ul>

### 3. Defining Adapters

Tag name	Occurrence count #1	Description
<getinfo-path>	0 or 1 time	<p>Specifies a tag that indicates the acquisition path of the information acquired from the FTP server to be connected. This path is to be specified when executing the command specified in &lt;getinfo-type&gt;.</p> <ul style="list-style-type: none"> <li>• If the tag and the value both are present Mention the specified value as the path and execute the command specified in &lt;getinfo-type&gt;.</li> <li>• The tag is not present or if the tag is present and the value is not present Execute the command specified in &lt;getinfo-type&gt; without specifying the path.</li> </ul>
<request-id>	0 or 1 time	<p>Specifies a tag that indicates the request ID created in FTP reception.<sup>#5</sup></p> <ul style="list-style-type: none"> <li>• If the tag and the value both are present The working folder is used as the local folder. The specified value is used as the request ID. This is valid only when a value other than common="true" is specified in the attribute of &lt;local-folder&gt;.</li> <li>• If the tag is present and the value is not present a system exception is to be returned.</li> <li>• If the tag is not present If &lt;local-folder&gt; tag is omitted, it is not output to a response message.</li> </ul>
<local-folder>	0 or 1 time	<p>Specifies a tag that indicates a local folder.<sup>#5</sup></p> <ul style="list-style-type: none"> <li>• If common="true" is specified in the tag attribute Use a common folder as the local folder and specify "Name of common folder definition" in this tag. When you cannot acquire the common folder from the specified name of the common folder definition, a system exception is to be returned.</li> <li>• If common="false" is specified in the tag attribute Use a working folder as the local folder.</li> <li>• If common attribute is not present or the value is neither "true" nor "false" A system exception is to be returned.</li> <li>• If the tag is not present Use a working folder as the local folder. However, if &lt;request-id&gt; is omitted, it is output to a response message.</li> </ul>
<local-file-name>	0 or 1 time	<p>Specifies a tag that indicates the name of a local file.<sup>#5</sup></p> <p>When using a working folder Specification of the tag is ignored and the process continues.</p> <p>When using a common folder Specify name of the file present just under the common folder. If you specify name of a file which does not exist, a file is created with the specified name. You can use a delimiting character such as slash (/) or yen mark (\) only at the beginning of a file name. If you use a slash (/) at the beginning of a file name, slash (/) is ignored. Also, specify a file not having a symbolic link.</p> <ul style="list-style-type: none"> <li>• If the tag is present and the value follows the above-mentioned limitation The tag is used as file name.</li> <li>• If the tag is present and the value does not follow the above-mentioned limitation A system exception is to be returned.</li> <li>• If the tag is not present or if the tag is present and the value is not present A system exception is to be returned.</li> </ul>

Tag name	Occurrence count #1	Description
<local-file-name>	0 or 1 time	When the tag is to be output to a response message Specification of the tag is ignored and the process continues.
<ftp-commands-before>#4	0 or 1 time	Specifies a tag that indicates the FTP command and its arguments to be executed before executing the list command.  When executing multiple commands, separate the commands in the sequence in which they will be executed using a semicolon (;) and set them (Example: MKD transdir;CWD transdir).  <ul style="list-style-type: none"> <li>• If the tag and the value both are present The command is used as the command to be executed before executing the list command.</li> <li>• If the tag is not present or the tag is present and the value is not present There is no command to be executed before executing the list command.</li> </ul>
<ftp-commands-after>#4	0 or 1 time	Specifies a tag that indicates the FTP command and its arguments to be executed after executing the list command.  When executing multiple commands, separate the commands in the sequence in which they will be executed using a semicolon (;) ,and set them (Example: RNFR oldfile.txt;RNTO newfile.txt).  <ul style="list-style-type: none"> <li>• If the tag and the value both are present The command is used as the command to be executed after executing the list command.</li> <li>• If the tag is not present or if the tag is present and the value is not present There is no command to be executed after executing the list command.</li> </ul>

## Legend:

-: There is no corresponding item.

## Note#1

If the specified occurrence count is exceeded, the operation is not guaranteed.

## Note#2

If you specify false in the value of the <ftps-enable> tag or the <ftps-implicit-mode> tag, when the connection destination FTP server operates in Implicit mode of FTPS, waiting status might appear after connecting to the FTP server till the timeout occurs.

The smallest value from among the following values is the duration till the timeout occurs:

- Value specified in ftpadp.control-con.connect.timeout property of FTP adapter runtime-environment property file
- Time-out value set in connection destination FTP server

If the time-out occurs as per the value specified in ftpadp.control-con.connect.timeout property of FTP adapter runtime-environment property file, after you perform retry operation for the count specified in ftpadp.control-con.retry.count property of FTP adapter runtime-environment property file, error of KDEK30407-E occurs.

If the time-out occurs as per the value set in the connection destination FTP server, error of KDEK30428-E occurs.

## Note#3

You can use only the following characters:

one-byte alphanumeric characters, one-byte space, exclamation mark (!), double-quotation mark ("), number sign (#), dollar sign (\$), percentage sign (%), ampersand (&), apostrophe ('), left parenthesis ((), right parenthesis ()), asterisk (\*), positive sign (+), comma (,), hyphen (-), period (.), slash (/), colon (:), semi-colon (;), left-angle bracket (<), right-angle bracket (>), equal sign (=), question mark (?), at sign (@), left-square bracket ([), right-square bracket (]), yen mark (\), circumflex accent (^), underscore (\_), accent grave (`), left curly bracket ({), right curly bracket (}), pipeline (|), wave dash (~)

## Note#4

The command specified in the <ftp-commands-before> tag and the <ftp-commands-after> tag the <ftp-commands-after> tag is sent as is to FTP server. The FTP adapter is not affected by the impact of the execution of the command and the validity of the command option.

Therefore, if you specify OPTS UTF {ON|OFF} command in the <ftp-commands-before> tag and the <ftp-commands-after> tag, you must specify the character set that is applied to the communication with FTP server in either of the following:

- <file-name-charset> tag of a request message

### 3. Defining Adapters

- ftpadp.file-name.charset property of FTP adapter runtime-environment property file

Note#5

For details on how to define GETINFO operation, see (c) *Request message and response message to be used at the time of using GETINFO operation.*

#### (b) Response message format for FTP adapters

##### • PUT operation

The following table describes the response message format for PUT operation, which is returned to the calling business process by the FTP adapter. The file name is "ftpadp\_put\_response.xsd". The name of the namespace is "http://www.hitachi.co.jp/soft/xml/cosminexus/csaftp/put\_response".

Table 3–49: Response message format (PUT operation)

Tag name	Occurrence count	Description
<response>	1 time	-
<local-folder>	1 time	Specifies a tag that indicates a local folder. If you use a working folder, "common="false" is set in the attribute. Null is set in the value. If you use a common folder, "common="true" is set in the attribute. The name of the common folder definition specified in the <local-folder> tag of a request message is set in the value.
<local-file-name>	1 time	Specifies a tag that indicates name of the local file. The value specified in <local-file-name> of a request message is set.
<command-info>	More than 1 time	-
<command>	1 time	Specifies a tag that indicates the command executed by an FTP adapter. #1 All the FTP commands, which are executed are output.
<command-message>	More than 1 time	Specifies a tag that indicates display message of the command executed by an FTP adapter. #2 The message of the FTP command consists of FTP response code and a message. In case of a message having multiple lines, if the response code of these lines is same, 1 tag is generated. In case of FTP response code of multiple types, multiple tags are generated.

Legend:

-: There is no corresponding item.

Note#1

For details on the types of the commands executed by an FTP adapter, see 8.6.2 *Executing the FTP commands* in the manual *Service Platform Overview*.

For details on the output format of the commands executed by an FTP adapter and examples of output, see Table 3-52.

Note#2

For details on the message format of FTP commands and examples of output, see Table 3-53.

##### • GET operation

The following table describes the response message format of GET operation, which is returned to the calling business process by an FTP adapter. The file name is "ftpadp\_get\_response.xsd". The name of the namespace is "http://www.hitachi.co.jp/soft/xml/cosminexus/csaftp/get\_response".

Table 3–50: Response message format (GET operation)

Tag name	Occurrence count	Description
<response>	1 time	-

Tag name	Occurrence count	Description
<local-folder>	1 time	Specifies a tag that indicates a local folder. If you use a working folder, "common=false" is set in the attribute. Null is set in the value. If you use a common folder, "common=true" is set in the attribute. The name of the common folder definition specified in the <local-folder> tag of a request message is set in the value.
<local-file-name>	1 time	Specifies a tag that indicates name of a local file. When using a working folder Specify name of the file, which is stored just under the working folder. File name will be in the format of "csc_<characters generated internally>". When using a common folder The value specified in <local-file-name> of a request message is set.
<file-size>	1 time	Specifies a tag that indicates the size of the remote file, which is sent and received. Size of the local file acquired from an FTP server is set.
<command-info>	More than 1 time	-
<command>	1 time	Specifies a tag that indicates the FTP command executed by an FTP adapter. <sup>#1</sup> All the FTP commands, which are executed are output.
<command-message>	More than 1 time	Specifies a tag that indicates the display message of the command executed by an FTP adapter. <sup>#2</sup> The message of the FTP command consists of the FTP response code and a message. In case of a message having multiple lines, if the response code of these lines is same, 1 tag is generated. In case of the FTP response code of multiple types, multiple tags are generated.

## Legend:

-. There is no corresponding item.

## Note#1

For details on the types of the commands executed by an FTP adapter, see 8.6.2 *Executing the FTP commands* in the manual *Service Platform Overview*.

For details on the output format of the commands executed by an FTP adapter and examples of output, see Table 3-52.

## Note#2

For details on the message format of FTP commands and examples of output, see Table 3-53.

### ● GETINFO operation

The following table describes the response message format of GETINFO operation, which is returned by the calling business process by an FTP adapter. The file name is "ftpadp\_getinfo\_response.xsd". The name of the name space is "http://www.hitachi.co.jp/soft/xml/cosminexus/csaftp/getinfo\_response".

Table 3-51: Response message format (GETINFO operation)

Tag name	Occurrence count	Description
<response>	1 time	-
<local-folder>	0 or 1 time	Specifies a tag that indicates a local folder. This tag appears when the output is to be generated in a common folder. This tag does not appear when the output is to be generated in a working folder or a response message. When using a common folder "common=true" is set in the attribute. The name of the common folder definition specified in the <local-folder> tag of a request message is set in the value.

### 3. Defining Adapters

Tag name	Occurrence count	Description
<local-file-name>	0 or 1 time	<p>Specifies a tag that indicates the name of a local file. This tag appears, when the output is to be generated in a working folder or common folder. This tag does not appear, when the output is to be generated in a response message.</p> <p>When using a working folder Specify name of the file, which is stored just under the working folder. File name will be in the format of "csc_&lt;characters generated internally&gt;". The acquired information is output using the character set of "UTF-16". Also, the linefeed code is output using CRLF.</p> <p>When using a common folder The value specified in &lt;local-file-name&gt; of a request message is set.</p> <p>The acquired information is written in this file using character set of UTF-16. Also, the linefeed code will be CRLF.</p>
<reply-code>	1 time	<p>Specifies a tag that indicates reply code of the LIST command or the NLST command.</p> <p>When you execute LIST command or NLST command, either of the following values is set as the reply code which is sent at the end by FTP server:</p> <ul style="list-style-type: none"> <li>• 200~299 Specifies a code indicating that the process has completed successfully.</li> <li>• 450 or 550 Specifies a tag indicating that the file operation could not be performed.</li> </ul> <p>Note that if the path specified in the LIST command or the NLST command does not exist, the reply code which is returned differs according to the connection destination FTP server.</p>
<reply-message>	1 time	<p>Specifies a tag that indicates the reply message of the LIST command or the NLST command.</p> <p>When you execute the LIST command or the NLST command, the reply message which is sent at the end is set by the FTP server.</p>
<command-info>	More than 1 time	-
<command>	1 time	<p>Specifies a tag that indicates an FTP command executed by an FTP adapter.<sup>#1</sup></p> <p>All the FTP commands, which are executed are output.</p>
<command-message>	More than 1 time	<p>Specifies a tag that indicates a display message of the command executed by an FTP adapter.<sup>#2</sup></p> <p>The message of the FTP command consists of FTP response code and a message.</p> <p>In case of a message having multiple lines, if the response code of these lines is same, 1 tag is generated. In case of the FTP response code of multiple types, multiple tags are generated.</p>
<list-info>	0 or 1 time	Appears when the output is to be generated in a response message. <sup>#3</sup>
<list>	More than 0	Appears when the output is to be generated in a response message. <sup>#3</sup>

**Legend:**

-: There is no corresponding item.

**Note#1**

For details on the types of the commands executed by an FTP adapter, see *8.6.2 Executing the FTP commands* in the manual *Service Platform Overview*.

For details on the output format of the commands executed by an FTP adapter and examples of output, see Table 3-52.

**Note#2**

For details on the message format of FTP commands and examples of output, Table 3-53.

## Note#3

For details on how to define GETINFO operation, see (c) *Request message and response message to be used at the time of using GETINFO operation.*

● **Output format of the commands executed by an FTP adapter and examples of output**

The following table describes the output format of the FTP commands which is output to the <command> tag of a response message and examples of output:

Table 3–52: Output format of the commands executed by an FTP adapter and examples of output

Item	FTP command	Output format	Example of output	Remarks
1	(Connection process)	Connected to <xxxxxx>.	Connected to 10.208.180.254.	<xxxxxx> is the IP address of the FTP server or the host name.
2	USER	USER <user name>	USER user1	-
3	PASS	PASS *****	PASS *****	The password is displayed with 8 * symbols.
4	ACCT	ACCT <Billing information>	ACCT acct1	-
5	TYPE	TYPE <format option>	TYPE A	Either "A" or "I" is displayed in <format option>.
6	MODE	MODE <transfer mode option>	MODE S	Either "S" or "C" is displayed in <transfer mode option>.
7	PASV	PASV	PASV	-
8	PORT	PORT <aaa>,<bbb>,<ccc>,<ddd>,<mmm>,<nnn>	PORT 10,208,182,27,14,160	<aaa>,<bbb>,<ccc>,<ddd> is the local IP address. <mmm>,<nnn> is the port number. The local data connection port number appears in the format of 256*mmm+nnn.
9	SITE	SITE FSIZE [<file size>]	SITE FSIZE 4096	For GET operation, <file size> is not output. Also, for PUT operation, size of the file present in the transfer path of the source file is output to <file size>.
10	Command prior to transfer or command prior to list	<command name> [<command argument>]	CWD ./workdir	-
11	STOR	STOR <path name>	STOR work/fileA.txt	-
12	APPE	APPE <path name>	APPE work/fileA.txt	-
13	RETR	RETR <path name>	RETR fileA.txt	-
14	LIST	LIST [<option>] [<path name>]	LIST -l work	-
15	NLST	NLST [<option>] [<path name>]	NLST -l work	-
16	Command after transfer or command	<command name> [<command argument>]	RNFR fileA.txt RNTO fileB.txt	-

Item	FTP command	Output format	Example of output	Remarks
16	after list	<command name> [<command argument>]	RNFR fileA.txt RNTO fileB.txt	-
17	QUIT	QUIT	QUIT	-

Legend:

-: There is no corresponding item.

● **Message format of FTP commands and examples of output**

The following table describes the message format of FTP commands, which is output to <command-message> tag of a response message and examples of output:

Table 3–53: Message format of FTP commands and examples of output

Condition	Message format	Example of output
Single response code	<FTP response code> <message>	226 File receive OK.
	<FTP response code>-<message 1> [<message 2>] : <FTP response code> <message n#>	214-The following commands are recognized. ACCT ALLO APPE CDUP ... MDTM MKD NLST NOOP OPTS PASS ... RMD RNFR 214 Help OK.
Multiple response code	<FTP response code 1> <message 1> <FTP response code 2> <message 2>	150 File status okay; about to open data connection. 226 Transfer complete.

Note

For details on the string contents of a response code and a message, see a document on the FTP server in use.

Note#

n is a natural number.

(c) Request message and response message to be used at the time of using GETINFO operation

The following table describes the tags which must be specified in a request message, and the tags which appear in a response message, when you want to output the result of the LIST command or the NLIST command to a response message, a working folder, and a common folder.

For details on the setup contents of each tag and the tags which are not described in the table, see (a) Request message format for FTP adapters and (b) Response message format for FTP adapters.

Table 3–54: Relation between an output destination and a request message

Output destination	Tag to be specified in a request message		
	<request-id>	<local-folder>	<local-file-name>
Response message	-	-	-
Working folder	Y	Y	-
Common folder	-	Y	Y

Legend:

Y: Specify.

-: Do not specify

Table 3–55: Relation between an output destination and a response message

Output destination	Tag to be specified in a response message			
	<local-folder>	<local-file-name>	<list-info>	<list>
Response message	-	-	Y	Y
Working folder	-	Y	-	-
Common folder	Y	Y	-	-

Legend:

Y: Appears.

--: Does not appear.

## (2) Creating a definition file of an FTP adapter

Types of the definition file to be created are as follows:

- FTP adapter command definition file  
This file is used to define output destinations and output levels of messages of the user management commands used in FTP adapters.
- FTP adapter account definition file  
This file is used to define the user account information such as FTP authorized user, billing information. This file is referenced from FTP-adapter runtime-environment property file.
- FTP command permission list definition file (for FTP adapters)  
This file is used to define the execution permissions of the FTP commands to be executed before and after transfer. This file is referenced from the FTP reception config file and FTP-adapter runtime-environment property file.
- FTP-adapter runtime-environment property file  
This file is used to define the configuration information for each FTP adapter.

The following is the procedure for creating each definition file. You can create a definition file using the template file provided by the FTP adapter.

### (a) Procedure for creating the FTP adapter command definition file

1. Copy the template file (*Installation directory of the service platform*\CSC\custom-adapter\FTP\config\templates\adpftpcommand.properties) and save this file in the following directory:  
*Installation directory of the service platform*\CSC\custom-adapter\FTP\config
2. Edit and save the definition contents.  
For details on the definition contents that you can edit in the FTP adapter command definition file, see *FTP adapter command definition file* in the manual *Service Platform Reference Guide*.

### (b) Procedure for creating the FTP adapter account definition file

1. Execute the `csaftpuseradd` command to create the FTP adapter account definition file.  
You must specify the path of the FTP adapter account definition file in the `ftpadp.account-inf-filepath` key of the FTP-adapter runtime-environment property file. For details on the `csaftpuseradd` command, see *csaftpuseradd (Registering and updating users of FTP adapters)* in the manual *Service Platform Reference Guide*.

### (c) Procedure for creating the FTP command permission list definition file (for FTP adapters)

1. Copy the template file (*Installation directory of the service platform*\CSC\custom-adapter\FTP\config\templates\cscthrough.properties) and save this file in any directory.
2. Change the name of the template file, which is copied with any name.
3. Edit and save the definition contents.

### 3. Defining Adapters

For details on the definition contents that can be edited in the FTP command permission list definition file (for FTP adapters), see *FTP command permission list definition file* in the manual *Service Platform Reference Guide*.

#### (d) Procedure for creating FTP-adapter runtime-environment property file

1. Copy the template file (*Installation directory of the service adapter\CSC\custom-adapter\FTP\config\templates\adpftp.properties*) and save this file in the following directory:  
*Installation directory of the service platform\CSC\custom-adapter\FTP\config*
2. Change the name of the template file, which is copied with "<Service ID#>.properties".
3. Edit and save the definition contents.

For details on the definition contents that can be edited in the FTP-adapter runtime-environment property file, see *FTP-adapter runtime-environment property file* in the manual *Service Platform Reference Guide*.

#### Important note

A service ID is any string, which is to be specified on the service adapter settings screen, when adding a new FTP adapter.

---

### (3) Defining data transformation

Define data transformation of the message format definition file of transformation source and the message format definition file of the transformation destination using the data transformation definition screen.

For details on how to define data transformation, see *6. Defining Data Transformation* in the manual *Service Platform Basic Development Guide*.

### (4) Operations to be performed on the service adapter settings screen

The procedure for defining FTP adapters is as follows:

1. Display the service adapter settings screen.  
For details on how to display the service adapter settings screen, see *3.3.1(4) Displaying the service adapter settings screen*.
2. Set up the definition information.  
For details on the items for which setting is required in the service adapter settings screen (basic), see *3.3.15(8) For FTP adapters* and *1.2.2 Service adapter settings screen* in the manual *Service Platform Reference Guide*.
3. Click the [Service adapter settings (details)] tab.  
The service adapter settings screen (details) is displayed.
4. Set up the definition information.  
For details on the items for which setting is required in the service adapter settings screen (details), see *3.3.15(8) For FTP adapters* and *1.2.2 Service adapter settings screen* in the manual *Service Platform Reference Guide*.

## 3.3.11 Defining file operations adapters

This section describes the method of defining file operations adapters. Note that for details on definition examples in case of using file transformation operation, see *Appendix E Example for defining file operations adapter*.

### (1) Creating a message format

For the message format definition file of file operations adapters, use the schema provided by the service platform. Therefore, you need not create the message format definition file.

The contents of the message format used by file operations adapters are described here.

The storage location for the file is "*Installation directory of the service platform\CSC\schema\adpfop*".

#### (a) Request message format for file operations adapters

The operation-wise request message format for file operations adapters is described here.

### ● File transformation operation

The following table describes the request message format to be passed on from a business process to a file operations adapter. The file name is "adpfop\_transform\_request.xsd". The name of the name space is "http://www.hitachi.co.jp/soft/xml/cosminexus/csc/adapter/fileoperation/transform\_request".

Table 3–56: Request message format (file transformation operation)

Tag name	Occurrence count#	Description
<request>	1 time	-
<request-id>	0 or 1 time	Specifies a request ID created in the FTP reception. Use this ID to generate the path of a working folder. You can omit this tag, if you do not specify a working folder in the input folder and the output folder.
<input-folder-name>	1 time	Specify a folder in which the file to be transformed is stored. <ul style="list-style-type: none"> <li>When specifying a working folder Specify <code>common="false"</code> in tag attribute and specify null in the value.</li> <li>When specifying a common folder Specify <code>common="true"</code> in tag attribute and specify name of the common folder definition in the value.</li> </ul>
<input-file-name>	1 time	Specify the name of the file to be transformed. You can use a slash (/) only at the beginning of a file name. Also, the slash (/) set at the beginning of a file name is ignored. Following are the limitations for Windows: <ul style="list-style-type: none"> <li>Case-insensitive.</li> <li>You cannot specify a file name in UNC format.</li> <li>You cannot specify a stream name of NTFS.</li> <li>You cannot specify a file name that includes the name of a reserved device.</li> </ul>
<output-folder-name>	1 time	Specify a folder to which the transformed file is to be output. <ul style="list-style-type: none"> <li>When specifying a working folder Specify <code>common="false"</code> in the tag attribute and specify null in the value.</li> <li>When specifying a common folder Specify <code>common="true"</code> in the tag attribute and specify the name of the common folder definition in the value.</li> </ul>
<output-file-name>	0 or 1 time	Specify name of the transformed file. However, if the output destination of the transformed file is a working folder (if <code>common="false"</code> is specified in the attribute of the output-folder-name tag), the specification is ignored, since a unique file is created in the working folder using <code>createTempFile()</code> of <code>java.io.File</code> . You can use a slash (/) only at the beginning of a file name. Also, the slash (/) set up at the beginning of a file name is ignored. Following are the limitations for Windows: <ul style="list-style-type: none"> <li>Case-insensitive.</li> <li>You cannot specify a file name in UNC format.</li> <li>You cannot specify a stream name of NTFS.</li> <li>You cannot specify a file name that includes the name of a reserved device.</li> </ul>

Legend:

-: There is no corresponding item.

Note#

If the specified occurrence count is exceeded, the operation is not guaranteed.

### • File replication operation

The following table describes the request message format to be passed on from a business process to a file operations adapter. The file name is "adpfop\_copy\_request.xsd". Name of the name space is "http://www.hitachi.co.jp/soft/xml/cosminexus/csc/adapter/fileoperation/copy\_request".

Table 3–57: Request message format (file replication operation)

Tag name	Occurrence count#	Description
<request>	1 time	-
<request-id>	0 or 1 time	Specifies a request ID created in the FTP reception. Use this ID to generate the path of a working folder. You can omit this tag, if you do not specify a working folder in the input folder and the output folder.
<input-folder-name>	1 time	Specify the folder in which the file to be copied is stored. <ul style="list-style-type: none"> <li>When specifying a working folder Specify <code>common="false"</code> in tag attribute and specify null in the value.</li> <li>When specifying a common folder Specify <code>common="true"</code> in tag attribute and specify the name of the common folder definition in the value.</li> </ul>
<input-file-name>	1 time	Specify the name of the file to be copied. You can use a slash (/) only at the beginning of a file name. Also, the slash (/) set at the beginning of a file name is ignored. Following are the limitations for Windows: <ul style="list-style-type: none"> <li>Case-insensitive.</li> <li>You cannot specify a file name in UNC format.</li> <li>You cannot specify a stream name of NTFS.</li> <li>You cannot specify a file name that includes the name of a reserved device.</li> </ul>
<output-folder-name>	1 time	Specify the folder to which the copied file is to be output. <ul style="list-style-type: none"> <li>When specifying a working folder Specify <code>common="false"</code> in the tag attribute and specify null in the value.</li> <li>When specifying a common folder Specify <code>common="true"</code> in the tag attribute and specify the name of the common folder definition in the value.</li> </ul>
<output-file-name>	0 or 1 time	Specify the name of the copied file. However, if the output destination of a transformed file is a working folder (if you specify <code>common="false"</code> in the tag attribute of <code>output-folder-name</code> ), the specification is ignored, since a unique file is created in the working folder using <code>createTempFile()</code> of <code>java.io.File</code> . You can use a slash (/) only at the beginning of a file name. Also, the slash (/) set at the beginning of a file name is ignored. Following are the limitations for Windows: <ul style="list-style-type: none"> <li>Case-insensitive.</li> <li>You cannot specify a file name in UNC format.</li> <li>You cannot specify a stream name of NTFS.</li> <li>You cannot specify a file name that includes the name of a reserved device.</li> </ul>

Legend:

-: There is no corresponding item.

Note#

If the specified occurrence count is exceeded, the operation is not guaranteed.

#### ● File and folder deletion operation

The following table describes the request message format to be passed on from a business process to a file operations adapter. The file name is "adpfop\_delete\_request.xsd". Name of the name space is "http://www.hitachi.co.jp/soft/xml/cosminexus/csc/adapter/fileoperation/delete\_request".

Table 3–58: Request message format (file and folder deletion operation)

Tag name	Occurrence count#	Description
<request>	1 time	-
<request-id>	0 or 1 time	Specifies a request ID created in the FTP reception. Use this ID to generate the path of a working folder. You can omit this tag, if you do not specify a working folder in the input folder and the output folder.
<folder-name>	1 time	Specify the folder in which the file to be deleted is stored or specify the folder to be deleted. <ul style="list-style-type: none"> <li>When specifying a working folder Specify <code>common="false"</code> in the tag attribute and specify null in the value.</li> <li>When specifying a common folder Specify <code>common="true"</code> in the tag attribute and specify the name of the common folder definition in the value.</li> </ul>
<file-name>	0 or 1 time	Specify the name of the file to be deleted. However, if the output destination of the file to be deleted is a working folder (if the attribute value of folder-name is <code>common="false"</code> ), specification of the <file-name> element is ignored. You can use a slash (/) only at the beginning of a file name. Also, the slash (/) set at the beginning of a file name is ignored. Following are the limitations for Windows: <ul style="list-style-type: none"> <li>Case-insensitive.</li> <li>You cannot specify a file name in UNC format.</li> <li>You cannot specify a stream name of NTFS.</li> <li>You cannot specify a file name that includes the name of a reserved device.</li> </ul>

Legend:

-: There is no corresponding item.

Note#

If the specified occurrence count is exceeded, the operation is not guaranteed.

#### ● File compression operation

The following table describes the request message format to be passed on from a business process to a file operations adapter. The file name is "adpfop\_compress\_request.xsd". The name of the name space is "http://www.hitachi.co.jp/soft/xml/cosminexus/csc/adapter/fileoperation/compress\_request".

Table 3–59: Request message format (file compression operation)

Tag name	Occurrence count#1	Description
<request>	1 time	-
<request-id>	0 or 1 time	Specifies a request ID created in the FTP reception. Use this ID to generate the path of a working folder. You can omit this tag, if you do not specify a working folder in the input folder and the output folder.
<input-folder-name>	1 time	Specify the folder in which the file to be compressed is stored.

### 3. Defining Adapters

Tag name	Occurrence count#1	Description
<input-folder-name>	1 time	<ul style="list-style-type: none"> <li>When specifying a working folder Specify <code>common="false"</code> in the tag attribute and specify null in the value.</li> <li>When specifying a common folder Specify <code>common="true"</code> in the tag attribute and specify the name of the common folder definition in the value.</li> </ul>
<input-files>	1 time	-
<input-file>	1 to 999 times	-
<input-file-name>	1 time	<p>Specify the name the file to be compressed. The following are the limitations for Windows:</p> <ul style="list-style-type: none"> <li>Case-insensitive.</li> <li>You cannot specify a file name in UNC format.</li> <li>You cannot specify a stream name of NTFS.</li> <li>You cannot specify a file name that includes name of a reserved device.</li> </ul>
<entry-name>	1 time	<p>Specify the entry name after the compression of a file specified in the <code>input-file-name</code> tag in the same <code>input-file</code> tag. If you specify null, the value of the <code>input-file-name</code> tag in the same <code>input-file</code> tag is specified. Here, entry name is not changed. The following are the limitations for Windows:</p> <ul style="list-style-type: none"> <li>Case-insensitive.</li> <li>You cannot specify a file name in UNC format.</li> <li>You cannot specify a stream name of NTFS.</li> <li>You cannot specify a file name that includes the name of a reserved device.</li> </ul>
<output-folder-name>	1 time	<p>Specify the folder to which the compressed file is to be output.</p> <ul style="list-style-type: none"> <li>When specifying a working folder Specify <code>common="false"</code> in the tag attribute and specify null in the value.</li> <li>When specifying a common folder Specify <code>common="true"</code> in the tag attribute and specify name of the common folder definition in the value.</li> </ul>
<output-file-name>	0 or 1 time	<p>Specify the name of the compressed file. However, if the output destination of the compressed file is a working folder (if you specify <code>common="false"</code> in the attribute of <code>output-folder-name</code> tag), specification is ignored, since a unique file is created in the working folder using <code>createTempFile()</code> of <code>java.io.File</code>. The following are the limitations for Windows:</p> <ul style="list-style-type: none"> <li>Case-insensitive.</li> <li>You cannot specify a file name in UNC format.</li> <li>You cannot specify a stream name of NTFS.</li> <li>You cannot specify a file name that includes name of a reserved device.</li> </ul>

Legend:

-: There is no corresponding item.

## Note#1

If the specified occurrence count is exceeded, the operation is not guaranteed.

## Note#2

Specify a file name by considering that input-file-name tag is case-insensitive. If the same file name is specified multiple times, the operation is not guaranteed.

## Note#3

When extracting a compressed file in Windows, the file name might be same, since the name of the extracted file is case-insensitive. Specify the entry name by considering this point.

### ● File extraction operation

The following table describes the request message format to be passed on from a business process to a file operations adapter. The file name is "adpfop\_extract\_request.xsd". The name of the name space is "http://www.hitachi.co.jp/soft/xml/cosminexus/csc/adapter/fileoperation/extract\_request".

Table 3–60: Request message format (file extraction operation)

Tag name	Occurrence count#	Description
<request>	1 time	-
<request-id>	0 or 1 time	Specifies a request ID created in the FTP reception. Use this ID to generate the path of a working folder. You can omit this tag, if you do not specify a working folder in the input folder and the output folder.
<input-folder-name>	1 time	Specify the folder in which the file to be compressed is stored. <ul style="list-style-type: none"> <li>When specifying a working folder Specify <code>common="false"</code> in the tag attribute and specify null in the value.</li> <li>When specifying a common folder Specify <code>common="true"</code> in the tag attribute and specify name of the common folder definition in the value.</li> </ul>
<input-file-name>	1 time	Specify the name of the file to be extracted. The following are the limitations for Windows: <ul style="list-style-type: none"> <li>Case-insensitive.</li> <li>You cannot specify a file name in UNC format.</li> <li>You cannot specify a stream name of NTFS.</li> <li>You cannot specify a file name that includes the name of a reserved device.</li> </ul>
<output-folder-name>	1 time	Specify the folder to which the extracted file is to be output. <ul style="list-style-type: none"> <li>When specifying a working folder Specify <code>common="false"</code> in the tag attribute and specify null in the value.</li> <li>When specifying a common folder Specify <code>common="true"</code> in the tag attribute and specify name of the common folder definition in the value.</li> </ul>
<output-file-prefix-name>	0 or 1 time	Specify the prefix of the name of the extracted file. <ul style="list-style-type: none"> <li>If the output destination of the transformed file is a common folder (If you specify <code>common="true"</code> in the attribute of output-folder-name tag) File name is automatically given by using this specified tag.</li> <li>If the output destination of the transformed file is a working folder (If you specify <code>common="false"</code> in the attribute of output-folder-name tag) The specification is ignored, since a unique file is created in the working folder using <code>createTempFile()</code> of <code>java.io.File</code>.</li> </ul> The following are the limitations for Windows: <ul style="list-style-type: none"> <li>Case-insensitive.</li> </ul>

### 3. Defining Adapters

Tag name	Occurrence count#	Description
<output-file-prefix-name>	0 or 1 time	<ul style="list-style-type: none"> <li>You cannot specify a file name in UNC format.</li> <li>You cannot specify a stream name of NTFS.</li> <li>You cannot specify a file name that includes the name of a reserved device.</li> </ul>

Legend:

-: There is no corresponding item.

Note#

If the specified occurrence count is exceeded, the operation is not guaranteed.

- **If a file operations adapter fails to acquire a request message**

An error message (KDEC80002-E) is to be output and the process is to be terminated.

#### (b) Response message format for file operations adapters

The operation wise response message format for file operations adapters is described here

- **File transformation operation**

The following table describes the format of a response message, which is returned to the calling business process by a file operations adapter. The file name is "adpfop\_transform\_response.xsd". The name of the name space is "http://www.hitachi.co.jp/soft/xml/cosminexus/csc/adapter/fileoperation/transform\_response".

Table 3–61: Response message format (file transformation operation)

Tag name	Occurrence count#	Description
<response>	1 time	-
<output-folder-name>	1 time	The folder to which the transformed file is to be output is set. <ul style="list-style-type: none"> <li>When specifying a working folder Specify <code>common="false"</code> in the tag attribute and specify null in the value.</li> <li>When specifying a common folder Specify <code>common="true"</code> in the tag attribute and specify the name of the common folder definition in the value.</li> </ul>
<output-file-name>	1 time	The name of the transformed file is set.
<output-file-size>	1 time	The size of the transformed file (byte) is set.

Legend:

-: There is no corresponding item.

Note#

If the specified occurrence count is exceeded, the operation is not guaranteed.

- **File replication operation**

The following table describes the format of a response message, which is returned to the calling business process by a file operations adapter. The file name is "adpfop\_copy\_response.xsd". The name of the name space is "http://www.hitachi.co.jp/soft/xml/cosminexus/csc/adapter/fileoperation/copy\_response".

Table 3–62: Response message format (file replication operation)

Tag name	Occurrence count#	Description
<response>	1 time	-
<output-folder-name>	1 time	<p>The folder to which the copied file is to be output is set.</p> <ul style="list-style-type: none"> <li>When specifying a working folder Specify <code>common="false"</code> in the tag attribute and specify null in the value.</li> <li>When specifying a common folder Specify <code>common="true"</code> in the tag attribute and specify the name of the common folder definition in the value</li> </ul>
<output-file-name>	1 time	The name of the copied file is set.

## Legend:

-: There is no corresponding item.

## Note#

If the specified occurrence count is exceeded, the operation is not guaranteed.

● **File and folder deletion operation**

The following table describes the format of a response message, which is returned to the calling business process by a file operations adapter. The file name is "adpfop\_delete\_response.xsd". The name of the namespace is "http://www.hitachi.co.jp/soft/xml/cosminexus/csc/adapter/fileoperation/delete\_response".

Table 3–63: Response message format (file and folder deletion operation)

Tag name	Occurrence count#	Description
<response>	1 time	-
<folder-name>	1 time	<p>The name of the folder in which the deleted file is stored is set.</p> <ul style="list-style-type: none"> <li>When specifying a working folder Specify <code>common="false"</code> in the tag attribute and specify null in the value.</li> <li>When specifying a common folder Specify <code>common="true"</code> in the tag attribute and specify the name of the common folder definition in the value.</li> </ul>
<file-name>	1 time	<p>The name of the deleted file is set.</p> <p>The file name is null, if you delete a working folder.</p>

## Legend:

-: There is no corresponding item.

## Note#

If the specified occurrence count is exceeded, the operation is not guaranteed.

● **File compression operation**

The following table describes the format of a response message, which is returned to the calling business process by a file operations adapter. The file name is "adpfop\_compress\_response.xsd". Name of the namespace is "http://www.hitachi.co.jp/soft/xml/cosminexus/csc/adapter/fileoperation/compress\_response".

Table 3–64: Response message format (file compression operation)

Tag name	Occurrence count#	Description
<response>	1 time	-
<output-folder-name>	1 time	The name of the folder in which the compressed file is stored is set. <ul style="list-style-type: none"> <li>When specifying a working folder Specify <code>common="false"</code> in the tag attribute and specify null in the value.</li> <li>When specifying a common folder Specify <code>common="true"</code> in the tag attribute and specify the name of the common folder definition in the value</li> </ul>
<output-file-name>	1 time	The name of the compressed file is set.

Legend:

-: There is no corresponding item.

Note#

If the specified occurrence count is exceeded, the operation is not guaranteed.

#### • File extraction operation

The following table describes the format of a response message, which is returned to the calling business process by a file operations adapter. The file name is "adpfop\_extract\_response.xsd". Name of the name space is "http://www.hitachi.co.jp/soft/xml/cosminexus/csc/adapter/fileoperation/extract\_response".

Table 3–65: Response message format (file extraction operation)

Tag name	Occurrence count#	Description
<response>	1 time	-
<output-folder-name>	1 time	The name of the folder in which the extracted file is stored is set. <ul style="list-style-type: none"> <li>When specifying a working folder Specify <code>common="false"</code> in the tag attribute and specify null in the value.</li> <li>When specifying a common folder Specify <code>common="true"</code> in the tag attribute and specify the name of the common folder definition in the value.</li> </ul>
<output-files>	1 time	-
<output-file>	1 to 999 times	-
<output-file-name>	1 time	The name of the extracted file is set.
<entry-name>	1 time	The entry name prior to extraction of the file set in output-file-name tag is set.

Legend:

-: There is no corresponding item.

Note#

If the specified occurrence count is exceeded, the operation is not guaranteed.

#### • If a file operations adapter fails to generate a response message

An error message (KDEC80003-E) is to be output and the process is to be terminated.

## (2) Operations to be performed on the service adapter settings screen

The procedure for defining file operations adapters is as follows:

1. Display the service adapter settings screen.  
For details on how to display the service adapter settings screen, see 3.3.1(4) *Displaying the service adapter settings screen*.
2. Set up the definition information in the service adapter settings screen (basic) and the service adapter settings screen (details).  
For details on the items to be set up in each operation, see the following locations:
  - For file transformation operation  
3.3.15(9)(a) *File transformation operation*
  - For file replication operation  
3.3.15(9)(b) *File replication operation*
  - For file and folder deletion operation  
3.3.15(9)(c) *File and folder deletion operation*
  - For file compression operation  
3.3.15(9)(d) *File compression operation*
  - For file extraction operation  
3.3.15(9)(e) *File extraction operation*

Also, for details on the service adapter settings screen (basic), see 1.2.2 *Service adapter settings screen* in the manual *Service Platform Reference Guide*.

---

### Reference note

You can save, edit, and validate the file operations adapter which is created, if necessary.

For details on how to edit a file operations adapter, see 3.5 *Editing Adapters*. For details on how to validate a file operations adapter, see 3.6 *Validating Adapters*.

---

## (3) Creating and editing the file operations adapter definition file

Types of the definition file to be created and edited are as follows:

- File operations adapter runtime-environment property file  
This file is required to define the configuration information of each file operations adapter.
- File operations adapter definition file  
This file is required to define the operation contents of a file operations adapter.

The procedures for creating and editing the definition file used by a file operations adapter are as follows:

### (a) Procedure for creating a file operations adapter runtime-environment property file

The procedure for creating a file operations adapter runtime-environment property file is as follows:

1. Copy the template file (*Installation directory of the service platform\CSC\config\adpfop\templates\adpfop.properties*), and save this file in the following directory:  
*Installation directory of the service platform\CSC\config\adpfop*
2. Change the name of the template file, which is copied as "<Service ID>.properties".  
A service ID is any string to be specified when adding a new file operations adapter.
3. Edit and save the definition contents.  
For details on the items which you can edit in the file operations adapter runtime-environment property file, see *File operations adapter runtime-environment property file* in the manual *Service Platform Reference Guide*.

The file operations adapter runtime-environment property file is reflected in the runtime-environment, when starting a file operations adapter. To change the contents of the file operations adapter runtime-environment property file, stop the file operations adapter and perform the operation. If you restart the file operations adapter, the changed contents are reflected in the execution environment.

(b) Procedure for editing the file operations adapter definition file

The procedure for editing the file operations adapter definition file is as follows:

1. Select "escFileOperation.properties" in the "Self-defined file" of the service adapter settings screen (details), and click the [Edit] button.

The editor used for editing the file operations adapter definition file starts.

2. Edit the file operations adapter definition file using the editor.

For details on the file operations adapter definition file, see *File operations adapter definition file* in the manual *Service Platform Reference Guide*.

3. Select [File] - [Save] from the Eclipse menu, and save the definition contents.

If you start a file operations adapter, the contents set in the file operations adapter definition file become valid.

(4) Points to be considered when using the file transformation operation

When you select the segmentation method in the file transformation operation and handle an input file in the linefeed separator format, do not set LineFeed in the separator of the complex content element of the following files:

- Binary format definition file of header code from input side
- Binary format definition file of data code from input side
- Binary format definition file of trailer code from input side

If you set LineFeed, a message (KDEC80043-E) is output, and the process after file transformation is canceled.

The following figure shows an example of the separator settings screen in which correct settings are performed:

Figure 3–21: Example of separator settings screen (without LineFeed settings)

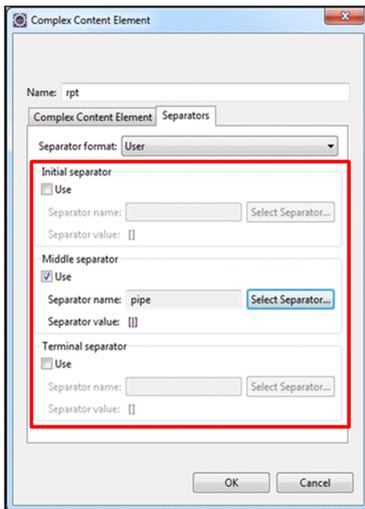
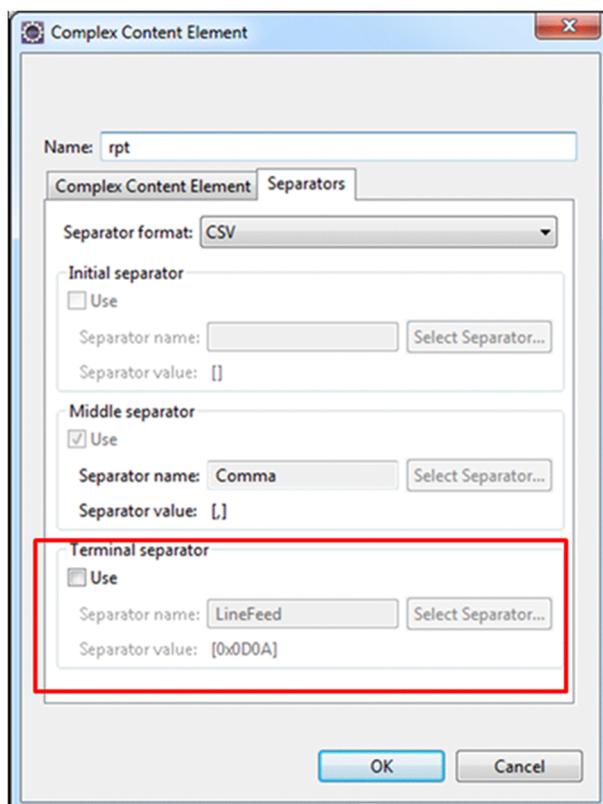


Figure 3–22: Example of the separator settings screen (for CSV format)



### 3.3.12 Defining mail adapters

This subsection describes how to define mail adapters.

#### (1) Notes on using mail adapters

Please note the following points before setting mail adapters.

- Set a valid address for the destination of a mail adapter.
- If you set an invalid address for a mail adapter, all emails sent to that address are bounced by the mail server function.
- If you use a file forwarded from an FTP client or FTP server as an attachment, linkage with the FTP inbound adapter, FTP reception, and FTP adapter is required in addition to linkage with the mail adapter. For details about the function for linking with FTP, see *Chapter 8. Functionality to Transfer Files by Integrating with FTP (FTP Integration)* in the manual *Service Platform Overview*.

#### (2) Creating a message format

The following shows the types and forms of the message formats used by mail adapters.

##### (a) Message format type

Mail adapters use either of the following two message formats:

- Request message format
- Response message format

For the request message format, set the mail server information or user input information (such as the email body, address, and subject). For the response message format, the message ID of the sent mail is set.

A mail adapter uses the XML schema provided by the service platform as the message format. The XML schemata used for request messages and response messages are `adpmail_smtp_request.xsd` and `adpmail_smtp_response.xsd`, respectively.

### (b) Message format

The following describes the request message format and response message format used by mail adapters.

The files are stored in `installation-directory-of-the-service-platform\CSC\schema\mail`.

- `adpmail_smtp_request.xsd` (request message format file)

The format of the request message passed to the mail adapter when the mail adapter is called from a business process is shown below. The name space name is `http://www.hitachi.co.jp/soft/xml/cosminexus/csc/adapter/mail/send_request`.

Table 3–66: Request message format

Tag name	Number of occurrences	Description
<code>&lt;request&gt;</code>	Once	--
<code>&lt;smtp-host-name&gt;</code>	0 or once	<p>A tag that indicates the IP address or host name of the mail server to connect with.</p> <ul style="list-style-type: none"> <li>• When the tag and the value exist: Connection is performed by using the specified character string.</li> <li>• When the tag exists but the value does not exist: A system exception is returned.</li> <li>• When the tag does not exist: The value in the mail adapter runtime environment property file becomes valid.</li> </ul>
<code>&lt;request-id&gt;</code>	0 or once	<p>A tag that indicates the request ID created at the reception.</p> <ul style="list-style-type: none"> <li>• When the tag and the value exist: The value is used as the request ID.</li> <li>• When the tag does not exist, or when the tag exists but the value does not exist: If a work folder is used, a system exception is returned. If the common folder is used, this tag is ignored.</li> </ul>
<code>&lt;smtp-port&gt;</code>	0 or once	<p>A tag that indicates the port number of the mail server to connect with.</p> <ul style="list-style-type: none"> <li>• When the tag and the value exist: Specify a port number in the range from 1 to 65535. If you specify a value outside this range, a system exception is returned.</li> <li>• When the tag exists but the value does not exist: The default value 25 is specified.</li> <li>• When the tag does not exist: The value in the mail adapter runtime environment property file becomes valid. If this tag is also not specified in the mail adapter runtime environment property file, the default value 25 is specified.</li> </ul>
<code>&lt;mail-user&gt;</code>	0 or once	<p>A tag that indicates the login user name of the mail server to connect with.</p> <ul style="list-style-type: none"> <li>• When the tag and the value exist: Characters specified in the range from 0x20 to 0x7E are valid. If you specify characters outside this range, a system exception is returned.</li> <li>• When the tag exists but the value does not exist: A system exception is returned.</li> <li>• When the tag does not exist: The value in the mail adapter runtime environment property file becomes valid.</li> </ul>

Tag name	Number of occurrences	Description
<code>&lt;mail-addr-to&gt;#1</code>	0 or once	<p>A tag that indicates the email address of the user the email is sent to. If you specify multiple email addresses, use a comma ( , ) as the delimiter for the email addresses.</p> <ul style="list-style-type: none"> <li>When the tag and the value exist: The value is used as the address for the <code>TO</code> field. The address is not validated.</li> <li>When the tag exists but the value does not exist: A system exception is returned.</li> <li>When the tag does not exist: The value in the mail adapter runtime environment property file becomes valid.</li> </ul>
<code>&lt;mail-addr-from&gt;</code>	0 or once	<p>A tag that indicates the email address of the user who sends the email. You can specify only one email address.</p> <ul style="list-style-type: none"> <li>When the tag and the value exist: The value is used as the address for the <code>FROM</code> field. The address is not validated.</li> <li>When the tag exists but the value does not exist: A system exception is returned.</li> <li>When the tag does not exist: The value in the mail adapter runtime environment property file becomes valid.</li> </ul>
<code>&lt;mail-addr-cc&gt;#1</code>	0 or once	<p>A tag that indicates the email address of the user who is set for the <code>CC</code> field. If you specify multiple email addresses, use a comma ( , ) as the delimiter for the email addresses.</p> <ul style="list-style-type: none"> <li>When the tag and the value exist: The value is used as the address for the <code>CC</code> field. The address is not validated.</li> <li>When the tag exists but the value does not exist: A system exception is returned.</li> <li>When the tag does not exist: The value in the mail adapter runtime environment property file becomes valid.</li> </ul>
<code>&lt;mail-addr-bcc&gt;#1</code>	0 or once	<p>A tag that indicates the email address of the user who is set for the <code>BCC</code> field. If you specify multiple email addresses, use a comma ( , ) as the delimiter for the email addresses.</p> <ul style="list-style-type: none"> <li>When the tag and the value exist: The value is used as the address for the <code>BCC</code> field. The address is not validated.</li> <li>When the tag exists but the value does not exist: A system exception is returned.</li> <li>When the tag does not exist: The value in the mail adapter runtime environment property file becomes valid.</li> </ul>
<code>&lt;mail-subject-text&gt;</code>	0 or once	<p>A tag that indicates the subject of the email.</p> <ul style="list-style-type: none"> <li>When the tag and the value exist: The value is used as the subject of the email. Note, however, that if the number of characters for the subject exceeds the size specified in the mail adapter runtime environment property file, the exceeded characters will be discarded.</li> <li>When the tag exists but the value does not exist: The email is sent without a subject.</li> <li>When the tag does not exist: The value in the mail adapter runtime environment property file becomes valid. If this tag is also not specified in the mail adapter runtime environment property file, the email is sent without a subject.</li> </ul>

### 3. Defining Adapters

Tag name	Number of occurrences	Description
<mail-body>#2	0 or once	<p>A tag that indicates the body of the email. Specify the email body by using &lt;message&gt; and &lt;file&gt;.</p> <ul style="list-style-type: none"> <li>When the tag and the value exist: The value is used as the body of the email.<sup>#3</sup></li> <li>When the tag exists but the value does not exist: A system exception is returned.</li> <li>When the tag does not exist: The value in the mail adapter runtime environment property file becomes valid. If this tag is also not specified in the mail adapter runtime environment property file, the email is sent without a body.</li> </ul>
<message>	0 or once	<p>A tag that indicates the character string used as the email body. Before you use a character such as &lt; or &gt;, sanitize the character string.</p> <ul style="list-style-type: none"> <li>When the tag and the value exist: The value is used as the body of the email.</li> <li>When the tag exists but the value does not exist, or there is no &lt;file&gt; specification: The email is sent without a body.</li> <li>When the tag does not exist: The value in the mail adapter runtime environment property file becomes valid.</li> </ul>
<file>	0 or once	<p>A tag that indicates the path to the file whose contents are used as the email body and the character set. Specify the file path and the character set by &lt;folder-name&gt;, &lt;mail-body-file-name&gt;, and &lt;file-charset&gt;.</p> <ul style="list-style-type: none"> <li>When the tag and the value exist: The value is used as the email body. If you use the contents of the file as the email body, you do not have to sanitize the character string.</li> <li>When the tag exists but the value does not exist: A system exception is returned.</li> <li>When the tag does not exist: The value in the mail adapter runtime environment property file becomes valid.</li> </ul>
<folder-name>	Once	<p>A tag that indicates the folder in which the file is to be stored.</p> <ul style="list-style-type: none"> <li>When <code>common="true"</code> is specified as the attribute of the tag: Use a common folder as the folder to which the file is to be stored, and specify the common folder definition name for this tag. If the common folder cannot be acquired from the specified common folder definition name, a system exception is returned.</li> <li>When <code>common="false"</code> is specified as the attribute of the tag: Use a work folder as the folder to which the file is to be stored.</li> <li>When the <code>common</code> attribute does not exist, the value of the <code>common</code> attribute is neither <code>true</code> nor <code>false</code>, or there is no tag: A system exception is returned.</li> </ul>
<mail-body-file-name>	Once	<p>A tag that indicates the file name to be acquired.</p> <p>Specify the file directly under the work folder or common folder. You can use a delimiter, such as a slash (/) or back slash (\) only for the beginning of the file name. If you use a slash (/) at the beginning of the file name, the slash (/) is ignored. Specify a file that is not a symbolic link. You cannot specify multiple file names.</p> <ul style="list-style-type: none"> <li>When the tag exists and the value conforms to the above restriction: The value is used as the file name.</li> </ul>

Tag name	Number of occurrences	Description
<mail-body-file-name>	Once	<ul style="list-style-type: none"> <li>When the tag exists and the value violates the above restriction: A system exception is returned.</li> <li>When the tag does not exist, or when the tag exists but the value does not exist: A system exception is returned.</li> </ul>
<file-charset>	0 or once	<p>Specify the character set of the file containing the email body text.</p> <ul style="list-style-type: none"> <li>When the tag and the value exist: The value is used as the character set.</li> <li>When the tag exists but the value does not exist: A system exception is returned.</li> <li>When the tag does not exist: The value in the mail adapter runtime environment property file becomes valid. If this tag is also not specified in the mail adapter runtime environment property file, UTF-8 is used as the character set.</li> </ul>
<attachment-file>	0 or once	<p>A tag that indicates the email attachment. Specify the attachment by using &lt;item&gt;. You can specify multiple attachments.</p> <ul style="list-style-type: none"> <li>When the tag and the value exist: The value is used as the email attachment.</li> <li>When the tag exists but the value does not exist: A system exception is returned.</li> <li>When the tag does not exist: The value in the mail adapter runtime environment property file becomes valid.</li> </ul>
<item>	Once or more	<p>A tag that indicates the details of the email attachment. Specify the details of the attachment by using &lt;attach-folder&gt; and &lt;attach-file-name&gt;.</p> <ul style="list-style-type: none"> <li>When the tag and the value exist: The value is used as the email attachment.</li> <li>When the tag does not exist, or when the tag exists but the value does not exist: A system exception is returned.</li> </ul>
<attach-folder>	Once	<p>A tag that indicates the folder in which the attachment is stored.</p> <ul style="list-style-type: none"> <li>When <code>common="true"</code> is specified for the attribute of the tag: Use a common folder as the folder to which the file is to be stored, and specify the common folder definition name for this tag. If the common folder cannot be acquired from the specified common folder definition name, a system exception is returned.</li> <li>When <code>common="false"</code> is specified for the attribute of the tag: Use a work folder as the folder to which the file is to be stored.</li> <li>When the <code>common</code> attribute does not exist, the value of the <code>common</code> attribute is neither <code>true</code> nor <code>false</code>, or there is no tag: A system exception is returned.</li> </ul>
<attach-file-name>	Once	<p>A tag that indicates the name of the attachment.</p> <p>Specify the file directly under the work folder or common folder. You can use a delimiter, such as a slash (/) or back slash (\), only for the beginning of the file name. If you use a slash (/) at the beginning of the file name, the slash (/) is ignored. Specify a file that is not a symbolic link. You cannot specify multiple file names.</p> <ul style="list-style-type: none"> <li>When the tag exists and the value conforms to the above restriction: The value is used as the file name.</li> <li>When the tag exists and the value violates the above restriction:</li> </ul>

### 3. Defining Adapters

Tag name	Number of occurrences	Description
<div style="border: 1px solid black; padding: 2px; width: fit-content;"> <span style="font-family: monospace;">&lt;attach-file-name&gt;</span> </div>	Once	A system exception is returned. <ul style="list-style-type: none"> <li>When the tag does not exist, or when the tag exists but the value does not exist: A system exception is returned.</li> </ul>
<div style="border: 1px solid black; padding: 2px; width: fit-content;"> <span style="font-family: monospace;">&lt;mail-format&gt;</span> </div>	0 or once	A tag that indicates the email format. Specify one of the following: <code>text/plain</code> : Creates the email body in plain text. <code>text/html</code> : Creates the email body in html format. <code>text/xml</code> : Creates the email body in xml format. <sup>#4</sup> <ul style="list-style-type: none"> <li>When the tag exists and the value is <code>text/plain</code>, <code>text/html</code>, or <code>text/xml</code>: The value is used as the email format.</li> <li>When the tag exists and the value is not <code>text/plain</code>, <code>text/html</code>, or <code>text/xml</code>: A system exception is returned.</li> <li>When the tag exists but the value does not exist: The value depends on the JavaMail settings.</li> <li>When the tag does not exist: The value in the mail adapter runtime environment property file becomes valid. If this tag is also not specified in the mail adapter runtime environment property file, the email body is created by using <code>text/plain</code>.</li> </ul>
<div style="border: 1px solid black; padding: 2px; width: fit-content;"> <span style="font-family: monospace;">&lt;mail-encoding&gt;</span> </div>	0 or once	A tag that indicates the encoding of the email. <ul style="list-style-type: none"> <li>When the tag and the value exist: The value is used as the encoding of the email.</li> <li>When the tag exists but the value does not exist: The value depends on the JavaMail settings.</li> <li>When the tag does not exist: The value in the mail adapter runtime environment property file becomes valid. If this tag is also not specified in the mail adapter runtime environment property file, ISO-2022-JP is used.</li> </ul>
<div style="border: 1px solid black; padding: 2px; width: fit-content;"> <span style="font-family: monospace;">&lt;mail-auth&gt;</span> </div>	0 or once	A tag that indicates the SMTP authentication format. Specify either of the following: <code>PLAIN</code> : Authentication is not performed. <code>LOGIN</code> : The user name and password are used for authentication when the user logs in. <ul style="list-style-type: none"> <li>When the tag exists and the value is <code>PLAIN</code> or <code>LOGIN</code>: The value is used as the SMTP authentication format.</li> <li>When the tag exists and the value is not <code>PLAIN</code> or <code>LOGIN</code>: A system exception is returned.</li> <li>When the tag exists but the value does not exist: The value depends on the JavaMail settings.</li> <li>When the tag does not exist: The value in the mail adapter runtime environment property file becomes valid. If this tag is also not specified in the mail adapter runtime environment property file, the email is sent in <code>PLAIN</code> format.</li> </ul>
<div style="border: 1px solid black; padding: 2px; width: fit-content;"> <span style="font-family: monospace;">&lt;mail-option&gt;</span> </div>	0 or more	A tag that indicates the user-defined email header. Specify the header name and value by combining <code>&lt;option-key&gt;</code> and <code>&lt;option-value&gt;</code> .

Tag name	Number of occurrences	Description
<mail-option>	0 or more	<ul style="list-style-type: none"> <li>When the tag and the value exist: The value is used as the user-defined email header.</li> <li>When the tag exists but the value does not exist: A system exception is returned.</li> <li>When the tag does not exist: The value in the mail adapter runtime environment property file becomes valid. If this tag is also not specified in the mail adapter runtime environment property file, the email is sent without a user-defined email header.</li> </ul>
<option-key>	Once	Specify the key name of the user-defined email header. <ul style="list-style-type: none"> <li>When the tag and the value exist: The value is used as the key name of the email header.</li> <li>When the tag does not exist, or when the tag exists but the value does not exist: A system exception is returned.</li> </ul>
<option-value>	Once	Specify the value of the user-defined email header. <ul style="list-style-type: none"> <li>When the tag and the value exist: The value is used as the value of the mail header.</li> <li>When the tag does not exist, or when the tag exists but the value does not exist: A system exception is returned.</li> </ul>

## Legend:

--: There is no relevant item.

## #1

If no address is specified for <mail-addr-to>, <mail-addr-cc>, and <mail-addr-bcc>, and this tag is not specified in the mail adapter runtime environment property file, a system exception is returned.

## #2

The settings of <mail-body> in a request message and the settings of the `mailadp.mail.body-text` key and `mailadp.mail.body-filepath` key in the mail adapter runtime environment property file are not linked. For example, say that in <mail-body> of a request message, you specify only <message> and do not specify <file>. In this case, even if you specify a file for the `mailadp.mail.body-filepath` key, <file> will not be set for the body.

## #3

When you specify both <message> and <file>, the character string specified for <message> is set for the email body first. Then, the contents of the file specified for <file> are set for the email body.

## #4

When you specify `text/xml`, the email body is displayed in plain text format in the mail adapter.

- adpmail\_smtp\_response.xsd (response message format file)

The format of the response message that the mail adapter returns to the caller business process is shown below. The name space name is `http://www.hitachi.co.jp/soft/xml/cosminexus/csc/adapter/mail/send_response`.

Table 3-67: Response message format

Tag name	Number of occurrences	Description
<response>	Once	--
<message-id>	Once	A tag that indicates the message ID of the sent email message. When an email is sent, the internally created MimeMessage is sent to the mail server. The ID of this MimeMessage will be the message ID.

## Legend:

--: There is no relevant item.

#### (3) Creating the mail adapter definition file

The following are the types of definition files to be created:

- Mail adapter command definition file  
This file is used to set the output destination of the command message log and the output level.
- Mail adapter account definition file  
This file is used to manage the user name and password that are required when connecting from the mail adapter to the mail server.
- Email header definition file  
This file is used to write user-defined email header options, such as the importance of an email and whether to respond to a request.
- Mail adapter runtime environment property file  
This file is used to define configuration information of the mail adapter.

The following describes how to create individual definition files.

##### (a) Creating the mail adapter command definition file

1. Copy the template file (*installation-directory-of-the-service-platform\CSC\config\mail\templates\adpmailcommand.properties*), and store it in the following directory:  
*installation-directory-of-the-service-platform\CSC\config\mail*
2. Edit and save the definitions.

For details about the definitions that can be edited in the mail adapter command definition file, see *Mail adapter command definition file* in the manual *Service Platform Reference Guide*.

##### (b) Creating the mail adapter account definition file

1. Execute the `csmmailaddusr` command to create a mail adapter account definition file.  
Specify the path to the created mail adapter account definition file in the `mailadp.account.file.path` key in the mail adapter runtime environment property file. For details about the `csmmailaddusr` command, see *csmmailaddusr (Registering and updating the mail adapter user information)* in the manual *Service Platform Reference Guide*.

##### (c) Creating the email header definition file

1. Copy the template file (*installation-directory-of-the-service-platform\CSC\config\mail\templates\mail-header.properties*), and store it in any directory.
2. Rename the copied template file to any name.
3. Edit and save the definitions.

For details about the definitions that can be edited in the email header definition file, see *Mail header definition file* in the manual *Service Platform Reference Guide*.

##### (d) Creating the mail adapter runtime environment property file

1. Copy the template file (*installation-directory-of-the-service-platform\CSC\config\mail\templates\adpmail.properties*), and store it in the following directory:  
*installation-directory-of-the-service-platform\CSC\config\mail*
2. Rename the copied template file to *service-ID#.properties*.
3. Edit and save the definitions.

For details about the definitions that can be edited in the mail adapter runtime environment property file, see *Mail adapter runtime environment property file* in the manual *Service Platform Reference Guide*.

#

*service-ID* is a character string specified in the Service Adapter Settings window when a new mail adapter is added.

The contents of the mail adapter runtime environment property file are applied to the execution environment when the mail adapter starts operating. Therefore, if you change the contents of the mail adapter runtime environment property file, you need to temporarily stop the mail adapter.

#### (4) Operations in the Service Adapter Settings window

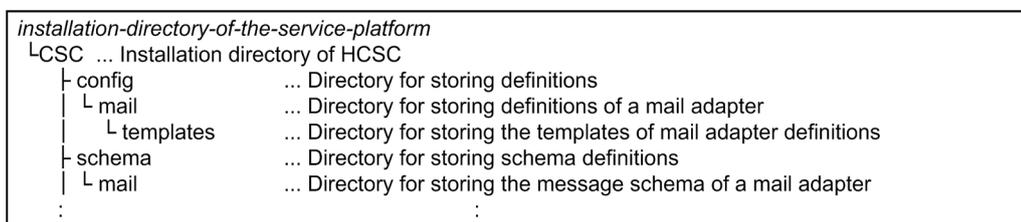
To define a mail adapter:

1. Open the Service Adapter Settings window.  
For details about how to open the Service Adapter Settings window, see 3.3.1(4) *Displaying the service adapter settings screen*.
2. In the Service Adapter Settings (standard) window, set the definition information.  
For details about the items that need to be set in the Service Adapter Settings (standard) window, see 3.3.15(10) *For mail adapters* and 1.2.2 *Service Adapter Definition Window* in the manual *Service Platform Reference Guide*.
3. Click the **Service-adapter definition (details)** tab.  
The Service Adapter Settings (details) window appears.
4. In the Service Adapter Settings (details) window, set the definition information.  
For details about the items that need to be set in the Service Adapter Settings (details) window, see 3.3.15(10) *For mail adapters*.

#### (5) Directory structure of a mail adapter

The figure below shows the directory structure after the service platform is installed. Note that only the directories for storing the definitions and message schema of a mail adapter are shown in this figure.

Figure 3–23: Directory structure of a mail adapter



For details about directory structures other than the above, see 2.1.2 *Installing Service Platform* in the manual *Service Platform Setup and Operation Guide*.

#### (6) Example of defining mail adapters

Below are examples of setting a mail adapter and business process. In these examples, the message passed to the standard reception is set to the email body, and the email is sent to the destination set in the mail adapter runtime environment property file.

##### (a) Example of setting a mail adapter

Below are examples of setting a mail adapter in the development environment and the mail adapter runtime environment property file.

##### ■ Setting a mail adapter in the development environment

Set the individual items in the Service Adapter Settings window in the development environment as shown below. For details about the storage location of the email format and definition file, and how to operate in the development environment, see 3.2.12 *Adding a new mail adapter*.

Figure 3–24: Example of configuring the Service Adapter Settings (standard) window

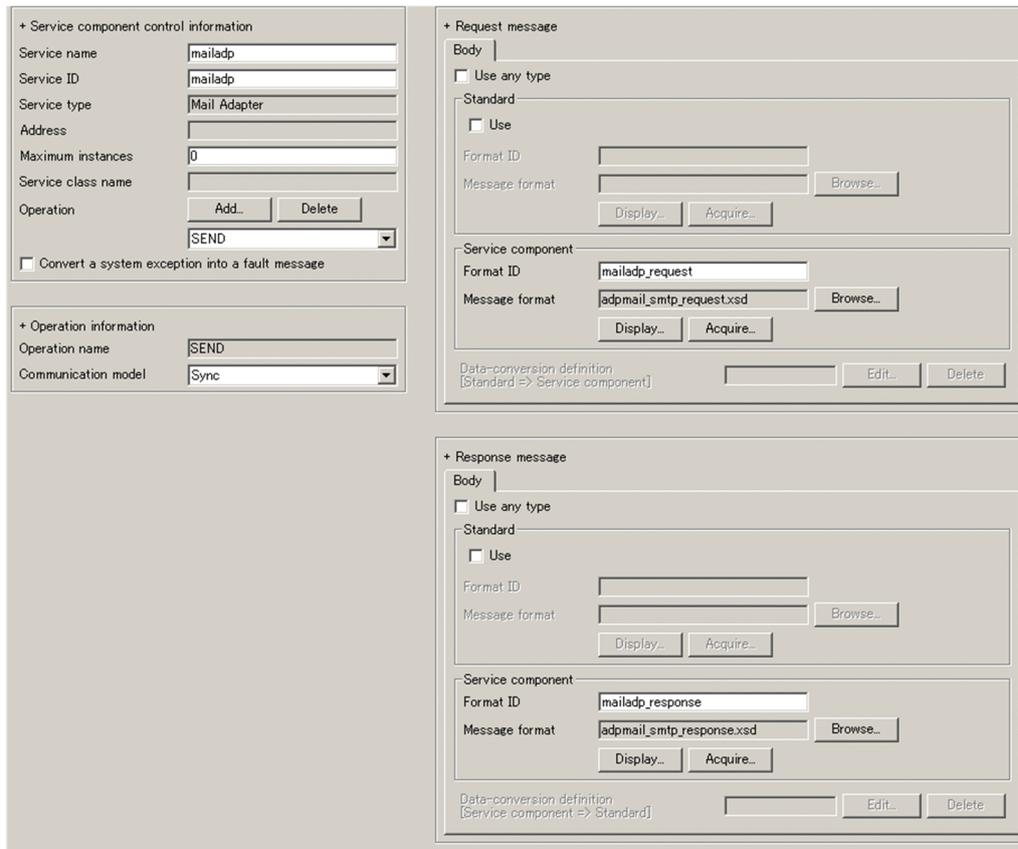


Table 3–68: Settings in the Service Adapter Settings (standard) window

Category	Item	Setting	
<b>Service component control information</b>	<b>Service name</b>	mailadp	
	<b>Service ID</b>	mailadp	
	<b>Service type</b>	Mail adapter	
	<b>Address</b>	--	
	<b>Maximum instances</b>	0	
	<b>Service class name</b>	--	
	<b>Operation</b>	SEND	
<b>Operation information</b>	<b>Operation name</b>	SEND	
	<b>Communication model</b>	Sync	
<b>Request message</b>	<b>Use any type</b> check box	Do not use (leave the check box cleared)	
	<b>Standard</b>	<b>Use</b> check box	Do not use (leave the check box cleared)
		<b>Format ID</b>	--
		<b>Message format</b>	--
	<b>Service component</b>	<b>Format ID</b>	mailadp_request

Category	Item		Setting
<b>Request message</b>	<b>Service component</b>	<b>Message format</b>	adpmail_smtp_request.xsd
	<b>Data-conversion definition</b>		--
<b>Response message</b>	Use any type check box		Do not use (leave the check box cleared)
	<b>Standard</b>	Use check box	Do not use (leave the check box cleared)
		<b>Format ID</b>	--
		<b>Message format</b>	--
	<b>Service component</b>	<b>Format ID</b>	mailadp_response
		<b>Message format</b>	adpmail_smtp_response.xsd
<b>Data-conversion definition</b>		--	
<b>Fault message</b>	--		--

Legend:

--: The item is not set, or there is no relevant item.

Figure 3–25: Example of setting the Service Adapter Settings (details) window

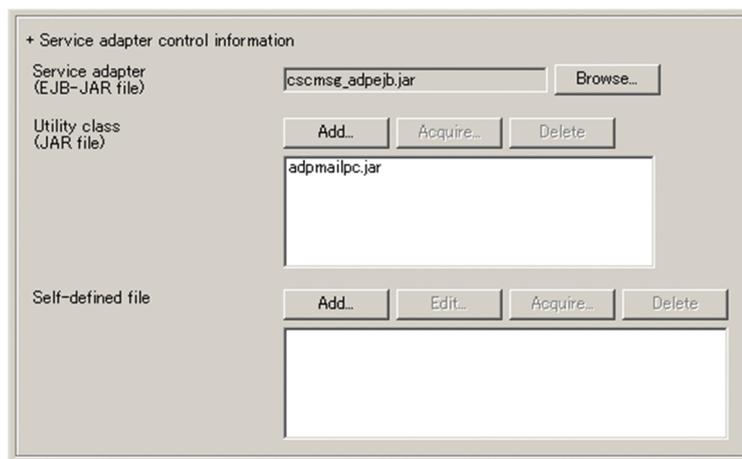


Table 3–69: Setting items in the Service Adapter Settings (details) window

Category	Item	Setting
<b>Service adapter control information</b>	<b>Service adapter (EJB-JAR file)</b>	cscmsg_adpejb.jar
	<b>Utility class (JAR file)</b>	adpmailpc.jar
	<b>Self-defined file</b>	--

Legend:

--: The item is not set.

#### ■ Creating the mail adapter runtime environment property file

Create the mail adapter runtime environment property file that corresponds to the mail adapter defined in the development environment. In the example below, the file name `mailadp.properties` is used.

For details about how to create the mail adapter runtime environment property file, see 3.3.12(3) *Creating the mail adapter definition file*.

### 3. Defining Adapters

An example of creating the mail adapter runtime environment property file is shown below. Italics indicate modified parts.

---

```
# All Rights Reserved. Copyright (C) 2011, Hitachi, Ltd.

mailadp.smtp.host.name = mail-server
#mailadp.smtp.port = 25
#mailadp.user =
mailadp.addr.from = sampleuser2@XXXXXXX.com
mailadp.addr.to = sampleuser@XXXXXXX.com
#mailadp.addr.cc =
#mailadp.addr.bcc =
#mailadp.addr.count.max = 768
#mailadp.mail.body.size = 1024000
#mailadp.attach.path.1 =
#mailadp.attach.file.size = 5242880
#mailadp.max.attach.count.per.mail = 10
#mailadp.mail.format = text/plain
#mailadp.mail.body-text =
#mailadp.mail.body-filepath =
#mailadp.mail.body-filecharset = UTF-8
#mailadp.subject.size = 80
#mailadp.subject.text =
mailadp.smtp.encoding = ISO-2022-JP
#mailadp.smtp.auth = PLAIN
#mailadp.user.header.path =
#mailadp.account.file.path =
#mailadp.methodtrace.level = 3
#mailadp.methodtrace.filepath =
#mailadp.methodtrace.fileenum =
#mailadp.methodtrace.filesize = 2097152
#mailadp.messagelog.level = 10
#mailadp.smtp.timeout = 180
#mailadp.smtp.connectiontimeout = 180
#mailadp.read-lock.retry.count = 0
#mailadp.read-lock.retry.interval = 1
```

---

The created mail adapter runtime environment property file is stored in the *installation-directory-of-the-service-platform\CSC\config\mail* folder.

#### (b) Example of setting a business process

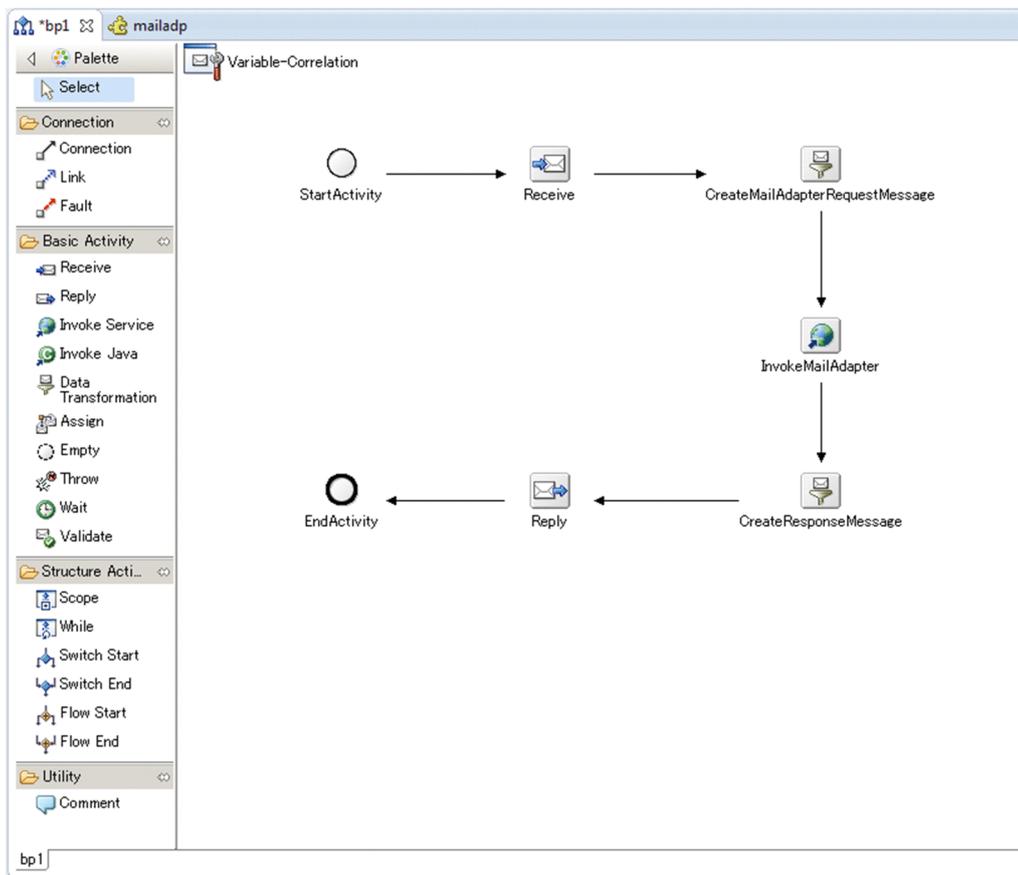
Below are examples of setting a business process in the development environment and data transformation definition.

For details about the window used in the development environment, see *Chapter 1. Windows (Development Environment)* in the manual *Service Platform Reference Guide*. For details about how to operate in the development environment, see *Chapter 5. Defining Business Processes* and *Chapter 6. Defining Data Transformation* in the manual *Service Platform Basic Development Guide*.

- Setting the Define Business Process window

In the Define Business Process window in the development environment, define an activity as follows.

Figure 3–26: Example of setting a business process



- Setting variables

Create variables for service component messages and standard messages, that are necessary for passing messages within a business process. Note that this setting example does not use a correlation set.

- Creating variables for service component messages

In the List Of Variables And Correlation Sets dialog box of the Define Business Process window, add the following variables.

Table 3–70: Variables for service component message

Variable name	Description
mail_input	A variable for request messages of the mail adapter
mail_output	A variable for response messages of the mail adapter

Always set the type XML. Set each variable as follows in the Take In Message Format dialog box.

Table 3–71: Setting values in the Take In Message Format dialog box

Item name	Variable name	
	mail_input	mail_output
<b>Service/Reception</b>	Service name	Service name
<b>Service name/Reception name</b>	mailadp	mailadp
<b>Operation name</b>	SEND	SEND
<b>Message type</b>	Request message (Body)	Response message (Body)

Item name	Variable name	
	mail_input	mail_output
Message format	mail_input	mail_output

- Creating variables for standard messages

Create the message formats for request messages and response messages for standard messages, by using the following names and content:

- input.xsd

```
<?xml version="1.0" encoding="UTF-8" ?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="SampleSOAPinput">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="MAIL_BODY" type="xsd:string"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

- output.xsd

```
<?xml version="1.0" encoding="UTF-8" ?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="SampleSOAPoutput">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="MESSAGE_ID" type="xsd:string"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

Then, in the List Of Variables And Correlation Sets dialog box of the Define Business Process window, add the following variables.

Table 3–72: Variables for standard messages

Variable name	Description
input	Variable for request messages for standard reception
output	Variable for response messages for standard reception

Always set the type XML.

Then, in the List Of Variables And Correlation Sets dialog box, click the ... button, and then specify the created message format as follows for each variable.

Table 3–73: Message format to be specified

Variable name	Message format to be specified
input	input.xsd
output	output.xsd

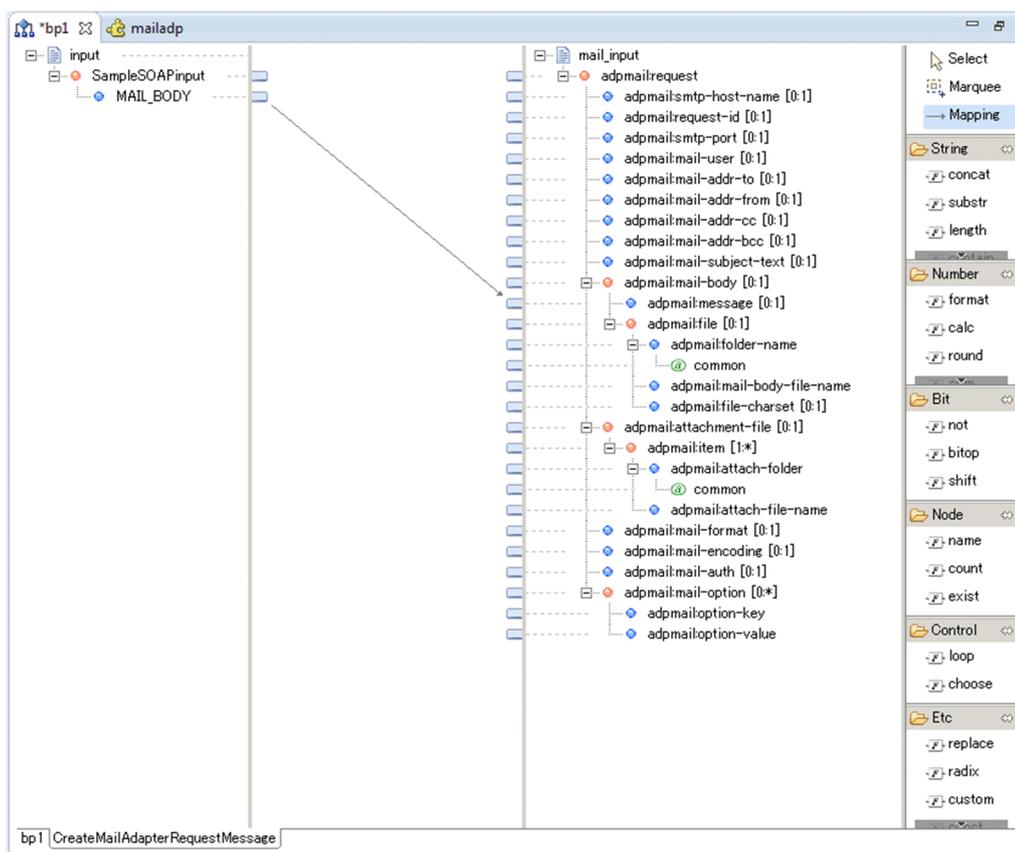
- Setting the data transformation definition

In the data transformation activity (CreateMailAdapterRequestMessage), you can perform mapping for creating a request message of the mail adapter from a request message of the standard reception. In the same way, in the data transformation activity (CreateResponseMessage), you can perform mapping for creating a response message of the standard adapter from a response message of the mail adapter.

First, right-click the data transformation activity (CreateMailAdapterRequestMessage) in the Define Business Process window, and then select **Launch & mapping definition**.

Map the MAIL\_BODY element in the request message of the standard reception to the message element, which is the child element of the mail-body element in the request message of the mail adapter.

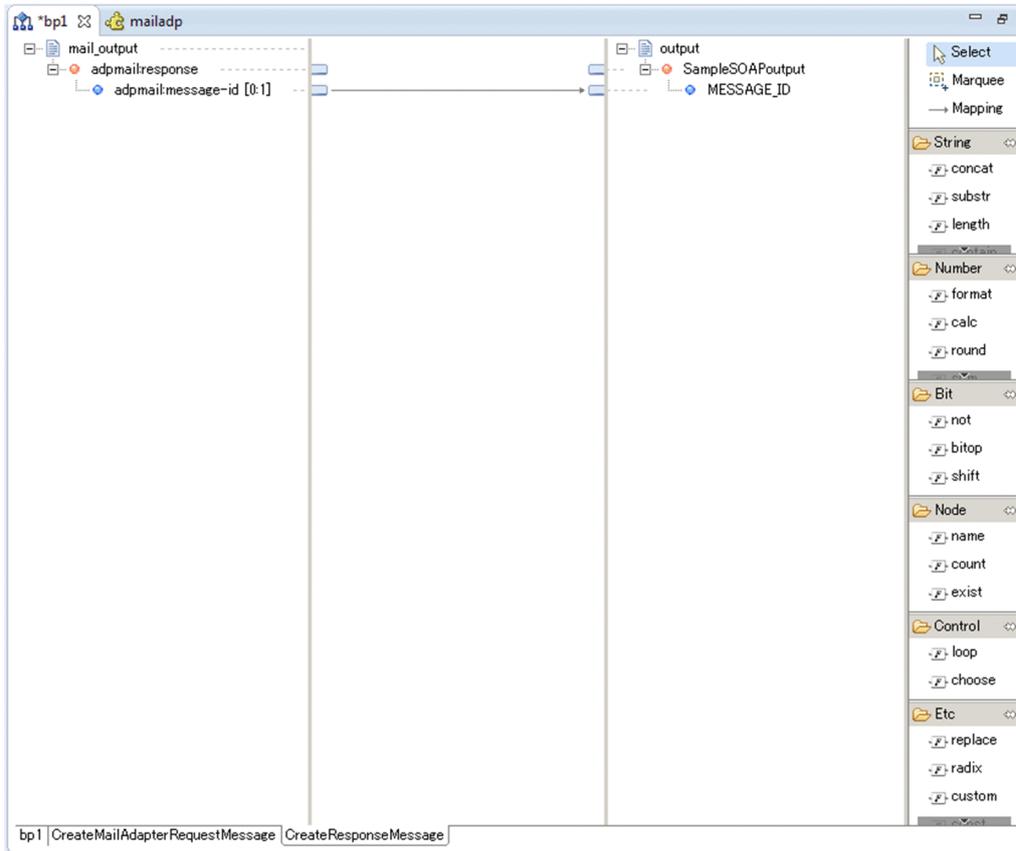
Figure 3–27: Mapping for the request message



In the same way, right-click the data transformation activity (CreateResponseMessage) in the Define Business Process window, and then select **Launch & mapping definition**.

Map the `message-id` element in the response message of the mail adapter to the `MESSAGE_ID` element in the response message of the standard reception.

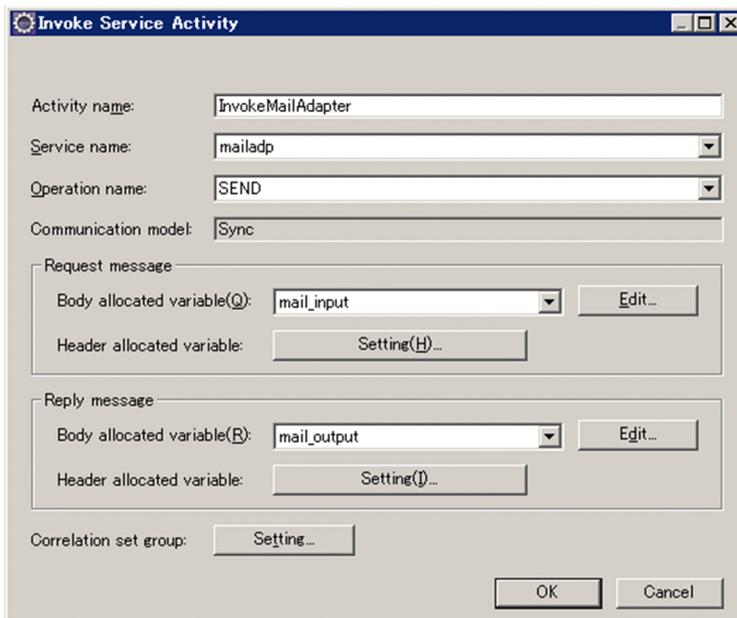
Figure 3–28: Mapping for the response message



- Setting the invoke service activity

In the Define Business Process window, double-click the invoke service activity (InvokeMailAdapter), and then make the following settings in the displayed Invoke Service Activity dialog box.

Figure 3–29: Setting values for the invoke service activity



The following table describes the settings.

Table 3–74: Settings of the invoke service activity

Category	Item	Setting value
<b>Activity name</b>		InvokeMailAdapter
<b>Service name</b>		mailadp
<b>Operation name</b>		SEND
<b>Communication model</b>		Sync
<b>Request message</b>	<b>Body allocated variable</b>	mail_input
	<b>Header allocated variable</b>	--
<b>Response message</b>	<b>Body allocated variable</b>	mail_output
	<b>Header allocated variable</b>	--
<b>Correlation set group</b>		--

Legend:

--: The item is not set.

This completes the mail adapter and business process settings.

To execute the business process in this example setting, you need to specify a request message for the standard reception from the service requester, and then send the message. For details about how to create a service requester, see *8.2 Service Requester That Sends Requests to a Standard Synchronous Reception (Web Services) (SOAP communication infrastructure)* in the manual the *Service Platform Basic Development Guide*.

After the above operation, perform validation and packaging of the component, perform deployment definition in the test environment, and then check the operation. For details, see *Chapter 7. Packaging HCSC Components and Defining Deployment* in the manual *Service Platform Basic Development Guide*.

### 3.3.13 Defining HTTP adapters

This subsection describes how to define HTTP adapters. To confirm the operations of HTTP adapter on the basis of an actual setup example, see *Appendix G Example for setting up business processes using HTTP reception and HTTP adapter*.

#### (1) Setting up the option definition file for J2EE servers

To operate an HTTP adapter in the execution environment, add the library (`cjjaxrs.jar`) of JAX-RS to the class path of a J2EE server. Edit the option definition file for a J2EE server (`usrconf.cfg`) and describe the following contents:

---

```
add.class.path=Installation directory of the service platform\jaxrs\lib\cjjaxrs.jar
```

---

For details on the option definition file for the J2EE server (`usrconf.cfg`), see *2.3 usrconf.cfg (option definition file for a J2EE server)* in the manual *Application Server Definition Reference Guide*.

#### (2) Creating message formats

The types, formats, and methods to create message formats to be used in the HTTP adapter are described here.

##### (a) Message format type

The following table describes the message format types to be used in the HTTP adapter:

Table 3–75: Message format types

Major type	Intermediate type	Minor type	Description	Editable
Request message format	For header variable	Request message format(for header variables)	This is the message format of header assigned variables that specify the information on the HTTP request header. Use the following schema files provided by the service platform: <ul style="list-style-type: none"> <li>• <code>adphttp_header_request1.xsd</code> Use to set Cookie elements in a batch.</li> <li>• <code>adphttp_header_request2.xsd</code> Use to set Cookie elements individually.</li> </ul> Both the schema files include <code>adphttp_header_query_detail.xsd</code> .	N
		Detailed message format for a query	This is the message format to set the query information. The service platform provides the schema file <code>adphttp_header_query_detail.xsd</code> . Use the file by editing the parameter settings.	A
	For body variable	Message format for the file data	This is the message format to be set if the HTTP request body is not handled when you send only file data. The service platform provides the schema file <code>adphttp_body_empty.fdx</code> .	N
		Message format for the form data	This is the message format used to set the form data in the HTTP request body. The service platform provides the schema file <code>adphttp_body_form-data.xsd</code> . You use the file by editing the parameter settings.	A
		Message format for the pass-through mode	Create this message format to send any HTTP request body with the pass-through mode. In this case, schema files are not provided in the service platform.	Y
	Response message format	For header variable	Response message format(for header variables)	This is the message format to assign the received HTTP response header and the status line. Use the following schema files provided by the service platform: <ul style="list-style-type: none"> <li>• <code>adphttp_header_response1.xsd</code> Use to set Set-Cookie elements in a batch.</li> <li>• <code>adphttp_header_response2.xsd</code> Use to set Set-Cookie individually.</li> </ul>
For body variable			Message format for the file data	This is the message format to be set if the HTTP response body is not handled when you send only file data. The service platform provides the schema file <code>adphttp_body_empty.fdx</code> .
		Message format for the pass-through mode	Create this message format to send any HTTP response body with the pass-through mode. No schema files are provided in the service platform.	Y
Fault message format			This is the message format to assign any received status code to the fault information The service platform provides the schema file <code>adphttp_fault.xsd</code> .	N

Legend:

Y: Created by User.

A: Can be edited, if necessary.

N: Cannot be edited. Operations to be performed on editing the message format are not guaranteed.

### (b) Message format

The formats of a request message, a response message, and a fault message used in the HTTP adapter are described here.

#### • **adphttp\_header\_request1.xsd/adphttp\_header\_request2.xsd (request message format (for header variable))**

The following table describes the request message format of a header variable, which is passed on to an HTTP adapter by a business process. Name of the name space is "http://www.hitachi.co.jp/soft/xml/cosminexus/csc/adapter/http/header\_request".

Table 3–76: Request message format (for header variables)

Tag name	Type	Occurrence count	Description
<http-header-request>	-	1 time	-
<request-id>	string	0 or 1 time	Set the request ID generated in the HTTP reception or the FTP reception. Required to send the file data in the working folder and to receive a file in the working folder.
<method>#1	string	0 or 1 time	Specify the HTTP method of the request line. You can specify any of the following values: <ul style="list-style-type: none"> <li>• GET</li> <li>• HEAD</li> <li>• POST</li> <li>• OPTIONS</li> <li>• PUT</li> <li>• DELETE</li> </ul> If you set any other value, operations are not guaranteed.
<uri-scheme-authority>#1	string	0 or 1 time	Specify the scheme and the authority (upto the port number of URI) of URI of the request line. URL encoding of the value is not performed. It becomes the first character string when composing the URI.
<uri-path>#1	string	0 or 1 time	Specify the path of the URI of the request line. As a general rule, you must specify "/" at the beginning of the specified character string, since this string is connected immediately after the uri-scheme-authority element. You need not specify "/" in the case of root path ("/").
<uri-query>	string	#2	Specify the request line query.
<http-header-Authorization>#1	string	0 or 1 time	Specify the authentication information corresponding to the Authorization header of the HTTP request header. Specify the authentication type in the type attribute: <ul style="list-style-type: none"> <li>• When you send the authentication information Specify the type attribute in raw, and then specify the header field value of the Authorization header in this element.</li> <li>• When you do not send the authentication information Specify none in the type attribute. Even if you specify a value in this element, it is ignored. Authorization header is not set in the HTTP request header.</li> </ul> If you specify any other value in the type attribute, an error occurs.

### 3. Defining Adapters

Tag name		Type	Occurrence count	Description
	<http-header-Content-Type>#1	string	0 or 1 time	Specify the media type corresponding to the Content-Type header of the HTTP request header. Specify the character code corresponding to the charset attribute of the Content-Type header in the charset attribute.
	<http-header-Cookies>	- (any)	0 or 1 time	Specify the Cookie information of the HTTP request. Schema files to be used differ depending on whether you set Cookie elements in a batch or individually: <ul style="list-style-type: none"> <li>When you set Cookie elements in a batch Use the schema file <code>adphttp_header_request1.xsd</code>. Specify the Cookie information with any type.</li> <li>When you set Cookie elements individually Use the <code>adphttp_header_request2.xsd</code> schema file. Specify respective Cookie information in the Cookie elements.</li> </ul>
	<Cookie>	string	0 or more	Specify the Cookie information of the HTTP request in the format of the Set-Cookie header passed from the HTTP response for each Cookie information. Specify the Cookie name of the Set-Cookie header of the HTTP response in the name attribute. Specify the host name (domain) of the server that sends the Cookie of the Set-Cookie of the HTTP response in the host attribute. Specify the path of the server that sends the Cookie of the Set-Cookie header of the HTTP response in the path attribute. For the value of each attribute and element of Cookie elements, use the value of Cookie elements of the response message (header) in which the Cookie information to be inherited is stored. If you set any other value, operations to be performed are not guaranteed.
	<http-header>	-	0 or 1 time	Specify the HTTP header information.
	Any element#1	string	0 to 1,024 times	Specify the HTTP header information such that the header field name is to be specified in the element name and the header field value is to be specified in the element value. This is ignored if the hierarchical structure and attribute value are specified, and also when some particular header information is specified. For details, see <i>2.14.3 Relationship between HTTP requests and request messages</i> in the manual <i>Service Platform Overview</i> .
	<http-part>	-	0 or 1 time	Specify the meta information of parts.
	<message>	-	0 or 1 time	Specify the information related to request messages (body).
	<binding>#1	string	1 time	Specify any of the following methods as the method to process a request message (body): <ul style="list-style-type: none"> <li>none Sends HTTP requests when the request message (body) is not present. When you specify this value the Content-Type header of the HTTP request is not set.</li> <li>raw Sends the contents of the request message (body) as-is as the pass-through mode. The Media type and the character code specified in the http-header-Content-Type element are set in the Content-Type header.</li> <li>form-data</li> </ul>

Tag name			Type	Occurrence count	Description
		<binding>#1	string	1 time	<p>As the normal mode, this method converts and sends the request message (body) to the form data format. <code>application/x-www-form-urlencoded</code> is set in the <code>Content-Type</code> header of the HTTP request. The character code specified in the <code>http-header-Content-Type</code> element is set in the character code.</p> <ul style="list-style-type: none"> <li>Other than the above</li> </ul> <p>A system exception occurs.</p>
		<files>	-	0 or 1 time	<p>Specify the file information sent with the HTTP request.</p> <p>If you specify this element, even if the request message (body) is already set, the content specified in this element takes precedence.</p>
		<file>	-	0 or more	<p>Specify the information related to the file to be sent.</p> <p>If you specify this element for more than 2 times, the element acquired first is considered as valid.</p>
		<input-folder-name>#1	string	1 time	<p>Specify any of the following folders as the folder to store the file to be sent:</p> <ul style="list-style-type: none"> <li>For a working folder Specify null. (An error does not occur even if you specify a null string).</li> <li>For a common folder Specify the common folder definition name.</li> </ul> <p>You can specify the following values to identify the folder in which the file to be sent exists in the common attribute.</p> <ul style="list-style-type: none"> <li>For a working folder Specify <code>"common=false"</code>.</li> <li>For a common folder Specify <code>"common=true"</code>.</li> </ul> <p>Note that an error occurs at the time of reading if the specified folder is invalid or is a symbolic link.</p>
		<local-file-name>#1	string	1 time	<p>Specify name of the file to be sent:</p> <ul style="list-style-type: none"> <li>When you specify a working folder in the <code>input-folder-name</code> element Specify the intermediate file name in the working folder.</li> <li>When you specify a common folder in the <code>input-folder-name</code> element Specify the file name in the common folder.</li> </ul> <p>If a file with the specified name does not exist, a system exception occurs.</p> <p>In addition, an error occurs in the following cases:</p> <ul style="list-style-type: none"> <li>If specification ("<code>/..</code>") related to password change is included</li> <li>If the specified file name is invalid</li> <li>If symbolic link is included</li> </ul>
		<output-folder-name>#1	string	0 or 1 time	<p>Specify any of the following folders as the folder to output the downloaded file at the time of the HTTP response:</p> <ul style="list-style-type: none"> <li>For a working folder Specify null. (An error does not occur even if you specify a null string).</li> </ul>

Tag name	Type	Occurrence count	Description
<output-folder-name>#1	string	0 or 1 time	<ul style="list-style-type: none"> <li>For a common folder Specify the common folder definition name.</li> </ul> <p>You can specify the following values to identify the folder in which the file to be sent exists in the common attribute.</p> <ul style="list-style-type: none"> <li>For a working folder Specify "common=false".</li> <li>For a common folder Specify "common=true".</li> </ul> <p>Note that an error occurs at the time of writing if the specified folder is invalid or is a symbolic link.</p>

Legend:

-: There is no corresponding item.

Note#1

If there is no specification of elements, the specification of the HTTP-adapter runtime-environment property file or the HTTP-adapter runtime-environment common property file becomes valid.

For the corresponding relationship between a request message and a property file, see 2.14.13 *Corresponding relationship between request message (header) and HTTP adapter execution environment (common) properties file* in the manual *Service Platform Overview*.

Note#2

Follows the specification of the detailed message format for the query of include destination.

● **adphhttp\_header\_query\_detail.xsd (detailed message format for a query)**

The following table describes the detailed message format for the query included in the request message format (for header variables). Name of the name space is "http://www.hitachi.co.jp/soft/xml/cosminexus/csc/adapter/http/header\_request".

Table 3–77: Detailed message format for a query

Tag name	Type	Occurrence count	Description
<uri-query>	-	0 or 1 time	-
Any element #	string	0 to 1,024	Set the detailed information of the request line query.

Legend:

-: There is no corresponding item.

Note#

If there is no specification of elements, the specification of the HTTP-adapter runtime-environment property file or the HTTP-adapter runtime-environment common property file becomes valid.

For the corresponding relationship between the request message and the property file, see 2.14.13 *Corresponding relationship between request message (header) and HTTP adapter execution environment (common) properties file* in the manual *Service Platform Overview*.

● **adphhttp\_body\_empty.fdx (message format for file data)**

The following table describes the message format for the file data indicating an empty HTTP message body:

Table 3–78: Message format for file data

Simple content element	Type	Occurrence count	Description
<http-body-common-empty>	-	1 time	-

Simple content element	Type	Occurrence count	Description
<data>	String	0 or more	Specifies an element used to pass an empty data. You can omit this element.

Legend:

-: There is no corresponding item.

● **adphttp\_body\_form-data.xsd (message format for the form data)**

The following table describes the message format for the form data passed on to an HTTP adapter to a business process. Name of the name space is "http://www.hitachi.co.jp/soft/xml/cosminexus/csc/adapter/http/body\_form\_data".

Table 3–79: Message format for the form data

Simple content element	Type	Occurrence count	Description
<http-body-form-data>	-	1 time	-
Any element	String	0 or more	Specify any element according to the form data sent with the HTTP request body.

Legend:

-: There is no corresponding item.

● **adphttp\_header\_response1.xsd/adphttp\_header\_response2.xsd (response message format (for header variable))**

The following table describes the response message format of header variables returned to a calling business process by an HTTP adapter. Name of the name space is "http://www.hitachi.co.jp/soft/xml/cosminexus/csc/adapter/http/header\_response".

Table 3–80: Response message format (for header variable)

Tag name	Type	Occurrence count	Description
<http-header-response>	-	1 time	-
<request-id>	string	0 or 1 time	The request ID specified in the request message is set. If you do not specify a request ID in the request message, this element is not generated.
<status-code>	string	1 time	The status code of the status line of the HTTP response is set.
<reason-phrase>	string	1 time	The reason phrase part of the status line is set.
<http-header-Content-Type>	string	0 or 1 time	The media type of the Content-Type header is set.
<http-header-Cookies>	- (any)	1 time	The Set-Cookie header of the HTTP response is set. Schema files to be used differ depending on whether you set Cookie elements in a batch or individually. <ul style="list-style-type: none"> <li>When you set Cookie elements in a batch Use the schema file <code>adphttp_header_response1.xsd</code>. Specify the Cookie information in a batch with any type.</li> <li>When you set Cookie elements individually Use the schema file <code>adphttp_header_response2.xsd</code>. Specify each Cookie information in the Cookie elements.</li> </ul>

### 3. Defining Adapters

Tag name	Type	Occurrence count	Description
<Cookie>	string	0 or more	<p>The <code>Set-Cookie</code> header of the HTTP response is stored for each Cookie information.</p> <p>The Cookie name of the <code>Set-Cookie</code> header of the HTTP response is set in the name attribute.</p> <p>The host (domain) name of the server that sends the Cookie of the <code>Set-Cookie</code> header of the HTTP response is set in the host attribute.</p> <p>The path of the server that sends the Cookie of the <code>Set-Cookie</code> header of the HTTP response is set in the path attribute.</p>
<http-header>	-	1 time	The HTTP response header is stored.
Any element	string	0 or more	The HTTP header information is set such that the header field name is specified in the element name and the header field value is specified in the element value.
<http-body>	-	1 time	The HTTP response body information is set.
<output-folder-name>	string	0 or 1 time	<p>The output destination folder of the files downloaded with the HTTP response is set.</p> <p>The value of the output destination folder is same as the value specified in the <code>output-folder-name</code> element of the request message format (for header variables)</p> <ul style="list-style-type: none"> <li>For a working folder Null is set.</li> <li>For a common folder The common folder definition name is specified.</li> </ul> <p>The values to identify the folder in which the file to be sent exists are set in the common attribute.</p> <ul style="list-style-type: none"> <li>For a working folder "common="false" is set.</li> <li>For a common folder "common="true" is set.</li> </ul> <p>If there are no files to be downloaded, this element is not generated.</p>
<http-part>	-	1 time	The meta information of parts is set.
<files>	-	0 or 1 time	The file information received with the HTTP response is set.
<file>	-	1 time	The information of the file to be received is stored.
<local-file-name>	string	1 time	<p>Name of the file output to a common or a working folder is set.</p> <ul style="list-style-type: none"> <li>If the output folder is a common folder A name that is unique in the HTTP adapter is set for the file, which is output.</li> <li>If the output folder is a working folder Name of an intermediate file, which is output, is set.</li> </ul>
<file-name>	string	0 or 1 time	<p>The value specified in the filename attribute of the <code>Content-Disposition</code> header is set.</p> <p>If the filename attribute is not present, this element is not generated.</p>
<context>	-	1 time	The execution information of the HTTP adapter is stored.
<adapter-name>	string	1 time	The name of the adapter that has sent an HTTP request that is the basis of this HTTP response is set.

Tag name	Type	Occurrence count	Description
<operation-name>	string	1 time	Name of the operation of sending an HTTP request that is the basis of this HTTP response is set.
<request-method>	string	1 time	The method type of the HTTP request that is the basis of this HTTP response is set.
<request-uri>	string	1 time	URI (Authorization, Path excluding Scheme and UserInfo) of the HTTP request that is the basis of this HTTP response is set.

Legend:

-: There is no corresponding item.

● **adphttp\_fault.xsd (fault message format)**

The following table describes the fault message format of header variables returned by the HTTP adapter to a calling business process. Name of the name space is "http://www.hitachi.co.jp/soft/xml/cosminexus/csc/adapter/http/fault".

Table 3–81: Fault message format

Tag name	Type	Occurrence count	Description
<http-fault>	-	1 time	-
<status-code>	string	1 time	The status code of the status line of an HTTP response is set.
<reason-phrase>	string	1 time	The reason phrase part of the status line is set.
<http-header>	-	1 time	The HTTP response header is stored.
Any element	string	0 or more	The HTTP header information is set such that the header field name is specified in the element name and the header field value is specified in the element value.
<context>	-	1 time	The execution information of the HTTP adapter is stored.
<adapter-name>	string	1 time	The name of the adapter that has sent the HTTP request which is the basis of this HTTP response is set.
<operation-name>	string	1 time	The name of the operation of sending an HTTP request which is the basis of this HTTP response is set.
<request-method>	string	1 time	The method type of the HTTP request that is the basis of this HTTP response is set.
<request-uri>	string	1 time	URI (Authorization, Path excluding Scheme and UserInfo) of the HTTP request that is the basis of this HTTP response is set.

Legend:

-: There is no corresponding item.

(c) Method to create the message format

The method to create the message format used in the HTTP adapter is described here.

The XML schema of the message format provided by the service platform is stored in "*Installation directory of the service platform*\CSC\custom-adapter\HTTP\schema". To edit, first copy folder-wise contents at any location, and then edit.

The method to create the following formats is as follows:

- Detailed message format for a query

- Message format for form data
- Message format for the pass-through mode

#### • Creating Detailed message format for a query

1. Open the XML schema file (adphttp\_header\_query\_detail.xsd) with the XML editor, and edit in the following manner:

- XML schema file before corrections

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema
targetNamespace="http://www.hitachi.co.jp/soft/xml/cosminexus/csc/adapter/http/
header_request"
elementFormDefault="qualified"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:tns="http://www.hitachi.co.jp/soft/xml/cosminexus/csc/adapter/http/header_request">
  <xsd:complexType name="uri-query-type">
    <xsd:sequence>
      <xsd:any namespace="##any" processContents="skip" minOccurs="0" maxOccurs="1024" />
    </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```

- XML schema file after corrections (Example)

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema
targetNamespace="http://www.hitachi.co.jp/soft/xml/cosminexus/csc/adapter/http/
header_request"
elementFormDefault="qualified"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:tns="http://www.hitachi.co.jp/soft/xml/cosminexus/csc/adapter/http/header_request">
  <xsd:complexType name="uri-query-type">
    <xsd:sequence>
      <xsd:element name="parameter1" type="xsd:string" minOccurs="0"/>
      <xsd:element name="parameter2" type="xsd:string" minOccurs="0"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```

Legend:

 : Locations to be edited

Change the xsd:any type element to the xsd:string type element for which the query name is the local name. You can specify any value as the occurrence count.

#### ! Important note

The upper limit of the query elements that you can specify directly under the uri-query element is upto 1,024. If you specify 1,025 or more elements, operations are not guaranteed.

2. Overwrite and save the XML schema file.

#### • Creating the Message format for form data

1. Open the XML schema file (adphttp\_body\_form-data.xsd) with the XML editor , and edit in the following manner:

- XML schema file before corrections

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema
targetNamespace="http://www.hitachi.co.jp/soft/xml/cosminexus/csc/adapter/http/body_form_data"
elementFormDefault="qualified" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:tns="http://www.hitachi.co.jp/soft/xml/cosminexus/csc/adapter/http/body_form_data">
<xsd:element name="http-body-form-data" type="tns:http-body-form-data-type"></xsd:element>
<xsd:complexType name="http-body-form-data-type">
  <xsd:sequence>
    <xsd:any namespace="##any" processContents="skip" minOccurs="0" maxOccurs="unbounded" />
  </xsd:sequence>
</xsd:complexType>
</xsd:schema>
```

- XML schema file after corrections (Example)

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema
targetNamespace="http://www.hitachi.co.jp/soft/xml/cosminexus/csc/adapter/http/body_form_data"
elementFormDefault="qualified" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:tns="http://www.hitachi.co.jp/soft/xml/cosminexus/csc/adapter/http/body_form_data">
<xsd:element name="http-body-form-data" type="tns:http-body-form-data-type"></xsd:element>
<xsd:complexType name="http-body-form-data-type">
  <xsd:sequence>
    <xsd:element name="parameter1" type="xsd:string" minOccurs="0"/>
    <xsd:element name="parameter2" type="xsd:string" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>
</xsd:schema>
```

Legend:

 : Locations to be edited

Change the xsd:any type element to the xsd:string type element for which the local name is the key name of form data. You can specify any value as the occurrence count.

2. Overwrite and save the XML schema file.
- **Creating the message format for the pass-through mode**

1. Create any message format with the editor.
2. Save with any file name.

#### Important note

- You can specify any one type among XML type, on-XML type, and any type for the variable type. However, in the case of XML type, you must create the message format of the XML format definition file and in the case of non-XML type, you must create the message format of the binary format definition file.
- You can specify the character code of the HTTP request body with the charset attribute of the Content-Type header. However, do not convert this character code to the character code specified in the HTTP adapter. Therefore, you must perform the character code conversion yourself, before setting up the body data in the request message (body).

### (3) Operations to be performed on the service adapter settings screen

The procedure for defining HTTP adapters is as follows:

1. Display the service adapter settings screen.  
For details on how display the service adapter settings screen, see 3.3.1(4) *Displaying the service adapter settings screen*.
2. Set the definition information on the service adapter settings screen (basic).

For details on the items that you must set on the service adapter settings screen (basic), see 3.3.15(11) *For HTTP adapters* and 1.2.2 *Service Adapter Definition Window* in the manual *Service Platform Reference Guide*.

3. Click the [Service adapter definition (details)] tab.

The service adapter settings screen (details) is displayed.

4. Set the definition information on the service adapter settings screen (details).

For details on the items that you must set in the service adapter settings screen (details), see 3.3.15(11) *For HTTP adapters*.

#### (4) Creating and editing the HTTP-adapter definition file

The following are the types of the definition files to be created and edited:

- HTTP-adapter definition file  
This file is used for setting the operation information of the HTTP adapter.
- HTTP-adapter runtime-environment property file  
This file is used for setting the configuration information for each HTTP adapter.
- HTTP-adapter runtime-environment common property file  
This file is used for setting the common configuration information in all HTTP adapters.

Note that the creation and editing of these definition files is optional. If omitted, the default value for each definition file is used.

The procedure for creating and editing each definition file is as follows:

##### (a) HTTP-adapter definition file

Edit the HTTP-adapter definition file by using the template file provided by the service platform.

The procedure for editing the HTTP-adapter definition file is as follows:

1. In the [Self-defined file] of the Service adapter settings screen (details), select "cscadphhttp.properties", and click the [Edit] button.  
The editor used for editing the HTTP-adapter definition file is started.
2. Edit the HTTP-adapter definition file using the editor.  
For details on the definition contents that you can edit in the HTTP-adapter definition file, see *HTTP-adapter definition file* in the manual *Service Platform Reference Guide*.
3. From the Eclipse menu, select [File]- [Save], and then save the definition content.

##### (b) HTTP-adapter runtime-environment property file

1. Copy the template file (*Installation directory of the service platform*\CSC\custom-adapter\HTTP\config\templates\serviceid.properties), and save this file in the following directory:  
*Installation directory of the service platform*\CSC\custom-adapter\HTTP\config
2. Change the file name of the copied template file to "<Service ID#>.properties".
3. Edit and save the definition content.

For details on the definition contents that you can edit in the HTTP-adapter runtime-environment property file, see *HTTP-adapter runtime-environment property file* in the manual *Service Platform Reference Guide*.

#### ! Important note

The service ID is any string that you specify in the service adapter setting screen when adding a new HTTP adapter.

The HTTP-adapter runtime-environment property file is reflected in the execution environment when you start an HTTP adapter. Therefore, to change the contents of the HTTP-adapter runtime-environment property file, you must stop the HTTP adapter once.

**(c) HTTP-adapter runtime-environment common property file**

1. Copy the template file (*installation directory of the service platform\CSC\custom-adapter\HTTP\config\templates\adphttpcom.properties*), and save this file in the following directory:

*Installation directory of the service platform\CSC\custom-adapter\HTTP\config\common*

Do not change the file name.

2. Edit and save the definition contents.

The definition contents that you can edit in the HTTP-adapter runtime-environment common property file are same as the contents of the HTTP-adapter runtime-environment property file. For details, see *HTTP-adapter runtime-environment common property file* in the manual *Service Platform Reference Guide*.

HTTP-adapter runtime-environment common property file is reflected in the execution environment, when you start an HTTP adapter. Therefore, to change the contents of the HTTP-adapter runtime-environment common property file, you must stop the HTTP adapter once.

**(5) Setting the system property file**

If settings such as the proxy server or connection inheritance are required, set it using the system property file (usrconf.cfg). For details on the properties to be set in the HTTP adapter, see *2.14.6 Communication through proxy servers* in the manual *Service Platform Overview* and *2.14.7 Communication using connection continuity (Keep-Alive)* in the manual *Service Platform Overview*.

**(6) Setting business processes**

If you use an HTTP adapter, you must set the invoke service activities and fault processing, if necessary.

The following is the method to set the invoke service activities and fault processing:

**(a) Settings of the invoke service activities**

Defines the invoke service activities to invoke the HTTP adapter. For details on how to define the invoke service activities, see *5.6.4 Invoke service activity* in the manual *Service Platform Basic Development Guide*.

The settings of the [Invoke service activities] dialog box are as follows:

Item		Setting value
Activity name		Set any name.
Service name		Specify the service name set at the time of adding a new HTTP adapter.
Operation name		Specify the operation name specified in (3) <i>Operations to be performed on the service adapter settings screen</i> .
Request message	Body assigned variable	Specify a variable in the following manner according to the format of the data to be sent: <ul style="list-style-type: none"> <li>• When sending in the normal mode</li> <li>• Specify the variable that has assigned the message format for sending the form data created in (2)(c).</li> <li>• When sending in the pass-through mode</li> <li>• Specify the variable that has assigned the message format for sending data in the pass-through mode created in (2)(c).</li> <li>• When sending the file data</li> </ul> Specify the variable that has assigned <code>adphttp_body_empty.fdx</code> . <sup>#1</sup>
	Assign header variable <sup>#2</sup>	Specify a variable in the following manner according to the method to set the Cookie: <ul style="list-style-type: none"> <li>• When setting Cookie in a batch</li> </ul> Specify the variable that has assigned <code>adphttp_header_request1.xsd</code> .

### 3. Defining Adapters

Item		Setting value
Request message	Assign header variable <sup>#2</sup>	<ul style="list-style-type: none"> <li>When setting Cookies individually Specify the variable that has assigned <code>adphttp_header_request2.xsd</code>.</li> </ul>
Response message	Assign body variable	Specify a variable in the following manner according to the format of the data received: <ul style="list-style-type: none"> <li>When receiving in the pass-through mode</li> <li>Specify the variable that has assigned the message format for receiving data in the pass-through mode created in (2)(c).</li> <li>When receiving the file data Specify the variable that has assigned <code>adphttp_body_empty.fdx</code>.<sup>#3</sup></li> </ul>
	Assign header variable <sup>#2</sup>	Specify a variable in the following manner according to the method to set the Cookie: <ul style="list-style-type: none"> <li>When setting Cookie in a batch Specify the variable that has assigned <code>adphttp_header_response1.xsd</code>.</li> <li>When setting Cookies individually Specify the variable that has assigned <code>adphttp_header_response2.xsd</code>.</li> </ul>
Assign correlation set suite		Do not set.

Note#1

You must initialize the variable before invoking the service.

Note#2

If you do not want to use the request message (header) or the response message (header), setting is not required.

Note#3

Null string is set after invoking the service.

#### (b) Settings of the fault processing

If you process a system exception that is converted to a status code or a fault that you cannot process as a normal node, as an error in the business process, define the fault processing in the invoke service activity and in the upper-level scope. For details on the method to define the fault processing, see *5.4.3 Defining Fault Handling* in the manual *Service Platform Basic Development Guide*.

The settings of the [Assign the fault processing] dialog box are as follows:

Item	Setting value
Assigned variable	<ul style="list-style-type: none"> <li>To cache faults of an HTTP adapter Specify any variable that has assigned <code>adphttp_fault.xsd</code>.</li> <li>To cache faults of system exceptions Specify any variable that has assigned <code>cscfault.xsd</code>.</li> </ul>
Transition destination	Specify any activity that is transitioned when a fault has occurred.

## 3.3.14 Defining custom adapters

This subsection describes how to define custom adapters. You can define custom adapters in the Service Adapter Settings window.

### (1) Designing operations

One custom adapter can have multiple operations. For each operation, you need to design the operation name, communication model, request message, and response message.

**(a) Determining the operation name**

The operation name is an identifier used for distinguishing operations when a service requester calls a service component, or when the invoke service activity of a business process calls a service component.

Specify a name (255 bytes or less) that is unique within the custom adapter. Do not specify the same name as another operation in the same custom adapter.

You can specify the following characters and symbols:

- Alphanumeric characters
- Underscores ( \_ )
- Periods ( . )
- Hyphens ( - )

The protocol converter in a custom adapter uses the operation name to identify which operation was called when the custom adapter was called.

**(b) Determining the communication model**

For a custom adapter, you can select the communication model for each operation. Select **Sync** or **Async**, depending on the service component to be called.

**(c) Designing the request message and response message**

Design the message forms and message formats for the request message received by the protocol converter and for the response message used by the response protocol converter. However, if the communication model is **Async**, no response message is returned, so you do not have to define the response message.

For the message format, you can select XML or binary format.

The message format depends on whether the XML-format data or binary-format data (other than XML-format data) is used for the message used for executing the service component. Select either XML or binary format depending on the service component to be called, and create a message format depending on the service component to be called.

For details about the difference between XML and binary format, see *4.2 Message Format Types* in the manual *Service Platform Basic Development Guide*.

For details about how to create a message format when XML-format data is used, see *4.3 Creating Message Formats (XML Format Definition File)* in the manual *Service Platform Basic Development Guide*. For details about how to create a message format when binary-format data (other than XML-format data) is used, see *4.4 Creating Message Formats (Binary Format Definition File)* in the manual *Service Platform Basic Development Guide*.

**(2) Data transformation**

Data transformation is required if the message format received by the protocol converter designed in *3.3.14(1) Designing operations* is different from the message format of a request from the service requester or business process. You can set the message format in the Service Adapter Settings window.

To perform data transformation of a message, set the transformation-source message format definition file and transformation-destination message format definition file, and then define data transformation in the Data-conversion definition screen.

For details about data transformation, see *Chapter 6. Defining Data Transformation* in the manual *Service Platform Basic Development Guide*.

**(3) Creating and setting self-defined files**

Create definition files that are used by the protocol converter and custom adapter development framework.

- Self-defined file for the protocol converter
- Custom adapter development framework action definition file
- Custom adapter property file
- HITACHI Application Integrated Property File for the custom adapter
- Custom adapter definition file

### 3. Defining Adapters

Set the created definition files in the Service Adapter Settings window in the development environment.  
The following describes the contents of the files to be created.

#### (a) Self-defined file for the protocol converter

The protocol converter can read the contents of the resources set for a general custom adapter, via the adapter context.  
Create this file if a file is necessary for operation of the protocol converter.

#### (b) Custom adapter development framework action definition file

Specify the class name (including the package name) of the protocol converter used by the custom adapter development framework. Specify `framework_properties.xml` for the file name.

For details about the settings in the custom adapter development framework action definition file, see *B.2(1) Custom adapter development framework action definition file*.

#### (c) Custom adapter property file

Create this file as necessary (for example, when you want to have the protocol converter settings in an external file). This is a kind of self-defined file, and is used when a custom adapter is defined. Specify `customadapter_properties.xml` for the file name.

For details about the settings in the custom adapter property file, see *B.2(2) Custom adapter property file*.

#### (d) HITACHI Application Integrated Property File for the custom adapter

You can use the HITACHI Application Integrated Property File for the custom adapter to set the parameters of the custom adapter according to your environment. Specify `cscadapter_property.xml` for the name of the HITACHI Application Integrated Property File for a custom adapter.

For details about the settings in the HITACHI Application Integrated Property File, see *B.2(3) HITACHI Application Integrated Property File for custom adapter*.

If you specify a HITACHI Application Integrated Property File, the value of `<pooled-instance>` of the `Session Bean` attribute will be the value specified in the HITACHI Application Integrated Property File.

#### (e) Custom adapter definition file

You can use the custom adapter definition file to use the function for skipping message structure transformation (structure transformation skip function) during data transformation. Specify `csccustomadapter.properties` for the name of the custom adapter definition file.

For details about the settings in the custom adapter definition file, see *B.2(4) Custom adapter definition file*.

The following describes how to set the structure transformation skip function:

In data transformation for a request message, you might want to create a request message in binary format directly from the DOM tree received from a business process, and then pass the request message to a custom adapter. If so, you can enable the structure transformation skip function.

In data transformation for a response message, you might want to pass a DOM tree generated from the response message (in binary format, returned from a custom adapter) to the business process as is. If so, you can enable the structure transformation skip function.

To set the structure transformation skip function, in the Service Adapter Settings (details) window in the development environment, set the custom adapter definition file for **Self-defined file**.

For details about the Service Adapter Settings window, see *1.2.2 Service Adapter Definition Window* in the manual *Service Platform, Reference Guide*.

The following are notes on using the structure transformation skip function:

- In the Service Adapter Settings window in the development environment, specify the XML format definition file for the standard message format for a request message and response message. Also, specify the binary format definition file for the service message format of the request message and response message.
- In the Service Adapter Settings window in the development environment, specify the XML format definition file for the standard message format for the request message and response message. This XML format definition file

must be generated by using the `cscdfx2xsd` command based on the reception message format (binary format definition file).

#### (4) Creating a class file

Create a class file that implements the protocol converter (protocol conversion processing). The class file to be created implements the `CSCMsgCustomProtocolConverter` interface provided by the custom adapter development framework.

The protocol converter operates as part of the Stateless Session Bean running on the service platform. Therefore, when you create a class file, see the descriptions regarding EJB and Stateless Session Bean in the *Application Server EJB Container Functionality Guide*.

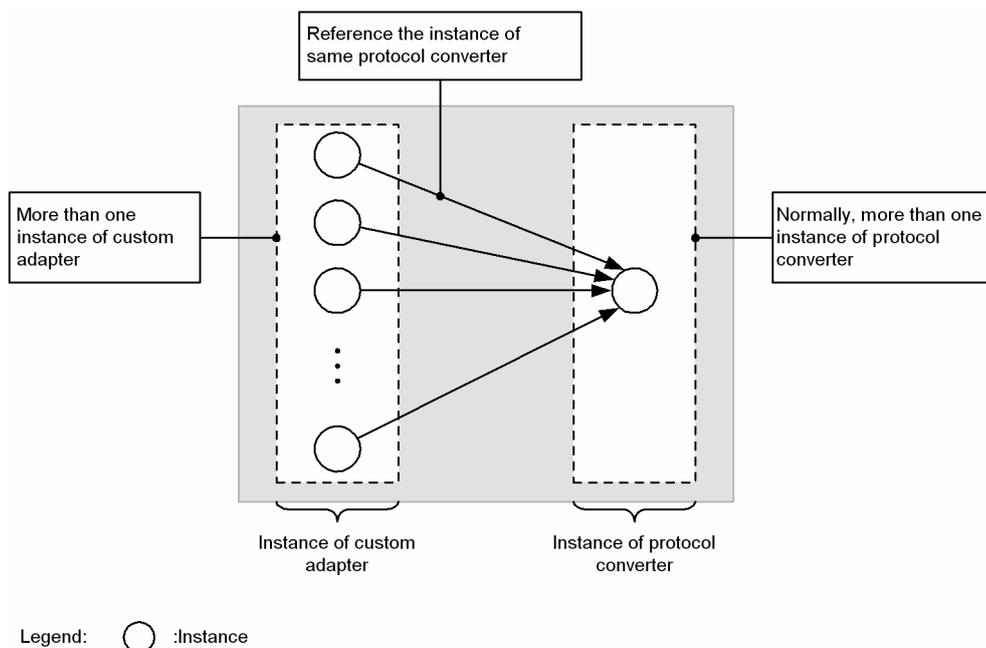
For details about the `CSCMsgCustomProtocolConverter` interface, see *B.1(1)(a) CSCMsgCustomProtocolConverter interface*. For details about example implementations, see *B.3 Sample program of the custom adapter development framework*.

The following describes what should be considered when you implement the protocol converter.

##### (a) Relationship between custom adapter instances and the protocol converter instance

One custom adapter can have multiple instances. However, the protocol converter always has one instance. Therefore, when you implement the protocol converter, note that the protocol converter will be accessed by multiple threads.

Figure 3–30: Relationship between custom adapter instances and the protocol converter instance



##### (b) How to allocate resources

The custom adapter development framework recommends the following methods to allocate resources for the protocol converter:

- Acquiring static resources: `start` method of `CSCMsgCustomProtocolConverter`  
For resources only referenced when a service component is called, the framework recommends that you use the `start` method to allocate the resources to the instance field. Such resources include the header information of the protocol necessary for service component calls, or the address of the service component.  
The reason for the above is that if a resource is acquired once when a custom adapter starts, the same resource can continue to be used.
- Acquiring dynamic resources (non-thread-safe resources): `invoke` method of `CSCMsgCustomProtocolConverter`  
For resources for which thread-safe access is not available (such as streaming file access, and connection to a service component), the framework recommends that you acquire the resources each time the `invoke` method calls a service component.

If you allocate resources for which thread-safe access is not available to the instance field, exclusive control among threads will be required.

If you acquire resources each time a service component is called, you can suppress the deterioration of processing performance caused by exclusive control among threads.

**! Important note**

Resources that are allocated once are retained, even after processing finishes. Therefore, if a high load is applied to the whole system, an `OutOfMemoryError` might occur due to a shortage of Java heap or Perm heap. Therefore, you must implement a process for adequately releasing the resources in the event of an `OutOfMemoryError`, or error processing such as rollback.

---

#### (c) How to specify a resource file name

The protocol converter can access the resource files in the archive file by using the adapter context. When accessing a resource file in the archive file, note the following:

- Specify a file name that includes the relative path in the archive.
- If the name of a self-defined file is the same as a resource file in the archive file, the system assumes that you specified the self-defined file.
- Do not place resource files of the same name in multiple archive files. If there are files of the same name, the system cannot identify which file belongs to which archive.

#### (d) Difference between fault information and exceptions

If an error occurs while the `invoke` method of `CSCMsgCustomProtocolConverter` is being executed, the protocol converter generates and issues an exception, or stores fault information in the response message.

Fault information is used to report a fault in a business operation that occurs while service component call processing (`invoke` method of `CSCMsgCustomProtocolConverter`) is being executed, to the custom adapter development framework.

An exception is used to report a system error (such as a communication error that occurs during the service component call processing) to the custom adapter development framework.

In the service adapter and business process that receive error information, there are the following differences between when a fault is used and when an exception is used:

- Fault information
  - Response to the service requester  
For details about how to acquire fault information, see *8.2.8 Acquiring Error Information* or *8.4.7 Acquiring Error Information* in the manual *Service Platform Basic Development Guide*.
  - Response to the business process  
A business process can process fault information in the same way as a fault of another service adapter provided by the service platform. For details about fault processing in a business process, see *5.4.3 Defining Fault Handling* in the *Service Platform Basic Development Guide*.

---

#### Reference note

Fault information reported to the custom adapter development framework has elements that correspond to the SOAP Fault information of SOAP. The method of using each element is also conforms to the SOAP specifications. Write the elements in accordance with the SOAP and SAAJ specifications supported by the SOAP Communication Infrastructure. For details about the SOAP and SAAJ specifications supported by the SOAP Communication Infrastructure, see *Chapter 12* in the manual *Application Server SOAP Application Development Guide*.

---

- Exception
  - Response to the service requester  
For details about how to acquire exception information, see the description of acquiring error information when a service requester is created.
  - Response to the business process  
The business process cannot perform fault processing, so an exception is treated as an error (exception).

**(e) Compiling**

You must specify, for the class path, the following JAR file that is used for the custom adapter development framework, and then compile the file:

```
installation-directory-of-the-service-platform\CSC\lib\csc_adapter.jar.
```

If you use a service platform API, separately add and compile the necessary JAR files.

**(5) Creating a JAR file**

The general names of the JAR files created during custom adapter development are classified as follows:

- Protocol converter JAR files  
JAR files including the protocol converter class file and custom adapter development framework action definition file
- Library JAR files  
JAR files other than protocol converter JAR files

The following describes how to create individual JAR files.

**(a) Creating protocol converter JAR files**

Use the `jar` command to archive (to a JAR file) the protocol converter class file, the class file used by the protocol converter class file, and the custom adapter development framework action definition file.

The following shows the directory structure when necessary files are archived to a JAR file, and gives an example of specifying the `jar` command.

- Directory structure of the JAR file  
The following table describes the directory structure of the JAR file when files are archived by the `jar` command.

Table 3–82: Directory structure of JAR file when files are archived by the `jar` command

Directory name or file name	Description	Specification
/	Root directory in the archive	R
*.class	Stores the protocol converters and the class files (*.class) of classes (interfaces) that the protocol converters depend on, into the directory hierarchy that conforms to the package name.	R
framework_properties.xml	Specifies the class name (including the package name) of the protocol converter used by the custom adapter development framework.	R <sup>#</sup>
*	Any resource files used by the protocol converter. You can create a directory, in which to store files.	--
/META-INF/	Directory that stores management information. This directory is automatically created by the <code>jar</code> command.	--
MANIFEST.MF	The manifest file. This file is automatically created in the archive if the <code>jar</code> command is executed with the <code>m</code> option specified.	--

Legend:

R: Indicates that specification is required when files are archived.

--: Indicates that specification is not required (or optional) when files are archived.

#

If there are multiple JAR files, store this file only in the JAR file in containing the protocol converter.

### 3. Defining Adapters

- Example of specifying the jar command when creating the JAR file

The following is an example of specifying the `jar` command when files are archived to a JAR file.

- Prerequisites

Assume that the following files are in the current directory and sub directories:

```
framework_properties.xml
protocolconverter/CustomProtocolConverter.class
lib/MyLib.class
data/table.dat
```

- Example specification

The following is an example of specifying the `jar` command when files are archived to a JAR file whose name is `MyProtocolConverter.jar`.

```
jar cf ../MyProtocolConverter.jar .\
```

#### (b) Creating library JAR files

Create library JAR files in the same way as general JAR files.

The following are notes on creating library JAR files.

- File structure of a JAR file

Do not place files of the same name in multiple JAR files.

If there are files of the same name, the system cannot identify which used file belongs to which JAR file. Note that it is not a problem if only the directory name is the same.

The following table describes an example where two JAR files (`A.jar` and `B.jar`) are used, and their directory and file structures in a directory are the same.

Table 3–83: Example for when directory and file structures in the archive are the same

Structure of A.jar	Structure of B.jar	Description
myfile.txt	--	The file <code>myfile.txt</code> in <code>A.jar</code> is used.
samedata.dat	samedata.dat	It cannot be determined whether the used file <code>samedata.dat</code> is the file in <code>A.jar</code> or <code>B.jar</code> . Do not use the same file name.
dir/	dir/	You can use the same directory name.
--	dir/b.class	The file <code>dir/b.class</code> in <code>B.jar</code> is used.
dir/same.class	dir/same.class	Whether the used file <code>def/same.class</code> is the file in <code>A.jar</code> or <code>B.jar</code> cannot be determined. Do not use the same file name.

Legend:

--: Indicates that the file does not exist.

- Arranging JAR files

Do not place a file with the name `framework_properties.xml` in the root directory of a JAR file.

If there is a JAR file with the name `framework_properties.xml` in the root directory, the file `framework_properties.xml` in the protocol converter JAR file might not be used.

#### (6) Creating an EAR file

Create an EAR file (necessary for registering a new custom adapter).

To create an EAR file, use the `jar` command to archive JAR files provided by the service platform to an EAR file.

The following shows the directory structure when JAR files are archived to an EAR file, and gives an example of specifying the `jar` command.

- Directory structure of the EAR file

The following table describes the directory structure of the EAR file to which JAR files are archived by the `jar` command.

Table 3–84: Directory structure of the EAR file to which JAR files are archived by the jar command

Directory name or file name	Description	Specification
/	Root directory in the archive	R
cscmsg_adpejb.jar	Store the copy of the following file: <i>installation-directory-of-the-service-platform\CSC\lib\cscmsg_adpejb.jar</i>	R
/META-INF/	Directory that stores management information. This directory is automatically created by the jar command.	--
MANIFEST.MF	The manifest file. This file is automatically created in the archive if the jar command is executed with the m option specified.	--

Legend:

- R: Indicates that specification is required when files are archived.
- : Indicates that specification is not required (or optional) when files are archived.

#### Important note

Do not store files other than the above. If multiple JAR files are stored, you will not be able to register a custom adapter.

- Example of specifying the jar command when creating an EAR file

The following is an example of specifying the jar command when files are archived to an EAR file.

- Prerequisites

Store a copy of the following file in the current directory:

*installation-directory-of-the-service-platform\CSC\lib\cscmsg\_adpejb.jar.*

- Example specification

The following is an example of specifying the jar command when files are archived to an EAR file whose name is `CustomAdapter.ear`.

```
jar cf ../CustomAdapter.ear .\
```

## (7) Replacing the EAR file

Set the EAR file when adding a custom adapter. If you need to modify the EAR file after adding a custom adapter, replace the EAR file in the Service Adapter Settings window of the custom adapter.

The EJB-JAR file names contained in the replacement EAR file must be the same as those (set when adding the custom adapter) in the EAR file. If the EJB-JAR file names are different, an error occurs when the EAR file is specified, and the EAR file cannot be set.

To replace the EAR file in the Service Adapter Settings window of the custom adapter:

1. On the Service Definition List in the tree view, double-click the custom adapter for which the EAR file is to be replaced.  
The Service Adapter Settings window appears.
2. Click the **Service-adapter definition (details)** tab at the bottom of the window.  
The Service Adapter Settings (details) window appears.
3. In the **Service Adapter Definition (details)** window, click the **Browse** button.
4. Specify the EAR file to be replaced.

## (8) Operations in the Service Adapter Settings window

To define a custom adapter:

### 3. Defining Adapters

1. Open the Service Adapter Settings window.  
For details about how to open the Service Adapter Settings window, see *3.3.1(4) Displaying the service adapter settings screen*.
2. In **Service component control information**, edit **Service name**, **Service ID**, and **Maximum instances**, as necessary.
3. In **Service component control information**, click the **Add** button to add an operation.  
Specify the operation name designed in *3.3.14(1) Designing operations*.
4. From the **Operation** drop-down list in **Service component control information**, select the operation you want to edit.
5. If necessary, select the **Convert a system exception into a fault message** check box in **Service component control information**.
6. From the **Communication model** drop-down list of the operation information, select **Sync** or **Async**.  
Specify the communication model designed in *3.3.14(1) Designing operations*.
7. Perform the following operations:
  - When specifying the standard message format  
Perform steps 8 to 12, and go to step 13.
  - When not specifying the standard message format  
Go to step 13.
8. Click the >> button for the request message.  
The information fields for the standard message and data transformation definition are displayed.
9. Select the **Use** check box for the standard message.
10. Specify the **Format ID** for the standard message.
11. Click the **Browse** button for the standard message, and then specify the standard message format for **Message format**.  
For **Message format**, you can specify an XML message or binary message.
12. Click the **Display** button for the standard message.  
The format of the standard message is displayed. If necessary, check the format of the specified standard message.
13. Specify the **Format ID** for the service component message.
14. Click the **Browse** button for the service component message, and then specify the service component message format for **Message format**.  
Specify the message format designed in *3.3.14(1) Designing operations*.  
For **Message format**, you can specify the XML message or binary message.
15. Click the **Display** button for the service component message.  
The format of the service component message is displayed. If necessary, check the format of the specified service component message.
16. Perform the following operations:
  - When selecting the **Use** check box for the standard message  
Perform steps 17 to 19, and then go to step 20.
  - When you do not select the **Use** check box for the standard message  
Go to step 20.
17. Enter the file name for the data transformation definition.
18. Click the **Edit** button.  
The Data-conversion definition screen appears.  
Note that, for the first definition, the Select Root Element dialog box appears.  
If you changed the message format, a dialog box appears, confirming whether to apply the change. For details, see *6.3.2 Procedure for defining changed message formats* in the manual *Service Platform Basic Development Guide*.
19. Map the contents of the standard message and the contents of the service component message.
20. If, in step 6, you select **Sync** for **Communication model**, also perform steps 7 to 19 for the response message.
21. Click the **Service-adapter definition (details)** tab.

22. Check the name of the service adapter (EJB-JAR file).  
Make sure that the name is `cscmsg_adpejb.jar`.
23. Click the **Add** button for the utility class (JAR file), and register the JAR file.  
The file to be added is the JAR file you created in 3.3.14(5)(a) *Creating protocol converter JAR files*.  
If necessary, you can also add JAR files used by the protocol converter (such as library JAR files).
24. Click the **Add** button for the self-defined file, and add the following files, as necessary:
  - Self-defined file for protocol converter
  - Custom adapter property file
  - HITACHI Application Integrated Property File for custom adapter
25. If necessary, select a user-defined file, click the **Edit** button, and then edit the contents of the file.
26. Make sure that the definitions are correct, and from the Eclipse menu, select **File** and then **Save** to save the definitions.

### 3.3.15 List of Settings in Adapter Definition

This sub section describes the settings to be performed on the "Service adapter settings screen" to be used in the service adapter definition.

#### Important note

- Do not specify a name that begins with "format" in the [Message format ID]. If you specify a name that begins with "format", it might overlap with the ID that is used internally by the system, and an error might occur.
- If the message format of the message format definition file specified in the service adapter settings screen references an external XML schema, without fail, set a file that corresponds to the root schema. The external XML schema file referenced by the root schema is automatically taken.
- The message format definition file specified in the service adapter settings screen must fulfill certain conditions. For details on the conditions of the schema, see 2.6.5 *Scoping of XML schema* in the manual *Service Platform Basic Development Guide*.

#### (1) For SOAP adapters

The following table lists the settings to be performed on the service adapter settings screen at the time of defining SOAP adapters:

Table 3–85: Settings to be performed on the service adapter settings screen of SOAP adapters

Classification		Item	Settings
Basic screen	Service component control information	Service name	B
		Service ID	B
		Service type	N
		Address	N
		Maximum instance count	B
		Service class name	N
		Operation	B
		Transform the system exception to a fault (check box)	Y
	Operation information	Operation name	N
		Communication model	N
	Message format	Header	Root element

Classification			Item	Settings	
<b>Basic screen</b>	<b>Message format</b>	<b>Header</b>	Name space	N	
			Message format	N	
		<b>Body</b>	Use (Check box)	Y	
			Standard format ID	A <sup>#1</sup>	
			Standard message format	A <sup>#1</sup>	
			Service component format ID	B	
			Service component message format	B	
			Data transformation definition	A <sup>#1</sup>	
	<b>Response message</b>	<b>Header</b>	Root element	Y	
			Name space	N	
			Message format	N	
		<b>Body</b>	Use (Check box)	Y	
			Standard format ID	A <sup>#2</sup>	
			Standard message format	A <sup>#2</sup>	
			Service component format ID	B	
			Service component message format	B	
	<b>Fault message</b>	Fault name		A <sup>#3</sup>	
		Fault message format		A <sup>#3</sup>	
	<b>Detailed screen</b>	<b>Web service control information</b>	Client definition file		B
			Use (Check box)		Y
			User name		A <sup>#4</sup>
			Password		A <sup>#4</sup>
			Password (confirm)		A <sup>#4</sup>
<b>SessionBean control information</b>		Client definition file		-	
		User-defined class (JAR file)		-	
<b>MDB control information</b>		JMS message type		-	
		Remote invocation (radio button)		-	
		Local invocation (radio button)		-	
		Maximum message count		-	
		Destination URL		-	
		Basic authentication (Check box)		-	
		User name		-	
		Password		-	
Password (confirm)		-			

Classification		Item	Settings
<b>Detailed screen</b>	<b>MDB control information</b>	Transfer queue creation destination RD area name	-
		Order guarantee (Check box)	-

Legend:

B: Must be set.

Y: Setting is optional.

A: Must be set as per the condition.

N: Confirm the displayed contents.

-: Not applicable.

Note# 1

You must set this item, if you have checked the [Use] (checkbox) of the request message.

Note#2

You must set this item, if you have checked the [Use] (checkbox) of the response message.

Note#3

Set this item only in case of the fault message of the service component message.

Note#4

You must set this item, if you have checked the [Use] (checkbox) of basic authentication.

#### Important note

You cannot specify single byte spaces or double byte spaces before and after the input item in the service adapter setting screen.

## (2) For the SessionBeanadapters

The following table lists the settings to be performed on the service adapter settings screen at the time of defining SessionBean adapters:

Table 3–86: Settings to be performed on the service adapter settings screen of SessionBean adapters

Classification		Item	Settings	
<b>Basic screen</b>	<b>Service component control information</b>	Service name	B	
		Service ID	B	
		Service type	N	
		Address	B	
		Maximum instance count	B	
		Service class name	N	
		Operation	B	
		Transform the system exception to a fault (Check box)	Y	
	<b>Operation information</b>	Operation name	N	
		Communication model	N	
	<b>Message format</b>	<b>Header</b>	Root element	-
			Name space	-
			Message format	-
		<b>Body</b>	Use (Checkbox)	Y
			Standard format ID	A <sup>#1</sup>

### 3. Defining Adapters

Classification			Item	Settings	
<b>Basic screen</b>	<b>Message format</b>	<b>Body</b>	Standard message format	A <sup>#1</sup>	
			Service component format ID	B	
			Service component message format	B	
			Data transformation definition	A <sup>#1</sup>	
	<b>Response message</b>	<b>Header</b>	Root element	-	
			Name space	-	
			Message format	-	
		<b>Body</b>	Use (Checkbox)	Y	
			Standard format ID	A <sup>#2</sup>	
			Standard message format	A <sup>#2</sup>	
			Service component format ID	B	
			Service component message format	B	
	<b>Fault message</b>		Fault name	-	
			Fault message format	-	
	<b>Detailed screen</b>	<b>Web service control information</b>		Client definition file	-
				Use (Checkbox)	-
			User name	-	
			Password	-	
			Password (confirm)	-	
<b>SessionBean control information</b>			Client definition file	B	
			User-defined class (JAR file)	Y	
<b>MDB control information</b>			JMS message type	-	
			Remote invocation (radio button)	-	
			Local invocation (radio button)	-	
			Maximum message count	-	
			Destination URL	-	
			Basic authentication (Checkbox)	-	
			User name	-	
			Password	-	
			Password (confirm)	-	
			Transfer queue creation destination RD area name	-	
			Order guarantee (Checkbox)	-	

Legend:

B: Must be set.

Y: Setting is optional.

A: Must be set as per the condition.

N: Confirm the displayed contents.

-: Not applicable.

Note#1

You must set this item, if you have checked the [Use] (checkbox) of the request message.

Note# 2

You must set this item, if you have checked the [Use] (checkbox) of the response message.

#### Important note

You cannot specify single byte spaces or double byte spaces before and after the input item in the service adapter setting screen.

### (3) For MDB (WS-R) adapters

The following table lists the settings to be performed on the service adapter settings screen at the time of defining MDB (WS-R) adapters:

Table 3–87: Settings to be performed on the service adapter settings screen of MDB (WS-R) adapters

Classification		Item	Settings	
<b>Basic screen</b>	<b>Service component control information</b>	Service name	B	
		Service ID	B	
		Service type	N	
		Address	B	
		Maximum instance count	B	
		Service class name	N	
		Operation	B	
		Transform the system exception to a fault (Check box)	Y	
	<b>Operation information</b>	Operation name	N	
		Communication model	N	
	<b>Request message</b>	<b>Header</b>	Root element	-
			Name space	-
			Message format	-
		<b>Body</b>	Use (Check box)	Y
			Standard format ID	A <sup>#1</sup>
			Standard message format	A <sup>#1</sup>
			Service component format ID	B
			Service component message format	B
	Data transformation definition	A <sup>#1</sup>		
	<b>Response message</b>	<b>Header</b>	Root element	-
Name space			-	
Message format			-	
<b>Body</b>		Use (Check box)	-	
		Standard format ID	-	

### 3. Defining Adapters

Classification		Item	Settings	
<b>Basic screen</b>	<b>Response message</b>	<b>Body</b>	Standard message format	-
			Service component format ID	-
			Service component message format	-
			Data transformation definition	-
	<b>Fault message</b>	Fault name	-	
		Fault message format	-	
<b>Detailed screen</b>	<b>Web service control information</b>	Client definition file	-	
		Use (Check box)	-	
		User name	-	
		Password	-	
		Password (confirm)	-	
	<b>Sessionbean control information</b>	Client definition file	-	
		User-defined class (JAR file)	-	
	<b>MDB control information</b>	JMS message type	B	
		Remote invocation (radio button)	B <sup>#2</sup>	
		Local invocation (radio button)	B <sup>#2</sup>	
		Maximum message count	A <sup>#3</sup>	
		Destination URL	A <sup>#3</sup>	
		Basic authentication (Check box)	Y <sup>#4</sup>	
		User name	A <sup>#5</sup>	
		Password	A <sup>#5</sup>	
		Password (confirm)	A <sup>#5</sup>	
		Transfer queue creation destination RD area name	Y <sup>#4</sup>	
		Order guarantee (Check box)	Y <sup>#4</sup>	

**Legend:**

- B: Must be set.
- Y: Setting is optional.
- A: Must be set as per condition.
- N: Confirm the displayed contents.
- : Not applicable.

**Note#1**

You must set this item, if you have checked the [Use] (checkbox) of the request message.

**Note#2**

You must select either the [Remote invocation] or the [Local invocation] radio button.

**Note#3**

You must set this item, if you have selected the [Remote invocation] radio button.

**Note#4**

Setup is optional only if you have selected the [Remote invocation] radio button.

## Note#5

You must set this item, if you have selected the [Remote Invocation] (radio button), and checked the [Use] (checkbox) of basic authentication.

 **Important note**

You cannot specify single byte spaces or double byte spaces before and after the input item in the service adapter setting screen. However, you can specify single byte space at the end in [Transfer queue creation destination RD area name] of MDB (WS-R) of the detailed screen.

## (4) For MDB (DB queue) adapters

The following table lists the settings to be performed on the service adapter settings screen at the time of defining MDB (DB queue) adapters:

Table 3–88: Settings to be performed on the service adapter settings screen of MDB (DB queue) adapters

Classification		Item	Settings	
Basic screen	Service component control information	Service name	B	
		Service ID	B	
		Service type	N	
		Address	B	
		Maximum instance count	B	
		Service class name	N	
		Operation	B	
		Transform the system exception to a fault (Check box)	Y	
	Operation information	Operation name	N	
		Communication model	N	
	Request message	Header	Root element	-
			Name space	-
			Message format	-
		Body	Use (Check box)	Y
			Standard format ID	A <sup>#</sup>
			Standard message format	A <sup>#</sup>
			Service component format ID	B
			Service component message format	B
			Data transformation definition	A <sup>#</sup>
			Response message	Header
	Name space	-		
	Message format	-		
	Body	Use (Check box)		-
		Standard format ID		-
		Standard message format		-
		Service component format ID		-

### 3. Defining Adapters

Classification			Item	Settings
<b>Basic screen</b>	<b>Response message</b>	<b>Body</b>	Service component message format	-
			Data transformation definition	-
	<b>Fault message</b>		Fault name	-
			Fault message format	-
<b>Detailed screen</b>	<b>Web service control information</b>		Client definition file	-
			Use (Check box)	-
			User name	-
			Password	-
			Password (confirm)	-
	<b>Sessionbean control information</b>		Client definition file	-
			User-defined class (JAR file)	-
	<b>MDB control information</b>		JMS message type	-
			Remote invocation (radio button)	-
			Local invocation (radio button)	-
			Maximum message count	-
			Destination URL	-
			Basic authentication (Check box)	-
			User name	-
			Password	-
			Password (confirm)	-
			Transfer queue creation destination RD area name	-
		Order guarantee (Check box)	-	

**Legend:**

- B: Must be set.
- Y: Setting is optional.
- A: Must be set as per the condition.
- N: Confirm the displayed contents.
- :Not applicable.

**Note#**

You must set this item, if you have checked the [Use] (checkbox) of the request message.

**! Important note**

You cannot specify single byte spaces or double byte spaces before and after the input item in the service adapter setting screen.

### (5) For DB adapters

The following table lists the settings to be performed on the service adapter settings screen at the time of defining DB adapters:

Table 3–89: Settings to be performed on the service adapter settings screen of DB adapters

Classification	Item	Settings			
		Synchronous	Asynchronous		
<b>Basic screen</b>	<b>Service component control information</b>	Service name	B	B	
		Service ID	B	B	
		Service type	N	N	
		Address	N	N	
		Maximum instance count #1	B	B	
		Service class name	N	N	
		Operation	B	B	
		Transform the system exception to a fault (Check box)	Y	Y	
	<b>Operation information</b>	Operation name	N	N	
		Communication model	B #2	B #3	
	<b>Request message</b>	Use (Check box) for any type	-	-	
		Use (Check box)	Y	Y	
		Standard format ID	A #4	A #4	
		Standard message format	A #4	A #4	
		Service component format ID	B	B	
		Service component message format #6	B	B	
		Data transformation definition	A #4	A #4	
	<b>Response message</b>	Use (Check box) for any type	-	-	
		Use (Check box)	Y	-	
		Standard format ID	A #5	-	
		Standard message format	A #5	-	
		Service component format ID	B	-	
		Service component message format #6	B	-	
		Data transformation definition	A #5	-	
	<b>Fault message</b>	Fault name	-	-	
		Service component message format	-	-	
	<b>Detailed screen</b>	<b>Service adapter control information</b>	Service adapter (EJB-JAR file)	N	N
			Utility class (JAR file)	-	-
			Self-defined file #7	B	B

Legend:

B: Must be set.

Y: Setting is optional.

A: Must be set as per the condition.

N: Confirm the displayed contents.

### 3. Defining Adapters

-: Not applicable.

Note#1

The value set in the service adapter definition screen is not applied.

For the maximum instance count, the value set in the HITACHI Application Integrated Property File element "hitachi-application-all-property/ejb-jar/hitachi-session-bean-property/session-runtime/stateless/pooled-instance/maximum" is set.

Note#2

Set "Synchronous".

Note#3

Set "Asynchronous".

Note#4

You must set this item, if you have checked the [Use] (checkbox) of the request message.

Note#5

You must set this item, if you have checked the [Use] (checkbox) of the response message.

Note#6

Set the message format of the XML message for each message type.

Note#7

cscadapter\_property.xml that is set is a template file. On selecting cscadapter\_property.xml, click the [Edit] button to modify the contents, and then save the file. For details on the property file, see 3.3.5(4) *Editing the HITACHI Application Integrated Property File*.

Create the SQL operation definition file and add it to the self-defined file. For details on the SQL operation definition file, see 3.3.5(1) *Creating the SQL operation definition file*.

### (6) For file adapters

The following table describes the settings to be performed on the service adapter settings (basic) screen at the time of defining file adapters. Settings are not required for the items that are not described in the table.

Table 3-90: Settings to be performed on the service adapter settings screen (basic) of file adapters

Classification	Item	Description	Settings
<b>Service component control information</b>	-	The information on service components specified at the time of creating a file adapter is displayed.	-
	Service name	The HCSC component name specified in the [Add service adapter settings (file adapter)] wizard is displayed.	B
	Service ID	The HCSC component ID is displayed.#1	A
	Service type	The "File adapter" is displayed as the type of service component.	N
	Address	Do not use this item.	N
	Maximum instance count	The maximum instance count of the service component is displayed.	B
	Service class name	Do not use this item.	N
	Operation	The operation name is displayed. Select the operation name of service components from the drop-down list.	B
<b>Operation information</b>	-	The information on service component operations is displayed.	-
	Operation name	The name of the operation selected in the operation column of the service component control information is displayed.	N
	Communication model	Select the communication model of the operation selected in the operation column of the service	B

Classification	Item		Description	Settings
<b>Operation information</b>	Communication model		component control information. Select "Synchronous" in the file adapter.	B
<b>Request message</b>	-		The information of the request message sent from the business process is displayed.	-
	Standard	Use (Checkbox)	Check this option when you transform the data of the standard message into the service component message format.	Y
		Format ID	Enter the format ID of the standard message received from the business process.	A <sup>#2</sup>
		Message format	Enter the format name of the standard message received from the business process.	A <sup>#2</sup>
	Service components	Format ID	Enter the format ID (message format definition file for the file adapter) of the service component message.	B
		Message format	Enter the format (message format definition file for the file adapter) of the service component message. adpff_read.xsd is set, if the definition pattern is "Read XML" or "Read Binary".	A
	Data transformation definition		Enter the definition file name to be used to transform the data from the standard message to the service components.	A <sup>#2</sup>
	<b>Response message</b>	-		The information of the response message sent from a service component is displayed. You can set this item in the communication model column of the operation information only if you have selected "Synchronous". You must set this item, since you select "Synchronous" in the file adapter.
Standard		Use (Checkbox)	Check this option when you transform the data of the standard message into the service component message format.	Y
		Format ID	Enter the format ID of the standard message received from service components.	A <sup>#3</sup>
		Message format	Enter the format name of the standard message received from service components.	A <sup>#3</sup>
Service component		Format ID	Enter the message format ID (message format definition file for the file adapter) of service components.	B
		Message format	Enter the message format (message format definition file for the file adapter) name of service components. adpff_result.xsd is set if the definition pattern is "Write XML" or "Write Binary".	A
Data transformation definition		Enter the definition file name to be used to transform the data from the service component message to the standard message.	A <sup>#3</sup>	

## Legend:

- B: Must be set.
- Y: Setting is optional.
- A: Must be set as per the condition.
- N: Confirm the displayed contents.
- : Not applicable.

### 3. Defining Adapters

**Note#1**

In the file adapter, the output path of the trace is decided on the basis of the service ID. Therefore, to edit the service ID, do not specify the service ID same as the service ID of the TP1 adapter that is already created once. Note that service ID is case-insensitive. When you specify the same service ID, since the trace is output to the trace information that was created in the past, it might become difficult to investigate the cause of failure.

**Note#2**

You must set this item, if you have checked the [Use] (checkbox) of the request message.

**Note#3**

You must set this item, if you have checked the [Use] (checkbox) of the response message.

The following table describes the settings to be performed on the service adapter settings (details) screen at the time of defining file adapters:

**Table 3–91: Settings to be performed on the service adapter settings screen (details) of file adapters**

Classification	Item	Description	Settings
<b>Service adapter control information</b>	Service adapter (EJB-JAR file)	The information on service components specified at the time of creating a file adapter is displayed.	N
	Utility class (JAR file)	The name of the EJB-JAR file of the file adapter is displayed. In the file adapter, the following utility class is set: <ul style="list-style-type: none"> <li>adpffpc.jar(protocol converter archive file)#</li> </ul>	N
	Self-defined file	Do not use the self-defined file in the file adapter.	-

**Legend:**

N: Confirm the displayed contents.

-: Not applicable.

**Note#**

The destination directory for storing files is "*Installation directory of the service platform*\CSC\custom-adapter\File\lib".

### (7) For Message Queue adapters

The following table describes the settings to be performed on the service adapter settings (basic) screen at the time of defining Message Queue adapters. Settings are not required for the items that are not described in the table.

**Table 3–92: Settings to be performed on the service adapter settings screen (basic) of Message Queue adapters**

Classification	Item	Description	Settings
<b>Service component control information</b>	-	The information on service components specified at the time of creating a Message Queue adapter is displayed.	-
	Service name	The HCSC component name specified in the [Add service adapter settings (Message Queue adapter)] wizard is displayed.	B
	Service ID	The HCSC component ID is displayed.#1	B
	Service type	The "Message Queue adapter" is displayed as the type of service component.	N
	Address	Does not use this item.	N
	Maximum instance count	The maximum instance count of the service component is displayed.	B
	Service class name	Does not use this item.	N

Classification	Item		Description	Settings
<b>Service component control information</b>	Operation		The operation name is displayed. Select the operation name of service components from the drop-down list.	B
<b>Operation information</b>	-		The information on service component operations is displayed.	-
	Operation name		The name of the operation selected in the operation column of the service component control information is displayed.	N
	Communication model		Select the communication model of the operation selected in the operation column of the service component control information. Select "Synchronous" in the Message Queue adapter.	B
<b>Request message</b>	-		The information of the request message sent from the business process is displayed.	-
	Standard	Use (Checkbox)	Check this option when you transform the data of the standard message into the service component message format.	Y
		Format ID	Enter the format ID of the standard message received from the business process.	A <sup>#2</sup>
		Message format	Enter the format name of the standard message received from the business process.	A <sup>#2</sup>
	Service component	Format ID	Enter the format ID (message format definition file for the Message Queue adapter) of the service component message.	B
		Message format	Enter the format name (message format definition file for the Message Queue adapter) of the service component message.  When the definition pattern is "Receive message", adpmq_rcv_request.xsd <sup>#3</sup> is set.  When the definition pattern is "Browse and receive message", adpmq_browse_request.xsd <sup>#3</sup> is set.	A
	Data transformation definition		Enter the definition file name to be used to transform the data from the standard message to the service components.	A <sup>#2</sup>
	<b>Response message</b>	-		The information of the response message sent from service components is displayed. You can set this item in the communication model column of the operation information, only if you have selected "Synchronous". Because you select "Synchronous" in the Message Queue adapter, set this item without fail
Standard		Use (Checkbox)	Check this option when you transform the data of the standard message into the service component message format.	Y
		Format ID	Enter the format ID of the standard message received from service components.	A <sup>#4</sup>
		Message format	Enter the format name of the standard message received from service components.	A <sup>#4</sup>
Service component		Format ID	Enter the message format ID (message format definition file for the Message Queue adapter) of service components.	B

### 3. Defining Adapters

Classification	Item		Description	Settings
<b>Response message</b>	Service component	Message format	Enter the message format name (message format definition file for the Message Queue adapter) of service components.  When the definition pattern is "Send message", adpmq_snd_response.xsd <sup>#3</sup> is set.	A
	Data transformation definition		Enter the definition file name to be used to transform the data from the service component message to the standard message.	A <sup>#4</sup>
<b>Fault message</b>	-		The information on the fault message of service components is displayed.	-
	Fault name		The fault name of service components is displayed.	B
	Message format		"adpmq_fault_response.xsd" <sup>#3</sup> is set when the definition pattern of operation is as follows: <ul style="list-style-type: none"> <li>• Receive message</li> <li>• Browse and receive message</li> <li>• Send and receive message</li> </ul>	N

**Legend:**

- B: Must be set.
- Y: Setting is optional.
- A: Must be set as per the condition.
- N: Confirm the displayed contents.
- : Not applicable.

**Note#1**

In the Message Queue adapter, the output path of the trace is decided on the basis of the service ID. Therefore, to edit the service ID, do not specify the service ID same as the service ID of the Message Queue adapter that is already created once. Note that service ID is case-insensitive. When you specify the same service ID, since the trace is output to the trace information that was created in the past, it might become difficult to investigate the cause of failure.

**Note#2**

You must set this item, if you have checked the [Use] (checkbox) of the request message.

**Note#3**

The destination directory for storing files is "*Installation directory of the service platform*\CSC\custom-adapter\MQ\schema".

**Note#4**

You must set this item, if you have checked the [Use] (checkbox) of the response message.

The following table describes the settings to be performed on the service adapter settings (details) screen at the time of defining Message Queue adapters:

**Table 3–93: Settings to be performed on the service adapter settings screen (details) of Message Queue adapters**

Classification	Item	Description	Settings
<b>Service adapter control information</b>	Service adapter (EJB-JAR file)	The information on service components specified at the time of creating a Message Queue adapter is displayed.	N
	Utility class(JAR file)	The name of the EJB-JAR file of the Message Queue adapter is displayed.  The following utility class is set: <ul style="list-style-type: none"> <li>• adpmqpc.jar(protocol converter archive file)<sup>#1</sup></li> </ul>	N
	Self-defined file	The following self-defined file is set: <ul style="list-style-type: none"> <li>• customadapter_properties.xml</li> </ul>	B

Classification	Item	Description	Settings
<b>Service adapter control information</b>	Self-defined file	<ul style="list-style-type: none"> <li>• ( Message Queue adapter environment-definition file) #2</li> <li>• adpmq_communication.xml</li> <li>• (Message Queue adapter communication-configuration definition file)#3</li> <li>• cscadapter_property.xml</li> <li>• (HITACHI Application Integrated Property File)#4</li> </ul>	B

Legend:

B: Must be set.

N: Confirm the displayed contents.

Note#1

The destination directory for storing files is "*Installation directory of the service platform*\CSC\custom-adapter\MQ\lib".

Note#2

customadapter\_properties.xml (Message Queue adapter environment-definition file) that is set is a template file. On selecting customadapter\_properties.xml, click the [Edit] button to modify the contents, and then save the file. For details on the Message Queue adapter environment-definition file Message Queue, see *Message Queue adapter environment-definition file* in the manual *Service Platform Reference Guide*.

Note#3

adpmq\_communication.xml (Message Queue adapter communication-configuration definition file) that is set is a template file. On selecting adpmq\_communication.xml, click the [Edit] button to modify the contents, and then save the file, if necessary. For details on the Message Queue adapter communication-configuration definition file, see *Message Queue adapter communication-configuration definition file* in the manual *Service Platform Reference Guide*.

Note#4

cscadapter\_property.xml (HITACHI Application Integrated Property File) that is set is a template file. On selecting cscadapter\_property.xml, click the [Edit] button to modify the contents, and then save the file. For details on the HITACHI Application Integrated Property File, see 3.3.9(6) *Editing HITACHI Application Integrated Property File*.

## (8) For FTP adapters

The following table describes the settings to be performed on the service adapter settings (basic) screen at the time of defining FTP adapters. Settings are not required for the items that are not described in the table.

Table 3–94: Settings to be performed on the service adapter settings screen (basic) of FTP adapters

Classification	Item	Description	Settings
<b>Service component control information</b>	Operation	Add the operation to be used from among the following operations: PUT: PUT (STOR or APPE) process GET:GET (RETR) process GETINFO:GETINFO (LIST or NLST) process	B
<b>Operation information</b>	Communication model	Set "Synchronous".	B
<b>Request message</b>	<b>Service component</b> Message format	The following message format definition files are set as per [Operation] settings: <ul style="list-style-type: none"> <li>• When the definition pattern is "Send file (PUT operation)" ftpadp_put_request.xsd#1</li> <li>• When the definition pattern is "Receive and Send file (GET operation)" ftpadp_get_request.xsd#1</li> <li>• When the definition pattern is "Acquire file information (GETINFO operation)"</li> </ul>	N

### 3. Defining Adapters

Classification		Item	Description	Settings
<b>Request message</b>	<b>Service component</b>	Message format	ftpadp_getinfo_request.xsd <sup>#1#2</sup>	N
<b>Response message</b>	<b>Service component</b>	Message format	<p>The following message format definition files are set as per [Operation] settings:</p> <ul style="list-style-type: none"> <li>When the definition pattern is "Send file (PUT operation)" ftpadp_put_response.xsd<sup>#1</sup></li> <li>When the definition pattern is "Receive and Send file (GET operation)" ftpadp_get_response.xsd<sup>#1</sup></li> <li>When the definition pattern is "Acquire file information (GETINFO operation)" ftpadp_getinfo_response.xsd<sup>#1#2</sup></li> </ul>	N

**Legend:**

B: Must be set.

N: Confirm the displayed contents.

**Note#1**

The destination directory for storing files is "*Installation directory of the service platform\CSC\custom-adapter\FTP\schema*".

**Note#2**

To edit the FTP adapter defined in Service Platform version 09-50-01 or earlier, you must re-specify the message format. Click the [Browse] button of the message format, specify `ftpadp_getinfo_request.xsd` (or `ftpadp_getinfo_response.xsd`), and re-set in the message format.

The following table describes the settings to be performed on the service adapter settings (details) screen at the time of defining FTP adapters. Settings are not required for the items that are not described in the table.

**Table 3–95: Settings to be performed on the service adapter settings screen (details) of FTP adapters**

Classification	Item	Description	Settings
<b>Service adapter control information</b>	Utility class(JAR file)	<p>In the FTP adapter, the following utility class is set:</p> <ul style="list-style-type: none"> <li>adpftpc.jar<sup>#</sup></li> </ul>	N
	Self-defined file	Do not use self-defined file in the FTP adapter.	-

**Legend:**

N: Confirm the displayed contents.

-: Do not set.

**Note#**

The destination directory for storing files is "*Installation directory of the service platform\CSC\custom-adapter\FTP\lib*".

## (9) For file operations adapters

The operation-wise settings to be performed on the service adapter settings screen at the time of defining file operations adapters are described here.

### (a) File transformation operation

The following table describes the settings to be performed on the service adapter settings (basic) screen in the case of the file transformation operation. Settings are not required for the items that are not described in the table.

Table 3–96: Settings (file transformation operation) on the service adapter settings screen (basic) of the file operations adapter

Classification		Item	Description	Settings
<b>Service component control information</b>		Operation	Add "Transform file (TRANSFORM operation)".	B
<b>Operation information</b>		Communication model	Set "Synchronous".	B
<b>Request message</b>	<b>Standard</b>	[Use] checkbox	Do not check this check box.	-
	<b>Service component</b>	Message format	adpfop_transform_request.xsd <sup>#</sup> is set.	N
<b>Response message</b>	<b>Standard</b>	[Use] checkbox	Do not check this check box.	-
	<b>Service component</b>	Message format	adpfop_transform_response.xsd <sup>#</sup> is set.	N
<b>Fault message</b>		Message format	Do not set. (Fault message is not used).	-

Legend:

B: Must be set.

N: Confirm the displayed contents.

-: Do not set.

Note#

The destination directory for storing files is "*Installation directory of the service platform*\CSC\schema\adpfop".

The following table describes the settings to be performed on the service adapter settings (details) screen in the case of the file transformation operation. Settings are not required for the items that are not described in the table.

Table 3–97: Settings (file transformation operation) on the service adapter settings screen (details) of file operations adapters

Classification	Item	Description		Settings
<b>Service adapter control information</b>	Utility class (JAR file)	adpfoppc.jar <sup>#1</sup> is set.		N
	Self-defined file	Common settings in batch processing method and segmentation method	cscFileOperation.properties (file operations adapter definition file) <sup>#2</sup> is set. Add the following files when the character code conversion UOC references the self-defined file: <ul style="list-style-type: none"> <li>csc_owncodeconvert.properties</li> <li>csc_owncodeconvert_in.properties (for input file)</li> <li>csc_owncodeconvert_out.properties (for output file)</li> </ul>	B
		When batch processing method is selected	Add the files specified in the following keys of the file operations adapter definition file: <ul style="list-style-type: none"> <li>csc.adapter.fileOperation.transform.all.inFormat</li> <li>csc.adapter.fileOperation.transform.all.outFormat</li> <li>csc.adapter.fileOperation.transform.all.styleSheet</li> </ul>	B
When segmentation method is selected		Add the files specified in the following keys of the file operations adapter definition file: <ul style="list-style-type: none"> <li>csc.adapter.fileOperation.transform.data.inFormat</li> <li>csc.adapter.fileOperation.transform.data.outFormat</li> <li>csc.adapter.fileOperation.transform.data.styleSheet</li> </ul>	B	

Classification	Item	Description		Settings
<b>Service adapter control information</b>	Self-defined file	When segmentation method is selected	<p>When you use the header record part (csc.adapter.fileOperation.transform.headerRecord=O N),add the files specified in the following keys of the file operations adapter definition file:</p> <ul style="list-style-type: none"> <li>csc.adapter.fileOperation.transform.header.inFormat</li> <li>csc.adapter.fileOperation.transform.header.outFormat</li> <li>csc.adapter.fileOperation.transform.header.styleSheet</li> </ul> <p>When you use the trailer record part (csc.adapter.fileOperation.transform.trailerRecord=O N),add the files specified in the following keys of the file operations adapter definition file:</p> <ul style="list-style-type: none"> <li>csc.adapter.fileOperation.transform.trailer.inFormat</li> <li>csc.adapter.fileOperation.transform.trailer.outFormat</li> <li>csc.adapter.fileOperation.transform.trailer.styleSheet</li> </ul> <p>When the output file is in XML format (csc.adapter.fileOperation.transform.output=xml), add the output template XML file specified with the csc.adapter.fileOperation.transform.outputTemplateXMLFile key.</p>	B

Legend:

B: Must be set.

N: Confirm the displayed contents.

Note#1

The destination directory for storing files is "*Installation directory of the service platform*\CSC\lib".

Note#2

cscFileOperation.properties (file operations adapter definition file) that is set is a template file. On selecting cscFileOperation.properties, click the [Edit] button to modify the contents, and then save the file, if necessary. For details on the file operations adapter definition file, see *File operations adapter definition file* in the manual *Service Platform Reference Guide*.

(b) File replication operation

The following table describes the settings to be performed on the service adapter settings (basic) screen for the file replication operation. Settings are not required for the items that are not described in the table.

Table 3–98: Settings (file replication operation) on the service adapter settings screen (basic) of file operations adapters

Classification	Item	Description	Settings	
<b>Service component control information</b>	Operation	Add "Copy file (COPY operation)".	B	
<b>Operation information</b>	Communication model	Set "Synchronous".	B	
<b>Request message</b>	<b>Standard</b>	[Use] checkbox	Do not check this check box.	-
	<b>Service component</b>	Message format	adpfop_copy_request.xsd#is set.	N
<b>Response message</b>	<b>Standard</b>	[Use] checkbox	Do not check this check box.	-
	<b>Service component</b>	Message format	adpfop_copy_response.xsd#is set.	N

Classification	Item	Description	Settings
<b>Fault message</b>	Message format	Do not set. (Fault message is not used).	-

Legend:

B: Must be set.

N: Confirm the displayed contents.

-: Do not set.

Note#

The destination directory for storing files is "*Installation directory of the service platform*\CSC\schema\adpfop".

The following table describes the settings to be performed on the service adapter settings (details) screen for the file replication operation. Settings are not required for the items that are not described in the table.

Table 3–99: Settings (file replication operation) on the service adapter settings screen (details) of file operations adapters

Classification	Item	Description	Settings
<b>Service adapter control information</b>	Utility class (JAR file)	adpfoppc.jar <sup>#1</sup> is set.	N
	Self-defined file	cscFileOperation.properties (file operations adapter definition file) <sup>#2</sup> is set.	Y

Legend:

Y: Setting is optional.

N: Confirm the displayed contents.

Note#1

The destination directory for storing files is "*Installation directory of the service platform*\CSC\lib".

Note#2

cscFileOperation.properties (file operations adapter definition file) that is set is a template file. On selecting cscFileOperation.properties, click the [Edit] button to modify the contents, and then save the file, if necessary. For details on the file operations adapter definition file, see *File operations adapter definition file* in the manual *Service Platform Reference Guide*.

### (c) File and folder deletion operation

The following table describes the settings to be performed on the service adapter settings (basic) screen for the file and folder deletion operation. Settings are not required for the items that are not described in the table.

Table 3–100: Settings (file and folder deletion operation) on the Service adapter settings screen (basic) of file operations adapters

Classification	Item	Description	Settings
<b>Service component control information</b>	Operation	Add "Delete file (DELETE Operation)".	B
<b>Operation information</b>	Communication model	Set "Synchronous".	B
<b>Request message</b>	<b>Standard</b>	[Use] checkbox	Do not check this check box.
	<b>Service component</b>	Message format	adpfop_delete_request.xsd <sup>#</sup> is set.
<b>Response message</b>	<b>Standard</b>	[Use] checkbox	Do not check this check box.
	<b>Service component</b>	Message format	adpfop_delete_response.xsd <sup>#</sup> is set.
<b>Fault message</b>	Message format	Do not set. (Fault message is not used).	-

Legend:

B: Must be set.

N: Confirm the displayed contents.

-: Do not set.

### 3. Defining Adapters

Note#

The destination directory for storing files is "*Installation directory of the service platform\CSC\schema\adpfop*".

The following table describes the settings to be performed on the service adapter settings (details) screen for the file and folder deletion operation. Settings are not required for the items that are not described in the table.

Table 3–101: Settings (file and folder deletion operation) on the service adapter settings screen (details) of file operations adapters

Classification	Item	Description	Settings
<b>Service adapter control information</b>	Utility class(JAR file)	adpfop.jar <sup>#1</sup> is set.	N
	Self-defined file	cscFileOperation.properties (file operations adapter definition file) <sup>#2</sup> is set.	Y

Legend:

Y: Setting is optional.

N: Confirm the displayed contents.

Note#1

The destination directory for storing files is "*Installation directory of the service platform\CSC\lib*".

Note#2

cscFileOperation.properties (file operations adapter definition file) that is set is a template file. On selecting cscFileOperation.properties, click the [Edit] button to modify the contents, and then save the file, if necessary. For details on the file operations adapter definition file, see *File operations adapter definition file* in the manual *Service Platform Reference Guide*.

#### (d) File compression operation

The following table describes the settings to be performed on the service adapter settings (basic) screen for the file compression operation. Settings are not required for the items that are not described in the table.

Table 3–102: Settings (file compression operation) on the Service adapter settings screen (basic) of file operations adapters

Classification	Item	Description	Settings
<b>Service component control information</b>	Operation	Add "Compress file (COMPRESS Operation)".	B
<b>Operation information</b>	Communication model	Set "Synchronous".	B
<b>Request message</b>	<b>Standard</b>	[Use] checkbox	Do not check this check box.
	<b>Service component</b>	Message format	adpfop_compress_request.xsd <sup>#</sup> is set.
<b>Response message</b>	<b>Standard</b>	[Use] checkbox	Do not check this check box.
	<b>Service component</b>	Message format	adpfop_compress_response.xsd <sup>#</sup> is set.
<b>Fault message</b>	Message format	Do not set. (Fault message is not used).	-

Legend:

B: Must be set.

N: Confirm the displayed contents.

-: Do not set.

Note#

The destination directory for storing files is "*Installation directory of the service platform\CSC\schema\adpfop*".

The following table describes the settings to be performed on the service adapter settings (details) screen for the file compression operation. Settings are not required for the items that are not described in the table.

Table 3–103: Settings (file compression operation) on the service adapter settings screen (details) of file operations adapters

Classification	Item	Description	Settings
<b>Service adapter control information</b>	Utility class(JAR file)	adpfoppc.jar <sup>#1</sup> is set.	N
	Self-defined file	cscFileOperation.properties (File operations adapter definition file) <sup>#2</sup> is set.	Y

Legend:

Y: Setting is optional.

N: Confirm the displayed contents.

Note#1

The destination directory for storing files is "*Installation directory of the service platform\CSC\lib*".

Note#2

cscFileOperation.properties (file operations adapter definition file) that is set is a template file. On selecting cscFileOperation.properties, click the [Edit] button to modify the contents, and then save the file, if necessary. For details on the file operations adapter definition file, see *File operations adapter definition file* in the manual *Service Platform Reference Guide*.

#### (e) File extraction operation

The following table describes the settings to be performed on the service adapter settings (basic) screen for the file extraction operation. Settings are not required for the items that are not described in the table.

Table 3–104: Settings (file extraction operation) on the Service adapter settings screen (basic) of file operations adapters

Classification	Item	Description	Settings
<b>Service component control information</b>	Operation	Add "Extract file (EXTRACT Operation)".	B
<b>Operation information</b>	Communication model	Set "Synchronous".	B
<b>Request message</b>	<b>Standard</b>	[Use] checkbox	Do not check this check box.
	<b>Service component</b>	Message format	adpfop_extract_request.xsd <sup>#</sup> is set.
<b>Response message</b>	<b>Standard</b>	[Use] checkbox	Do not check this check box.
	<b>Service component</b>	Message format	adpfop_extract_response.xsd <sup>#</sup> is set.
<b>Fault message</b>	Message format	Do not set. (Fault message is not used).	-

Legend:

B: Must be set.

N: Confirm the displayed contents.

-: Do not set.

Note#

The destination directory for storing files is "*Installation directory of the service platform\CSC\schema\adpfop*".

The following table describes the settings to be performed on the service adapter settings (details) screen for the file extraction operation. Settings are not required for the items that are not described in the table.

Table 3–105: Settings (file extraction operation) on the Service adapter settings screen (details) of file operations adapters

Classification	Item	Description	Settings
<b>Service adapter control information</b>	Utility class(JAR file)	adpfoppc.jar <sup>#1</sup> is set.	N

### 3. Defining Adapters

Classification	Item	Description	Settings
<b>Service adapter control information</b>	Self-defined file	cscFileOperation.properties (file operations adapter definition file) #2 is set.	Y

Legend:

Y: Setting is optional.

N: Confirm the displayed contents.

Note#1

The destination directory for storing files is "*Installation directory of the service platform\CSC\lib*".

Note#2

cscFileOperation.properties (file operations adapter definition file) that is set is a template file. On selecting cscFileOperation.properties, click the [Edit] button to modify the contents, and then save the file, if necessary. For details on the file operations adapter definition file, see *File operations adapter definition file* in the manual *Service Platform Reference Guide*.

### (10) For mail adapters

The following table describes the settings to be performed on the service adapter settings (basic) screen at the time of defining mail adapters. Settings are not required for the items that are not described in the table.

Table 3–106: Settings to be performed on the service adapter settings screen (basic) of mail adapters

Classification	Item	Description	Settings	
<b>Service component control information</b>	Operation	Add "SEND".	B	
<b>Operation information</b>	Communication model	Set "Synchronous".	B	
<b>Request message</b>	<b>Service component</b>	Format ID	Specify any format ID.	B
		Message format	adpmail_smtp_request.xsd # is set.	N
<b>Response message</b>	<b>Service component</b>	Format ID	Specify any format ID.	B
		Message format	adpmail_smtp_response.xsd # is set.	N
<b>Fault message</b>		Do not set. (Fault message is not used).	-	

Legend:

B: Must be set.

N: Confirm the displayed contents.

-: Do not set.

Note#

The destination directory for storing files is "*Installation directory of the service platform\CSC\schema\mail*".

The following table describes the settings to be performed on the service adapter settings (basic) screen at the time of defining mail adapters:

Table 3–107: Settings to be performed on the service adapter settings screen (details) of mail adapters

Classification	Item	Description	Settings
<b>Service adapter control information</b>	Service adapter (EJB-JAR file)	cscmsg_adpejb.jar is set.	N
	Utility class(JAR file)	The following utility class is set: • adpmailpc.jar#	N
	Self-defined file	Self-defined file is not used in the mail adapter.	-

Legend:

N: Confirm the displayed contents.

-: Do not set.

Note#

The destination directory for storing files is "*Installation directory of the service platform\CSC\lib*".

To edit the added mail adapter, select the corresponding mail adapter from the service definition list in the tree view, and then double-click.

### (11) For HTTP adapters

The following table lists and describes the settings to be performed on the service adapter settings (basic) screen at the time of defining the HTTP adapter. Settings are not required for the items that are not described in the table.

Table 3–108: Settings to be performed on the service adapter settings screen (basic) of HTTP adapters

Classification		Item	Description	Settings
<b>Service component control information</b>		Transform the system exception to a fault (Check box)	Check when the system exception such as communication error is processed as a fault.	Y
<b>Operation information</b>		Communication model	Set "synchronous".	B
<b>Request message</b>	<b>Body</b>	Service component message format	Specify following message formats as per the data to be sent: <ul style="list-style-type: none"> <li>For the normal mode Specify adphttp_body_form-data.xsd.</li> <li>For the pass-through mode</li> <li>Specify any schema file created in "<i>3.3.13(2)(c) Method to create the message format</i>".</li> <li>For the file data Specify adphttp_body_empty.fdx.</li> </ul>	B
<b>Response message</b>	<b>Body</b>	Service component message format	Specify the message format according to the received data: <ul style="list-style-type: none"> <li>For the pass-through mode</li> <li>Specify any schema file created in "<i>3.3.13(2)(c) Method to create the message format</i>".</li> <li>For the file data Specify adphttp_body_empty.fdx.</li> </ul>	B
<b>Fault message</b>		Fault message format	adphttp_fault.xsd # is set.	N

Legend:

B: Must be set.

Y: Setting is optional.

N: Confirm the displayed contents.

Note#

The destination directory for storing files is "*Installation directory of the service platform\CSC\custom-adapter\HTTP\schema*".

The following table lists and describes the settings to be performed on the service adapter settings (details) screen at the time of defining HTTP adapters:

Table 3–109: Settings to be performed on service adapter settings screen (details) of HTTP adapters

Classification	Item	Description	Settings
<b>Service adapter control information</b>	Service adapter (EJB-JAR file)	cscmsg_adpejb.jar is set.	N

### 3. Defining Adapters

Classification	Item	Description	Settings
<b>Service adapter control information</b>	Utility class(JAR file)	adphttpc.jar <sup>#1</sup> is set.	N
	Self-defined file	cscadhttp.properties <sup>#2</sup> is set.	Y
		Add the customer adapter definition file (csccustomadapter.properties) <sup>#3</sup> , if necessary.	Y

Legend:

Y: Set this item, if necessary.

N: Confirm the displayed contents.

Note#1

The destination directory for storing files is "*Installation directory of the service platform*\CSC\custom-adapter\HTTP\lib".

Note#2

cscadhttp.properties (HTTP-adapter definition file) that is set is a template file. On selecting cscadhttp.properties, click the [Edit] button to modify the contents, and then save the file if necessary. For details on the HTTP-adapter definition file, see *HTTP-adapter definition file* in the manual *Service Platform Reference Guide*.

Note#3

For details on the custom adapter definition file (csccustomadapter.properties), see "Appendix B.2(4) Custom adapter definition file".

To edit the added HTTP adapter, select the corresponding HTTP adapter from the service definition list in the tree view, and then double-click.

### (12) For custom adapters

The following table describes the settings to be performed on the service adapter settings screen at the time of defining custom adapters:

Table 3–110: Settings to be performed on the service adapter settings screen of custom adapters

Classification	Item	Settings		
		Synchronous	Asynchronous	
<b>Basic screen</b>	<b>Service component control information</b>	Service name	B	B
		Service ID	B	B
		Service type	N	N
		Address	N	N
		Maximum instance count <sup>#1</sup>	A	A
		Service class name	N	N
		Operation	B	B
		Transform the system exception to a fault (Check box)	Y	Y
	<b>Operation information</b>	Operation name	A	A
		Communication model	B <sup>#2</sup>	B <sup>#3</sup>
	<b>Request message</b>	Use (Checkbox) for any type	Y	Y
		Use (Checkbox)	Y	Y
		Standard format ID	A <sup>#4</sup>	A <sup>#4</sup>
		Standard message format	A <sup>#4</sup>	A <sup>#4</sup>

Classification		Item	Settings	
			Synchronous	Asynchronous
<b>Basic screen</b>	<b>Request message</b>	Service component format ID	B	B
		Service component message format	B	B
		Data transformation definition	A <sup>#4</sup>	A <sup>#4</sup>
	<b>Response message</b>	Use (Checkbox) for any type	Y	-
		Use (Checkbox)	Y	-
		Standard format ID	A <sup>#5</sup>	-
		Standard message format	A <sup>#5</sup>	-
		Service component format ID	B	-
		Service component message format	B	-
		Data transformation definition	A <sup>#5</sup>	-
	<b>Fault message</b>	Fault name	-	-
Service component message format		-	-	
<b>Detailed screen</b>	<b>Service adapter control information</b>	Service adapter (EJB-JAR file)	N	N
		Utility class(JAR file)	B	B
		Self-defined file	B	B

## Legend:

- B: Must be set.
- Y: Setting is optional.
- A: Must be set as per the condition.
- N: Confirm the displayed contents.
- : Not applicable.

## Note#1

If you have added the element of the HITACHI Application Integrated Property File as the self-defined file, the value set in the service adapter setting screen is not applied.

For the maximum instance count, the value specified in the element "hitachi-application-all-property/ejb-jar/hitachi-session-bean-property/session-runtime/stateless/pooled-instance/maximum" in the HITACHI Application Integrated Property File is set.

## Note#2

Set "Synchronous".

## Note#3

Set "Asynchronous".

## Note#4

You must set this item, if you have checked the [Use] (checkbox) of the request message.

## Note#5

You must set this item, if you have checked the [Use] (checkbox) of the response message.

## 3.4 Saving Adapters

---

You can save the service adapter and DB adapter contents that are being edited in the Service Adapter Definition screen during the editing process and after the editing ends. Adapter contents are saved in repositories.

The `Data transformation definition` file generated in the Data transformation definition is saved simultaneously with the Service Adapter Definition screen.

The procedure for saving adapters is described below.

### Method 1

On the Eclipse File menu, choose Save.

### Method 2

On the Eclipse File menu, choose Save All.

### Method 3

Press `Ctrl+S` in the Service Adapter Definition screen.

If an adapter has not been saved when you complete adapter definition by closing the screen for adapter definition, the Save Resource dialog box opens. In this dialog box, you can save the adapter that has not been saved.

### Important note

If invalid data is entered, you may not be able to save the adapter. In such a case, take measures as per the displayed message.

---

## 3.5 Editing Adapters

---

You can modify the contents of service adapters and database adapters that have been saved. To change the adapter contents, display the Service Adapter Definition screen and edit contents. The Service Adapter Definition screen is displayed if you choose a relevant adapter from the service definition list in the tree view and double click on the adapter.

For details about how to define adapter contents, see *3.3 Defining the Contents of Service Adapters*.

If the message format changes, adapter contents must be edited and data transformation must be redefined. For details about the procedure of the changed message format, see *6.3.2 Procedure for defining changed message formats* in the manual *Service Platform Basic Development Guide*.

## 3.6 Validating Adapters

You can check the contents of the service adapters and database adapters that you have created to verify that they are valid.

If the required service adapter definitions or database adapter definitions are missing, or if the definition relationship is invalid, the adapters cannot operate normally. Therefore, before executing the adapters in the execution environment, you can validate whether all of the required items are in the service adapters and database adapters that you have created and whether their relationship is valid. You can perform validation at any time as and when required.

### 3.6.1 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.

If an adapter has not been saved before validation, the Save Resource dialog box opens. In this dialog box, you can save the adapter that has not been saved.

### 3.6.2 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 3–111: Types of messages

Type	Explanation
Error	Displayed in either of the following cases: <ul style="list-style-type: none"> <li>• The definition content is invalid.</li> <li>• Although the syntax for the definition is correct, the definition cannot be executed.</li> </ul>
Warning	Displayed when a definition might cause an error during execution.
Information	Displays additional information.

## 3.7 Deleting Adapters

---

You can delete the service adapters and database adapters that have been created.

### (1) Deletion methods

The following two methods are available for deletion:

#### **Method 1**

1. Choose a service from the service definition list in the tree view.
2. Click Delete.  
A deletion confirmation dialog box opens.
3. Click Yes.  
The specified service adapter or database adapter is deleted.

#### **Method 2**

1. Choose and right-click a service 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 service adapter or database adapter is deleted.

### (2) Adapters that cannot be deleted

The following service adapters and database adapters cannot be deleted:

- Adapters that have already been defined for deployment
- Adapters that are set as the invocation targets of a service invocation activity of a business process



# Appendixes

## A. Custom Reception

This appendix describes custom receptions, which are a way for users to create reception processes based on the specifications of the service requester.

### A.1 Custom reception overview

Service Platform provides a custom reception framework that allows users to create custom receptions to suit their requirements. Users can develop receptions to support any protocol by implementing them as user-developed J2EE applications in the custom reception framework.

#### (1) Structure of custom reception

A custom reception is a J2EE application that consists of reception processes created by developers, and the custom reception framework itself.

The reception processes and custom reception framework processing in a custom reception are as follows:

Processing in reception processes created by developers

- Processing that receives service component invocation requests from the service requester
- Processing that passes request messages to the custom reception framework
- Processing that passes to the service requester any response messages returned by the custom reception framework

Processing executed by the custom reception framework

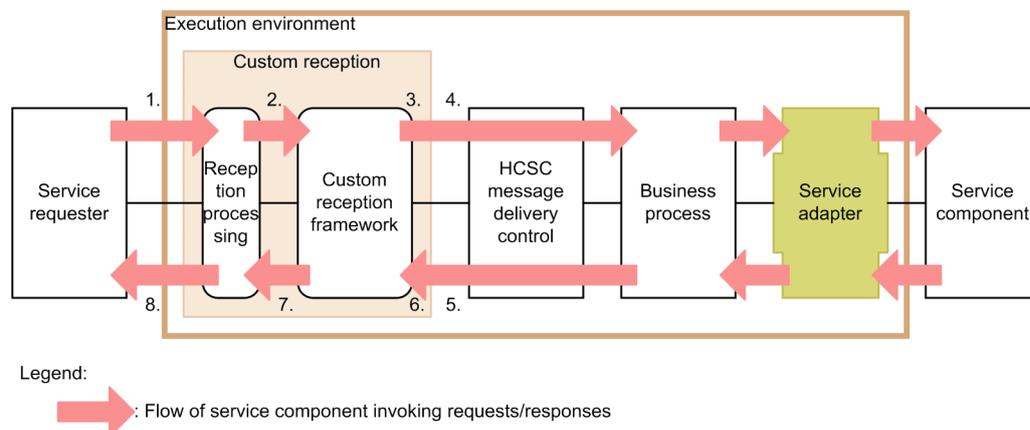
- Processing that passes to HCSC message delivery control any request messages received from reception processes
- Processing that returns to the reception processes any response messages received from HCSC message delivery control
- Processing that performs data transformation of request messages and response messages according to HSCS data transformation definitions

#### (2) General overview of custom reception framework operation

The custom reception framework allows users to develop and operate custom receptions. The framework provides APIs for creating custom receptions. For details on these APIs, see *A.3 APIs of the custom reception framework*.

The custom reception framework provides a mechanism by which custom receptions created by developers can operate as receptions in the execution environment of Service Platform. The following figure shows the flow of custom reception framework operation when a custom reception receives service component invocation requests:

Figure A–1: Flow of custom reception framework operation



1. The reception process receives a request to invoke the service component from the service requester.
2. The reception process passes the request message received from the service requester to the custom reception framework.
3. The custom reception framework performs data transformation of the request message.
4. The custom reception framework invokes HCSC message delivery control.
5. The custom reception framework receives a response message from HCSC message delivery control.
6. The custom reception framework performs data transformation of the response message.
7. The reception process receives the response message from the custom reception framework.
8. The reception process returns the response message to the service requester.

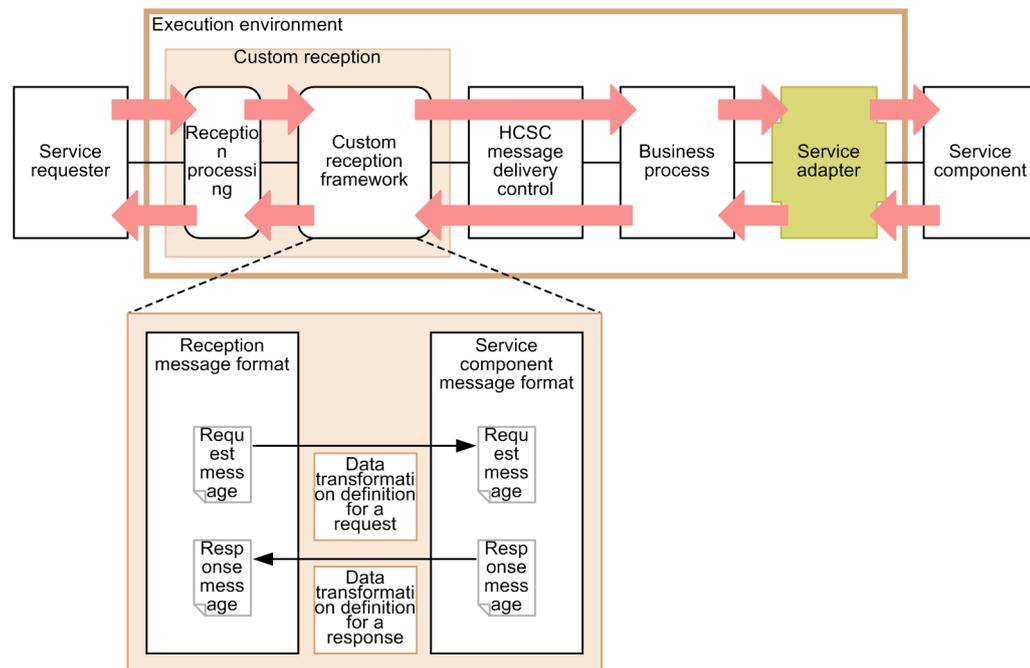
### (3) Data transformation in custom receptions

In a custom reception, request and response messages are subjected to data transformation in the custom reception framework. However, in an asynchronous communication model, data transformation of response messages does not occur. In this scenario, a null value is returned to the reception process.

The custom reception framework performs data transformation of the request messages received from the reception process before invoking HCSC message delivery control. The request messages transformed by the data transformation process are passed to the relevant business process via HCSC message delivery control. Response messages returned by the business process are again subjected to data transformation by the custom reception framework before being returned to the reception process. The response messages transformed by the data transformation process are returned to the service requester via the reception process.

The following figure shows the relationship between the data flow and data transformation in a custom reception.

Figure A–2: Relationship between data flow and data transformation in a custom reception



Legend:

- : Flow of service component invoking requests/responses
- : Data transformation processing

The tables below show the combinations of formats for which data transformation of request messages and response messages is supported.

(a) Format combinations supported for data transformation of request messages

The following table shows the request message formats the custom reception framework can subject to data transformation when a business process is invoked:

Table A–1: Request message formats for which custom reception framework supports data transformation

Request message format		Data transformation supported
Reception message format	Service component message format	
Binary	-- (not specified)	Yes
	Binary	Yes
	XML	Yes
XML	-- (not specified)	Yes
	Binary	Yes
	XML	Yes

Legend:

Yes: Can be transformed.

(b) Format combinations supported for data transformation of response messages

The following table shows the response message formats the custom reception framework can subject to data transformation when a business process is invoked:

Table A–2: Response message formats for which the custom reception framework supports data transformation

Response message format		Data transformation supported
Service component message format	Reception message format	
Binary	-- (not specified)	Yes
	Binary	Yes
	XML	Yes
XML	-- (not specified)	Yes
	Binary	Yes
	XML	Yes

Legend:

Yes: Can be transformed.

## A.2 Custom reception development

This section describes the development of custom receptions.

Custom receptions are developed in the development environment.

### (1) File structure used in custom reception development

The structure of the custom reception EAR file created in the development environment is as follows:

```

|- META-INF
| |- application.xml           ... 1.
| |- cosminexus.xml          ... 2.
| |- MANIFEST.MF             ... 3.
|- Reception process file     ... 4.
    
```

1. Deployment descriptor for EAR file
2. Application property file for EAR file
3. Manifest file for EAR file
4. Archive file of created reception processes

Do not change the structure of the custom reception EAR file. If you want to use certain definition information required to operate reception processes, define the information in a custom reception user file created for that purpose. For details on custom reception user files, see *A.2(3)(d) Custom reception user file*.

## (2) Creating a reception process file

Developers create reception process files by archiving the implementation files of the reception process together with the necessary definition files.

### ! Important note

The resources reserved during execution of a reception process are not automatically released when the process has finished. When the system is under a heavy load, this might cause an `OutOfMemoryError` due to insufficient memory in the Java heap or Permanent heap area. To avoid this issue, you need to implement processing that releases resources as appropriate or performs error handling such as rollback when an `OutOfMemoryError` occurs.

The reception process file is described in detail below.

#### (a) Format of reception process file

You can select the WAR or EJB-JAR file format for a reception process file. Select the format that is appropriate for the protocol supported by the reception process.

#### (b) Libraries used to implement reception processes

Use the library JAR files of the APIs provided by the custom reception framework to implement reception processes. When implementing a reception process, the following JAR files must be specified in the classpath:

---

```
Service-Platform-installation-directory\CSC\lib\cscmsg.jar
Service-Platform-installation-directory\CSC\lib\cscmsg_urecp_custom.jar
```

---

#### (c) Processing implemented in a reception process

The following processing is implemented in a reception process:

1. Processing that receives service component invocation requests from the service requester
2. Processing that passes request messages to the custom reception framework
3. Processing that returns to the service requester any response messages returned by the custom reception framework

For details on the custom reception framework APIs used during implementation, see *A.3 APIs of the custom reception framework*.

#### (d) Setting deployment descriptors

Set the deployment descriptor for the reception process file.

Set the content as appropriate for the custom reception you are defining.

#### (e) Creating reception process files

Create a reception process file as an archive containing the implementation files for the reception process and the necessary definition files.

The creation procedure depends on the format of the reception process file.

Do not give a reception process file a name that starts with `csc`. If the file name starts with `csc`, normal operation is not guaranteed.

### (3) Creating definition files

The following four definition files are set in a custom reception:

- Deployment descriptor (`application.xml`)
- Application property file (`cosminexus.xml`)
- Manifest file (`MANIFEST.MF`)
- Custom reception user file

Each file is described below.

#### (a) Deployment descriptor (`application.xml`)

Set the deployment descriptor for the custom reception EAR file.

To set the deployment descriptor, add the settings for the reception process to the template provided by Service Platform. The template file is located in the following directory:

`Service-Platform-installation-directory\CSC\samples\urecp\application.xml`

The table below shows which elements and values in the deployment descriptor template file are editable. Note that the DTD of the deployment descriptor is `application_1_3.dtd`.

Table A-3: Elements in deployment descriptor template file

Element	Editable
<code>&lt;?xml version="1.0" encoding="UTF-8"?&gt;</code>	No
<code>&lt;!DOCTYPE application PUBLIC "-//Sun Microsystems, Inc.//DTD J2EE Application 1.3//EN" "http://java.sun.com/dtd/application_1_3.dtd"&gt;</code>	No
<code>&lt;!-- All Rights Reserved. Copyright (C) 2010, Hitachi, Ltd. --&gt;</code>	No
<code>&lt;application&gt;</code>	No
<code>&lt;display-name&gt;CSCMsgCustomServiceDelivery&lt;/display-name&gt;</code>	No
<code>&lt;description&gt;&lt;/description&gt;</code>	Yes
<code>&lt;module&gt;</code>	Yes
<code>&lt;web&gt;</code>	Yes
<code>&lt;web-uri&gt;web-uri&lt;/web-uri&gt;</code>	Yes
<code>&lt;context-root&gt;context-root&lt;/context-root&gt;</code>	Yes
<code>&lt;/web&gt;</code>	Yes
<code>&lt;/module&gt;</code>	Yes
<code>&lt;module&gt;</code>	Yes
<code>&lt;ejb&gt;ejb&lt;/ejb&gt;</code>	Yes
<code>&lt;/module&gt;</code>	Yes
<code>&lt;module&gt;</code>	Yes
<code>&lt;java&gt;java&lt;/java&gt;</code>	Yes
<code>&lt;/module&gt;</code>	Yes
<code>&lt;/application&gt;</code>	No

Legend:

Yes: Can be edited.

No: Cannot be edited.

**(b) Application property file (cosminexus.xml)**

Set the application property file for the custom reception EAR file.

To set the application property file, add the settings for the reception process to the template file provided by Service Platform. The template file is located in the following directory:

*Service-Platform-installation-directory\CSC\samples\urecp\cosminexus.xml*

The table below shows which elements and values in the template of the application property file are editable. Note that the DTD of the application property file is `cosminexus_8_0.dtd`.

**Table A-4: Elements in the application property template file**

Element	Editable
<code>&lt;?xml version="1.0" encoding="UTF-8"?&gt;</code>	No
<code>&lt;!DOCTYPE cosminexus-app PUBLIC "-//Hitachi, Ltd.//DTD Cosminexus 8.0//EN" "file:///C:/Program%20Files/Hitachi/Cosminexus/CC/admin/dtds/cosminexus_8_0.dtd"&gt;</code>	No
<code>&lt;!-- All Rights Reserved. Copyright (C) 2010, Hitachi, Ltd. --&gt;</code>	No
<code>&lt;cosminexus-app&gt;</code>	No
<code>&lt;start-order&gt;180&lt;/start-order&gt;</code>	No
<code>&lt;ejb-jar&gt;</code>	No
<code>&lt;module-name&gt;cscmsg_urecp_custom.jar&lt;/module-name&gt;</code>	No
<code>&lt;session&gt;</code>	No
<code>&lt;ejb-name&gt;CSCMsgCustomFwInitializerEJB&lt;/ejb-name&gt;</code>	No
<code>&lt;maximum-sessions&gt;0&lt;/maximum-sessions&gt;</code>	No
<code>&lt;stateless&gt;</code>	No
<code>&lt;pooled-instance&gt;</code>	No
<code>&lt;minimum&gt;1&lt;/minimum&gt;</code>	No
<code>&lt;maximum&gt;1&lt;/maximum&gt;</code>	No
<code>&lt;/pooled-instance&gt;</code>	No
<code>&lt;/stateless&gt;</code>	No
<code>&lt;start-order&gt;10&lt;/start-order&gt;</code>	No
<code>&lt;/session&gt;</code>	No
<code>&lt;session&gt;</code>	No
<code>&lt;ejb-name&gt;CSCMsgCustomOperationCallerEJB&lt;/ejb-name&gt;</code>	No
<code>&lt;maximum-sessions&gt;0&lt;/maximum-sessions&gt;</code>	No
<code>&lt;stateless&gt;</code>	No
<code>&lt;pooled-instance&gt;</code>	No
<code>&lt;minimum&gt;0&lt;/minimum&gt;</code>	No
<code>&lt;maximum&gt;0&lt;/maximum&gt;</code>	No
<code>&lt;/pooled-instance&gt;</code>	No
<code>&lt;/stateless&gt;</code>	No
<code>&lt;start-order&gt;20&lt;/start-order&gt;</code>	No

## A. Custom Reception

Element	Editable
<code>&lt;ejb-transaction-timeout&gt;</code>	No
<code>&lt;method&gt;</code>	No
<code>&lt;method-name&gt;*&lt;/method-name&gt;</code>	No
<code>&lt;/method&gt;</code>	No
<code>&lt;transaction-timeout&gt;0&lt;/transaction-timeout&gt;</code>	No
<code>&lt;/ejb-transaction-timeout&gt;</code>	No
<code>&lt;/session&gt;</code>	No
<code>&lt;/ejb-jar&gt;</code>	No
<code>&lt;/cosminexus-app&gt;</code>	No

### Legend:

No: Cannot be edited.

### Note 1

Do not add settings for modules whose names start with `csc`. That is, do not specify a value starting with `csc` in a `<module-name>` element.

### Note 2

Do not set a method timeout period (a `<method-observation-timeout>` element and an `<ejb-method-observation-timeout>` element).

### Note 3

Set a value greater than 50 for the start/stop order (`<start-order>` element) of J2EE applications associated with reception processes.

### (c) Manifest file (MANIFEST.MF)

You can create a manifest file (`MANIFEST.MF`) for the custom reception EAR file.

Note that creating a manifest file is optional.

### (d) Custom reception user file

Create a custom reception user file in situations where, for example, you want to store the settings for a reception process in an external file. The reception process can use a custom reception framework API to read the custom reception user file. For details on the API used to read the custom reception user file, see *A.3 APIs of the custom reception framework*.

Register the custom reception user file as a self-defined file when defining the custom reception in the development environment. For details on defining custom receptions, see *A.4 Custom reception definition*.

## (4) Creating custom reception EAR files

Use commands such as `ant` and `jar` to create a custom reception EAR file that archives the reception process file and associated definition files.

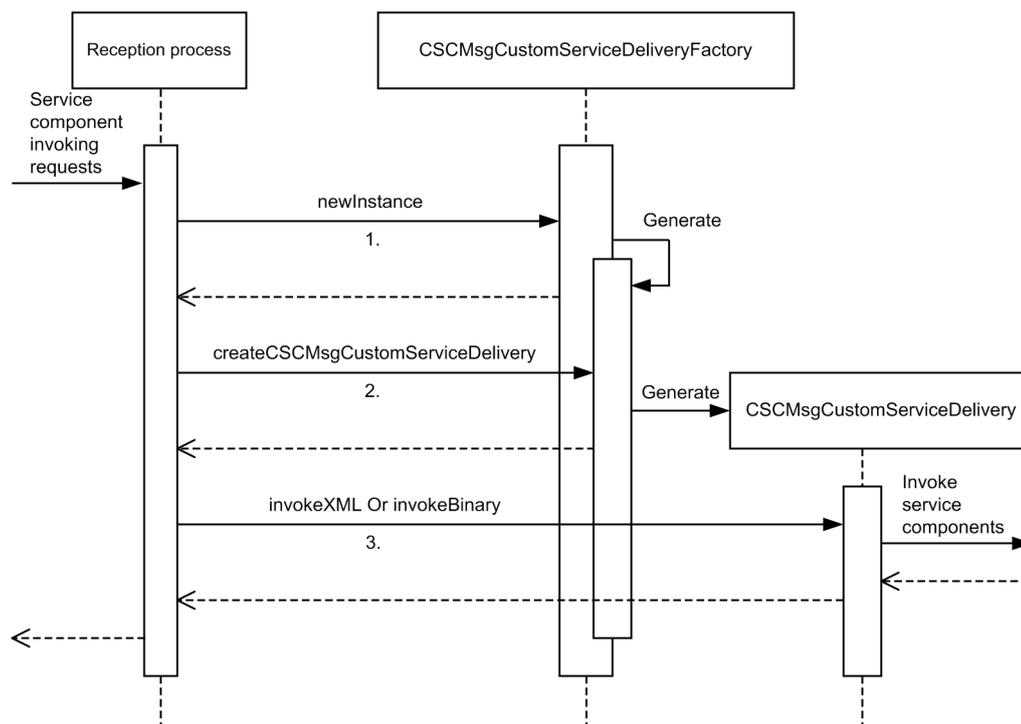
## A.3 APIs of the custom reception framework

This section describes the APIs provided by the custom reception framework.

### (1) Invocation order of custom reception framework APIs

The following figure shows the order in which a reception process invokes the custom reception framework APIs.

Figure A-3: Invocation order of custom reception framework APIs



1. Invoke the `newInstance` method of the `CSCMsgCustomServiceDeliveryFactory` class and acquire the `CSCMsgCustomServiceDeliveryFactory` object.
2. Invoke the `createCSCMsgCustomServiceDelivery` method of the `CSCMsgCustomServiceDeliveryFactory` object acquired in step 1, and then acquire the `CSCMsgCustomServiceDelivery` object.
3. Invoke the `invokeXML` method or `invokeBinary` method of the `CSCMsgCustomServiceDelivery` object acquired in step 2, and then pass the request message to the custom reception framework.

## (2) CSCMsgCustomServiceDeliveryFactory class

### (a) Description

The `CSCMsgCustomServiceDeliveryFactory` class is a factory class, whose functionality includes generating `CSCMsgCustomServiceDeliveryFactory` objects and `CSCMsgCustomServiceDelivery` objects.

### (b) Format

```
package jp.co.Hitachi.soft.csc.msg.message.reception.custom;
public abstract class CSCMsgCustomServiceDeliveryFactory {
    public static CSCMsgCustomServiceDeliveryFactory newInstance();
    public CSCMsgCustomServiceDelivery createCSCMsgCustomServiceDelivery();
}
```

### (c) Methods

The following table lists the methods associated with the `CSCMsgCustomServiceDeliveryFactory` class:

Method name	Function
<code>newInstance</code> method	Generates a <code>CSCMsgCustomServiceDeliveryFactory</code> object.
<code>createCSCMsgCustomServiceDelivery</code> method	Generates a <code>CSCMsgCustomServiceDelivery</code> object.

**newInstance method**

**Description**

This method generates a `CSCMsgCustomServiceDeliveryFactory` object.

**Format**

```
public static CSCMsgCustomServiceDeliveryFactory newInstance();
```

**Parameters**

None.

**Exceptions**

None.

**Return values**

- `CSCMsgCustomServiceDeliveryFactory`  
Returns a `CSCMsgCustomServiceDeliveryFactory` object.

**createCSCMsgCustomServiceDelivery method**

**Description**

This method generates a `CSCMsgCustomServiceDelivery` object.

**Format**

```
public CSCMsgCustomServiceDelivery createCSCMsgCustomServiceDelivery();
```

**Parameters**

None.

**Exceptions**

None.

**Return values**

- `CSCMsgCustomServiceDelivery`  
Returns a `CSCMsgCustomServiceDelivery` object.

### (3) `CSCMsgCustomServiceDelivery` interface

(a) **Description**

This interface provides the methods used by the reception process of a custom reception to pass messages to and from the custom reception framework. It also provides a method for acquiring custom reception user files registered in the development environment.

(b) **Format**

```
package jp.co.Hitachi.soft.csc.msg.message.reception.custom;
public interface CSCMsgCustomServiceDelivery {
    public String invokeXML(String cscCorrelationID,
        String cscServiceOperationName, String msg)
        throws CSCMsgServerException;
    public byte[] invokeXML(String cscCorrelationID,
        String cscServiceOperationName, byte[] msg)
        throws CSCMsgServerException;
    public String invokeXML(String cscCorrelationID, String msg)
        throws CSCMsgServerException;
    public byte[] invokeXML(String cscCorrelationID, byte[] msg)
        throws CSCMsgServerException;

    public byte[] invokeBinary(String cscCorrelationID,
        String cscServiceOperationName,
        int requestMessageLength, byte[] msg)
        throws CSCMsgServerException;
    public byte[] invokeBinary(String cscCorrelationID,
        int requestMessageLength, byte[] msg)
        throws CSCMsgServerException;
    public InputStream getUserFile(String fileName);
}
```

**(c) Methods**

The following table lists the methods provided by the `CSCMsgCustomServiceDelivery` interface:

Method name	Function
<code>invokeXML</code> method (format 1)	Sends XML messages to the service associated with the specified operation name, and returns the responses to those messages.
<code>invokeXML</code> method (format 2)	Sends byte-array XML messages to the service associated with the specified operation name, and returns the responses to those messages. Responses will also be in byte-array format.
<code>invokeXML</code> method (format 3) <sup>#1</sup>	Sends XML messages to the service associated with the default operation name, and returns the responses to those messages.
<code>invokeXML</code> method (format 4) <sup>#1</sup>	Sends byte-array XML messages to the service associated with the default operation name, and returns the responses to those messages. Responses will also be in byte-array format.
<code>invokeBinary</code> method (format 1)	Sends binary messages to the service associated with the specified operation name, and returns the responses to those messages.
<code>invokeBinary</code> method (format 2) <sup>#2</sup>	Sends binary messages to the service associated with the default operation name, and returns the responses to those messages.
<code>getUserFile</code> method	Returns the <code>InputStream</code> value of the custom reception user file registered in the development environment.

#1

Equivalent to the `invokeXML` method (format 1) with the operation name omitted.

#2

Equivalent to the `invokeBinary` method (format 1) with the operation name omitted.

#### `invokeXML` method (format 1)

##### Description

This method sends XML messages to the service associated with the specified operation name, and returns the responses to those messages.

##### Format

```
public String invokeXML(String cscCorrelationID,
    String cscServiceOperationName, String msg)
    throws CSCMsgServerException;
```

##### Parameters

- `cscCorrelationID` (client correlation ID)

The client correlation ID is a correlation ID that uniquely identifies request messages from the service requester.

A `CSCMsgServerException` exception is thrown if the client correlation ID does not comply with the following restrictions:

- Size: 255 or fewer characters

- Usable characters: One-byte alphanumeric characters, underscores (`_`), periods (`.`), and hyphens (`-`).

The client correlation ID is used to establish mapping between the request messages from the service requester and the trace information and log data managed by the HCSC server. Therefore, you need to specify a different ID for each request message sent to the HCSC server.

If you do not intend to specify a client correlation ID, specify `null` or an empty string (`""`).

- `cscServiceOperationName` (operation name)

The operation name associated with the service that is the recipient of the request.

This parameter specifies the operation name of the service component defined in the development environment.

A `CSCMsgServerException` exception is thrown if the operation name does not comply with the following restrictions:

## A. Custom Reception

- Size: 255 or fewer characters

- Usable characters: Characters defined in the NCName data type in the XML schema

A `CSCMsgServerException` exception is thrown if you specify an operation name that is not defined in the User-Defined Reception Definition window.

If you do not intend to specify an operation name (you want to use the default operation name), specify `null` or an empty string (`""`).

- `msg` (user message)

The request message from the service requester (XML format).

If there is no request message, specify `null` or an empty string (`""`).

### Exceptions

- `CSCMsgServerException`

Thrown in the following circumstances:

- An invalid value is specified in a method parameter.

- An error occurred during data transformation of a request message or response message.

- An error occurred in the service component targeted by the request.

### Return values

- `String`

- If the communication model is synchronous

Returns the response message (XML format) corresponding to the execution request of the service component.

If there is no response message, `null` is returned.

- If the communication model is asynchronous

`null` is returned.

## invokeXML method (format 2)

### Description

This method sends byte-array XML messages to the service associated with the specified operation name, and returns the responses to those messages. Responses will also be in byte-array format.

When a custom reception handles user messages in byte array format, you can use this method to skip the process of transforming messages to string format.

### Format

```
public byte[] invokeXML(String cscCorrelationID,  
    String cscServiceOperationName, byte[] msg)  
    throws CSCMsgServerException;
```

### Parameters

- `cscCorrelationID` (client correlation ID)

The client correlation ID is a correlation identifier that uniquely identifies request messages from the service requester.

A `CSCMsgServerException` exception is thrown if the client correlation ID does not comply with the following restrictions:

- Size: 255 or fewer characters

- Usable characters: One-byte alphanumeric characters, underscores (`_`), periods (`.`), and hyphens (`-`).

The client correlation ID is used to establish mapping between the request messages from the service requester and the trace information and log data managed by the HCSC server. Therefore, you need to specify a different ID for each request message sent to the HCSC server.

If you do not intend to specify a client correlation ID, specify `null` or an empty string (`""`).

- `cscServiceOperationName` (operation name)

The operation name associated with the service that is the recipient of the request.

This parameter specifies the operation name of the service component defined in the development environment.

A `CSCMsgServerException` exception is thrown if the operation name does not comply with the following restrictions:

- Size: 255 or fewer characters
  - Usable characters: Characters defined in the NCName data type in the XML Schema
- A `CSCMsgServerException` exception is thrown if you specify an operation name that is not defined in the User-Defined Reception Definition window.

If you do not intend to specify an operation name (you want to use the default operation name), specify `null` or an empty string (`""`).

- `msg` (user message)  
The request message from the service requester (XML format).  
If there is no request message, specify `null`.

#### Exceptions

- `CSCMsgServerException`  
Thrown in the following circumstances:
  - An invalid value is specified in a method parameter.
  - An error occurred during data transformation of a request message or response message.
  - An error occurred in the service component targeted by the request.

#### Return values

- `byte[]`
  - If the communication model is synchronous  
Returns the response message (XML format) corresponding to the execution request of the service component.  
If there is no response message, `null` is returned.
  - If the communication model is asynchronous  
`null` is returned.

#### invokeXML method (format 3)

##### Description

This method sends XML messages to the service associated with the default operation name, and returns the responses to those messages.

##### Format

```
public String invokeXML(String cscCorrelationID, String msg)
    throws CSCMsgServerException;
```

##### Parameters

- `cscCorrelationID` (client correlation ID)  
The client correlation ID is a correlation identifier that uniquely identifies request messages from the service requester.  
A `CSCMsgServerException` exception is thrown if the client correlation ID does not comply with the following restrictions:
  - Size: 255 or fewer characters
  - Usable characters: One-byte alphanumeric characters, underscores (`_`), periods (`.`), and hyphens (`-`).
 The client correlation ID is used to establish mapping between the request messages from the service requester and the trace information and log data managed by the HCSC server. Therefore, you need to specify a different ID for each request message sent to the HCSC server.  
If you do not intend to specify a client correlation ID, specify `null` or an empty string (`""`).
- `msg` (user message)  
The request message from the service requester (XML format).  
If there is no request message, specify `null` or an empty string (`""`).

#### Exceptions

- `CSCMsgServerException`  
Thrown in the following circumstances:
  - An invalid value is specified in a method parameter.
  - An error occurred during data transformation of a request message or response message.

- An error occurred in the service component targeted by the request.

Return values

- `String`

- If the communication model is synchronous

Returns the response message (XML format) corresponding to the execution request of the service component.

If there is no response message, `null` is returned.

- If the communication model is asynchronous  
`null` is returned.

`invokeXML` method (format 4)

Description

This method sends byte-array XML messages to the service associated with the default operation name, and returns the responses to those messages. Responses will also be in byte-array format.

When a custom reception handles user messages in byte array format, you can use this method to skip the process of transforming messages to string format.

Format

```
public byte[] invokeXML(String cscCorrelationID, byte[] msg)
    throws CSCMsgServerException;
```

Parameters

- `cscCorrelationID` (client correlation ID)

The client correlation ID is a correlation identifier that uniquely identifies request messages from the service requester.

A `CSCMsgServerException` exception is thrown if the client correlation ID does not comply with the following restrictions:

- Size: 255 or fewer characters
- Usable characters: One-byte alphanumeric characters, underscores (`_`), periods (`.`), and hyphens (`-`).

The client correlation ID is used to establish mapping between the request messages from the service requester and the trace information and log data managed by the HCSC server. Therefore, you need to specify a different ID for each request message sent to the HCSC server.

If you do not intend to specify a client correlation ID, specify `null` or an empty string (`""`).

- `msg` (user message)

The request message from the service requester (XML format).

If there is no request message, specify `null`.

Exceptions

- `CSCMsgServerException`

Thrown in the following circumstances:

- An invalid value is specified in a method parameter.
- An error occurred during data transformation of a request message or response message.
- An error occurred in the service component targeted by the request.

Return values

- `byte[]`

- If the communication model is synchronous

Returns the response message (XML format) corresponding to the execution request of the service component.

If there is no response message, `null` is returned.

- If the communication model is asynchronous  
`null` is returned.

## invokeBinary method (format 1)

## Description

This method sends binary messages to the service associated with the specified operation name, and returns the responses to those messages.

## Format

```
public byte[] invokeBinary(String cscCorrelationID,
    String cscServiceOperationName, int requestMessageLength, byte[] msg)
    throws CSCMsgServerException;
```

## Parameters

- **cscCorrelationID** (client correlation ID)  
The client correlation ID is a correlation identifier that uniquely identifies request messages from the service requester.  
A `CSCMsgServerException` exception is thrown if the client correlation ID does not comply with the following restrictions:
  - Size: 255 or fewer characters
  - Usable characters: One-byte alphanumeric characters, underscores (`_`), periods (`.`), and hyphens (`-`).
 The client correlation ID is used to establish mapping between the request messages from the service requester and the trace information and log data managed by the HCSC server. Therefore, you need to specify a different ID for each request message sent to the HCSC server.  
If you do not intend to specify a client correlation ID, specify `null` or an empty string (`""`).
- **cscServiceOperationName** (operation name)  
The operation name associated with the service that is the recipient of the request.  
This parameter specifies the operation name of the service component defined in the development environment.  
A `CSCMsgServerException` exception is thrown if the operation name does not comply with the following restrictions:
  - Size: 255 or fewer characters
  - Usable characters: Characters defined in the `NCName` data type in the XML Schema
 A `CSCMsgServerException` exception is thrown if you specify an operation name that is not defined in the User-Defined Reception Definition screen.  
If you do not intend to specify an operation name (you want to use the default operation name), specify `null` or an empty string (`""`).
- **requestMessageLength** (user message length)  
The length of the request message.  
A `CSCMsgServerException` exception is thrown if the user message length does not comply with the following restrictions:
  - An integer value of 0 or greater
  - An integer value not exceeding the byte length of the user message (`msg` parameter)
 If there is no request message, specify 0.  
If you specify a smaller value than the byte length of the user message (`msg` parameter), the method uses the user message length (specified by the `requestMessageLength` parameter) as the length of the request message. In other words, the request message passed to the service component only includes the part of the user message (`msg` parameter) that spans the user message length (`requestMessageLength` parameter) + 1 byte.
- **msg** (user message)  
The request message from the service requester (binary format).  
If there is no request message, specify `null` or a 0-byte byte array.

## Exceptions

- `CSCMsgServerException`  
Thrown in the following circumstances:
  - An invalid value is specified in a method parameter.
  - An error occurred during data transformation of a request message or response message.

- An error occurred in the service component targeted by the request.

Return values

- `byte[]`

- If the communication model is synchronous

Returns the response message (binary format) corresponding to the execution request of the service component.

If there is no response message, `null` is returned.

- If the communication model is asynchronous  
`null` is returned.

`invokeBinary` method (format 2)

Description

This method sends binary messages to the service associated with the default operation name, and returns the responses to those messages.

Format

```
public byte[] invokeBinary(String cscCorrelationID,  
    int requestMessageLength, byte[] msg)  
    throws CSCMsgServerException;
```

Parameters

- `cscCorrelationID` (client correlation ID)

The client correlation ID is a correlation identifier that uniquely identifies request messages from the service requester.

A `CSCMsgServerException` exception is thrown if the client correlation ID does not comply with the following restrictions:

- Size: 255 or fewer characters
- Usable characters: One-byte alphanumeric characters, underscores (`_`), periods (`.`), and hyphens (`-`).

The client correlation ID is used to establish mapping between the request messages from the service requester and the trace information and log data managed by the HCSC server. Therefore, you need to specify a different ID for each request message sent to the HCSC server.

If you do not intend to specify a client correlation ID, specify `null` or an empty string (`""`).

- `requestMessageLength` (user message length)

The length of the request message.

A `CSCMsgServerException` exception is thrown if the user message length does not comply with the following restrictions:

- An integer value of 0 or greater
- An integer value not exceeding the byte length of the user message (`msg` parameter)

If there is no request message, specify 0.

If you specify a smaller value than the byte length of the user message (`msg` parameter), the method uses the user message length (specified by the `requestMessageLength` parameter) as the length of the request message. In other words, the request message passed to the service component only includes the part of the user message (`msg` parameter) that spans the user message length (`requestMessageLength` parameter) + 1 byte.

- `msg` (user message)

The request message from the service requester (binary format).

If there is no request message, specify `null` or a 0-byte byte array.

Exceptions

- `CSCMsgServerException`

Thrown in the following circumstances:

- An invalid value is specified in a method parameter.
- An error occurred during data transformation of a request message or response message.
- An error occurred in the service component targeted by the request

## Return values

- `byte[]`
  - If the communication model is synchronous  
Returns the response message (binary format) corresponding to the execution request of the service component.
  - If there is no response message, `null` is returned.
  - If the communication model is asynchronous  
`null` is returned.

`getUserFile` method

## Description

This method reads the custom reception user file specified in the parameter, and returns the file contents as a stream.

The contents of the custom reception user file are returned as an unprocessed stream regardless of the file format. The appropriate processing for the file format must be applied at the invocation source. For example, you can use the `load` method of the `Properties` class to load the custom reception user file as a property file.

After using the stream acquired by this method, close the stream at the invocation source.

## Format

```
public InputStream getUserFile(String fileName);
```

## Parameters

- `fileName`  
The name of the custom reception user file.  
Specify the name of the custom reception user file you registered when defining the custom reception in the development environment.  
You can use any characters.

## Exceptions

None.

## Return values

- `InputStream`  
Returns the custom reception user file as a stream.  
If the method cannot locate the file specified in the `fileName` parameter, `null` is returned.

#### (4) Examples of API usage

The following are examples in which the APIs of the custom reception framework are used to implement reception processes in EJB (Stateless Session Bean) format:

- `MyReception.java`

---

```
package com.example.sample.reception.ejb;

import javax.ejb.EJBObject;

/**
 * Example of reception process (Stateless Session Bean) implementation (Remote interface)
 */
public interface MyReception extends EJBObject {
    public String invoke(String cscCorrelationID, String cscServiceOperationName, String msg)
        throws RemoteException;
}
```

---

- `MyReceptionHome.java`

---

```
package com.example.sample.reception.ejb;

import java.rmi.RemoteException;

import javax.ejb.CreateException;
```

---

## A. Custom Reception

---

```
import javax.ejb.EJBHome;

/**
 * Example of reception process (Stateless Session Bean) implementation (Home interface)
 */
public interface MyReceptionHome extends EJBHome {
    public MyReception create() throws RemoteException, CreateException;
}
```

---

- MyReceptionBean.java

---

```
package com.example.sample.reception.ejb;

import java.rmi.RemoteException;

import javax.ejb.CreateException;
import javax.ejb.EJBException;
import javax.ejb.SessionBean;
import javax.ejb.SessionContext;

import jp.co.Hitachi.soft.csc.msg.message.reception.CSCMsgServerException;
import jp.co.Hitachi.soft.csc.msg.message.reception.custom.CSCMsgCustomServiceDelivery;
import
jp.co.Hitachi.soft.csc.msg.message.reception.custom.CSCMsgCustomServiceDeliveryFactory;

/**
 * Example of reception process (Stateless Session Bean) implementation (Bean class)
 */
public class MyReceptionBean implements SessionBean {
    private static final CSCMsgCustomServiceDeliveryFactory factory =
        CSCMsgCustomServiceDeliveryFactory.newInstance();
    private CSCMsgCustomServiceDelivery delivery;
    private SessionContext context;

    public MyReceptionBean() {
    }

    public void ejbActivate() throws EJBException, RemoteException {
    }

    public void ejbCreate() throws CreateException {
        // Generate CSCMsgCustomServiceDelivery object
        delivery = factory.createCSCMsgCustomServiceDelivery();
    }

    public void ejbPassivate() throws EJBException, RemoteException {
    }

    public void ejbRemove() throws EJBException, RemoteException {
        // Discard CSCMsgCustomServiceDelivery object
        delivery = null;
    }

    public String invoke(final String cscCorrelationID, final String cscServiceOperationName,
final String msg) {
        try {
            // Invoke custom reception framework
            final String response = delivery.invokeXML(cscCorrelationID, cscServiceOperationName,
msg);
            return response;
        } catch (CSCMsgServerException e) {
            // Catch custom reception framework exceptions
            // Implement exception handling
        }
    }

    public void setSessionContext(final SessionContext sc) throws EJBException {
        context = sc;
    }
}
```

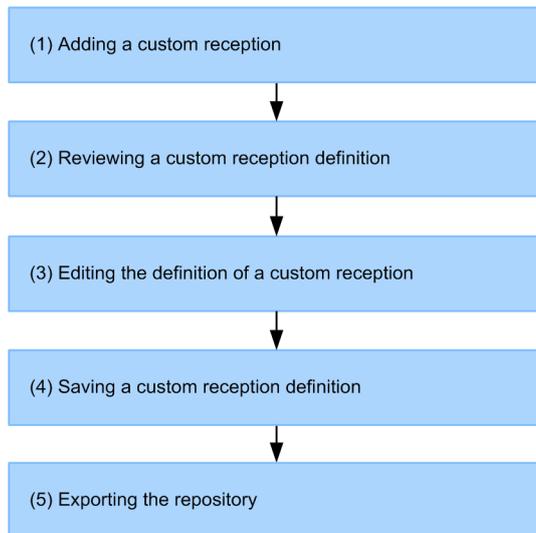
---

## A.4 Custom reception definition

This section describes how to define a custom reception.

The following figure shows the flow of custom reception definition.

Figure A-4: Flow of custom reception definition



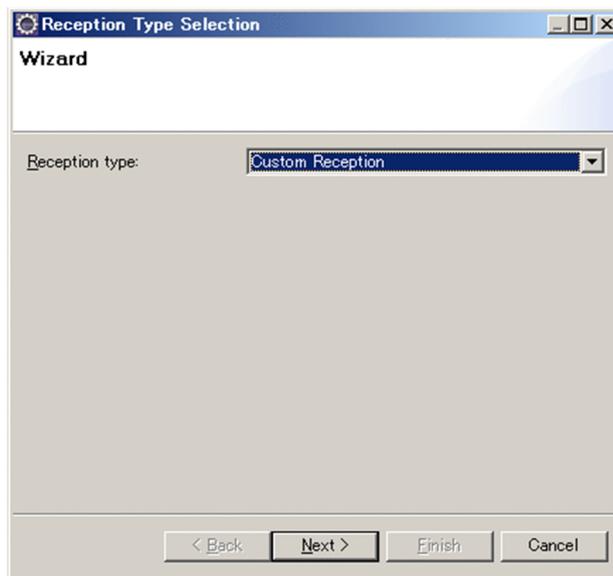
Each part of the process is described below.

### (1) Adding a custom reception

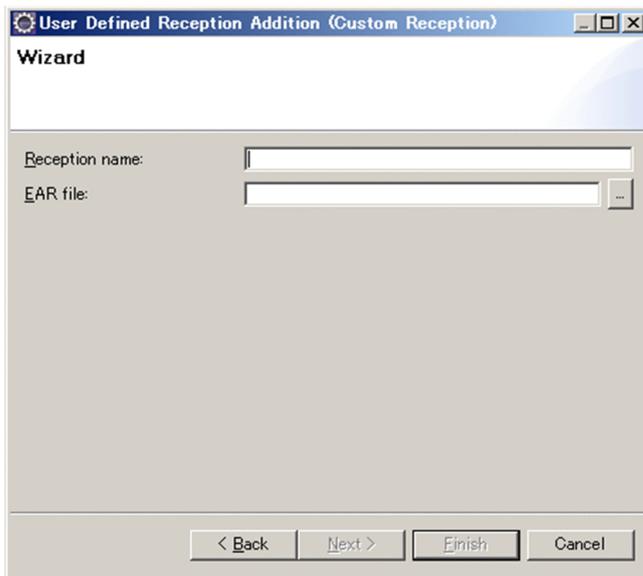
Import a custom reception EAR file you created in the development environment.

To add a new custom reception:

1. In the Service Definition List in the tree view, right-click the service (business process) for which you want to add a custom reception.  
The pop-up menu for the service list appears.
2. From the pop-up menu, select **Add User Defined Reception**.  
The Reception Type Selection wizard appears.



3. From the **Reception type:** drop-down list, select **Custom Reception**.
4. Click **Next**.  
The User Defined Reception Addition (Custom Reception) wizard appears.



5. Specify values in **Reception name** and **EAR file**.

- **Reception name**

Specify the name of the custom reception as a character string of 1 to 40 bytes. The reception name must be unique within the business process.

- **EAR file**

Specify the absolute path of the EAR file you want to import. Do not specify the file by using a relative path or in UNC format.

You can also select the EAR file in the dialog box that appears when you click the ... button.

6. Click **Finish**.

The custom reception is added to the business process, and then the User-defined Reception Definition window appears.

For details on the User-defined Reception Definition window, see *1.2.6 User-Defined Reception Definition Window* in the manual *Service Platform Reference Guide*.

## (2) Reviewing a custom reception definition

After importing a custom reception, check the contents of the custom reception definition. You can check the definition of a custom reception in the User-defined Reception Definition window. This window appears after you add a custom reception by using the Add Custom Reception wizard.

You can display the User-defined Reception Definition window at any other time by using the following step.

1. In the service definition list in the tree view, double-click the custom reception.

The User-defined Reception Definition window appears.

## (3) Editing the definition of a custom reception

In the User-defined Reception Definition window, you can edit aspects of the custom reception definition you imported, such as the reception name, reception ID, and operation name.

For details on how to display the User-defined Reception Definition window at any time, see *(2) Reviewing a custom reception definition*. For details on the User-defined Reception Definition window and the values you can enter, see *1.2.6 User-Defined Reception Definition Window* in the manual *Service Platform Reference Guide*.

## (4) Saving a custom reception definition

You can save the custom reception definition being edited in the User-defined Reception Definition window during or after the editing process. The contents of the custom reception are stored in the repository.

If you close the User-defined Reception Definition window without saving the custom reception definition, the Save Resource dialog box appears. You can save the unsaved custom reception definition in this dialog box.

You can use the following methods to save a custom reception:

Method 1

From the Eclipse menu, select **File** and then **Save**.

Method 2

From the Eclipse menu, select **File** and then **Save All**.

Method 3

In the Eclipse toolbar, click the Save icon.

Method 4

In the User-defined Reception Definition window, press **Ctrl + S**.

Note that you might be unable to save the custom reception if invalid data has been entered. In this case, take action according to the message displayed on the screen.

## (5) Exporting the repository

You can save the repository containing the custom reception you defined to a specified directory as a ZIP file.

To export the repository:

1. From the Eclipse menu, select **HCSC-Definer**, **Repository management**, and then **Export repository**. The Export Repository dialog box appears.
2. Specify the directory in which you want to save the repository information, and then enter the file name of the zip file to be saved.
3. Click **Save**.

The zip file is saved to the directory you specified.

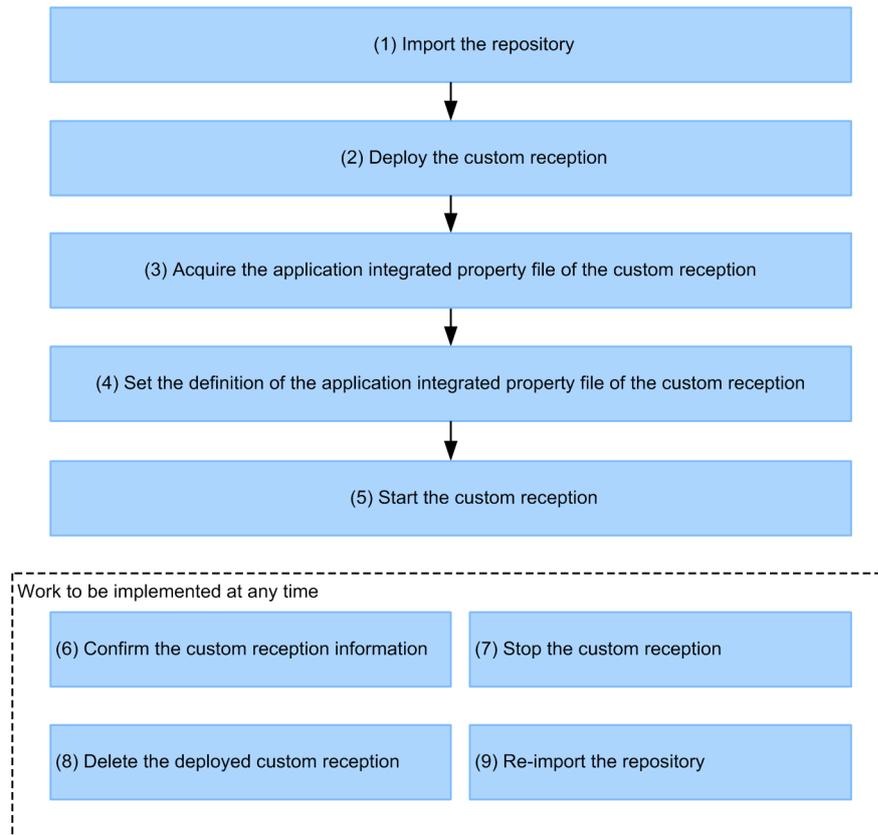
If a file with the same name already exists in the specified directory, a dialog box appears asking if you want to overwrite the file. To overwrite the file, click **Yes**.

## A.5 Custom reception operations

This section describes the operation of custom receptions.

The following figure shows the flow of custom reception operation.

Figure A–5: Flow of custom reception operation



Each aspect of custom reception operation is described below.

### (1) Importing the repository

You can make custom receptions you created in the development environment available in the operating environment by importing the zip file containing repository information to the repository. For details on how to import repository information, see *4.3 Importing a repository* in the manual *Service Platform Setup and Operation Guide*.

To import repository information to the repository, you use the `cscrepctl` command. For details, see *cscrepctl(Importing/Exporting)* in the manual *Service Platform Reference Guide*.

### (2) Deploying custom receptions

To deploy a custom reception, execute the `csccomposedeploy` command in the operating environment. For details, see *csccomposedeploy(Deploying HCSC components)* in the manual *Service Platform Reference Guide*.

The method of deploying custom receptions in the execution environment is the same as when deploying a user-defined reception. The prerequisites are also the same. For details, see *3.1.14 Setting up the HCSC server definition information* in the manual *Service Platform Setup and Operation Guide*.

### (3) Acquiring HITACHI Application Integrated Property files for custom receptions

The following describes the prerequisites and method for acquiring the HITACHI Application Integrated Property File associated with a custom reception.

#### (a) Prerequisites

To acquire the HITACHI Application Integrated Property File for a custom reception, the HCSC server must be running.

For details on how to check the status of the HCSC server, see 5.3.15 *Checking the HCSC server information* in the manual *Service Platform Setup and Operation Guide*.

#### (b) Acquisition method

To acquire the HITACHI Application Integrated Property File for a custom reception, execute the `cscocompoconfig` command in the operating environment.

For details, see *cscocompoconfig(Defining HCSC components)* in the manual *Service Platform Reference Guide*.

The format of the `cscocompoconfig` command is as follows:

---

```
cscocompoconfig -user login-user-ID -pass login-password -operation get -csc HCSC-server-name -name reception-ID-of-custom-reception
```

---

### (4) Setting HITACHI Application Integrated Property File definition information in custom receptions

The following describes how to apply the definition information in a HITACHI Application Integrated Property File to a custom reception, and the prerequisites for doing so.

#### (a) Prerequisites

To apply the definition information in a HITACHI Application Integrated Property File to a custom reception, the HCSC server must be running.

For details on how to check the status of the HCSC server, see 5.3.15 *Checking the HCSC server information* in the manual *Service Platform Setup and Operation Guide*.

#### (b) Method of applying settings

To apply the definition information in the HITACHI Application Integrated Property File to a custom reception, execute the `cscocompoconfig` command in the operating environment.

For details, see *cscocompoconfig(Defining HCSC components)* in the manual *Service Platform Reference Guide*.

The format of the `cscocompoconfig` command is as follows:

When applying definition information to a custom reception deployed on the HCSC server

Execute the command with the HCSC server name and the reception ID of the custom reception specified.

The definition information specified in the HITACHI Application Integrated Property File is applied to the custom reception on the specified HCSC server.

---

```
cscocompoconfig -user login-user-ID -pass login-password -operation set -propfile HITACHI-Application-Integrated-Property-file -csc HCSC-server-name -name reception-ID-of-custom-reception
```

---

When applying definition information to a custom reception deployed on an HCSC server in a cluster

Execute the command with the cluster name and the reception ID of the custom reception specified.

The definition information specified in the HITACHI Application Integrated Property File is applied to the custom reception deployed in the specified cluster.

---

```
cscocompoconfig -user login-user-ID -pass login-password -operation set -propfile HITACHI-Application-Integrated-Property-file -cluster cluster-name -name reception-ID-of-custom-reception
```

---

### (5) Starting custom receptions

The following describes how to start a custom reception, and the prerequisites for doing so.

#### (a) Prerequisites

The following conditions must be satisfied before you can start a custom reception:

- The HCSC server is active.
- The business process in which the custom reception is defined is active.

## A. Custom Reception

For details on how to check the status of the HCSC server, see 5.3.15 *Checking the HCSC server information* in the manual *Service Platform Setup and Operation Guide*.

### (b) Start method (from the GUI)

To start a custom reception from the GUI in the operating environment:

1. In the tree view, double-click the user-defined reception (custom reception) you want to start.  
Information about the custom reception you double-clicked appears on the **Operations** page of the editor area.  
For details on the information shown on the **Operations** page for a custom reception, see the description that relates to user-defined reception operation in 4.4 *Operations Page* in the manual *Service Platform Reference Guide*.
2. Review the information displayed on the **Operations** page, and then click **Start**.  
A dialog box appears in which you can confirm that you want to start the custom reception.
3. Review the information in the dialog box, and then click **OK**.  
The system begins the process of starting the custom reception. The log data generated during start processing is output to the console view.  
If you click **Cancel**, the custom reception does not start and you are returned to the **Operations** page.  
When the system has finished start processing of the custom reception, a dialog box that shows the execution results appears.
4. Review the execution results in the dialog box.
  - If start processing was successful:  
Click **OK** to return to the **Operations** page.  
The status of the custom reception on the **Operations** page is now **active**. At this time, the **Start** button becomes unavailable and the **Stop** button becomes available.
  - If start processing failed:  
A dialog box appears describing the nature of the error. Look up the message ID displayed in the error details and take the appropriate action.

### (c) Start method (from the command line)

To start a custom reception, execute the `cscocompostart` command in the operating environment.

For details, see `cscocompostart(Starting HCSC components)` in the manual *Service Platform Reference Guide*.

The format of the `cscocompostart` command is as follows:

When starting custom receptions deployed to all HCSC servers in the cluster as a batch (at the cluster level)

Execute the command with the cluster name and the `-all` option specified.

The system starts as a batch the custom receptions deployed on all HCSC servers in the cluster specified by the `-cluster` option. Any service adapters and business processes that were not already started also start at this time.

---

```
cscocompostart -user login-user-ID -pass login-password -cluster cluster-name -all
```

---

When starting custom receptions deployed on a specific HCSC server as a batch (at the HCSC server level)

Execute the command with the HCSC server name and the `-all` option specified.

The system starts as a batch the custom receptions deployed on the HCSC server specified by the `-csc` option. Any service adapters and business processes that were not already started also start at this time.

---

```
cscocompostart -user login-user-ID -pass login-password -csc HSCS-server-name -all
```

---

When starting a specific custom reception deployed to every HCSC server in a cluster

Execute the command with the cluster name and the reception ID of the custom reception specified.

---

```
cscocompostart -user login-user-ID -pass login-password -cluster cluster-name -name reception-ID-of-custom-reception
```

---

When starting a specific custom reception deployed on an HCSC server

Execute the command with the HCSC server name and the reception ID of the custom reception specified.

---

```
csccompstart -user login-user-ID -pass login-password -csc HCSC-server-name -name
reception-ID-of-custom-reception
```

---

## (6) Checking custom reception information

To view information about a custom reception, execute the `cscsvcls` command. For details, see *cscsvcls(Displaying the service information)* in the manual *Service Platform Reference Guide*.

By executing the `cscsvcls` command, you can view the file name and status of the definition file set in the development environment or operating environment. For details on how to check this information, see *3.1.7 Checking the service information* in the manual *Service Platform Setup and Operation Guide*.

## (7) Stopping custom receptions

The following describes how to stop a custom reception, and the prerequisites for doing so.

### (a) Prerequisites

The following conditions must be satisfied before you can stop a custom reception:

- The HCSC server is active.  
For details on how to check the status of the HCSC server, see *5.3.15 Checking the HCSC server information* in the manual *Service Platform Setup and Operation Guide*.
- The custom reception is active.  
The method of checking the status of a custom reception is the same as for a user-defined reception. For details, see *5.3.19 Checking the information of user-defined reception* in the manual *Service Platform Setup and Operation Guide*.

### (b) Stop method

To stop a custom reception, execute the `csccompstop` command in the operating environment.

For details, see *csccompstop(Stopping HCSC components)* in the manual *Service Platform Reference Guide*.

The format of the `csccompstop` command is as follows:

When stopping custom receptions deployed to all HCSC servers in a cluster as a batch (at the cluster level)

Execute the command with the cluster name and the `-all` option specified.

The system batch-stops the custom receptions deployed on all HCSC servers in the cluster specified by the `-cluster` option. Any service adapters and business processes that were not already stopped are also stopped at this time.

---

```
csccompstop -user login-user-ID -passlogin-password -cluster cluster-name -all
```

---

When batch-stopping the custom receptions deployed on a specific HCSC server (at the HCSC server level)

Execute the command with the HCSC server name and the `-all` option specified.

The system stops the custom receptions deployed on the HCSC server specified by the `-csc` option. Any service adapters and business processes that were not already stopped are also stopped at this time.

---

```
csccompstop -user login-user-ID -pass login-password -csc HCSC-server-name -all
```

---

When stopping a specific custom reception deployed to every HCSC server in a cluster

Execute the command with the cluster name and the reception ID of the custom reception specified.

---

```
csccompstop -user login-user-ID -pass login-password -cluster cluster-name -name reception-
ID-of-custom-reception
```

---

When stopping a specific custom reception deployed on an HCSC server

Execute the command with the HCSC server name and the reception ID of the custom reception specified.

---

```
csccompstop -user login-user-ID -pass login-password -csc HCSC-server-name -name reception-
ID-of-custom-reception
```

---

When stopping a custom reception, the `csccompostop` command first changes the status of the custom reception in CSC to **inactive**, and then stops the application associated with the custom reception. If a request is received while the custom reception is in the process of being stopped, the request will be processed normally if the custom reception maintained by CSC and the custom reception application are both still active.

### (8) Deleting deployed custom receptions

To delete a custom reception that has been deployed to the execution environment, execute the `csccomoundeploy` command in the operating environment. For details, see *csccomoundeploy(Deleting the deployed HCSC components)* in the manual *Service Platform Reference Guide*.

The method of deleting deployed custom receptions in the execution environment and the prerequisites for doing so are the same as when deleting a user-defined reception. For details on how to delete deployed custom receptions, see *3.3.3 Deleting a user-defined reception* in the manual *Service Platform Setup and Operation Guide*.

### (9) Re-importing the repository

If you edit the definition of a custom reception in the development environment after the repository has been imported to the operating environment, use the following procedure to import the repository again:

1. Export the repository.

Export the repository in which the edited definition is stored.

Export the repository in the development environment.

For details on how to export the repository, see *3.2.2 Exporting a Repository* in the manual *Service Platform Basic Development Guide*.

2. Import the repository.

Import the repository you exported in the development environment to the operating environment.

For details on how to import the repository, see *4.3 Importing a repository* in the *Service Platform Setup and Operation Guide*.

## A.6 Custom reception tuning

This section describes how to set the number of concurrent executions of custom receptions, and the timeout time for transactions.

### (1) Setting the maximum number of concurrent executions (custom receptions)

The following describes how to set the maximum number of concurrently executable custom receptions.

#### (a) Reception processes

You can set the number of concurrent reception process executions to suit the nature of the reception processes when developing and operating the custom reception.

- Setting during development  
Set values for elements in the application property file (`cosminexus.xml`) to suit the nature of the reception processes.
- Setting during operation  
Set values for elements in the HITACHI Application Integrated Property file to suit the nature of the reception processes.

For details on the application property file (`cosminexus.xml`), see *Chapter 2. Cosminexus Application Property File (cosminexus.xml)* in the manual *Application Server Application and Resource Definition Reference Guide*. For details on the HITACHI Application Integrated Property file, see *3.1 HITACHI Application Integrated Property file* in the manual *Application Server Application and Resource Definition Reference Guide*.

**(b) Custom reception framework**

You cannot set the maximum number of concurrent executions of the custom reception framework when developing the custom reception. You can only specify this setting when operating the custom reception.

Set values for the appropriate elements in the HITACHI Application Integrated Property File.

The elements to be set are shown below as Xpath expressions:

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

**(2) Setting the transaction timeout period (custom reception)**

The following describes how to set the transaction timeout period.

**(a) Reception processes**

You can set the transaction timeout period for a reception process during its development and operation to suit the implementation format of the reception process.

- **Setting during development**  
Set values for elements in the application property file (`cosminexus.xml`) to suit the nature of the reception processes.
- **Setting during operation**  
Set values for elements in the HITACHI Application Integrated Property file to suit the content of the reception process.

For details on the application property file (`cosminexus.xml`), see *Chapter 2. Cosminexus Application Property File (cosminexus.xml)* in the manual *Application Server Application and Resource Definition Reference Guide*. For details on the HITACHI Application Integrated Property file, see *3. HITACHI Application Integrated Property file* in the manual *Application Server Application and Resource Definition Reference Guide*.

**(b) Custom reception framework**

You cannot set the transaction timeout period of the custom reception framework when developing the custom reception. You can only specify this setting when operating the custom reception.

Set values for the appropriate elements in the HITACHI Application Integrated Property File.

The elements to be set are shown below as Xpath expressions:

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

**A.7 Method of notifying error in custom reception**

If a failure occurs when using a custom reception to invoke a service component from the service requester, an error is returned to the service requester. This appendix describes how the error reaches the service requester from its point of origin.

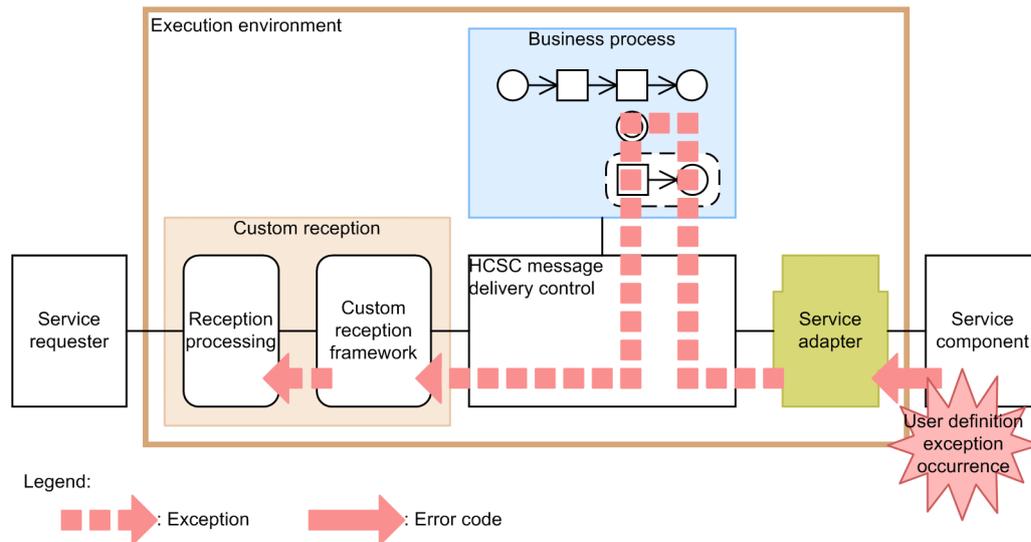
(1) When a user-defined exception error is returned from a service component (when using a business process)

The following describes how a user-defined exception error (when using a business process) propagates through the system when returned by a service component.

(a) When fault processing does not transform the error information into a service message

The following figure shows how a user-defined exception error returned by a service component (when using a business process) propagates (without fault processing) through the system in the context of a custom reception.

Figure A-6: Method of error propagation (without fault processing) in a custom reception when a user-defined exception error is returned from a service component (when using a business process)

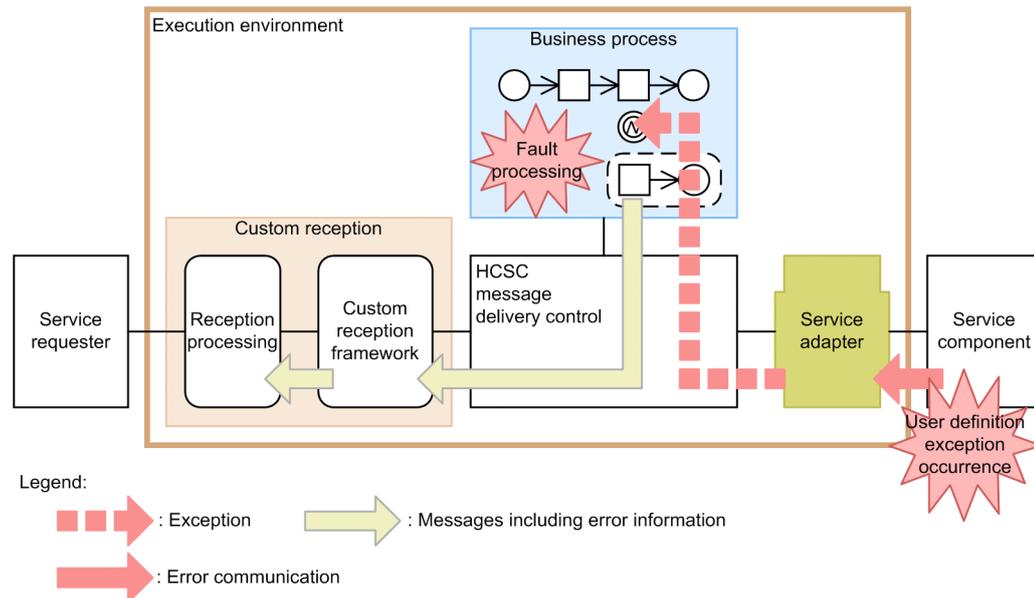


Exceptions that occur in the service component reach the custom reception framework unchanged as exceptions. The custom reception framework that catches the exception throws `CSCMsgServerException` unmodified to the reception process.

(b) When fault processing transforms the error information into a service message

The following figure shows how a user-defined exception error returned by a service component (when using a business process) propagates through the system (without fault processing) in the context of a custom reception.

Figure A-7: Method of error propagation (with fault processing) in a custom reception when a user-defined exception error is returned from a service component (when using a business process)



Exceptions that occur in the service component are forwarded as-is to the fault processing of the business process. The error information is then transformed into a response message by the fault processing of the business process, and then passed to subsequent processes in the form of a response message containing the error information. In this scenario, the response message is returned to the custom reception framework via HCSC message delivery control. The custom reception framework returns this response message to the reception process in the same way as a normal response message.

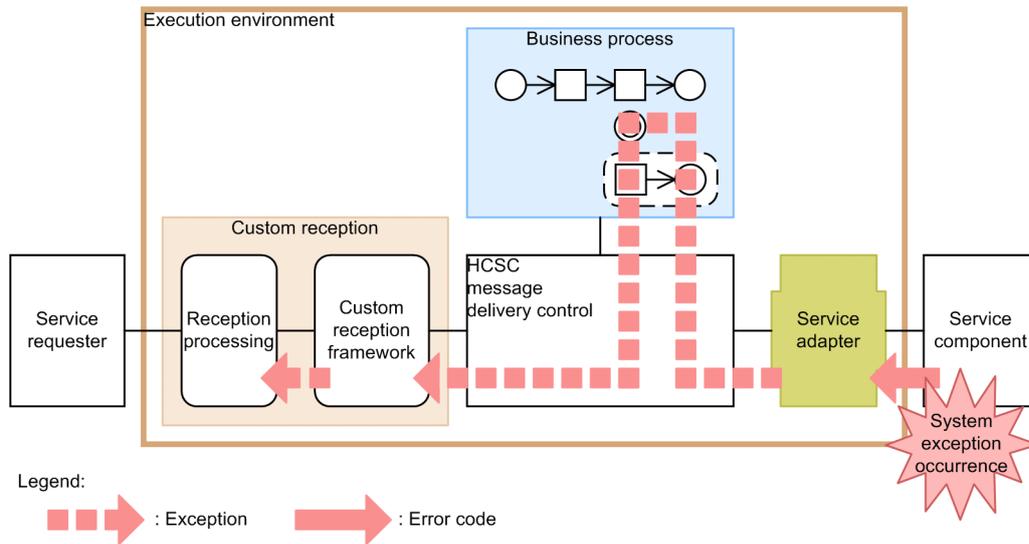
## (2) When an error other than a user-defined exception error is returned from a service component (when using a business process)

The following describes how an error other than a user-defined exception error propagates through the system when returned by a service component (when using a business process).

### (a) When fault processing does not transform the error information into a service message

The following figure shows how an error other than a user-defined exception error returned by a service component propagates through the system (without fault processing) in the context of a custom reception (when using a business process).

Figure A–8: Method of error propagation (without fault processing) in a custom reception when an error other than a user-defined exception error is returned from a service component (when using a business process)

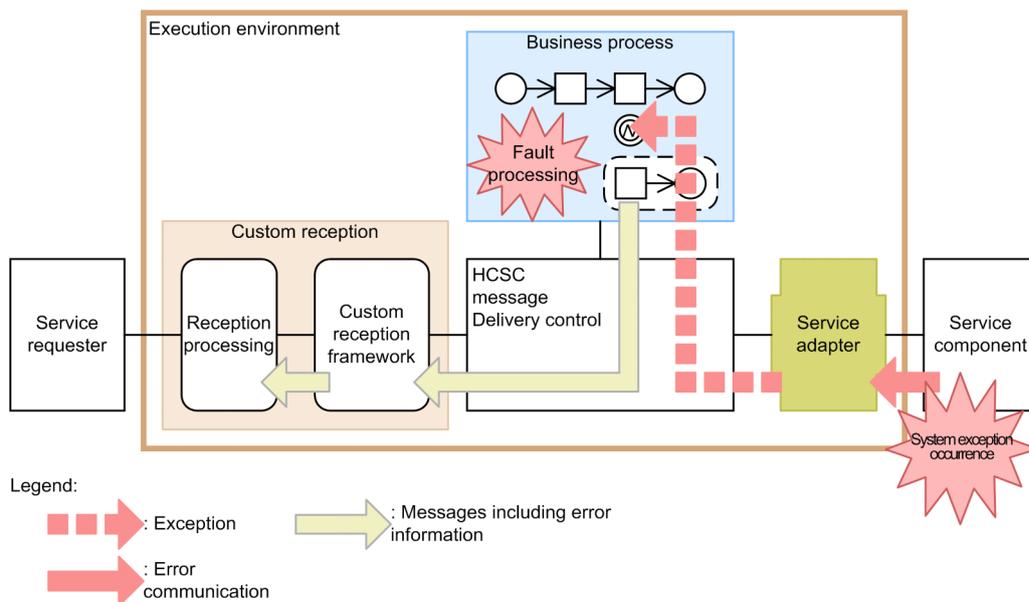


When an unexpected error occurs in a service component, the error reaches the custom reception framework as a `RuntimeException` error (system exception). The custom reception framework that catches the exception throws the caught `RuntimeException` unmodified to the reception process.

(b) When fault processing transforms the error information into a service message

The following figure shows how an error other than a user-defined exception error returned by a service component propagates through the system (with fault processing) in the context of a custom reception (when using a business process).

Figure A–9: Method of error propagation (with fault processing) in a custom reception when an error other than a user-defined exception error is returned from a service component (when using a business process)



Exceptions that occur in the service component are forwarded as-is to the fault processing of the business process. The error information is then transformed into a response message by the fault processing of the business process, and then passed to subsequent processes in the form of a response message containing the error information. In this scenario,

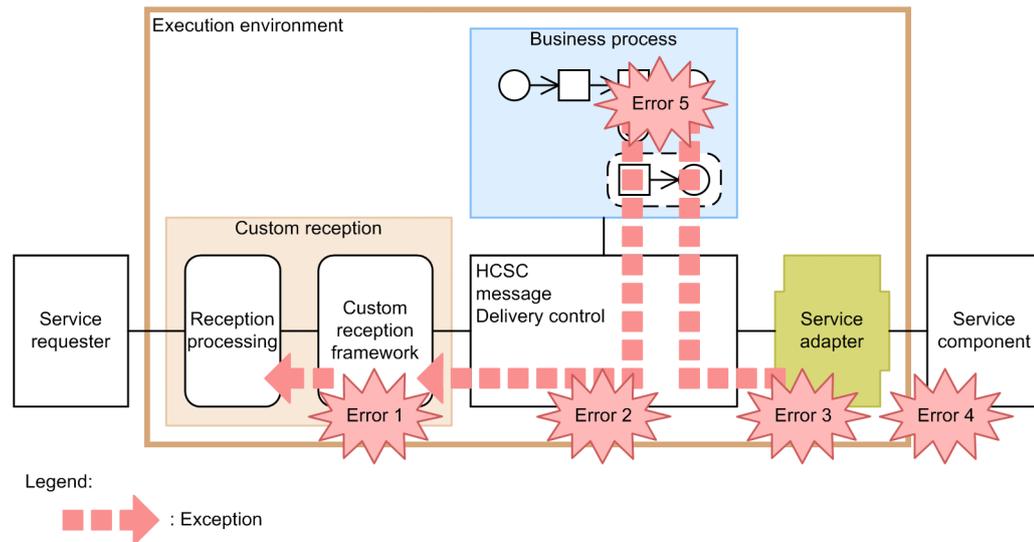
the response message is returned to the custom reception framework via HCSC message delivery control. The custom reception framework returns this response message to the reception process in the same way as a normal response message.

### (3) When an error is returned from the HCSC server (when using a business process)

The following describes how an error returned by the HCSC server propagates through the system (when using a business process).

The following figure shows how an error returned from the HCSC server propagates through the system in the context of a custom reception (when using a business process).

Figure A–10: Propagation of errors returned by HCSC server in a custom reception



The errors that might have occurred in the illustrated scenario are as follows:

- Error 1: Invalid request parameter
- Error 2: Location not found, service adapter is inactive
- Error 3: Data transformation failed
- Error 4: Invalid location, service component is inactive, communication error
- Error 5: Exception error in business process

If any of errors 1 to 5 in the figure is detected on the HCSC server, the custom reception framework throws the error information to the reception process as a `CSCMsgServerException` error.

## A.8 Acquiring the failure information (Custom reception)

If an error occurs during operation of a custom reception, the following information is output for troubleshooting purposes:

- Message log data
- Request trace information
- Performance analysis trace information
- User message trace information

This section describes how to acquire this log data and trace information.

### (1) Message log data (custom reception)

Message log data is output to the following logs:

- HCSC-Manager log
- Integrated message log
- J2EE server operation log

For details on how to acquire message log data and its output destination, see 7.4.1 *Message log* in the manual *Service Platform Setup and Operation Guide*. For further details, see *Chapter 2. List of Messages* in the manual *Service Platform Messages*.

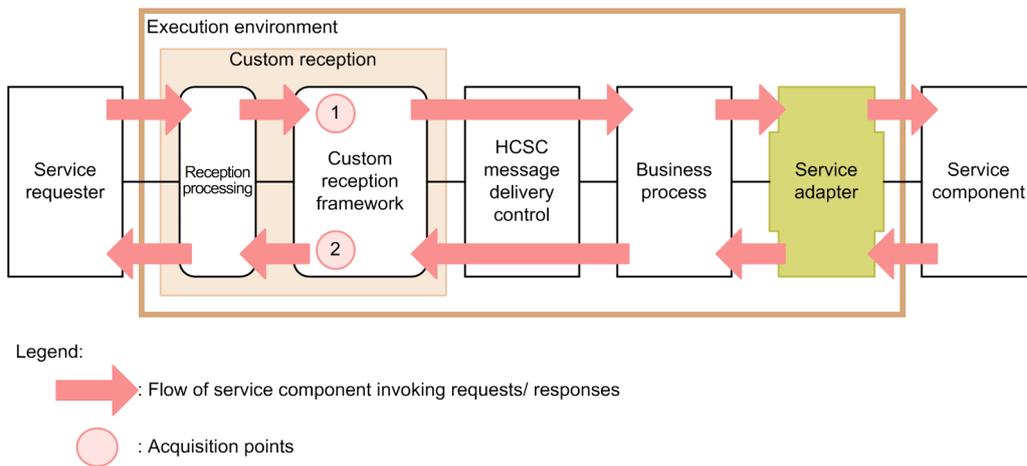
### (2) Request trace information (custom reception)

You can use request trace information to analyze the cause of an error in a request.

#### (a) Request trace information collection points

The following figure shows the collection points for request trace information:

Figure A-11: Request trace information collection points



The table below describes each trace information collection point and its detailed location. The number in the leftmost column of the table corresponds to the number in the figure.

Table A-5: Request trace collection points

Number in figure	Detailed location	Trace collection point
1	IN	Entry point of custom reception framework
2	OUT	Exit point of custom reception framework

#### (b) Format and contents of request trace output

##### Output format

For details on the output format of request trace information, see 7.4.2 *Request trace* in the manual *Service Platform Setup and Operation Guide*.

##### Output contents

The following table lists the items output in request trace information.

Table A-6: Items output in request trace information

Item	Description
No.	The output serial number of the trace record

Item		Description
Date		The date on which the trace record was acquired, in the format <i>yyyy/mm/dd</i>
Time		The time at which the trace record was acquired in local time, in the format <i>hh:mm:ss.sss</i>
Product ID		An ID that identifies the product. <ul style="list-style-type: none"> <li>• CSCMSG: Messaging infrastructure</li> </ul>
pid		An ID that identifies the process
tid		An ID that identifies the thread
ID		--
Common message ID		Information that identifies the request (parent ID)
Service requester ID		Information that identifies the request (child ID)
Collection point identification information	Collection point	Information about the collection point of the trace information. <ul style="list-style-type: none"> <li>• URCP: User-defined reception</li> </ul>
	Protocol type	Information about the collection point of the trace information (protocol type). <ul style="list-style-type: none"> <li>• CUSTM: Custom reception</li> </ul>
	Detailed location	Information about the collection point of the trace information (detailed location). <ul style="list-style-type: none"> <li>• IN: Reception</li> <li>• OUT: Response</li> </ul>
	Name	--
	Adapter type	--
Result		Outputs the response result type. This item is not output when the detailed location is IN. <ul style="list-style-type: none"> <li>• NORMAL: Normal termination</li> <li>• ERROR: Abnormal termination</li> </ul>
Additional information		Outputs the information appended to the response result. <ul style="list-style-type: none"> <li>• Reception name</li> <li>• Reception ID</li> <li>• Client correlation ID</li> <li>• Service name</li> <li>• Service operation name</li> </ul>
CRLF		The end of record code

Legend:

--: Nothing is output.

### (c) Method of request trace information collection and output destination of request traces

For details on how to collect request trace information and the output destination of request trace information, see *7.4.2 Request trace* in the manual *Service Platform Setup and Operation Guide*.

## (3) Performance analysis trace information (custom reception)

A performance analysis trace (PRF trace) is information that can be used to analyze the performance of the Service Platform system. A performance analysis trace file is a text file containing this information in CSV format.

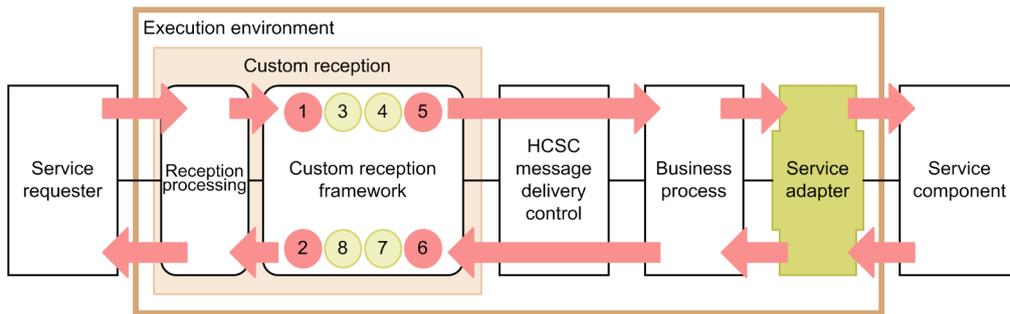
## A. Custom Reception

Performance analysis trace information is useful for analyzing performance bottlenecks across the entire system, including J2EE applications. You can use it to identify and troubleshoot performance issues. For details on the performance analysis trace functionality, see *Chapter 7. Performance Analysis by Using Trace based Performance Analysis* in the manual *Application Server Maintenance and Migration Guide*.

### (a) Trace information collection points

The following figure shows the collection points for performance analysis trace information.

Figure A–12: Collection points for performance analysis trace information



Legend:

-  : Flow of service component invoking requests/responses
-  : Indicates trace acquisition points. The performance analysis trace acquisition level is "Standard".
-  : Indicates trace acquisition points. The performance analysis trace acquisition level is "Detailed".

The table below shows the event IDs, trace information collection points, and performance analysis trace information collection levels. The number in the *Number in figure* column of the table corresponds to the number in the figure.

Table A–7: Collection points of performance analysis trace information

Event ID	Number in figure	Trace information collection point	Level
0x9860	1	Entry point of custom reception framework	A
0x9861	2	Exit point of custom reception framework	A
0x9862	3	Invocation of data transformation (request message)	B
0x9863	4	Reception of data transformation response (request message)	B
0x9864	5	Invocation of HCSC message delivery control	A
0x9865	6	Reception of HCSC message delivery control response	A
0x9866	7	Invocation of data transformation (response message)	B
0x9867	8	Reception of data transformation response (response message)	B

Legend:

- A: Indicates that the acquisition level is *Standard*.
- B: Indicates that the acquisition level is *Detailed*.

### (b) Format and contents of performance analysis trace information output

#### Output format

The output format of the performance analysis trace file is the same as that of a performance analysis trace for a J2EE server. For details on performance analysis trace files, see *7.3 Collecting the trace based performance analysis file by using Management Server* in the manual *Application Server Maintenance and Migration Guide*.

Output contents

The following table lists the information output in a performance analysis trace file.

Table A–8: Information output to a performance analysis trace file

Item	Description
Event ID	The event ID associated with the collection point
Return code	The collection point type. <ul style="list-style-type: none"> <li>• 0: Normal termination.</li> <li>• 1: Abnormal termination.</li> </ul>
Interface name	The class name is output. The package name is not shown.
Operation name	The method name
Optional information	The following optional information is output: <ul style="list-style-type: none"> <li>• Request ID information (parent ID)</li> <li>• Request ID information (child ID)</li> <li>• Reception name</li> <li>• Reception ID</li> <li>• Client correlation ID</li> <li>• Service name</li> <li>• Service operation name</li> <li>• Exception class name (abnormal termination only)</li> </ul>

(c) Method of performance analysis trace information collection and output destination of request trace information

The acquisition method and output destination for performance analysis trace files is common to the entire Service Platform. For details, see 7.3 *Collecting the trace based performance analysis file by using Management Server* in the manual *Application Server Maintenance and Migration Guide*.

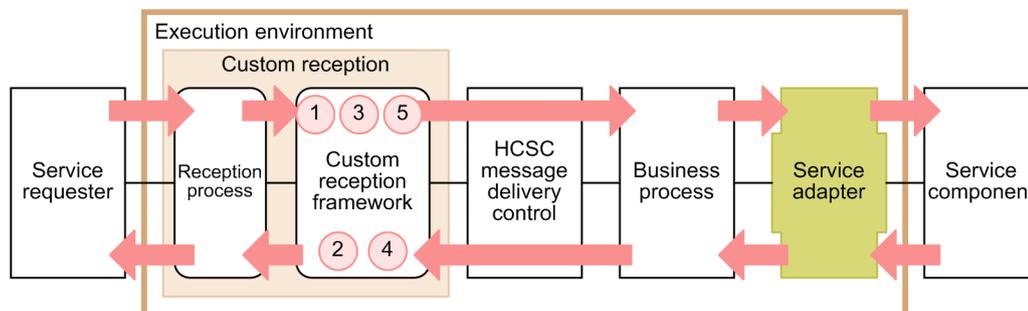
(4) User message trace (custom reception)

You can use user message trace information to check the status of a message.

(a) User message trace information collection points

The following figure shows the collection points for user message trace information.

Figure A–13: User message trace information collection points



Legend:

- : Flow of service component invocation request and response
- : Collection point

## A. Custom Reception

The table below shows the detailed location and collection points for user message trace information. The number in the leftmost column of the table corresponds to the number in the figure.

Table A–9: User message trace information collection points

Number in figure	Detailed location	Trace collection point
1	IN	Entry point of custom reception framework
2	OUT	Exit point of custom reception framework
3	RQA <sup>#1</sup>	Response to data transformation (request message)
4	RSB <sup>#1</sup>	Invocation of data transformation (response message)
5	ERR <sup>#2</sup>	Service invocation of custom reception framework

#1

Trace information is acquired only if the message is subject to data transformation.

#2

Request message trace information is acquired only if an error (system exception or fault) is detected in the service invocation.

### (b) Format and contents of performance user message trace information output

#### Output format

For details on the output format of user message trace information, see *7.4.4 User message trace* in the manual *Service Platform Setup and Operation Guide*.

#### Output contents

The following table lists the items output in user message trace information.

Table A–10: Items output in user message trace information (trace start)

Item	Description	
No.	The output serial number of the trace record	
Date	The date on which the trace record was acquired, in the format <i>yyyy/mm/dd</i>	
Time	The time at which the trace record was acquired in local time, in the format <i>hh:mm:ss.sss</i>	
Product ID	An ID that identifies the product. <ul style="list-style-type: none"> <li>CSCMSG: Messaging infrastructure</li> </ul>	
pid	An ID that identifies the process	
tid	An ID that identifies the thread	
ID	--	
Collection point identification information for message trace start	Message trace start	The string <code>telegramtrace started</code> which indicates the start of the user message trace is output.
	Collection point	Information about the collection point of the trace information. <ul style="list-style-type: none"> <li>URCP: User-defined reception</li> </ul>
	Protocol type	Information about the collection point of the trace information (protocol type). <ul style="list-style-type: none"> <li>CUSTOM: Custom reception</li> </ul>
	Detailed location	Information about the collection point of the trace information (detailed location). <ul style="list-style-type: none"> <li>IN: Reception</li> <li>OUT: Response</li> </ul>

Item		Description
Collection point identification information for message trace start	Detailed location	<ul style="list-style-type: none"> <li>• RQA: After data transformation of request in custom reception</li> <li>• RSB: Before data transformation of response in custom reception</li> </ul>
	Common message ID	Information that identifies the request (parent ID)
Additional information		The following additional information is output: <ul style="list-style-type: none"> <li>• Request ID information (child ID) (Service Request ID)</li> <li>• Reception name</li> <li>• Reception ID</li> <li>• Client correlation ID (Client ID)</li> <li>• Service name</li> <li>• Service operation name (OP)</li> <li>• PRF root application information (RootApInfo)</li> </ul>
CRLF		The end of record code

Legend:

--: Nothing is output.

Table A–11: Items output in user message trace information (trace data)

Item		Description
No.		The output serial number of the trace record
Date		The date on which the trace record was acquired, in the format <i>yyyy/mm/dd</i>
Time		The time at which the trace record was acquired in local time, in the format <i>hh:mm:ss.sss</i>
Product ID		An ID that identifies the product. <ul style="list-style-type: none"> <li>• CSCMSG: Messaging infrastructure</li> </ul>
pid		An ID that identifies the process
tid		An ID that identifies the thread
ID		--
Message trace data	Output position	The offset (hexadecimal format) from the start of the user message is output.
	Data (hexadecimal format)	The content of the user message (hexadecimal format) is output.
	Data (ASCII format)	The content of the user message (ASCII format) is output. ASCII characters are output for characters in the range from 0x20 to 0x7E. Periods (.) are output for values outside this range.
CRLF		The end of record code

Legend:

--: Nothing is output.

Table A–12: Items output in user message trace information (trace end)

Item		Description
No.		The output serial number of the trace record
Date		The date on which the trace record was acquired, in the format <i>yyyy/mm/dd</i>

## A. Custom Reception

Item		Description
Time		The time at which the trace record was acquired in local time, in the format <i>hh:mm:ss.sss</i>
Product ID		An ID that identifies the product. <ul style="list-style-type: none"> <li>CSCMSG: Messaging infrastructure</li> </ul>
pid		An ID that identifies the process
tid		An ID that identifies the thread
ID		--
Collection point identification information for message trace end	Message trace end	The string <code>telegramtrace_sended</code> which indicates the end of the user message trace is output.
	Collection point	Information about the collection point of the trace information. <ul style="list-style-type: none"> <li>URCP: User-defined reception</li> </ul>
	Protocol type	Information about the collection point of the trace information (protocol type). <ul style="list-style-type: none"> <li>CUSTM: Custom reception</li> </ul>
	Detailed location	Information about the collection point of the trace information (detailed location). <ul style="list-style-type: none"> <li>IN: Reception</li> <li>OUT: Response</li> <li>RQA: After data transformation of request in custom reception</li> <li>RSB: Before data transformation of response in custom reception</li> </ul>
	Common message ID	Information that identifies the request (parent ID)
Additional information		The length of the user message (decimal format) is output. If there is no user message, <code>null</code> is output.
CRLF		The end of record code

Legend:  
--: Nothing is output.

### (c) Method of user message trace information collection and output destination of user message trace information

For details on the collection method and output destination of user message trace information, see 7.4.4 *User message trace* in the manual *Service Platform Setup and Operation Guide*.

## B. Custom Adapter Development Framework

This appendix describes the development of General custom adapters using the custom adapter development framework.

### B.1 APIs of the custom adapter development framework

This section describes the custom adapter development framework APIs provided by Service Platform.

The interfaces and exception classes defined in the custom adapter development framework are stored in the following location:

*Service-Platform-installation-directory\CSC\lib\csc\_adapter.jar*

When compiling a General custom adapter, include this JAR file in the classpath.

#### (1) Protocol converter interfaces

##### (a) CSCMsgCustomProtocolConverter interface

###### Description

This interface is used to implement the protocol converter.

The package name of the `CSCMsgCustomProtocolConverter` interface is `jp.co.Hitachi.soft.csc.msg.adapter.custom`.

###### Format

```
public interface CSCMsgCustomProtocolConverter
{
    public void start()
        throws CSCMsgCustomAdapterException;
    public void stop();
    public void setCustomAdapterContext (CSCMsgCustomAdapterContext
adapterContext);
    public void invoke (CSCMsgRequestMessage requestMessage,
        CSCMsgResponseMessage responseMessage)
        throws CSCMsgCustomAdapterException;
}
```

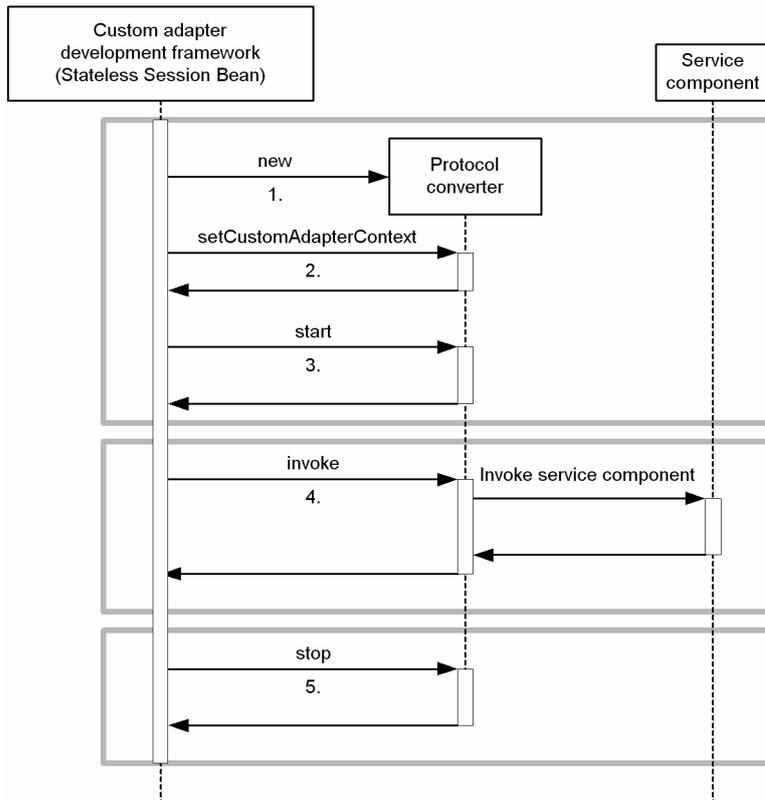
###### Methods

The following table lists the methods provided by the `CSCMsgCustomProtocolConverter` interface:

Method name	Description
<code>start</code> method	This method is invoked when the General custom adapter is started. This method implements start processing for the protocol converter. This method is invoked only once.
<code>stop</code> method	This method is invoked when the General custom adapter ends. This method implements end processing for the protocol converter. This method is invoked only once.
<code>setCustomAdapterContext</code> method	This method is invoked when the adapter context is set. This method describes the processing for storing the allocated adapter context in an instance field of the protocol converter. This method is invoked only once.
<code>invoke</code> method	This method is invoked when invoking a service component via the General custom adapter. This method describes the processing for invoking a service component. This method is invoked each time there is a request to invoke the service adapter.

The following figure shows the order in which each of these methods is invoked from the custom adapter development framework.

Figure B–1: Order of method invocation from the custom adapter development framework



1. Constructor

When the General custom adapter starts, the protocol converter is instantiated by the default constructor invoked from the custom adapter development framework.

2. setCustomAdapterContext method

The `setCustomAdapterContext` method is invoked from the custom adapter development framework immediately after the protocol converter class is instantiated. This method is invoked only once when the General custom adapter starts. Ensure that you maintain the adapter context received as an argument in an instance field of the protocol converter.

3. start method

The `start` method is invoked from the custom adapter development framework immediately after the `setServiceAdapterInfo` method. This method is invoked only once when the General custom adapter starts.

4. invoke method

The `invoke` method is invoked from the custom adapter development framework during processing to invoke a service component. This method is used to perform protocol conversion and service invocation processing.

5. stop method

The `stop` method is invoked from the custom adapter development framework immediately before the General custom adapter is deleted from the HCSC server. This method performs end processing for the General custom adapter. This method is invoked only once when the General custom adapter ends.

Notes

- Public default constructor  
The custom adapter development framework uses the default constructor to instantiate the classes that implement this interface. For this reason, you must implement a public default constructor in classes that implement this interface.
- Multi-threading

Because the protocol converter is shared by multiple General custom adapter instances, you must implement a thread safe class.

- Throwing exceptions

The methods of the `CSCMsgCustomProtocolConverter` class can only throw specific exceptions. Do not have the methods throw `java.lang.RuntimeException` or classes that derive from it. If a method throws such an exception, normal operation is not guaranteed.

- start method

Description

This method is invoked when the General custom adapter is started.

We recommend that you use this method to secure the resources that will be used throughout the life cycle of the General custom adapter.

Format

```
public void start()
    throws CSCMsgCustomAdapterException;
```

Parameters

None.

Exceptions

`jp.co.Hitachi.soft.csc.msg.adapter.custom.CSCMsgCustomAdapterException`  
Generated when execution of the `start` method fails.

Return values

None.

- stop method

Description

This method is invoked when the General custom adapter ends. It describes the end processing for the General custom adapter.

Format

```
public void stop();
```

Parameters

None.

Exceptions

None.

Return values

None.

- setCustomAdapterContext method

Description

This method is invoked when the adapter context is set.

The `start` method is always invoked after this method.

The adapter context interface obtained by this method must be maintained throughout the lifecycle of the General custom adapter.

Format

```
public void setCustomAdapterContext(CSCMsgCustomAdapterContext
    adapterContext);
```

Parameters

`adapterContext`

Passes the `CSCMsgCustomAdapterContext` reference.

Exceptions

None.

Return values

None.

- invoke method

Description

This method is invoked when invoking a service component via the General custom adapter. This method describes the processing for invoking a service component.

The `invoke` method invokes the service component based on the request message stored in the `requestMessage` parameter.

If the communication model is synchronous, the response message is stored in the `responseMessage` parameter.

If the communication model is asynchronous, `null` is passed for `responseMessage`.

Format

```
public void invoke (CSCMsgRequestMessage requestMessage,
                  CSCMsgResponseMessage responseMessage)
    throws CSCMsgCustomAdapterException;
```

Parameters

- `requestMessage`  
The `CSCMsgRequestMessage` reference that stores the request message is passed.
- `responseMessage`  
If the communication model is synchronous, the `CSCMsgResponseMessage` reference that stores the response message is passed. If the communication model is asynchronous, `null` is passed.

Exceptions

`jp.co.Hitachi.soft.csc.msg.adapter.custom.CSCMsgCustomAdapterException`  
Generated when execution of the `invoke` method fails.

Return values

None.

## (2) Adapter context interface

### (a) CSCMsgCustomAdapterContext interface

Description

This interface is used by the protocol converter to acquire adapter information. The package name of the `CSCMsgCustomAdapterContext` interface is `jp.co.Hitachi.soft.csc.msg.adapter.custom`.

The adapter context interface is input as an argument of the `setCustomAdapterContext` method immediately before processing to start the General custom adapter. The protocol converter maintains this argument as an instance field and acquires the required information.

Format

```
public interface CSCMsgCustomAdapterContext
{
    public String getAdapterName();
    public Properties getProperties();
    public byte[] getResourceAsBytes(String fileName)
        throws CSCMsgResourceAccessException;
    public java.io.InputStream getResourceAsStream(String fileName)
        throws CSCMsgResourceAccessException;
}
```

Methods

The following table lists the methods provided by the `CSCMsgCustomAdapterContext` interface:

Method name	Description
<code>getAdapterName</code> method	Acquires the name of the General custom adapter.

Method name	Description
<code>getProperties</code> method	Acquires the contents of the custom adapter property file as <code>Properties</code> .
<code>getResourceAsBytes</code> method	Acquires the contents of the resource file in binary format.
<code>getResourceAsStream</code> method	Acquires the stream for accessing the resource file.

- `getAdapterName` method

Description

This method acquires the name of the General custom adapter.

Format

```
public String getAdapterName();
```

Parameters

None.

Exceptions

None.

Return values

Returns the name of the General custom adapter.

- `getProperties` method

Description

This method acquires the contents of the custom adapter property file as `Properties`.

Format

```
public Properties getProperties();
```

Parameters

None.

Exceptions

None.

Return values

Returns the contents of the property file.

- `getResourceAsBytes` method

Description

This method acquires the contents of the resource file in binary format.

Format

```
public byte[] getResourceAsBytes(String fileName)
    throws CSCMsgResourceAccessException;
```

Parameters

`fileName`

Specifies the file name of the resource to be accessed.

Exceptions

`CSCMsgResourceAccessException`

The resource specified by the `fileName` parameter is not found, or `java.io.IOException` was generated.

Return values

Returns the contents of the resource file in binary format.

- `getResourceAsStream` method

Description

This method acquires the stream for accessing the resource file.

**Format**

```
public java.io.InputStream getResourceAsStream(String fileName)
    throws CSCMsgResourceAccessException;
```

**Parameters**

`fileName`

Specifies the file name of the resource to be accessed.

**Exceptions**

`CSCMsgResourceAccessException`

The resource specified by the `fileName` parameter is not found, or `java.io.IOException` was generated.

**Return values**

Returns the contents of the resource file in stream format.

**Notes**

Do not allocate a stream returned by this method to a static field. Doing so might cause a memory leak.

### (3) Message interfaces

The following table lists the interfaces used to exchange messages between the HCSC server and protocol converter:

Table B–1: List of APIs used to work with messages

Interface name	Description
<code>CSCMsgMessageConstant</code> interface	Defines the constants required to work with messages.
<code>CSCMsgRequestMessage</code> interface	Provides the methods that acquire the information (such as operation names and messages) required to invoke service components.
<code>CSCMsgResponseMessage</code> interface	Provides the methods that store the results of service component invocation (messages or fault information).

#### (a) `CSCMsgMessageConstant` interface

**Description**

This interface defines the constants required by the protocol converter to work with messages. The package name of the `CSCMsgMessageConstant` interface is `jp.co.Hitachi.soft.csc.msg.adapter.custom`.

The protocol converter uses these constants to check the type of message stored as the request message or response message.

**Format**

```
public interface CSCMsgMessageConstant
{
    public static final int MESSAGE_TYPE_NONE;
    public static final int MESSAGE_TYPE_XML;
    public static final int MESSAGE_TYPE_BINARY;
    public static final int MESSAGE_TYPE_ANY;
}
```

**Member attributes**

`MESSAGE_TYPE_NONE:`

Indicates that no message is stored.

`MESSAGE_TYPE_XML:`

Indicates that the message is in XML format (DOM format).

`MESSAGE_TYPE_BINARY:`

Indicates that the message is in binary format.

`MESSAGE_TYPE_ANY:`

Indicates that the message is in an arbitrary format.

## (b) CSCMsgRequestMessage interface

## Description

This interface provides the methods that acquire the information required to invoke service components (such as operation names and messages).

The package name of the CSCMsgRequestMessage interface is

`jp.co.Hitachi.soft.csc.msg.adapter.custom`.

## Format

```
public interface CSCMsgRequestMessage
{
    public byte[] getBytes()
        throws CSCMsgIllegalMessageTypeException,
            CSCMsgInvalidMessageException;
    public Map getMessageContext();
    public int getMessageType();
    public String getOperationName();
    public org.w3c.dom.Document getXMLDocument()
        throws CSCMsgIllegalMessageTypeException,
            CSCMsgInvalidMessageException;
    public byte[] getXMLBytes()
        throws CSCMsgIllegalMessageTypeException,
            CSCMsgInvalidMessageException;
}
```

## Methods

The following table lists the methods provided by the CSCMsgRequestMessage interface:

Method name	Description
getBytes method	Acquires messages in binary format.
getMessageContext method	When a General custom adapter is invoked via a business process, this method returns information related to the business process.
getMessageType method	Acquires the type of message stored as the request message.
getOperationName method	Acquires the name of the operation invoked when invoking the service component.
getXMLDocument method	Acquires messages in XML format.
getXMLBytes method	Acquires XML messages as a byte array.

- getBytes method

## Description

This method acquires messages in binary format.

## Format

```
public byte[] getBytes()
    throws CSCMsgIllegalMessageTypeException,
        CSCMsgInvalidMessageException;
```

## Parameters

None.

## Exceptions

- `jp.co.Hitachi.soft.csc.msg.adapter.custom.CSCMsgIllegalMessageTypeException`

The message format is not binary. Alternatively, a message is already stored. You cannot use this method to acquire messages that are in XML format.

- `jp.co.Hitachi.soft.csc.msg.adapter.custom.CSCMsgInvalidMessageException`

The stored message is invalid.

## Return values

This method returns the response as a byte array if the specified message type is binary (MESSAGE\_TYPE\_BINARY).

If no message is stored (MESSAGE\_TYPE\_NONE), the method returns a 0 byte array (an array of length 0).

Notes

- You can modify the binary messages (byte arrays) acquired by this method without causing any issues.
- A copy of the object is created when you execute this method. Executing the method again will cause the object to be re-acquired in its original form.

- **getMessageContext method**

Description

When a General custom adapter is invoked via a business process, this method returns information related to the business process.

It returns information in the `java.util.Map` format.

The Map returned by the `getMessageContext` method is the object returned by the `Collections#unmodifiableMap()` method.

It cannot be modified.

Because the Map is not a synchronized object, it needs to be exclusively locked when shared among multiple threads.

When the `getMessageContext` method is invoked without going through a business process, the Map it returns contains the information from the HCSC server name to the J2EE server name in the table below.

Table B–2: Information acquired by `getMessageContext` method

Information type	Map contents			Description
	Element		Value	
	Element name	Element type	Value type	
Business process name	CSCMsgCustomAdapterConstant.ContextType.BP_DEFINITION_NAME	enum	java.lang.String	The name of the business process from which the General custom adapter is invoked <sup>#1</sup>
Business process version	CSCMsgCustomAdapterConstant.ContextType.BP_VERSION	enum	java.lang.String	The version of the business process from which the General custom adapter is invoked <sup>#1</sup>
Invoke service activity name	CSCMsgCustomAdapterConstant.ContextType.BP_INVOKEACTIVITY_NAME	enum	java.lang.String	The name of the invoke service activity of the business process from which the General custom adapter is invoked <sup>#1</sup>
Service name of General custom adapter	CSCMsgCustomAdapterConstant.ContextType.BP_INVOKEACTIVITY_CUSTOMADAPTERNAME	enum	java.lang.String	The service name of the General custom adapter invoked from the invoke service activity <sup>#1</sup>
Operation name	CSCMsgCustomAdapterConstant.ContextType.BP_INVOKEACTIVITY_OPERATIONNAME	enum	java.lang.String	Operation name of invoke service activity <sup>#1, #2</sup>
Process instance ID	CSCMsgCustomAdapterConstant.ContextType.BP_PROCESSINSTANCEID	enum	java.lang.String	Process instance ID <sup>#1</sup>
HCSC server name	CSCMsgCustomAdapterConstant.ContextType.SERVERNAME_HCSC	enum	java.lang.String	HCSC server name
Cluster name	CSCMsgCustomAdapterConstant.ContextType.SERVERNAME_CLUSTER	enum	java.lang.String	Cluster name

Information type	Map contents			Description
	Element		Value	
	Element name	Element type	Value type	
J2EE server name	CSCMsgCustomAdapterConstant.ConstantType.SERVERNAME_J2EE	enum	java.lang.String	J2EE server name

#1

This element returns information about the business process that directly invokes the General custom adapter.

If a business process (BP1) invokes another business process (BP2) which then invokes the General custom adapter, the method will not acquire information about BP1.

#2

If multiple operations are defined in the General custom adapter invoked from the invoke service activity, the method only acquires the name of the operation invoked in the request processing.

Format

```
public Map getMessageContext();
```

Parameters

None.

Exceptions

None.

Return values

```
java.util.Map
```

- getMessageType method

Description

This method acquires the type of message stored as the request message.

Format

```
public int getMessageType();
```

Parameters

None.

Exceptions

None.

Return values

This method returns the type of message stored as the request message. For details on the return values, see *B.1(3)(a) CSCMsgMessageConstant interface*. Note that if you created a General custom adapter for an arbitrary message format, MESSAGE\_TYPE\_ANY is returned regardless of the message type.

- getOperationName method

Description

This method acquires the name of the operation invoked when invoking the service component.

Format

```
public String getOperationName();
```

Parameters

None.

Exceptions

None.

Return values

This method returns the name of the invoked operation.

- **getXMLDocument** method

**Description**

Acquires messages in XML format.

**Format**

```
public org.w3c.dom.Document getXMLDocument()  
    throws CSCMsgIllegalMessageTypeException,  
           CSCMsgInvalidMessageException;
```

**Parameters**

None.

**Exceptions**

- `jp.co.Hitachi.soft.csc.msg.adapter.custom.CSCMsgIllegalMessageTypeException`  
The message format is not XML. Alternatively, a message is already stored. You cannot use this method to acquire messages that are in binary format.
- `jp.co.Hitachi.soft.csc.msg.adapter.custom.CSCMsgInvalidMessageException`  
The stored message is invalid.

**Return values**

This method returns a response as an object in DOM format if the assigned message type is XML (MESSAGE\_TYPE\_XML).

If no message is stored (MESSAGE\_TYPE\_NONE), the method returns null.

**Notes**

- You can modify the XML messages (DOM) acquired by this method without causing any issues.
- A copy of the object is created when you execute this method. Executing the method again will cause the original object to be re-acquired.
- If you created a General custom adapter for an arbitrary message format, executing this method generates an error (KDEC03016-E) and messages cannot be acquired. In this scenario, use the `getBytes` method instead.

- **getXMLBytes** method

**Description**

This method acquires XML messages as a byte array.

When using StAX or other tools to read messages, you can reduce memory usage by using this method.

**Format**

```
public byte[] getXMLBytes()  
    throws CSCMsgIllegalMessageTypeException,  
           CSCMsgInvalidMessageException;
```

**Parameters**

None.

**Exceptions**

- `jp.co.Hitachi.soft.csc.msg.adapter.custom.CSCMsgIllegalMessageTypeException`  
The message format is not XML.  
You cannot use this method to acquire messages that are in binary format.
- `jp.co.Hitachi.soft.csc.msg.adapter.custom.CSCMsgInvalidMessageException`  
The stored message is invalid.

**Return values**

This method returns a response as an object in byte array format if the assigned message is in XML format (MESSAGE\_TYPE\_XML).

If no message is stored (MESSAGE\_TYPE\_NONE), the method returns null.

## Notes

- You can modify the XML messages (byte arrays) acquired by this method without causing any issues.
- A copy of the object is created when you execute this method. Executing the method again will cause the original object to be re-acquired.
- Note the following when using the StAX XML processor to read XML messages:
  - An exception occurs if a message in XML version 1.1 is set.
  - An exception occurs if you use ISO-10646-UCS-4 encoding.
  - The inclusion of element names starting with colons (:) in the XML document does not cause an exception.
  - The presence of a single CDATA end token (]]>) in the character data of the XML document does not cause an exception.
  - For empty elements, only the opening and closing tags are output.

Example: <sample></sample>

## (c) CSCMsgResponseMessage interface

## Description

This interface provides the methods that store the results of service component invocation (messages or fault information) in response messages.

The package name of the CSCMsgResponseMessage interface is  
jp.co.Hitachi.soft.csc.msg.adapter.custom.

## Format

```
public interface CSCMsgResponseMessage
{
    public int getMessageType();
    public void setBytes(byte[] message)
        throws CSCMsgIllegalMessageTypeException,
        CSCMsgMultipleMessageInsertionException,
        CSCMsgInvalidMessageException;
    public void setFault(String faultCode,
        String faultString,
        String faultActor,
        byte[] faultDetail)
        throws CSCMsgMultipleMessageInsertionException;
    public void setFault(String faultCode,
        String faultString,
        String faultActor,
        org.w3c.dom.Document faultDetail)
        throws CSCMsgMultipleMessageInsertionException,
        CSCMsgInvalidMessageException;
    public void setXMLDocument(org.w3c.dom.Document dom)
        throws CSCMsgIllegalMessageTypeException,
        CSCMsgMultipleMessageInsertionException,
        CSCMsgInvalidMessageException;
    public void setXMLBytes(byte[] byte)
        throws CSCMsgIllegalMessageTypeException,
        CSCMsgMultipleMessageInsertionException,
        CSCMsgInvalidMessageException;
}
```

## Methods

The following table lists the methods provided by the CSCMsgResponseMessage interface:

Method name	Description
getMessageType method	Acquires the type of message to be stored in the response message.
setBytes method	Stores binary format messages in response messages.
setFault method (format 1)	Stores fault information generated at service component invocation in response messages.
setFault method (format 2)	Stores fault information generated at service component invocation in response messages.
setXMLDocument method	Stores XML format messages in response messages.

Method name	Description
setXMLBytes method	Stores XML messages in response messages in binary array format.

- getMessageType method

## Description

This method acquires the type of message to be stored in the response message.

## Format

```
public int getMessageType();
```

## Parameters

None.

## Exceptions

None.

## Return values

This method returns the type of message to be stored in the response message. Note that if you create a General custom adapter for messages in an arbitrary format, MESSAGE\_TYPE\_ANY is returned regardless of the message type.

- MESSAGE\_TYPE\_NONE:  
No message is to be stored.
- MESSAGE\_TYPE\_XML:  
The message is to be stored in XML format.
- MESSAGE\_TYPE\_BINARY:  
The message is to be stored in binary format.

- setBytes method

## Description

This method stores a binary message in the response message.

## Format

```
public void setBytes(byte[] message)
    throws CSCMsgIllegalMessageTypeException,
           CSCMsgMultipleMessageInsertionException,
           CSCMsgInvalidMessageException;
```

## Parameters

message

Specifies a binary format message.

If the method stores a binary message, the message type is set to binary format (MESSAGE\_TYPE\_BINARY). If the method stores a byte array with a length of 0 or null, the message type is set to the type indicating that no message is stored (MESSAGE\_TYPE\_NONE).

## Exceptions

- jp.co.Hitachi.soft.csc.msg.adapter.custom.CSCMsgIllegalMessageTypeException  
The format of the set message is not binary. Alternatively, a message is already stored. If the message stored in the request message is in XML format, you cannot use binary as the format of the response message.
- jp.co.Hitachi.soft.csc.msg.adapter.custom.CSCMsgMultipleMessageInsertionException  
A message or fault information is already set.
- jp.co.Hitachi.soft.csc.msg.adapter.custom.CSCMsgInvalidMessageException  
The specified binary format contains an error.

## Return values

None.

## Notes

If you created a General custom adapter for messages in an arbitrary format, use this method regardless of the message format. This method sets the type of the stored message to any type (`MESSAGE_TYPE_ANY`). The message type is set to any type (`MESSAGE_TYPE_ANY`) even if the message is a byte array of 0 bytes or null.

- `setFault` method (format 1)

## Description

This method stores fault information generated when a service component is invoked.

Use this method to store the contents of `faultDetail` in binary format.

Because there will be no response message, the message type indicating that no message is stored (`MESSAGE_TYPE_NONE`) is set.

## Format

```
public void setFault(String faultCode,
                    String faultString,
                    String faultActor,
                    byte[] faultDetail)
    throws CSCMsgMultipleMessageInsertionException;
```

## Parameters

- `faultCode`  
Specifies `faultCode` information.  
If null is specified, `Server.ServiceExecutionError` is stored.
- `faultString`  
Specifies `faultString` information.  
If null is specified, `Service Execution Error at CustomAdapter` is stored.
- `faultActor`  
Specifies `faultActor` information.  
If null is specified, the service ID set in `HCSC-Definer` is stored.
- `faultDetail`  
Specifies detail information in XML format as a byte array.  
If a byte array with a length of 0 is specified, the byte array of 0 bytes is stored as is. If null is specified, null is stored as is.

## Exceptions

```
jp.co.Hitachi.soft.csc.msg.adapter.custom.CSCMsgMultipleMessageInsertionException
```

A response message or fault information is already stored.

## Return values

None.

## Notes

- `faultDetail` is the XML document converted to binary format. It must begin with an XML declaration. If this declaration is missing, an error will not occur but the method might not work correctly.
- If you intend to convert the XML document to a byte array, use UTF-8 as the encoding. If you specify any other encoding, the method might not work correctly.

- `setFault` method (format 2)

## Description

This method stores fault information generated when a service component is invoked.

Use this method to store the contents of `faultDetail` in XML format (DOM format).

Because there will be no response message, the message type indicating that no message is stored (`MESSAGE_TYPE_NONE`) is set.

Format

```
public void setFault(String faultCode,
                    String faultString,
                    String faultActor,
                    org.w3c.dom.Document faultDetail)
    throws CSCMsgMultipleMessageInsertionException,
           CSCMsgInvalidMessageException;
```

Parameters

- `faultCode`  
Specifies `faultCode` information.  
If `null` is specified, `Server.ServiceExecutionError` is stored.
- `faultString`  
Specifies `faultString` information.  
If `null` is specified, `Service Execution Error at CustomAdapter` is stored.
- `faultActor`  
Specifies `faultActor` information.  
If `null` is specified, the service ID set in `HCSC-Definer` is stored.
- `faultDetail`  
Specifies detail information in DOM format.  
If `null` is specified, `null` is stored as-is.

Exceptions

- `jp.co.Hitachi.soft.csc.msg.adapter.custom.CSCMsgMultipleMessageInsertionException`  
A response message or fault information is already stored.
- `jp.co.Hitachi.soft.csc.msg.adapter.custom.CSCMsgInvalidMessageException`  
The specified DOM format contains an error.

Return values

None.

• `setXMLDocument` method

Description

This method stores an XML format message (DOM format) in the response message.

Format

```
public void setXMLDocument(org.w3c.dom.Document dom)
    throws CSCMsgIllegalMessageTypeException,
           CSCMsgMultipleMessageInsertionException,
           CSCMsgInvalidMessageException;
```

Parameters

`dom`

Specifies a message in DOM format.

If the stored message is in XML format, the message type is set to XML format (`MESSAGE_TYPE_XML`). If `null` is stored, the message type is set to the type that indicates no message is stored (`MESSAGE_TYPE_NONE`).

Exceptions

- `jp.co.Hitachi.soft.csc.msg.adapter.custom.CSCMsgIllegalMessageTypeException`  
The format of the set message is not XML. Alternatively, a message is already stored. If the message stored in the request message is an XML message, you cannot change the format of the response message to binary.
- `jp.co.Hitachi.soft.csc.msg.adapter.custom.CSCMsgMultipleMessageInsertionException`  
A message or fault information is already set.

- `jp.co.Hitachi.soft.csc.msg.adapter.custom.CSCMsgInvalidMessageException`  
The specified DOM format contains an error.

## Return values

None.

## Notes

If you created a General custom adapter for an arbitrary message format, executing this method generates an error (KDEC03016-E) and messages cannot be stored. In this scenario, use the `setBytes` method instead.

- `setXMLBytes` method

## Description

This method stores XML format messages (binary array) in response messages.

If the General custom adapter handles XML format messages as byte arrays, the need to convert the messages to DOM format is eliminated. In this case, you can reduce memory usage by using the `setXMLBytes` method.

## Format

```
public void setXMLBytes(byte[] byte)
    throws CSCMsgIllegalMessageTypeException,
           CSCMsgMultipleMessageInsertionException,
           CSCMsgInvalidMessageException;
```

## Parameters

byte

Specifies a byte array format message.

Note that if `byte` is not null, the message type is set to XML (`MESSAGE_TYPE_XML`). If `byte` is null, the message type is set to the type indicating that no message is stored (`MESSAGE_TYPE_NONE`).

## Exceptions

- `jp.co.Hitachi.soft.csc.msg.adapter.custom.CSCMsgIllegalMessageTypeException`  
The format of the request message is not XML. If the request message type is binary, you cannot change the type of the response message to XML.
- `jp.co.Hitachi.soft.csc.msg.adapter.custom.CSCMsgMultipleMessageInsertionException`  
A message or fault information is already set.
- `jp.co.Hitachi.soft.csc.msg.adapter.custom.CSCMsgInvalidMessageException`  
The specified XML format contains an error.

## Return values

None.

## Notes

- Note the following when using StAX to create messages:
  - An exception occurs if a message in XML version 1.1 is set.
  - An exception occurs if you use ISO-10646-UCS-4 encoding.
  - The inclusion of element names starting with colons (:) in the XML document does not cause an exception.
  - The presence of a single CDATA end token (]]>) in the character data of the XML document does not cause an exception.
  - For empty elements, only the opening and closing tags are output.  
Example: `<sample></sample>`

(d) `CSCMsgCustomAdapterConstant` class

## Description

This interface defines the Map keys that can store the business process information acquired by the `getMessageContext` method of the `CSCMsgRequestMessage` interface.

The package name of the CSCMsgCustomAdapterConstant interface is `jp.co.Hitachi.soft.csc.msg.adapter.custom`.

**Format**

```
public interface CSCMsgCustomAdapterConstant {
    public enum ContextType {
        BP_DEFINITION_NAME,
        BP_VERSION,
        BP_INVOKEACTIVITY_NAME,
        BP_INVOKEACTIVITY_CUSTOMADAPTERNAME,
        BP_INVOKEACTIVITY_OPERATIONNAME,
        BP_PROCESSINSTANCEID,
        SERVERNAME_HCSC,
        SERVERNAME_CLUSTER,
        SERVERNAME_J2EE;
    }
}
```

**Enum constants**

- BP\_DEFINITION\_NAME:**  
The name of the business process from which the General custom adapter was invoked
- BP\_VERSION:**  
The version of the business process from which the General custom adapter was invoked
- BP\_INVOKEACTIVITY\_NAME:**  
The name of the invoke service activity of the business process from which the General custom adapter was invoked
- BP\_INVOKEACTIVITY\_CUSTOMADAPTERNAME:**  
The service name of the General custom adapter invoked from the invoke service activity
- BP\_INVOKEACTIVITY\_OPERATIONNAME:**  
The operation name of the invoke service activity
- BP\_PROCESSINSTANCEID:**  
The process instance ID
- SERVERNAME\_HCSC:**  
The HCSC server name
- SERVERNAME\_CLUSTER:**  
The cluster name
- SERVERNAME\_J2EE:**  
The J2EE server name

**(4) Exception classes**

The following table lists the exception classes generated during protocol converter development:

**Table B-3: List of protocol converter exception classes**

Class name	Description
CSCMsgCustomAdapterException class	The exception thrown when an exception occurs during processing to initialize the General custom adapter or invoke the service component
CSCMsgIllegalMessageTypeException class	The exception thrown when a method is invoked whose message format differs from the message type being acquired or stored
CSCMsgMultipleMessageInsertionException class	The exception thrown when a message or fault information is already stored.  You cannot set a message or fault information if this information has already been stored.

Class name	Description
CSCMsgInvalidMessageException class	The exception thrown when the type (format) of a stored request message is invalid. It is also thrown when an invalid message type is specified for the response message.
CSCMsgResourceAccessException class	The exception thrown when an error occurs during resource access. This exception occurs in the following situations: <ul style="list-style-type: none"> <li>• The resource you attempted to acquire does not exist</li> <li>• An I/O exception (<code>java.io.IOException</code>) occurred</li> </ul>

## B.2 Definition files of the custom adapter development framework

This appendix describes the contents of the files defined when developing General custom adapters in the custom adapter development framework.

### (1) Custom adapter development framework action definition file

The custom adapter development framework action definition file is a properties file in XML format that can be read by the `loadFromXML` method of the `java.util.Properties` class. It must have the file name `framework_properties.xml`.

#### Function

The purpose of this file is to specify the class name (including the package name) of the protocol converter loaded by the custom adapter development framework. The class name of the protocol converter is specified in the `entry` tag whose key attribute is `classname`.

#### File location

When you create a JAR file of the protocol converter, this file is created in the root directory of the JAR file. For details on how to create a JAR file of the protocol converter, see 3.3.14(5) *Creating a JAR file*.

#### Notes

- If the file does not exist or its file format cannot be read by the `loadFromXML` method of the `java.util.Properties` class, the General custom adapter cannot start.
- Make sure that the same key attribute is not specified in more than one `entry` element.

#### Specification example

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE properties SYSTEM "http://java.sun.com/dtd/properties.dtd">
<properties>
<entry key="classname">protocolconverter.MyProtocolConverter</entry>
</properties>
```

### (2) Custom adapter property file

The custom adapter property file is a properties file in XML format that can be read by the `loadFromXML` method of the `java.util.Properties` class. It must have the file name `customadapter_properties.xml`.

#### Function

This file sets the properties used by the protocol converter.

When the General custom adapter starts, the properties are read from the custom adapter development framework. The properties read in this process can be acquired as `Property` instances by the `getProperty` method from the adapter context.

#### File location

Register the file as a self-defined file in the General custom adapter when defining the General custom adapter. For details on how to define a General custom adapter, see 3.3.14 *Defining custom adapters*.

Notes

Even if the file exists, the General custom adapter cannot start if the file is in a format that cannot be read by the `loadFromXML` method of the `java.util.Properties`.

Specification example

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE properties SYSTEM "http://java.sun.com/dtd/properties.dtd">
<properties>
<entry key="init">1</entry>
<entry key="min">0</entry>
<entry key="max">10</entry>
</properties>
```

### (3) HITACHI Application Integrated Property File for custom adapter

A HITACHI Application Integrated Property File for a custom adapter is a file in XML format. It is a variant of a HITACHI Application Integrated Property File, used in the following situations:

- When acquiring the attributes of applications, EJB-JARs, and Session Beans as a batch
- When editing the attributes of applications, EJB-JARs, and Session Beans as a batch

A HITACHI Application Integrated Property File for custom adapters must have the file name `cscadapter_property.xml`.

For details on HITACHI Application Integrated Property files, see the following sections:

- *9.2 Property settings using the HITACHI Application Integrated Property File* in the manual *Application Server Application Setup Guide*.
- *3.1 HITACHI Application Integrated Property file* in the manual *Application Server Application and Resource Definition Reference Guide*.

Function

By using a HITACHI Application Integrated Property file, you can set the parameters of the General custom adapter according to its execution environment.

File location

Register the file as a self-defined file in the General custom adapter when defining the General custom adapter. For details on how to define a General custom adapter, see *3.3.14 Defining custom adapters*.

File contents

The table below shows the template of the HITACHI Application Integrated Property File for custom adapters. You can modify or add items in the variable parts of this template.

You can also add and edit the following items as elements under `<hitachi-application-all-property>/<ejb-jar>/<hitachi-session-bean-property>`:

- `<ejb-ref>` (defines reference information for an Enterprise Bean with a remote interface)
- `<ejb-local-ref>` (sets EJB reference information)
- `<env-entry>` (defines an environment entry)
- `<resource-ref>` (defines a resource reference)
- `<resource-env-ref>` (defines resource environment variables)
- `<linked-queue>` (queue name)

Table B-4: Template of HITACHI Application Integrated Property File for custom adapter

Tag	Description	Changeable
<code>&lt;?xml version="1.0" encoding="MS932"?&gt;</code>	XML declaration	Yes
<code>&lt;!DOCTYPE hitachi-application-all-property PUBLIC "-//Hitachi, Ltd.//DTD Application All Property 7.1//EN"</code>	DOCTYPE declaration	--

Tag	Description	Changeable
'http://localhost/hitachi-application-all-property_7_1.dtd'	DOCTYPE declaration	--
<hitachi-application-all-property>	Root tag	--
<hitachi-application-property>	Opening tag for the definition of application-related information	--
<description></description>	Description of the application	Yes
<icon>	Opening tag for the definition of a J2EE application icon displayed in the Deploy tool	--
<small-icon></small-icon>	File name of small icon (16 x 16)	Yes
<large-icon></large-icon>	File name of large icon (32 x 32)	Yes
</icon>	Closing tag for the definition of a J2EE application icon displayed in the Deploy tool	--
<lookup-name>CTMADP</lookup-name>	Name used when looking up EJBs from the client	Yes <sup>#1</sup>
<security-prop>	Opening tag for the definition of the Enterprise Bean security management method	--
<security-method>no_security_for_methods_without_roles</security-method>	Security management method	--
<default-security-role></default-security-role>	Default security role when map_methods_without_roles is specified in security-method	--
</security-prop>	Closing tag for the definition of the Enterprise Bean security management method	--
<start-order>120</start-order>	The order in which J2EE applications are started and stopped	Auto #2
<scheduling-unit>Application</scheduling-unit>	Specifies the queue scheduling unit	--
<scheduling>	Opening tag for the definition relating to CTM integration	--
<queue-name>CTMADP</queue-name>	Name of the queue used for scheduling	Yes <sup>#1</sup>
<parallel-count>1</parallel-count>	The number of threads prepared by CTM to invoke an application	--
<queue-length></queue-length>	Length of the queue used for scheduling	--
</scheduling>	Closing tag for the definition relating to CTM integration	--
<managed-by-ctm>>false</managed-by-ctm>	Specifies whether to perform CTM integration	--

Tag	Description	Changeable
<code>&lt;method-observation-recovery-mode&gt; &lt;/method-observation-recovery-mode&gt;</code>	Specifies the recovery mode of the J2EE application time monitoring functionality	--
<code>&lt;/hitachi-application-property&gt;</code>	Closing tag for the definition of application-related information	--
<code>&lt;ejb-jar&gt;</code>	Opening tag for the definition of EJB-related information	--
<code>&lt;hitachi-ejb-jar-property&gt;</code>	Opening tag for the definition of information relating to EJB-JAR attributes	--
<code>&lt;description xml:lang="en"&gt;&lt;/description&gt;</code>	Description of EJB-JAR	Yes
<code>&lt;display-name xml:lang="en"&gt;CSCMsgServiceAdapter&lt;/display-name&gt;</code>	Display name of EJB-JAR	--
<code>&lt;icon xml:lang="en"&gt;</code>	Opening tag for the definition of an EJB-JAR icon displayed in the GUI tool	--
<code>&lt;small-icon&gt;&lt;/small-icon&gt;</code>	File name of small icon (16 x 16)	Yes
<code>&lt;large-icon&gt;&lt;/large-icon&gt;</code>	File name of large icon (32 x 32)	Yes
<code>&lt;/icon&gt;</code>	Closing tag for the definition of an EJB-JAR icon displayed in the GUI tool	--
<code>&lt;ejb-client-jar&gt;&lt;/ejb-client-jar&gt;</code>	Name of client JAR file	--
<code>&lt;/hitachi-ejb-jar-property&gt;</code>	Closing tag for the definition of information relating to EJB-JAR attributes	--
<code>&lt;hitachi-session-bean-property&gt;</code>	Opening tag for the definition of Session Bean attributes	--
<code>&lt;description xml:lang="en"&gt;&lt;/description&gt;</code>	Description of Session Bean	Yes
<code>&lt;display-name xml:lang="en"&gt;CSCMsgServiceAdapterEJB&lt;/display-name&gt;</code>	Display name of Session Bean	--
<code>&lt;icon xml:lang="en"&gt;</code>	Opening tag for the definition of a Session Bean icon displayed in the GUI tool	--
<code>&lt;small-icon&gt;&lt;/small-icon&gt;</code>	File name of small icon (16 x 16)	Yes
<code>&lt;large-icon&gt;&lt;/large-icon&gt;</code>	File name of large icon (32 x 32)	Yes
<code>&lt;/icon&gt;</code>	Closing tag for the definition of a Session Bean icon displayed in the GUI tool	--
<code>&lt;session-type&gt;Stateless&lt;/session-type&gt;</code>	Session Bean type	--
<code>&lt;transaction-type&gt;Container&lt;/transaction-type&gt;</code>	Transaction management type	--
<code>&lt;env-entry&gt;</code>	Opening tag for the environment entry definition	Auto #3
<code>&lt;description xml:lang="en"&gt;&lt;/description&gt;</code>	Description of the environment entry	
<code>&lt;env-entry-name&gt;CscServerName&lt;/env-entry-name&gt;</code>	Environment entry name	

Tag	Description	Changeable
<code>&lt;env-entry-type&gt;java.lang.String&lt;/env-entry-type&gt;</code>	Data type of the environment entry	Auto #3
<code>&lt;env-entry-value&gt;HCSC&lt;/env-entry-value&gt;</code>	Value of the environment entry	
<code>&lt;/env-entry&gt;</code>	Closing tag for the environment entry definition	
<code>&lt;env-entry&gt;</code>	Opening tag for the environment entry definition	
<code>&lt;description xml:lang="en"&gt;&lt;/description&gt;</code>	Description of the environment entry	
<code>&lt;env-entry-name&gt;CscClusterName&lt;/env-entry-name&gt;</code>	Environment entry name	
<code>&lt;env-entry-type&gt;java.lang.String&lt;/env-entry-type&gt;</code>	Data type of the environment entry	
<code>&lt;env-entry-value&gt;Cluster&lt;/env-entry-value&gt;</code>	Value of the environment entry	
<code>&lt;/env-entry&gt;</code>	Closing tag for the environment entry definition	
<code>&lt;env-entry&gt;</code>	Opening tag for the environment entry definition	
<code>&lt;description xml:lang="en"&gt;&lt;/description&gt;</code>	Description of the environment entry	
<code>&lt;env-entry-name&gt;AdapterName&lt;/env-entry-name&gt;</code>	Environment entry name	
<code>&lt;env-entry-type&gt;java.lang.String&lt;/env-entry-type&gt;</code>	Data type of the environment entry	
<code>&lt;env-entry-value&gt;CTMADP&lt;/env-entry-value&gt;</code>	Value of the environment entry	
<code>&lt;/env-entry&gt;</code>	Closing tag for the environment entry definition	
<code>&lt;container-transaction&gt;</code>	Opening tag for the definition of the container transaction	--
<code>&lt;description xml:lang="en"&gt;&lt;/description&gt;</code>	Description of the container transaction	--
<code>&lt;method&gt;</code>	Opening tag for the method definition	--
<code>&lt;description xml:lang="en"&gt;&lt;/description&gt;</code>	Description of the method	--
<code>&lt;method-interfaces&gt;&lt;/method-interfaces&gt;</code>	Interface type to which the method belongs	--
<code>&lt;method-name&gt;*&lt;/method-name&gt;</code>	Method name	--
<code>&lt;/method&gt;</code>	Closing tag for the method definition	--
<code>&lt;trans-attribute&gt;Required&lt;/trans-attribute&gt;</code>	Transaction attribute allocated to the method	--
<code>&lt;/container-transaction&gt;</code>	Closing tag for the definition of the container transaction	--
<code>&lt;session-runtime&gt;</code>	Opening tag for the runtime definition	--
<code>&lt;lookup-name&gt;CSCMsgServiceAdapterEJB&lt;/lookup-name&gt;</code>	Name used when looking up EJB from the client	--
<code>&lt;optional-name&gt;CTMADP&lt;/optional-name&gt;</code>	Optional name for the remote interface	Auto #3

Tag	Description	Changeable
<maximum-sessions>0</maximum-sessions>	Maximum number of sessions	Yes
<stateless>	Opening tag for the stateless definition	--
<pooled-instance>	Opening tag for the definition relating to pooled instances	--
<minimum>1</minimum>	Minimum number of pooled instances	Yes
<maximum>0</maximum>	Maximum number of pooled instances	Yes
</pooled-instance>	Closing tag for the definition relating to pooled instances	--
<instance-timeout>0</instance-timeout>	Timeout period for instance acquisition	Yes
</stateless>	Closing tag for the stateless definition	--
<enable-scheduling>>false</enable-scheduling>	Specifies whether to enable scheduling for the bean (whether to use it as a scheduler Gate)	--
<pass-by-reference>>true</pass-by-reference>	Specifies whether to use pass by reference when invoking the bean	Auto #3
<scheduling>	Opening tag for the definition of information relating to CTM integration	--
<queue-name>CSCMsgServiceAdapterEJB</queue-name>	Name of the queue used for scheduling	--
<parallel-count>1</parallel-count>	The number of threads prepared by CTM to invoke an application	--
<queue-length></queue-length>	Length of the queue used for scheduling	--
</scheduling>	Closing tag for the definition of information relating to CTM integration	--
<front-ejb>>false</front-ejb>	Specifies whether the EJB is a front EJB (an EJB the client invokes directly)	--
</session-runtime>	Closing tag for the runtime definition	--
<start-order>10</start-order>	Specifies the order in which J2EE applications are started and stopped	--
</hitachi-session-bean-property>	Closing tag for the definition of Session Bean attributes	--
</ejb-jar>	Closing tag for the definition of EJB-related information	--
</hitachi-application-all-property>	Root tag	--

Legend:

Yes: Can be modified.

Auto: Modified automatically.

--: Cannot be modified.

#1

Specify the service ID.

#2

Set automatically when the General custom adapter is deployed.

#3

Added or overwritten automatically when the General custom adapter is deployed.

#### Notes

- Do not modify any parts of the template except those listed as modifiable in Table B-4 Template of HITACHI Application Integrated Property File for custom adapter.
- You are unable to deploy General custom adapters if the file is not correctly formatted as a HITACHI Application Integrated Property File.

### (4) Custom adapter definition file

The function of the custom adapter definition file and the properties you can set in the file are described below.

#### Function

Sets information about a General custom adapter.

#### File location

```
Service-Platform-installation-directory\CSC\samples\customadapter
\csccustomadapter.properties
```

#### Specifiable properties

```
custom-adapter.dt-skip={ true | false }
```

Specifies whether to skip conversion of the message structure during data transformation. If you omit this key or its value, structure conversion is not skipped.

- `true`  
Skips structure conversion.
- `false`  
Does not skip structure conversion.

When specifying a value, you must specify `true` or `false`. If you specify a value other than `true` or `false`, an error occurs during command execution.

## B.3 Sample program of the custom adapter development framework

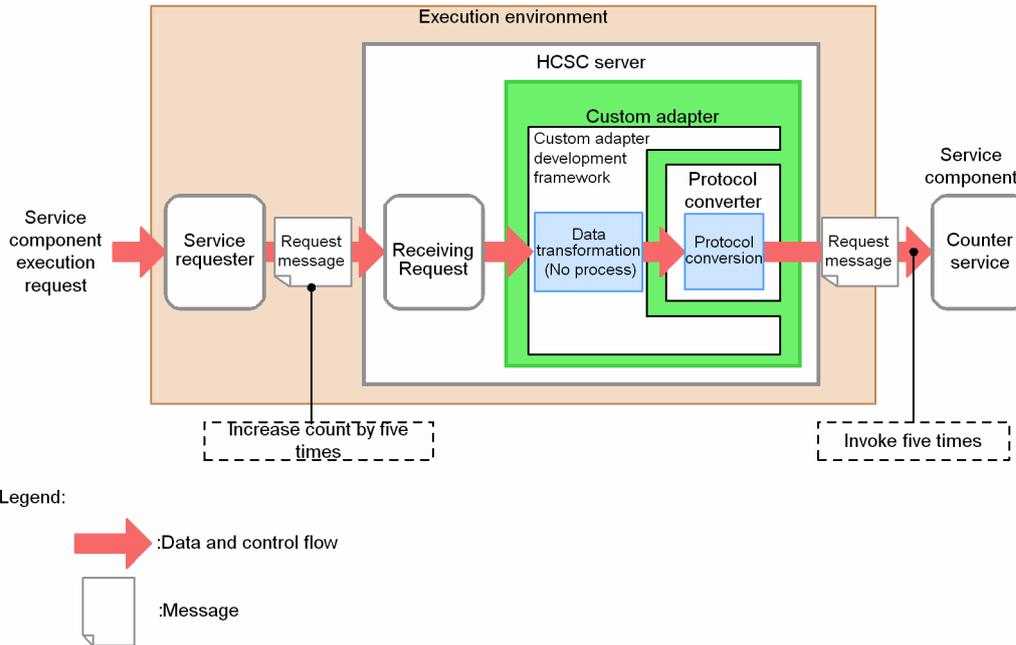
This section describes the sample program of the custom adapter that uses the custom adapter development framework.

### (1) Overview of the sample program

With the sample program, create a custom adapter that receives the count increment request (`increment`) or the count decrement request (`decrement`) and invokes the counter service.

The following figure shows an overview of the custom adapter that invokes the counter service:

Figure B-2: Overview of the custom adapter that invokes the counter service



The counter service and the protocol converter of the custom adapter are as follows:

(a) Counter service

The counter service maintains the count.

The counter service of the sample program is a normal Java class but is viewed as a service. Therefore, the storage location of the counter service is the same JAR as the protocol converter.

The counter service provides the following three services:

- Count increment service (*increment*)  
This service increases the count by one.
- Count decrement service (*decrement*)  
This service decreases the count by one. However, an exception occurs when the count is attempted to be brought down below 0.
- Count value reference service (*getCount*)  
This service returns the current count value.

(b) Protocol converter

The protocol converter invokes the counter service.

The protocol converter performs the following two operations:

- Count increment (*increment operation*)  
Invoke the count increment service of the counter service multiple times as specified in the request message. During asynchronous communication, a binary message is received as the request message. The increment amount can be specified in the request message, and count increment of the counter service is executed multiple times as the absolute value of the specified increment amount. The increment amount is specified as a binary number in one-byte encoding. For details about the request message format, see *Appendix B.3(2)(a) Request message format of increment*.
- Count decrement (*decrement operation*)  
Invoke the count decrement service of the counter service and reduce the count only by the specified number. During synchronous communication, an XML message is received as the request message, and an XML message is returned as the response message.

The increment amount can be specified in the request message, and the count is reduced only as much as the integer value of the specified decrement amount. The integer value of the decrement amount is provided as a character string.

The response message of the counter decrement request includes the count value after the count decrement.

When an exception occurs during the execution of the count decrement service of the counter service, a fault will occur. Also, if the format of the decrement amount of the counter provided with the request message is such that the decrement amount cannot be processed by the `parseInt` method of the `Integer` class, an exception will be thrown.

For details about request message formats, see *Appendix B.3(2)(b) Request message format for decrement*, and for details about the response message format, see *Appendix B.3(2)(c) Response message format for decrement*.

## (2) Message format

The message formats used in the sample program are as follows:

### (a) Request message format of increment

Create a binary format definition file that takes a single binary integer with one-byte encoding. Specify the file name as `BinaryRequest.fdx`.

For details about how to create a binary format definition file, see *4.4 Creating Message Formats (Binary Format Definition File)* in the manual *Service Platform Basic Development Guide*.

### (b) Request message format for decrement

Create the request message format (XML format definition file) for decrement as an XML document type by using a text editor. Specify the file name as `XMLRequest.xsd`.

For details about how to create an XML format definition file, see *4.3 Creating Message Formats (XML Format Definition File)* in the manual *Service Platform Basic Development Guide*.

The contents of the XML format definition file to be created are as follows:

---

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema" targetNamespace="http://
www.example.org/RequestXML" xmlns:tns="http://www.example.org/RequestXML">
  <element name="request" type="string"></element>
</schema>
```

---

### (c) Response message format for decrement

Create the response message format (XML format definition file) for decrement as an XML document type using a text editor. Specify the file name as `XMLResponse.xsd`.

For details about how to create an XML format definition file, see *4.3 Creating Message Formats (XML Format Definition File)* in the manual *Service Platform Basic Development Guide*.

The contents of the XML format definition file to be created are as follows:

---

```
<?xml version="1.0" encoding="UTF-8"?>
<schema xmlns="http://www.w3.org/2001/XMLSchema" targetNamespace="http://
www.example.org/RequestXML" xmlns:tns="http://www.example.org/RequestXML">
  <element name="response" type="string"></element>
</schema>
```

---

## (3) Sample codes

The counter service used in the sample program, and the sample code of the protocol converter are as follows:

### (a) Counter service

A sample code for counter service is described here. The class name is `Counter` and the package name is `service`.

---

```
/* All Rights Reserved. Copyright (C) 2008, Hitachi, Ltd. */
package service;
```

---

---

```

/**
 * Count value fluctuation and count value acquisition class (viewed as a service)
 * The count value is always 0 or more
 */
public class Counter {
/**
 * Count value (always 0 or more)
 */
protected int count;
/**
 * Constructor
 * Initial value of @param init count
 * @throws IllegalArgumentException An invalid value below 0 was specified.
 * @return void
 */
public Counter(int init) {

    if (init < 0) {
        throw new IllegalArgumentException();
    }
    this.count = init;
}
/**
 * Increment the count by 1
 * @return void
 */
public void increment() {
    ++this.count;
}

/**
 * Decrement the count by 1
 * @throws IllegalStateException Invoked when the count was 0 or less.
 * @return void
 */
public void decrement() {
    if (this.count <= 0) {
        throw new IllegalStateException();
    }
    --this.count;
}
/**
 * Acquire the count value
 * @return int Count value
 */
public int getCount() {
    return this.count;
}
}

```

---

**(b) Protocol converter**

A sample code for protocol converter is described below. The class name is `MyProtocolConverter` and the package name is `protocolconverter`.

---

```

/* All Rights Reserved. Copyright (C) 2008, Hitachi, Ltd. */
package protocolconverter;

import java.util.Properties;

import javax.xml.parsers.*;
import org.w3c.dom.*;
import jp.co.Hitachi.soft.csc.msg.adapter.custom.*;

import service.Counter;
/**
 * Implementation class of custom protocol converter interface
 */
public class MyProtocolConverter implements CSCMsgCustomProtocolConverter {
/**
 * Adapter context
 */
private CSCMsgCustomAdapterContext adapterContext;
/**
 * Counter class (viewed as a service)
 */
private Counter counter;
/**
 * Used to create a DOM. XML message
 */

```

---

---

```

private DocumentBuilderFactory docBuilderFactory;
private DocumentBuilder docBuilder;
/**
 * public default constructor (mandatory)
 */
public MyProtocolConverter() { }
/**
 * Invoked when the custom adapter is started (before start is invoked).
 * Acquire the adapter context.
 * @param adapterContext Adapter context
 */
public void setCustomAdapterContext(
    CSCMsgCustomAdapterContext adapterContext) {
    this.adapterContext = adapterContext;
}
/**
 * Invoked when the custom adapter is started (after setCustomAdapterContext is
invoked).
 * Implement the initialization processing.
 * @throws CSCMsgCustomAdapterException An exception occurred in the custom adapter.
 */
public void start() throws CSCMsgCustomAdapterException {
    try {
        log("start");

        Properties properties = this.adapterContext.getProperties();
        String init = properties.getProperty("init");
        if (init == null) {
            // The init value must definitely be specified
            throw new CSCMsgCustomAdapterException("init");
        }

        // Secure the resources (Resources used throughout the life cycle of the adapter)
        this.counter = new Counter(Integer.parseInt(init));

        docBuilderFactory = DocumentBuilderFactory.newInstance();
        //docBuilderFactory.setNamespaceAware(true);
        docBuilder = docBuilderFactory.newDocumentBuilder();
    } catch (CSCMsgCustomAdapterException e) {
        // Output the log required for failure analysis on the custom adapter side
        log(e);
        throw e;
    } catch (Exception e) { // Also catch RuntimeException
        // Output the log required for failure analysis on the custom adapter side
        log(e);
        // Throw again after wrapping with CSCMsgCustomAdapterException
        throw new CSCMsgCustomAdapterException("start failed", e);
    }
}
/**
 * Implement the service component invocation processing.
 * @param request Request message
 * @param response Response message
 * @throws CSCMsgCustomAdapterException An exception occurred in the custom adapter.
 */
public void invoke(CSCMsgRequestMessage request,
    CSCMsgResponseMessage response)
    throws CSCMsgCustomAdapterException {

    try {
        log("invoke [" + request.getOperationName() + "]");

        // Distribute by operation name
        if (request.getOperationName().equals("increment")) {
            // Count increment processing
            execIncrement(request, response);
        } else if (request.getOperationName().equals("decrement")) {
            // Count decrement processing
            try {
                execDecrement(request, response);
            } catch (IllegalStateException e) { // Failed to decrement count
                // Log output
                log(e);
                // Fault setting
                response.setFault("invoke error", "decrement failed",
                    this.adapterContext.getAdapterName(), (Document)null);
            }
        } else {
            throw new CSCMsgCustomAdapterException("unknown operation");
        }
        // Output current count
        log("Count:" + counter.getCount());
    } catch (CSCMsgCustomAdapterException e) {

```

---

---

```

// Output the log required for failure analysis on the custom adapter side
log(e);
throw e;
} catch (Exception e) { // Also catch RuntimeException
// Output the log required for failure analysis on the custom adapter side
log(e);
// Throw again after wrapping with CSCMsgCustomAdapterException
throw new CSCMsgCustomAdapterException("invoke failed", e);
}
}
/**
 * Count increment processing
 * @param request Request message
 * @param response Response message
 * @throws CSCMsgCustomAdapterException An exception occurred in the custom adapter.
 * @throws CSCMsgIllegalMessageTypeException The message is not in the binary format.
 * @throws CSCMsgInvalidMessageException An invalid message is specified.
 */
private void execIncrement(
CSCMsgRequestMessage request,
CSCMsgResponseMessage response)
throws CSCMsgCustomAdapterException,
CSCMsgIllegalMessageTypeException,
CSCMsgInvalidMessageException {
// Asynchronous operation: increment

if (response != null) {
// The communication model is synchronous
throw new CSCMsgCustomAdapterException("Invalid communication model");
}

if (request.getMessageType() != CSCMsgMessageConstant.MESSAGE_TYPE_BINARY) {
// The message is not in the binary format
throw new CSCMsgCustomAdapterException("Invalid message format (Request)");
}

byte[] data = request.getBytes();
if (data.length != 1) {
throw new CSCMsgCustomAdapterException("Invalid data");
}

int loop = Math.abs(data[0]);
for (int i = 0; i < loop; ++i) {
counter.increment();
}
}
/**
 * Count decrement processing
 * @param request Request message
 * @param response Response message
 * @throws CSCMsgCustomAdapterException An exception occurred in the custom adapter.
 * @throws CSCMsgIllegalMessageTypeException The message is not in the binary format.
 * @throws CSCMsgInvalidMessageException An invalid message is specified.
 * @throws CSCMsgMultipleMessageInsertionException A message or fault information is
already saved.
 */
private void execDecrement(
CSCMsgRequestMessage request,
CSCMsgResponseMessage response)
throws CSCMsgCustomAdapterException,
CSCMsgIllegalMessageTypeException,
CSCMsgInvalidMessageException,
CSCMsgMultipleMessageInsertionException {
// Synchronous operation: decrement

if (response == null) {
// The communication model is asynchronous
throw new CSCMsgCustomAdapterException("Invalid communication model");
}

if (request.getMessageType() != CSCMsgMessageConstant.MESSAGE_TYPE_XML) {
// The message is not in the XML format
throw new CSCMsgCustomAdapterException("Invalid message format (Request)");
}

if (response.getMessageType() != CSCMsgMessageConstant.MESSAGE_TYPE_XML) {
// The message is not in the XML format
throw new CSCMsgCustomAdapterException("Invalid message format (Response)");
}
// Request message analysis processing
NodeList nodeList = request.getXMLDocument().getElementsByTagName("request");
if (nodeList.getLength() != 1) {
// The root node cannot be found

```

---

---

```

throw new CSCMsgCustomAdapterException("Invalid request message");
}
Text text = (Text)nodeList.item(0).getChildNodes().item(0);

int loop = Math.abs(Integer.parseInt(text.getNodeValue()));
for (int i = 0; i < loop; ++i) {
this.counter.decrement();
}
// Build response message
Document responseDoc = docBuilder.newDocument();
Element rootElement = responseDoc.createElement("response");
responseDoc.appendChild(rootElement);
Text countTextNode =
responseDoc.createTextNode(Integer.toString(counter.getCount()));
rootElement.appendChild(countTextNode);

// Save response message
response.setXMLDocument(responseDoc);
}
/**
 * Perform termination processing and resource release
 */
public void stop() {
log("stop");

// Release resources
this.counter = null;
this.adapterContext = null;
this.docBuilderFactory = null;
this.docBuilder = null;
}
/**
 * Log output
 * @param message Message to be output
 */
public void log(String message) {
// The output results in standard output are output in the user output log
System.out.printf("[%s] %s\n",
this.adapterContext.getAdapterName(), message);
System.out.flush();
}
/**
 * Log output
 * @param message Exception to be output
 */
public void log(Exception e) {
// The output results in standard error output are output in the user error log
e.printStackTrace();
System.err.flush();
}
}

```

---

#### (4) Property files

The following are the property files used in the sample program:

##### (a) Custom adapter development framework operation definition file

with the custom adapter development framework operation definition file, specify the class name of the protocol converter along with the package name.

---

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE properties SYSTEM "http://java.sun.com/dtd/properties.dtd">
<properties>
<entry key="classname">protocolconverter.MyProtocolConverter</entry>
</properties>

```

---

##### (b) Custom adapter property file

With the custom adapter property file, set up properties to be used in the protocol converter.

A property file in which the initial value of the counter service count is set to 1 is as follows:

---

```

<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE properties SYSTEM "http://java.sun.com/dtd/properties.dtd">
<properties>

```

---

---

```
<entry key="init">1</entry>
</properties>
```

---

## (5) Protocol converter JAR file

Create a JAR file that includes a protocol converter class file and a custom adapter development framework operation definition file.

For creating a protocol converter JAR file, use the `jar` command and archive the protocol converter class file, the class files that uses the protocol converter class file, and the custom adapter development framework operation definition file into a JAR file.

For details about the directory configuration of JAR files when using the `jar` command to archive, see 3.3.14(5) *Creating a JAR file*.

### (a) Prerequisites

Arrange the created file in the current directory and sub directory as follows:

---

```
framework_properties.xml
protocolconverter/CustomProtocolConverter.class
service/Counter.class
```

---

### (b) Creating the file

Archive the created JAR file with the file name `MyProtocolConverter.jar`.

If you execute the following command, the JAR file will be created:

---

```
jar cf ..\MyProtocolConverter.jar .\
```

---

## (6) EAR file

Create the EAR file required for registering a new custom adapter. Archive the created EAR file with the file name `CustomAdapter.ear`.

When creating an EAR file, use the `jar` command and archive the JAR file provided with the Cosminexus Service Platform in the EAR file.

For details about the directory configuration of EAR files when using the `jar` command for archiving, see 3.3.14(6) *Creating an EAR file*.

### (a) Prerequisites

Save a copy of the following file in the current directory:

```
Service-Platform-installation-directory\CSC\lib\cscmsg_adpejb.jar
```

### (b) Creating a file

Set name of the file to be created as `CustomAdapter.ear`.

If you execute the following command, the EAR file will be created:

---

```
jar cf ..\CustomAdapter.ear .\
```

---

### (c) Registering the file

To enable the usage of the file as a custom adapter, you must register the created EAR file in HCSC-Definer using the Adapter Addition Wizard. For details about how to register, see 3.2.14 *Adding a new custom adapter*.

## B.4 Custom adapter definition screen

This section describes the service adapter definition screens you use to define General custom adapters.

## (1) Service adapter definition (standard) screen

The items in the Service adapter definition (standard) screen are described below.

Figure B–3: Service adapter definition (standard) screen

## (a) Service component control information

This area displays information about the service component specified when a General custom adapter is created.

**Service name**

Displays the service name set in the Service adapter definition addition screen.

**Service ID**

Enter a service ID. Specify the service ID using no more than 8 bytes of alphanumeric characters and underscores (\_).

**Service type**

Displays Custom adapter as the service component type.

**Address**

This item is not used.

**Maximum instances**

Displays the maximum number of instances of the service component.

**Service class name**

This item is not used.

**Operation**

Displays an operation name. You can select the operation name of the service component from the drop-down list.

**Convert a system exception into a fault message** check box

Select this check box if you want exceptions that occur in the service component to be converted to faults.

**Add** button

Displays a dialog box in which you can add an operation.

**Delete** button

Deletes the operation selected in the **Operation** field.

(b) Operation information

This area displays information about the operations of the service component.

**Operation name**

Displays the name of the operation selected in the **Operation** field in the **Service component control information** area.

**Communication model**

Displays the communication model of the operation selected in the **Operation** field in the **Service component control information** area. You can select **Sync** or **Async** from the drop-down list. The information displayed in the Service adapter definition (standard) screen changes according to the option you select.

- If you select **Sync**, information about the response message and fault message is displayed.
- If you select **Async**, information about the response message and fault message is not displayed. If you save the adapter definition with **Async** selected as the communication model, all fields relating to the response message and fault message are deleted.

(c) Request message

This area displays information about the request message sent from the service requester.

**Use any type** check box

Select this check box to create request messages without specifying a data type.

**Standard**

**Use** check box

Select this check box if you want the service to check the contents of standard messages received from the service requester, or to transform standard messages to the message format of the General custom adapter.

**Format ID**

Displays the format ID of standard messages received from the service requester.

**Message format**

Displays the format name of standard messages received from the service requester.

**Browse** button

Displays a dialog box in which you can set the standard message format.

**Display** button

Displays the format of the standard message.

**Acquire** button

Displays a dialog box in which you can specify the output destination of the standard message format.

**Service component****Format ID**

Displays the format ID of the message format of request messages received by the protocol converter of the General custom adapter.

**Message format**

Displays the name of the message format of request messages received by the protocol converter of the General custom adapter.

**Browse button**

Displays a dialog box in which you can set the message format of request messages received by the protocol converter of the General custom adapter.

**Display button**

Displays the message format of request messages received by the protocol converter of the General custom adapter.

**Acquire button**

Displays a dialog box in which you can specify the output destination of the message format of request messages received by the protocol converter of the General custom adapter.

**Data-conversion definition**

Enter the name of the definition file used to transform the standard message to the request message received by the protocol converter of the General custom adapter. This item is unavailable when the **Use** check box is cleared in the **Standard message** area.

**Edit button**

Displays the Data-conversion definition screen. In this screen, you can edit the contents of the definition file set in the **Data-conversion definition** field. If you are creating a new data transformation definition, the Select Root Element dialog box appears. This item is unavailable when the **Use** check box is cleared in the **Standard message** area.

**Delete button**

Deletes the data transformation definition set by using the **Edit** button. This item is unavailable when the **Use** check box is cleared in the **Standard message** area.

## (d) Response message

This area displays information about the response message sent from the service component.

You can set the information in this area when **Sync** is selected as the communication model in the **Operation information** area.

**Use any type check box**

Select this check box to create response messages without specifying a data type.

**Standard****Use check box**

Select this check box if you want to check the contents of messages for the General custom adapter received from the service component or to transform messages for the General custom adapter to standard message format.

**Format ID**

Displays the format ID of the standard message used when responding to the service requester.

**Message format**

Displays the name of the format of the standard message used when responding to the service requester.

**Browse button**

Displays a dialog box in which you can set the standard message format.

**Display button**

Displays the format of the standard message.

**Acquire button**

Displays a dialog box in which you can specify the output destination of the standard message format.

### Service component

#### Format ID

Displays the format ID of the message format of response messages sent by the protocol converter of the General custom adapter.

#### Message format

Displays the name of the message format of response messages sent by the protocol converter of the General custom adapter.

#### Browse button

Displays a dialog box in which you can set the message format of response messages sent by the protocol converter of the General custom adapter.

#### Display button

Displays the message format of response messages sent by the protocol converter of the General custom adapter.

#### Acquire button

Displays a dialog box in which you can specify the output destination of the message format of response messages sent by the protocol converter of the General custom adapter.

### Data-conversion definition

Enter the name of the definition file used to transform the response message sent from the protocol converter of the General custom adapter to a standard message. This item is unavailable when the **Use** check box is cleared in the **Standard message** area.

#### Edit button

Displays the Data-conversion definition screen. In this screen, you can edit the contents of the definition file set in the **Data-conversion definition** field. If you are creating a new data transformation definition, the Select Root Element dialog box appears. This item is unavailable when the **Use** check box is cleared in the **Standard message** area.

#### Delete button

Deletes the data transformation definition set by using the **Edit** button. This item is unavailable when the **Use** check box is cleared in the **Standard message** area.

### (e) Fault message

This area displays information about the fault message of the service component.

#### Add button

Displays a dialog box in which you can add a fault name. This button is unavailable when defining a General custom adapter.

#### Delete button

Deletes the file selected in the **Fault name** field. This button is unavailable when defining a General custom adapter.

#### Fault name

Displays the fault name of the service component. This field is unavailable when defining a General custom adapter.

#### Message format

Displays the name of the format of the fault message of the service component. This field is unavailable when defining a General custom adapter.

#### Browse button

Displays a dialog box in which you can set the fault message format. This button is unavailable when defining a General custom adapter.

#### Display button

Displays the format of the fault message. This button is unavailable when defining a General custom adapter.

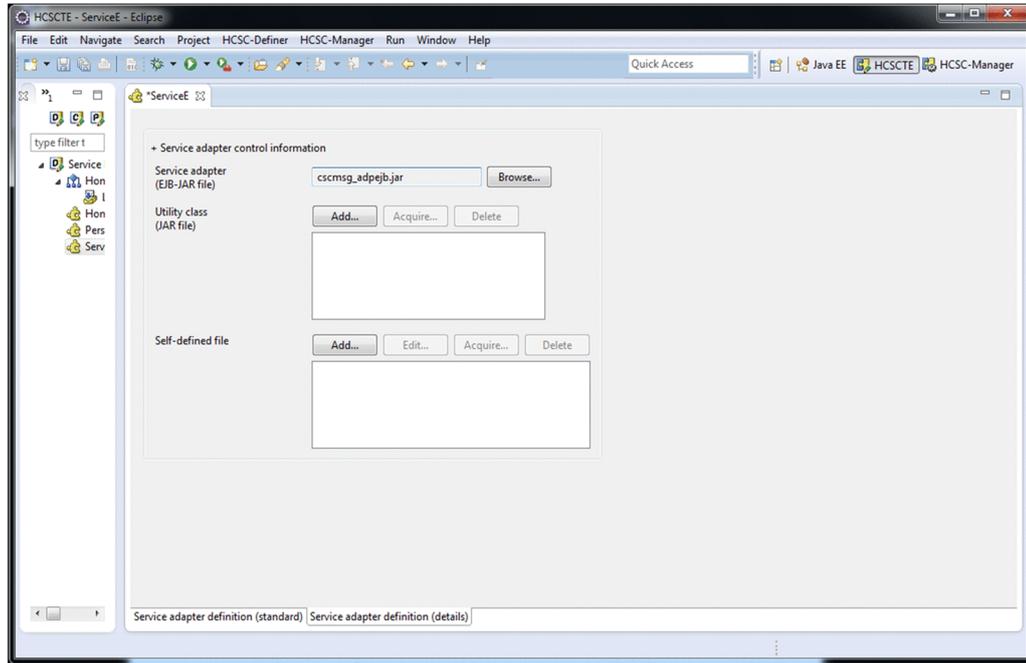
#### Acquire button

Displays a dialog box in which you can specify the output destination of the fault message format. This button is unavailable when defining a General custom adapter.

## (2) Service adapter definition (details) screen

The items in the Service adapter definition (details) screen are described below.

Figure B-4: Service adapter definition (details) screen



### (a) Service adapter control information

This area displays information about the service component specified when a General custom adapter is created.

#### Service adapter

##### Service adapter (EJB-JAR file)

Displays the name of the EJB-JAR file of the General custom adapter.

##### Browse button

Displays a dialog box in which you can set the file name of the EJB-JAR file of the General custom adapter.

#### Utility class

##### Utility class (JAR file)

Displays the utility classes set by the **Add** button.

##### Add button

Displays a dialog box from which you can add a utility class. Add the JAR file you created in *3.3.14 Defining custom adapters*.

##### Acquire button

Displays a dialog box in which you can specify the output destination of the utility class.

##### Delete button

Deletes the file selected in the **Utility class** field.

#### Self-defined file

##### Self-defined file

Displays the protocol converter self-defined file, custom adapter property file, HITACHI Application Integrated Property File for custom adapter, and custom adapter definition file you set using the **Add** button.

##### Add button

Displays a dialog box in which you can add self-defined files (protocol converter self-defined files, custom adapter property files, custom adapter HITACHI Application Integrated Property files, and custom adapter definition files).

**Edit** button

Displays an editor in which you can edit self-defined files (protocol converter self-defined files, custom adapter property files, custom adapter HITACHI Application Integrated Property files, and custom adapter definition files). If you click the **Edit** button with multiple self-defined files selected, an editor window opens for each file. After making changes in the editor, you can apply the changes to the repository by saving the self-defined file in the editor and then saving the General custom adapter. If you attempt to save the General custom adapter while editing a self-defined file, a dialog box appears asking if you want to save the self-defined file.

**Acquire** button

Displays a dialog box in which you can specify the output destination for self-defined files (protocol converter self-defined files, custom adapter property files, custom adapter HITACHI Application Integrated Property files, and custom adapter definition files).

**Delete** button

Deletes the file selected in the **Self-defined file** field.

## B.5 Environment Settings when using a custom adapter

This section describes the contents that you must set up in the execution environment of an HCSC server when using a custom adapter.

### (1) Setting up the protected area list file

To use a custom adapter on an HCSC server, set up the class in which the protocol converter (protocol conversion processing) is implemented as a class prohibiting the method cancellation, in the protected area list file.

Set up the implementation class name or prefix name of the custom adapter as elements of the *protected area list file*.

The set up protected area list file is read when the J2EE server is started, and becomes enabled in all the J2EE servers running on the machine on which Cosminexus is installed.

For details about the protected area list file, see *criticalList.cfg (protected area list file)* in the manual *Cosminexus Application Server Server Definitions*.

## B.6 Troubleshooting

This section describes the problems that might occur during operation of a General custom adapter, what might cause these problems, and the action you can take to resolve them.

### (1) Troubleshooting deployment of General custom adapters

The following describes what might cause deployment of a General custom adapter to fail, how to identify the cause of the failure, and what action you need to take.

#### (a) Causes

The following are the most likely causes of a General custom adapter failing to deploy:

1. The format or content of the HITACHI Application Integrated Property File for the custom adapter is invalid.
2. An invalid EAR file is registered because the correct procedure in 3.3.14 *Defining custom adapters* was not followed.

#### (b) Identifying the cause

Check the console log displayed during deployment:

- If the following message is displayed:

---

```
KEOS50007-E An exception occurred during the operation. (Cosminexus Manager
name = manager-ID, object name = jp.co.Hitachi.soft.csc.msg:type=
CSCMsgAdapterBuilder, operation = setup, details = The application attributes
file is invalid. (detailed-information) ErrorCode=KDEC10031-E)
```

---

The cause might be 1 or 2 above.  
View *detailed-information* for the specific cause.

### (c) Action to take

Take the following action:

- Check whether the format and content of the HITACHI Application Integrated Property File for the custom adapter are valid. If the format or content is invalid, create the file again. For details on the HITACHI Application Integrated Property File, see *B.2(3) HITACHI Application Integrated Property File for custom adapter*.
- Check whether the format and content of the EAR file you are using are valid. If the format or content is invalid, create the file again. For details on how to create an EAR file, see *3.3.14(6) Creating an EAR file*.

## (2) Troubleshooting startup of General custom adapters

The following describes what might cause a General custom adapter to fail to start, how to identify the cause of the failure, and what action you can take to resolve it.

### (a) Causes

The following are the most likely causes of a General custom adapter failing to start:

1. The custom adapter development framework action definition file has not been placed in the root folder of all registered JAR files.
2. The format of the custom adapter development framework action definition file is invalid.
3. The format of the custom adapter property file is invalid.
4. The protocol converter JAR file does not include the class file for the protocol converter.
5. The protocol converter is specified incorrectly in the custom adapter development framework action definition file, or the specified protocol converter does not exist.
6. The protocol converter does not have a public default constructor.
7. An exception occurred in the `start` method of the protocol converter.

### (b) Identifying the cause

Check the message log file.

- If a message like the following is recorded in the message log:

---

```
KDEC02513-E An attempt to read the file has failed. (adapter name =
adapter-name, file name =file-name, information =detailed-information)
```

---

The following table shows the correspondence between the characters in the error message and the cause of the error:

Table B–5: Causes of KDEC02513-E

File name	Detailed information	Cause
framework_properties.xml	InputStream is null	The cause might be 1.
	InvalidPropertiesFormatException	The cause might be 2.
defs/customadapter_properties.xml	InvalidPropertiesFormatException	The cause might be 3.

- If a message like the following is recorded in the message log:

---

```
KDEC03015-E An initialization error occurred in the protocol converter.
(adapter name = adapter-name, information =exception-that-caused-error)
```

---

The following table shows the correspondence between the characters in the error message and the cause of the error:

Table B–6: Causes of KDEC03015-E

Exception that caused error	Cause
ClassNotFoundException	The cause might be 4 or 5.
IllegalAccessException	The cause might be 6.
CSCMsgCustomAdapterException	The cause might be 7.

**(c) Action to be taken**

If the cause is 1 or 2, take the following action:

- Check whether the format and content of the custom adapter development framework action definition file are valid. If the format or content is invalid, create the file again. For details on the custom adapter development framework action definition file, see *B.2(1) Custom adapter development framework action definition file*.
- Check whether the custom adapter development framework action definition file is in the location specified in the JAR file for the protocol converter. If the file is not in this location, place it in the correct location.

If the cause is 3, take the following action:

- Check whether the format and content of the operation definition file for the custom adapter development framework are valid. If the format or content is invalid, create the file again. For details on the custom adapter property file, see *B.2(2) Custom adapter property file*.

If the cause is 4 or 5, take the following action:

- Check whether the protocol converter JAR file includes the class file for the protocol converter. If it does not include the class file, use the `jar` command to archive the protocol converter class file, the class files used by the protocol converter class file, and the custom adapter development framework action definition file in the JAR file. For details on how to create the protocol converter JAR file, see *3.3.14(5)(a) Creating protocol converter JAR files*.
- Check that the package name is specified together with the protocol converter name in the custom adapter development framework action definition file. If the name is invalid, specify the protocol converter name and package name correctly.
- Check whether `classname` is specified as the value of the `key` attribute of the `entry` element that specifies the protocol converter name in the custom adapter development framework action definition file. If this is not the case, enter the class name of the protocol converter in the `entry` tag whose `key` attribute is `classname`.

If the cause is 6, take the following action:

- Check whether a public default constructor is defined for the protocol converter. If none is defined, define a public default constructor.

If the cause is 7, take the following action:

- Check the processing of the `start` method of the protocol converter. If the processing is invalid, review its implementation. For details on the format of the `start` method, see *B.1(2)(a) CSCMsgCustomAdapterContext interface*.
- Make sure that the environment and preferences you are using permit the protocol converter to start.

**(3) Troubleshooting exceptions in operations performed from the converter**

This section describes what might cause a `java.security.AccessControlException` exception to occur when you attempt to perform file operations from the protocol converter, how to identify the cause, and what action to take.

**(a) Cause**

A Java security feature might have prevented file access.

**(b) Identifying the cause**

Make sure that `java.security.AccessControlException` is thrown in the protocol converter during file access.

**(c) Action to be taken**

Take the following action:

- Use the container extension library  
For details on how to use the container extension library, see *14.2 Using the container extension library* in the manual *Application Server Common Container Functionality Guide*. For details on how to implement the container extension library, see *14.3.2 Procedure of creating and using the container extension library* in the manual *Application Server Common Container Functionality Guide*.
- Edit the settings in the security policy file  
For details on the settings in the security policy, see *9.8.2 Setting security policy* in the manual *Application Server Expansion Guide*.
- Use the user log functionality (if the purpose of the file access is log output)  
For details on the user log functionality, see *9.12 Notes for using the user log functionality* in the manual *Application Server Expansion Guide*.

---

## C. Defining the DB adapter using the DB adapter definition support function

This appendix describes how to use the DB adapter definition support function to create a DB adapter definition.

### C.1 Overview of Overview of the DB adapter definition support function

The DB adapter definition support function has two key features:

- Creation of definition files for DB adapters

The DB adapter definition support function generates the files required when defining a DB adapter. It can generate the following files:

- SQL operation definition file
- HITACHI Application Integrated Property file
- Standard message format for request and response messages
- Service component message format for request and response messages

#### **!** Important note

The DB adapter definition support function cannot generate a standard message format for request messages associated with a function that executes multiple SQL statements in response to a single service component execution request from the service requester. You must create the standard message format for such request messages manually.

- 
- Debugging of SQL statements with argument elements

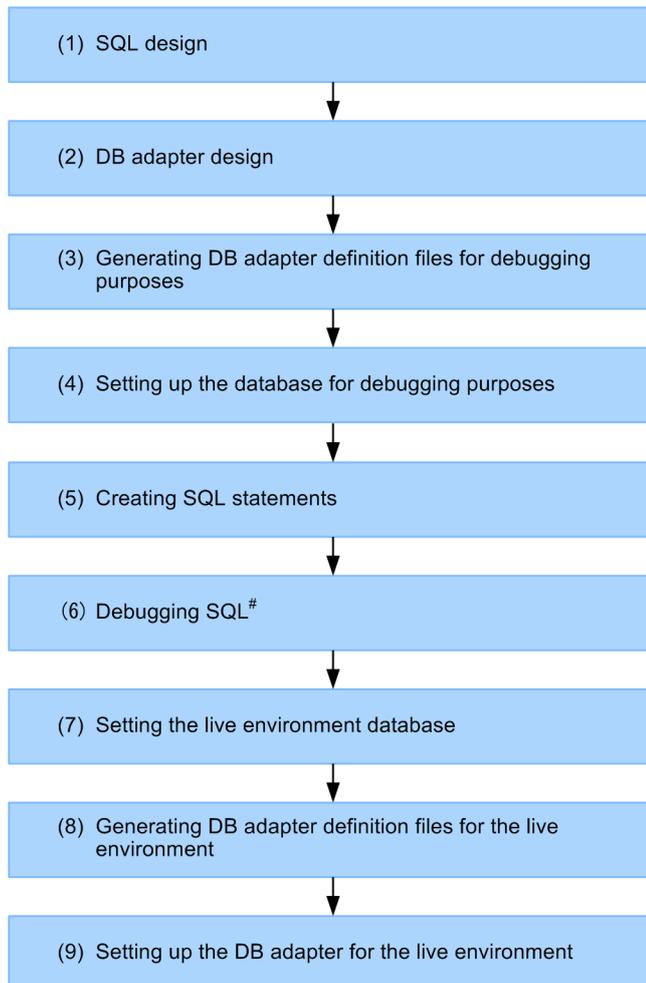
The DB adapter definition support function provides the following three features that support the debugging of SQL statements with argument elements:

- Extracting argument elements
- Executing SQL statements with values substituted for argument elements
- Checking SQL execution results

### C.2 Defining Defining the DB adapter

The following figure shows the flow of DB adapter definition in the development environment:

Figure C–1: Flow of DB adapter definition



#  
Only perform this step if a separate database from that of the live environment has been provided for debugging purposes.

Each step is described in detail below. The instructions assume that you are using HiRDB. If you are using Oracle, interpret each definition as its Oracle equivalent.

#### ! Important note

Close all editors associated with the DB adapter definition support function before closing the Eclipse project.

## (1) SQL design

Use the DB adapter definition support function to design the contents of the SQL statements executed in DB adapters. The variable components of SQL execution statements, such as data and the names of tables and columns, are designed as argument elements.

Note that the DB adapter definition support function allows you to specify the following SQL commands:

- SELECT
- INSERT

The data types you can specify are those supported by the JDBC driver used by the DB Connector.

## (2) DB adapter design

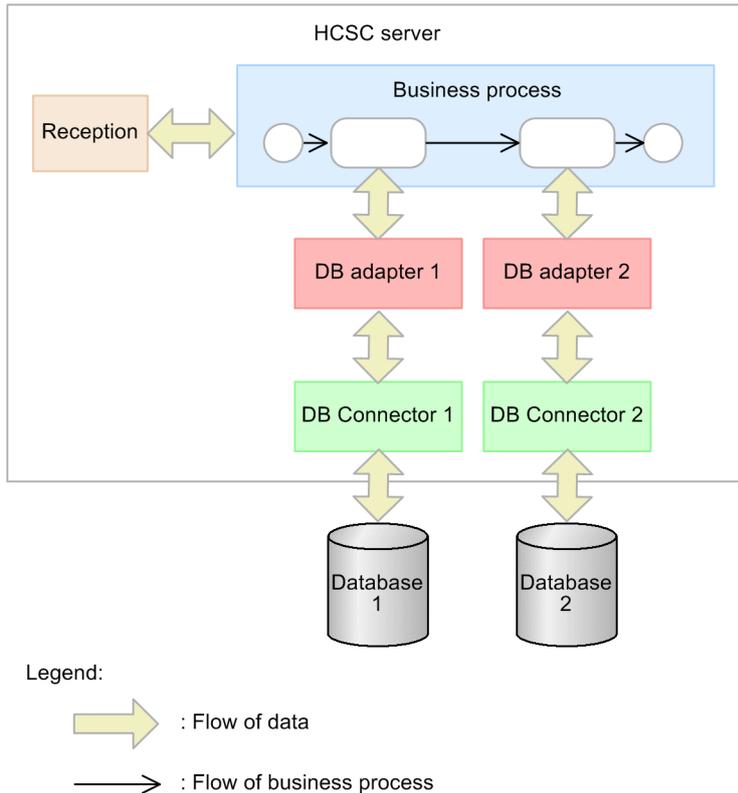
Design the content of the DB adapters. The aspects you need to design are described below.

### (a) Level of DB adapter creation

Because a DB adapter is required for each DB Connector, you need to prepare DB adapters according to how the DB Connectors are structured.

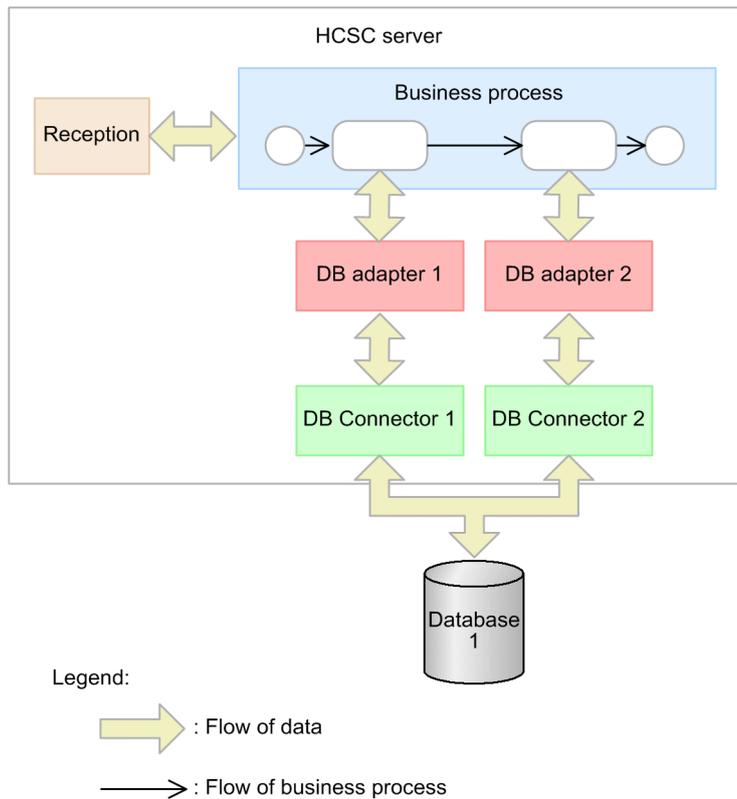
For example, a system that integrates multiple databases will have a DB Connector for each database. In this case, you will need to create a DB adapter for each DB Connector.

Figure C-2: When there is a DB Connector for each database



You can also prepare multiple DB Connectors for the same database. This is useful in situations where you want to use different connection pool settings for individual DB adapters. In this case, you need to create a DB adapter for each DB Connector regardless of the number of databases in the system.

Figure C-3: When a database has multiple DB Connectors



## (b) Level of SQL statement definition

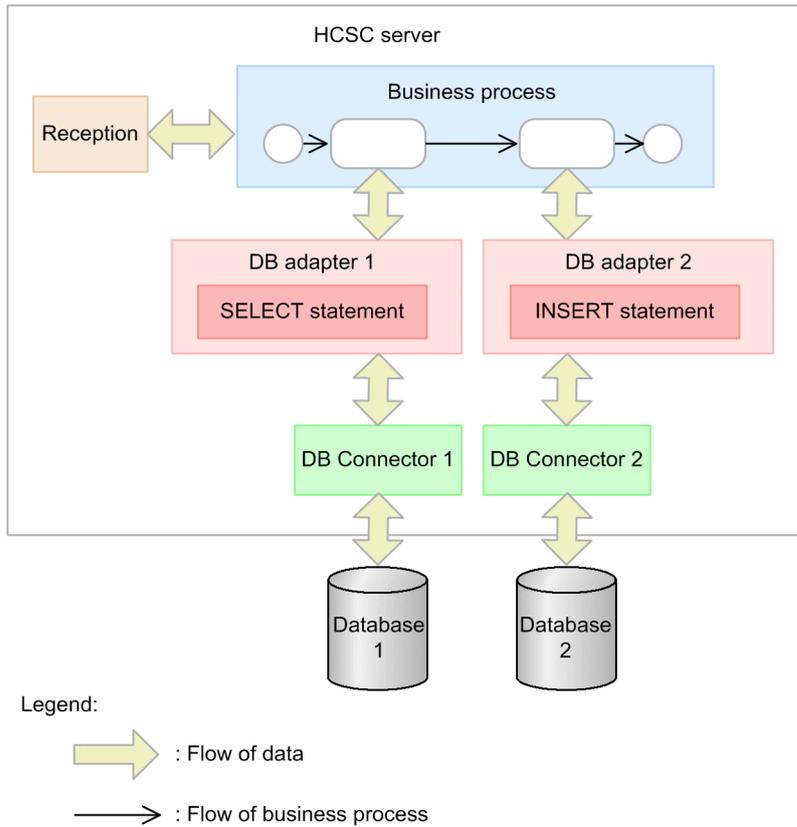
You can define one or more SQL statements in a DB adapter. Design the contents of the SQL statements to be defined in each DB adapter. Examples of SQL statement design are shown below.

- When executing different SQL statements for different databases

If you want to execute different SQL statements for different databases, define the appropriate SQL statements in the DB adapter for each database.

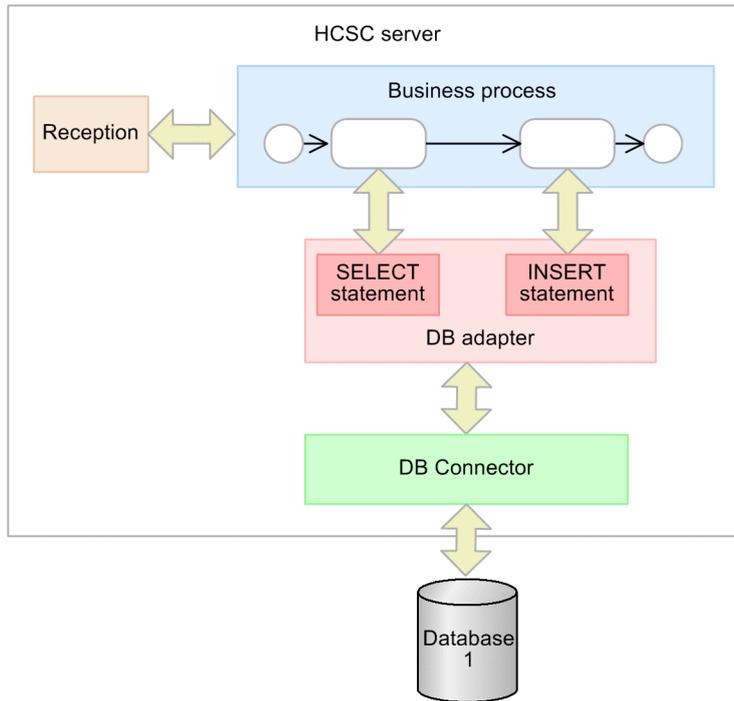
For example, if you want to find data in Database 1 in the figure and register that data in Database 2, define a `SELECT` statement in DB adapter 1 and an `INSERT` statement in DB adapter 2.

Figure C-4: Executing different SQL statements for different databases



- When executing different SQL statements for the same database  
If you want to execute different SQL statements for different activities in a database, define multiple SQL statements in the DB adapter.  
For example, if you want data found by the first activity to be registered by the next activity, define `SELECT` and `INSERT` statements in the same DB adapter.

Figure C–5: Executing different SQL statements for the same database



Legend:

 : Flow of data

 : Flow of business process

When using the DB adapter definition support function to define multiple SQL statements in a DB adapter, define an operation for each SQL statement to execute.

For details on how to specify operations, see (5) *Creating SQL statements*.

### (3) Generating DB adapter definition files for debugging purposes

Create sample DB adapter definition files to use for debugging purposes. DB adapter definition files are created in Eclipse. Use the procedure below to create a new DB adapter definition file.

#### ! Important note

- If the following message appears when editing a DB adapter definition file, synchronize the resources with the file system by right-clicking the project in Package Explorer view and selecting **Refresh**.  
Resource 'file-path' is out of sync with file system.
- Deleting a DB adapter definition file does not automatically delete the output directory that was created when outputting the file. Delete the directory manually as needed.
- When working with a DB adapter definition file, use the view that opens when you select the HCSCTE-DBEditor perspective.

#### 1. Create a project in Eclipse.

We recommend that you select HCSCTE as the project type. For details on how to create a HCSCTE project, see 3.1.1 *Creating a Project in the Service Platform Basic Development Guide*.

#### 2. From the Eclipse menu, select **Window, Open Perspective**, and then **Other**.

The Open Perspective dialog box appears.

#### 3. Select **HCSC-DBEditor**, and then click **OK**.

The HCSCTE-DBEditor perspective appears.

#### 4. From the Eclipse menu, select **File, New**, and then **Other**.

### C. Defining the DB adapter using the DB adapter definition support function

A dialog box appears in which you can select a wizard.

5. Under **HCSCTE-DBEDITOR**, select **DBEditor File** and then click **Next**.

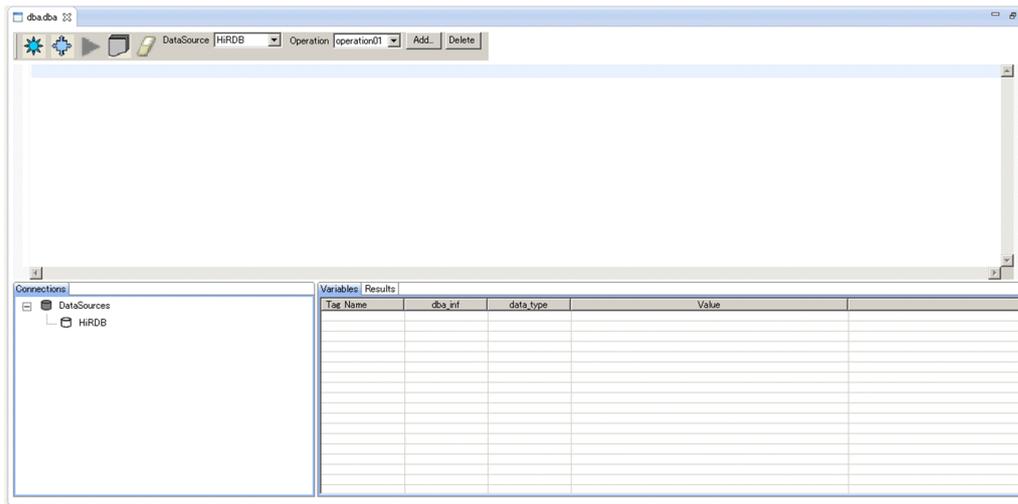
The **New DBA File** wizard appears.

6. In the **Enter or select the parent folder** field, enter or select the folder in which you want to save the definition file of the DB adapter.

7. In the **File name** field, enter the file name for the definition file.

8. Click **Finish**.

The new definition file is created, and the HCSCTE-DBEditor window appears.



### (4) Setting up the database for debugging purposes

In the DB adapter definition file, set the information for the database you want to use. This process differs depending on whether the database you are using for debugging differs from the database in the production environment. You cannot debug SQL statements without a debug database.

#### Reference note

---

Service Architect incorporates an embedded database you can use for debugging purposes.

---

#### (a) When providing a debug database separate from the production environment database

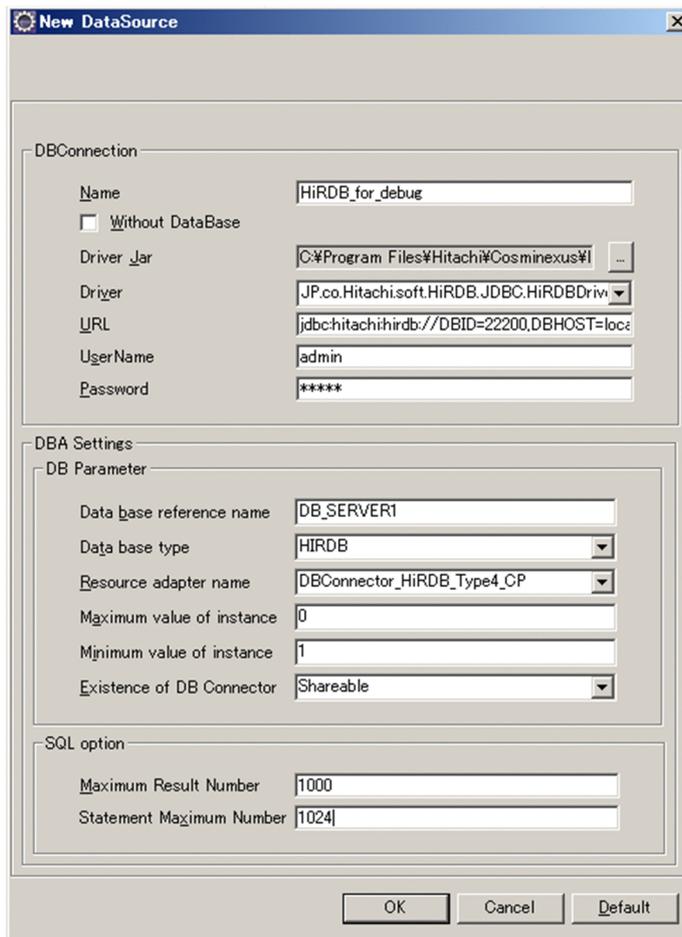
To set up the debug database in an environment where the debug database is separate from the database for the production environment:

1. Open the DB adapter definition file in Eclipse.

The HCSCTE-DBEditor window appears.

2. On the **Connections** tab, right-click **DataSources** and select **New DataSource**.

The New DataSource dialog box appears.



3. Enter values in the input fields in the **DBConnection** area.

Enter the settings for SQL debugging.

Here, you enter the settings required to connect to the database using the JDBC driver.

Table C-1: Information to define in input fields in **DBConnection** area (**New DataSource** dialog box)

Item	Description	Default value <sup>#</sup>
<b>Name</b>	The name displayed on the <b>Connections</b> tab of the HCSCTE-DBEditor window. Specify a name that is unique within the DB editor definition file.	HiRDB
<b>Without DataBase</b>	Select this check box if there is no debug database.	Cleared
<b>Driver Jar</b>	Specify the JAR file of the JDBC driver. Make sure that you obtain the appropriate JDBC driver in advance.	<i>service-platform-installation-directory</i> \DB\CLIENT\UTL\pdjdbc2.jar
<b>Driver</b>	Specify the package name (fully qualified Java class name) of the JDBC you want to use.	JP.co.Hitachi.soft.HiRDB.JDBC.HiRDBDriver
<b>URL</b>	Specify the database connection URL, including the port number and host name of the connection-target database.	jdbc:hitachi:hirdb://DBID=22200,DBHOST=localhost
<b>User Name</b>	Specify the login user name used when connecting to the database.	admin
<b>Password</b>	Specify the password used when connecting to the database.	admin (shown as **** on screen)

## C. Defining the DB adapter using the DB adapter definition support function

### Note:

The values in the input fields in the **DBConnection** area are only used when debugging SQL statements. They are not used in the SQL operation definition file and HITACHI Application Integrated Property file generated by the DB adapter definition support function.

### #

The default values are the values entered when you click the **Default** button in the **New DataSource** dialog box.

The initial values are the same as the environment created when you execute the HCSC Easy Setup function with the initial values.

### 4. Enter values in the input fields in the **DBA Settings** area.

The settings in this area are used to define the SQL operation definition file for the DB adapter and the HITACHI Application Integrated Property file. Specify the values according to the settings of the DB Connector you are using.

Table C-2: Information to define in input fields in **DBA Settings** area (**New DataSource** dialog box)

Item	Description	Default value <sup>#</sup>
<b>Data base reference name</b>	Specify the database reference name.	DB_SERVER1
<b>Data base type</b>	Specify the database type.	HIRDB
<b>Resource adapter name</b>	Specify the name of the resource adapter specified in the property definition ( <code>display-name</code> ) of the DB Connector. If you cannot select the name in the drop-down list, enter the value directly.	DBConnector_HiRDB_Type4_CP
<b>Maximum value of instance</b>	Specify the maximum number of instances of the DB adapter. To allow unlimited instances, specify 0.	0
<b>Minimum value of instance</b>	Specify the minimum number of instances of the DB adapter.	1
<b>Existence of DB Connector</b>	Specify whether to share the DB Connector referenced by the DB adapter.	Shareable
<b>Maximum Result Number</b>	Specify the maximum number of results to be returned by a <code>SELECT</code> statement. If the <code>SELECT</code> statement returns more than the specified number of results, only the number specified here is displayed. Example: If you specify 1000, results for a maximum of 1000 records is displayed.	1000
<b>Statement Maximum Number</b>	If you want to execute multiple SQL statements in a single service component execution request, specify the maximum number of SQL statements to execute.	1024

### #

The default values are the values entered when you click the **Default** button in the **New DataSource** dialog box.

The initial values are the same as the environment created when you execute the HCSC Easy Setup function using the initial values.

### ! Important note

When using HiRDB as the DBMS in the DB adapter configuration, use the following combination of database type and resource adapter name:

- When using HiRDB as the database type, set `DBConnector_DABJ_CP` or `DBConnector_DABJ_XA` as the resource adapter name.
- When using HIRDB-TYPE4 as the database type, set `DBConnector_HiRDB_Type4_CP` or `DBConnector_HiRDB_Type4_XA` as the resource adapter name.

- Click **OK**.

The settings are registered.

(b) When using the production environment database for debugging

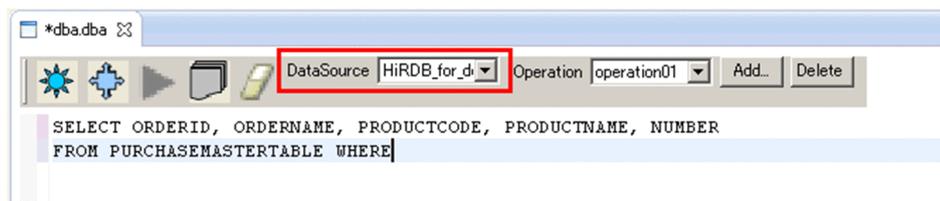
To set up the data source in an environment that does not use a separate database for debugging:

- Open the DB adapter definition file in Eclipse.  
The HCSCTE-DBEditor window appears.
- On the **Connections** tab, right-click **DataSources** and then select **New DataSource**.  
The **New DataSource** dialog box appears.
- Select the **Without DataBase** check box in the **DBConnection** area.
- In the **DBA Settings** area, enter the database settings for the production environment.
- Click **OK**.  
The settings are registered.

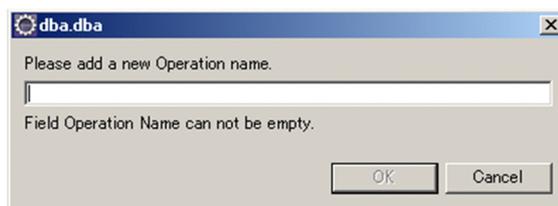
## (5) Creating SQL statements

To define the SQL statements executed in the DB adapter:

- Open the DB adapter definition file in Eclipse.  
The HCSCTE-DBEditor window appears.
- From the **DataSource** pull-down list, select the data source you want to use.



- Click the **Add** button next to the **Operation** pull-down list.  
A dialog box appears in which you can specify the name of the operation you are adding.



- Enter the name of the operation you want to add, and then click **OK**.  
If you intend to define multiple SQL statements in one DB adapter, define an operation for each SQL statement to execute.
- Define the SQL statement you want to execute in the appropriate fields.  
For details on how to define argument elements, see *Format of the SQL operation definition file*.

### ! Important note

- You can enter one SQL statement per operation.
- You cannot use wildcards (\*). You must enter the name of every column targeted by a `SELECT` statement.
- You do not need to terminate the SQL statement with a semicolon.

## (6) Debugging SQL

Debug the SQL statements you defined. When you execute the SQL statements during the debugging process, values are substituted for the argument elements. Note that you can only debug SQL when a separate database from that of the production environment has been provided for debugging purposes.

To debug SQL statements:

1. Create the tables to use for debugging.

In the database used for debugging, create the tables to use in the debug process. Create the tables according to the SQL statements you defined.

Example:

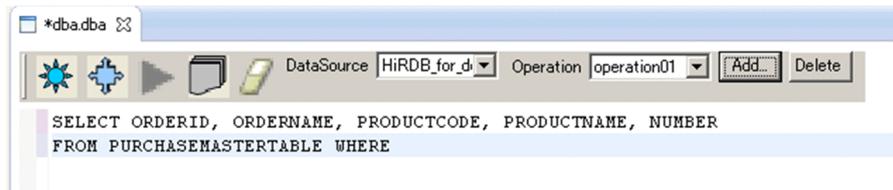
Table name: PURCHASEMASTERTABLE

Table structure:

ORDERID (VARCHAR)	ORDERNAME (VARCHAR)	PRODUCTCODE (VARCHAR)	PRODUCTNAME (VARCHAR)	NUMBER (INTEGER)
01	AA001	0A1	NPW	4
02	AA002	0A1	NPW	3
03	AA003	0E1	NPW	3
04	BB001	0E1	NPW	4
05	BB002	0B1	NPW	5

2. Open the DB adapter definition file in Eclipse.

The HCSCTE-DBEditor window appears.



3. With the SQL statement whose argument elements you want to extract selected in the **Operation** pull-down list, click the **Parse** icon (  ) in the toolbar. The variables appear on the **Variables** tab.

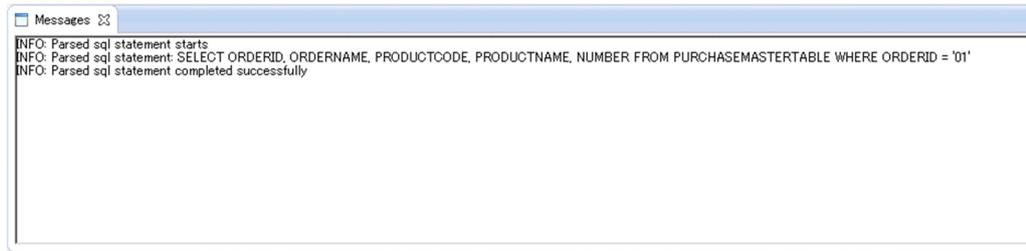
Tag Name	dba_inf	data_type	Value
KEYCOLUMN	column	--	
ENZANSHI	preset	--	
DATA	data	VARCHAR	

4. In the **Value** column on the **Variables** tab, enter the values you want to use for debugging.

Example:

Argument element name	DBa_inf	data_type	Value
KEYCOLUMN	column	--	ORDERID
OPERATOR	preset	--	=
DATA	data	VARCHAR	'01'

You cannot perform the subsequent steps unless a debug database has been set up. When you click the **Parsed SQL Statement** icon (  ), the SQL statement appears in Messages view with values substituted for its argument elements. Use the information in this window to make sure that the SQL statement has been defined correctly.



5. On the **Connections** tab, right-click the data source used for debugging, and then click **Connect**.

A connection with the database is established based on the parameters of the data source. If the connection is successful, the following message appears in Messages view: `INFO: Data Source <data-source-name> Connection established successfully.`

If you defined multiple SQL statements, repeat steps 3 to 5 for each statement.

6. Click the **Execute** icon (  ) in the toolbar.

If execution of the SQL statement fails, an error message appears in Messages view. Amend the SQL statement and argument element names according to the message contents.

The execution results of the SQL statement are displayed on the **Results** tab.

ORDERID	ORDERNAME	PRODUCTCODE	PRODUCTNAME	NUMBER
01	AA001	0A1	NPW	4

## (7) Setting the production environment database

Change the data source settings in the DB adapter definition file to the settings for the database in the production environment.

### Important note

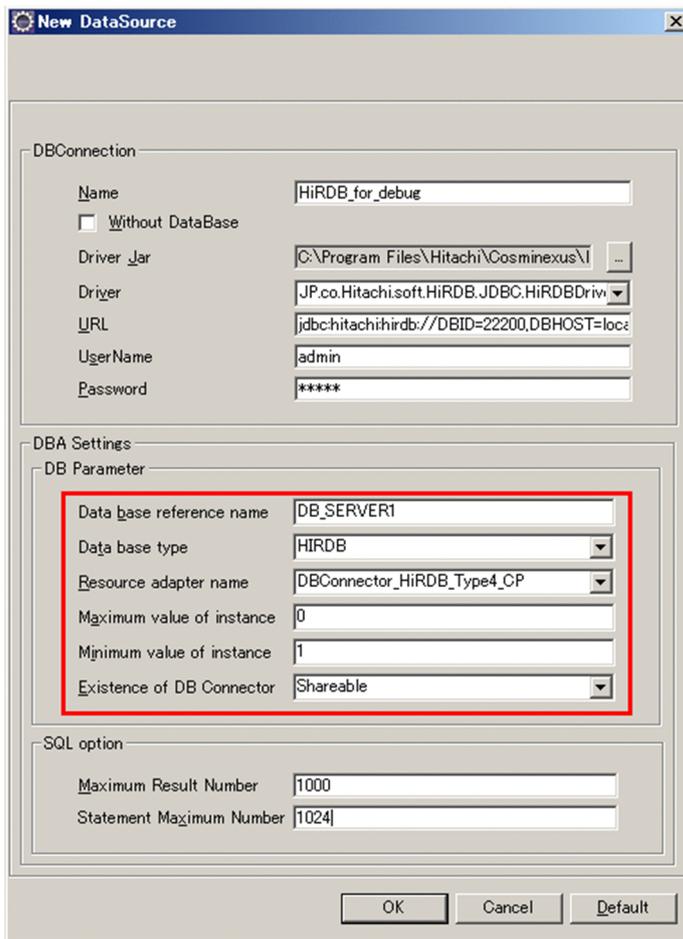
Before making this change, disconnect the data source you were using for debugging. To disconnect a data source:

1. Open the DB adapter definition file used for debugging in Eclipse.
2. On the **Connections** tab, right-click the data source used for debugging, and then click **Disconnect**.

To change the database settings:

1. Open the DB adapter definition file you used for debugging in Eclipse.  
The HCSCTE-DBEditor window appears.
2. On the **Connections** tab, right-click the data source and select **Edit DataSource**.  
The **Edit DataSource** dialog box appears.
3. In the fields in the **DBA Settings** area, enter the database information for the production environment.

C. Defining the DB adapter using the DB adapter definition support function

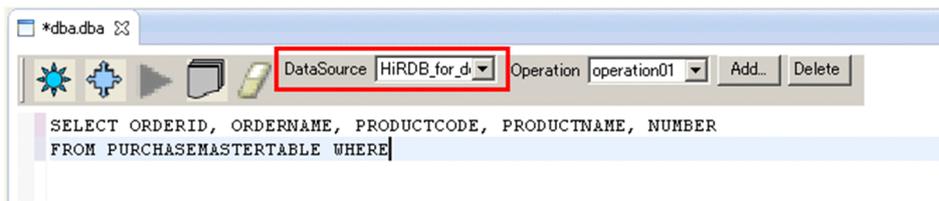


- 4. Click **OK**.  
The settings are registered.

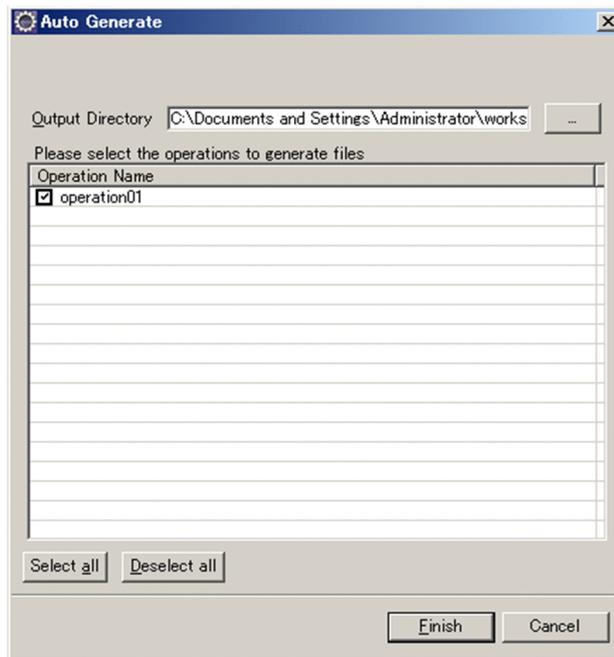
(8) Generating DB adapter definition files for the production environment

To generate the DB adapter definition files to be used in the production environment:

- 1. Open the DB adapter definition file for debugging in Eclipse.  
The HCSCTE-DBEditor window appears.
- 2. From the **DataSource** pull-down list, select the data source you want to apply to the DB adapter definition files for the production environment.



- 3. Click the **Auto generate** icon (  ) in the toolbar.  
The **Auto Generate** dialog box appears.



4. In the **Output Directory** field, specify the directory to which to output the DB adapter definition files for the production environment.

If you click the ... button, the following path is selected in the **Output Directory** field:

*Eclipse-project-path\DB-adapter-definition-file-name\_Output*

**! Important note**

- If you specify the path of the Eclipse project as the output directory, you can use Eclipse functionality to import and export DB adapter definition files and the results of definition file generation in a batch.
- Specify the output directory as an absolute path that contains no more than 155 bytes when converted to UTF-8 encoding.

5. Select the check box for the operation or operations for which you want to generate DB adapter definition files. If you defined multiple SQL statements for one DB adapter, select all the operations required to define the SQL.
6. Click **Finish**.

The DB adapter definition files for use in the production environment are output to the directory you specified. If you specified the path of the Eclipse project as the output directory, refresh the Eclipse project after the DB adapter definition files are generated. The DB adapter definition files appear in the Package Explorer view of Eclipse.

The following table lists the generated files:

Table C-3: Generated files

File type	File name	Number of files	Remarks
SQL operation definition file	<i>csa_sql_dbadapter.xml</i>	1	--
HITACHI Application Integrated Property file	<i>cscadapter_property.xml</i>	1	--
Standard request message <sup>#1</sup>	<i>operation-name_Input.xsd</i>	0 to <i>n</i> <sup>#2</sup>	<i>operation-name</i> is replaced with the specified operation name.
Standard response message <sup>#3</sup>	<i>operation-name_Output.xsd</i>	0 to <i>n</i> <sup>#2</sup>	<i>operation-name</i> is replaced with the specified operation name.

## C. Defining the DB adapter using the DB adapter definition support function

File type	File name	Number of files	Remarks
Service component message format for request and response messages	ServiceMessageFile.xsd	1	Use the same file as the service component message of the request message and response message.

### Legend:

--: No description.

#### #1

If there are no argument elements in the SQL statements, a standard message format for the request message is not generated. In this case, use the service component message. When invoking the DB adapter without argument elements, use an invoke Java activity or other means to generate the request message and invoke the DB adapter.

Example:

Operation name: `select_op`

SQL statement to execute: `SELECT ORDERID FROM PURCHASEMASTERTABLE`

Message generated by invoke java activity or similar:

```
<?xml version="1.0" encoding="UTF-8"?>
<DBadapter>
<DBA_MULTI_SQL>
<select_op>
<DBA_IN_DATA/>
</select_op>
</DBA_MULTI_SQL>
</DBadapter>
```

#### #2

The number of operations specified when generating the DB adapter definition files for the production environment.

#### #3

The standard response message is only output when the SQL command is `SELECT`. If the command is `INSERT`, use the service component message format of the response message.

## (9) Setting up the DB adapter for the production environment

Create the DB adapter using the DB adapter definition files generated for the production environment.

### (a) Creating the DB adapter

Define a new DB adapter using the HCSCTE interface integrated into Eclipse. For details on how to define a new DB adapter, see [3.2.5 Adding a new database adapter](#).

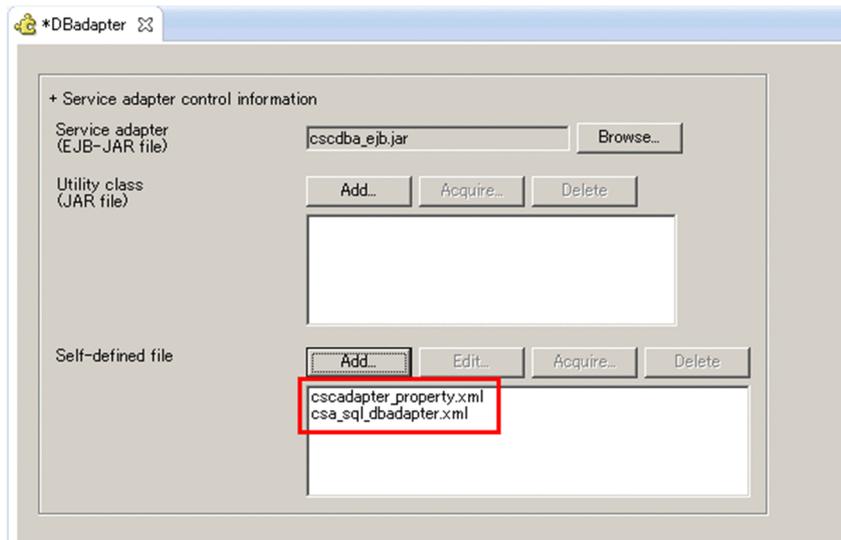
You also need to add the operations specified in [\(8\) Generating DB adapter definition files for the production environment](#) to the DB adapter you created. If you specified multiple operations, add each operation you specified.

### (b) Setting the SQL operation definition file and HITACHI Application Integrated Property File

Display the Service-adapter definition (details) window for the DB adapter you created. Add the following files to the **Self-defined file** area:

- `csa_sql_dbadapter.xml` (the generated SQL operation definition file)
- `cscadapter_property.xml` (the generated HITACHI Application Integrated Property File)

Figure C–6: Self-defined files to add to the DB adapter



## (c) Setting request message and response message information

Display the Service-adapter definition (standard) window for the DB adapter you created. Set the following information:

## Operation names

- **Operation** in the **Service component control information** area  
Select the operation name from the drop-down list.

## Request message for operation

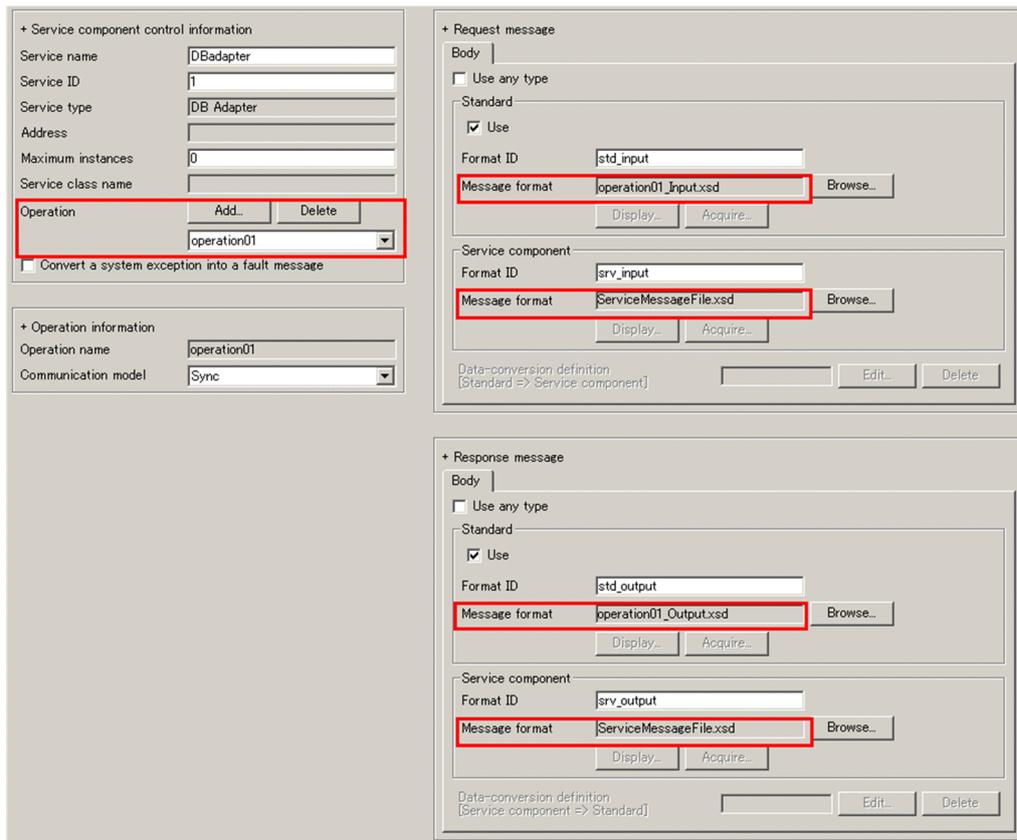
- **Message format** under **Standard** in the **Request message** area  
Specify *operation-name\_Input.xsd*.
- **Message format** under **Service component** in the **Request message** area  
Specify *ServiceMessageFile.xsd*.

## Response message for operation

- **Message format** under **Standard** in the **Response message** area  
Specify *operation-name\_Output.xsd*. Specify this file only if the SQL command in the specified operation is **SELECT**. You do not need to specify the file if the SQL command is **INSERT**.
- **Message format** under **Service component** in the **Response message** area  
Specify *ServiceMessageFile.xsd*.

The following figure shows an example of setting this information:

Figure C–7: Example of setting request message and response message



(d) Defining data transformation (mapping)

Define the source and destination mappings to use when transforming data.

- Defining data transformation (mapping) of request messages
 

You can use the auto mapping function to perform data transformation of the standard message and service component message in a request message. For details on the auto mapping function, see 6.4 Mapping in the manual *Service Platform Basic Development Guide*.
- Defining data transformation (mapping) of response messages
 

Data transformation of the standard message and service component message in a response message must be mapped manually.

For details on how to transform the results of a SELECT statement to a standard message, see 3.3.5(5) Data transformation.

A standard message format file for response messages is not output for INSERT statements. As such, you do not need to set up mapping for these statements. The results of an INSERT statement are stored in an XML message in the same message format as the service component side.

### C.3 Exporting and importing at the time of distributed development

You can use an approach called *distributed development* to develop a DB adapter in multiple development environments. When you develop a DB adapter in an environment other than the master development environment, you need to export it from the non-master environment and then import it to the master development environment.

The following describes how to export and import all of the definition files for a DB adapter, and how to export and import individual DB adapter definition files.

### (1) Exporting and importing all files for a DB adapter

When you use HCSCTE in Eclipse to develop a DB adapter, the DB adapter created by this process is stored in the repository. Therefore, when developing a DB adapter by distributed development, you can export and import it by exporting and importing the repository.

For details on how to export the repository, see *3.2.2 Exporting a Repository* in the manual *Service Platform Basic Development Guide*. For details on how to import the repository, see *3.2.3 Importing a Repository* in the *Service Platform Basic Development Guide*.

### (2) Exporting and importing individual DB adapter definition files

If the Eclipse project is specified as the output destination of the definition files for the DB adapter, you can use standard Eclipse functionality to export and import the DB adapter definition files and the results of definition file generation as a batch.

For details on how to export and import this information, see the Eclipse documentation.

---

## D. Example for setting up file adapter

This appendix describes how to build and operate the system using sample business processes.

### D.1 About the sample

**! Important note**

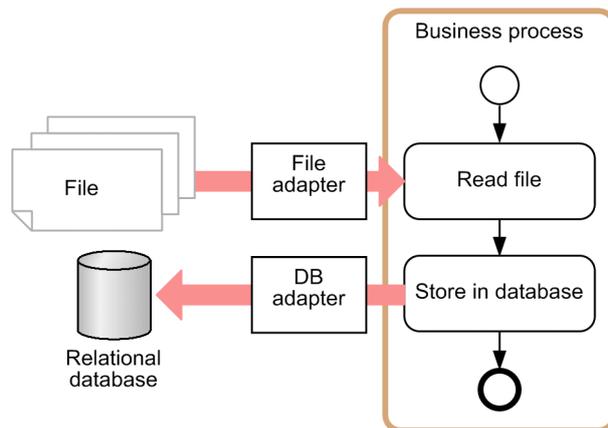
The explanation below assumes that you are executing sample business processes in the operation and execution environment (test environment) within the development environment.

The sample business processes used throughout this process use a file adapter and DB adapter to store data read from a file in the database, and to write data found in the database to a file. Different business processes are used to read and write data to and from each file format.

#### (1) Storing data read from a file in the database

The following figure shows an overview of the process:

Figure D-1: Sample business process overview (storing data read from a file in the database)



Legend:

- : Flow of business process
- ➡ : Flow of data

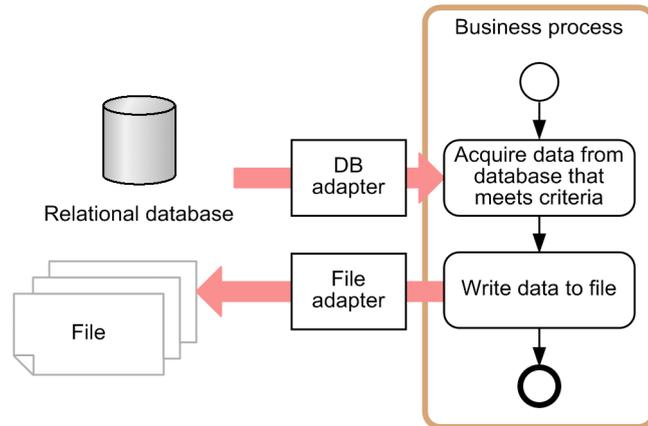
First, a file adapter is used to read data from a file transferred by FTP or other means from an existing system. This data is then stored in the database through a DB adapter. In this scenario, the business process is triggered by file transfer.

The files in the sample business process cover the definitions in the business process and execution of the business process (service requester).

#### (2) Writing database search results to a file

The following figure shows an overview of the process:

Figure D–2: Sample business process overview (writing database search results to a file)



Legend:

- ▶ : Flow of business process  
 ▶ : Flow of data

The service requester that executes the business process provides database search criteria. Based on this criteria, the data in the tables is acquired through the DB adapter. The acquired data is then written to a file using a file adapter.

The files in the sample business process cover the definitions in the business process and execution of the business process (service requester).

### (3) List of sample business processes

The following table lists the provided samples:

Table D–1: List of sample business processes

Sample business process	Description	Directory <sup>#1</sup>
Sample for reading XML data	Reads data from a file in XML format and stores it in the database.	read_xml
Sample for reading CSV data	Reads data from a file in CSV format and stores it in the database.	read_csv
Sample for reading fixed-length data <sup>#2</sup>	Reads data from a file in fixed-length format and stores it in the database.	read_fix
Sample for writing XML data	Writes database search results to a file in XML format.	write_xml
Sample for writing CSV data	Writes database search results to a file in CSV format.	write_csv
Sample for writing length tag data <sup>#3</sup>	Writes database search results to a file in length tag data format.	write_len

#1

The directory in which the files used by the sample business process are stored. These directories are located under *service-platform-installation-directory*\CSC\custom-adapter\File\sample.

#2

Because this sample can be used with the same setup as the sample for reading CSV data, it is not described in this manual.

#3

Because this sample business process can be used with the same setup as the sample business process for writing CSV data, it is not described in this manual.

## (4) Prerequisites for sample business processes

The required knowledge for using sample business processes and the prerequisites for the system environment are as follows:

### Required knowledge

You must have the following knowledge to work with sample business processes:

- Knowledge of building SOA-based systems using Service Platform (Service Architect and Service Platform)
- Fundamental knowledge of SQL and relational databases

### Required environment

The following products must have been installed and set up:

- Service Architect
- Service Platform
- HiRDB SQL Executer

The instructions in this appendix assume an environment that was built using the HCSC Easy Setup function. For details on the HCSC Easy Setup function, see *2.4 Easy Setup of the Test Environment* in the manual *Service Platform Basic Development Guide*.

When using the HCSC Easy Setup function, be sure to select **V7 compatible name** in the HCSC Easy Setup window (**Server Name** tab).

## D.2 DB adapter settings

This section describes how to set up a DB adapter for use with sample business processes.

After performing this setup process once, you can use the resulting DB adapter with multiple sample business processes. You do not need to set up a DB adapter separately for each one.

### (1) Defining the DB adapter

To define the DB adapter:

1. From the Eclipse menu, select **Window, Show View**, and then **Other**.  
The Show View dialog box appears.
2. Select **HCSC-Definer** and then **HCSCTE View**, and click **OK**.  
The Service Definition List appears in the tree view.
3. Right-click the Service Definition List in the tree view, and select **Add Service Adapter**.  
The Service adapter definition addition wizard appears.
4. From the **Service component type:** drop-down list, select **DB adapter**.
5. Click **Next**.  
The Service adapter definition addition (DB Adapter) wizard appears.
6. Enter the service name.  
Enter `CSAFF_SAMPLE_DB` as the service name.
7. Click **Finish**.  
The necessary files are created, and stored in the repository. The Service adapter definition (standard) window appears.
8. Enter settings in the Service adapter definition (standard) and Service adapter definition (details) windows.  
See *DB adapter settings* below for information about the settings.
9. Review the definition contents, and then save them by selecting **File** and then **Save** from the Eclipse menu.
10. Verify the DB adapter settings.  
Select **Verify** from the pop-up list displayed when you right-click the Service Definition List in the tree view. The results of the verification process appear in the Console View.

## DB adapter settings

The table below shows the items you need to set in the Service adapter definition (standard) window when defining the DB adapter for sample business processes. You do not need to set items that do not appear in this table.

Table D–2: Settings in Service adapter definition (standard) window (DB adapter)

Category	Item		Value to set	Requires setting
<b>Service component control information</b>	<b>Service name</b>		CSAFF_SAMPLE_DB	Y
	<b>Service ID</b>		CSAFF_DB	Y
	<b>Service type</b>		DB adapter	N
	<b>Maximum instances</b>		0	Y
	<b>Operation</b>		CSAFF_SAMPLE_DB	Y
<b>Operation information</b>	<b>Operation name</b>		CSAFF_SAMPLE_DB	N
	<b>Communication model</b>		Sync	Y
<b>Request message</b>	<b>Standard</b>	<b>Use check box</b>	Do not use (leave the check box cleared)	Y
	<b>Service component</b>	<b>Format ID</b>	SAMPLE_DB_Request	Y
		<b>Message format</b>	csaff_sample_sqlformat.xsd	Y
<b>Response message</b>	<b>Standard</b>	<b>Use check box</b>	Do not use (leave the check box cleared)	Y
	<b>Service component</b>	<b>Format ID</b>	SAMPLE_DB_Response	Y
		<b>Message format</b>	csaff_sample_sqlformat.xsd	Y

Legend:

Y: Must be set.

N: Check the content that is already displayed.

The table below shows the items you need to set in the Service adapter definition (details) window when defining the DB adapter for sample business processes. You do not need to set items that do not appear in this table.

Table D–3: Settings in Service adapter definition (details) window (DB adapter)

Category	Item		Value to set	Requires setting
<b>Service adapter control information</b>	<b>Service adapter (EJB-JAR file)</b>		cscdba_ejb.jar	N
	<b>Self-defined file</b>		csa_sql_csaff_sample.xml	Y
			cscadapter_property.xml	N

Legend:

Y: Must be set.

N: Check the content that is already displayed.

## (2) Defining DB adapter deployment

Deploy the DB adapter you defined to the server and start the DB adapter.

To deploy the DB adapter definition:

1. Right-click the Service Definition List in the tree view, and select **Deploy all services to server and start**.

## D. Example for setting up file adapter

If the DB adapter fails to start, a dialog box appears describing the nature of the error. Look up the message ID displayed in the error details and take the appropriate action.

If you are not logged in, the Account Verification window appears. In this case, perform step 2.

2. Enter `admin` in the **User ID** field and the **Password** field, and then click **OK**.

A message indicating that account verification is in progress appears, followed by a message indicating whether verification was successful.

## D.3 Sample for reading XML data

In the sample for reading XML data, a file adapter is used to read data from a file in XML format, and the data is stored in the database through a DB adapter.

The following table lists the items that are input and output:

Table D–4: Input and output items (sample for reading XML data)

Input or output item	Description
Input file of file adapter	<code>service-platform-installation-directory\CSC\custom-adapter\File\sample\read_xml\read_xml_data.xml</code>
Output table of DB adapter	<code>ADMIN.CSAFF_SAMPLE_RX#</code>

#

Do not change the schema name.

### (1) Defining the file adapter

#### (a) Adding and validating the file adapter

To add and validate a file adapter:

1. From the Eclipse menu, select **Window, Show View**, and then **Other**.  
The Show View dialog box appears.
2. Select **HCSC-Definer** and then **HCSCTE View**, and click **OK**.  
The Service Definition List appears in the tree view.
3. Right-click the Service Definition List in the tree view, and select **Add Service Adapter**.  
The Service adapter definition addition wizard appears.
4. From the **Service component type:** drop-down list, select **File adapter**.
5. Click **Next**.  
The Service adapter definition addition (File Adapter) wizard appears.
6. Enter the service name.  
Enter `CSAFF_SAMPLE_READ_XML` as the service name.
7. Click **Finish**.  
The necessary files are created, and stored in the repository. The Service adapter definition (standard) window appears.
8. Enter settings in the Service adapter definition (standard) and Service adapter definition (details) windows.  
For details on the settings you need to enter, see *File adapter settings (sample for reading XML data)*.
9. Review the content of the definition, and then save the definition by selecting **File** and then **Save** from the Eclipse menu.
10. Verify the file adapter settings.  
Select **Verify** from the pop-up list displayed when you right-click the Service Definition List in the tree view. The results of the verification process appear in Console View.

File adapter settings (sample for reading XML data)

The table below shows the items you need to set in the Service adapter definition (standard) window when defining the file adapter for the sample for reading XML data. You do not need to set items that do not appear in this table.

Table D–5: Settings in Service adapter definition (standard) window (sample for reading XML data)

Category	Item		Value to set	Requires setting
<b>Service component control information</b>	<b>Service name</b>		CSAFF_SAMPLE_READ_XML	Y
	<b>Service ID</b>		CSAFF_RX	Y
	<b>Service type</b>		<b>File adapter</b>	N
	<b>Maximum instances</b>		0	Y
	<b>Operation</b>		<b>Pattern:</b> XML file reading <b>Operation name:</b> CSAFF_SAMPLE_READ_XML	Y
<b>Operation information</b>	<b>Operation name</b>		CSAFF_SAMPLE_READ_XML	N
	<b>Communication model</b>		<b>Sync</b>	Y
<b>Request message</b>	<b>Standard</b>	<b>Use check box</b>	Do not use (leave the check box cleared)	Y
	<b>Service component</b>	<b>Format ID</b>	SAMPLE_READ_XML_Request	Y
		<b>Message format</b>	adpff_read.xsd	N
<b>Response message</b>	<b>Standard</b>	<b>Use check box</b>	Do not use (leave the check box cleared)	Y
	<b>Service component</b>	<b>Format ID</b>	SAMPLE_READ_XML_Response	Y
		<b>Message format</b>	csaff_sample_read_xml.xsd	Y

Legend:

Y: Must be set.

N: Check the content that is already displayed.

The table below shows the items you need to set in the Service adapter definition (details) window when defining the file adapter for the sample for reading XML data. You do not need to set items that do not appear in this table.

Table D–6: Settings in Service adapter definition (details) window (sample for reading XML data)

Category	Item		Value to set	Requires setting
<b>Service adapter control information</b>	<b>Service adapter (EJB-JAR file)</b>		cscmsg_adpejb.jar	N
	<b>Utility class (JAR file)</b>		adpffpc.jar	N

Legend:

N: Check the content that is already displayed.

#### (b) Creating the file adapter runtime-environment property file

To create the file adapter runtime-environment property file:

1. Open the sample file adapter runtime-environment property file in a text editor.  
The sample adapter runtime-environment property file is *service-platform-installation-directory\CSC\custom-adapter\File\sample\read\_xml\CSAFF\_RX.properties*.
2. Add the following content to the file:

## D. Example for setting up file adapter

```
fileaccess.path1 = service-platform-installation-directory\CSC\custom-adapter\File\sample\read_xml
```

### ! Important note

Use \\ as the delimiting characters between directories.

3. Save the file in the following directory: *service-platform-installation-directory\CSC\custom-adapter\File\config*

## (2) Defining the business process

The task of defining a business process involves adding a business process, defining the contents of the business process, and then validating the business process.

### (a) Adding a business process

To add a business process:

1. From the Eclipse menu, select **Window, Show View**, and then **Other**.  
The Show View dialog box appears.
2. Select **HCSC-Definer** and then **HCSCTE View**, and click **OK**.  
The Service Definition List appears in the tree view.
3. Right-click the Service Definition List in the tree view, and select **Add Business Process**.  
A dialog box appears in which you can add a business process definition.
4. Enter the name of the business process, and select whether to make its status persistent (whether to save the execution status of the business process in the database).  
Enter `CSAFF_SAMPLE_BP_READ_XML` as the business process name.  
Select **yes** for **Status Persistence**.

#### Reference note

Do not use a BPEL file.

5. Click **Finish**.  
The necessary files are created, and stored in the repository. The Define Business Process window appears.

### (b) Setting variables

Set the following variables:

- `CSAFF_DB_RequestMessage`
- `CSAFF_DB_ResponseMessage`
- `CSAFF_RequestMessage`
- `CSAFF_ResponseMessage`

#### Reference note

The sample business process does not use a correlation set.

To set the variables:

1. On the canvas of the Define Business Process window, double-click the Variable-Correlation icon.  
The List Of Variables And Correlation Sets dialog box appears.
2. Select **Variable List** from the list.
3. Enter the variable name.
4. From the **Type:** drop-down list, select **XML**.
5. Perform whichever of the following operations is relevant for the variable you are defining:

For CSAFF\_DB\_ResponseMessage, CSAFF\_RequestMessage, and CSAFF\_ResponseMessage

In the Take In Message Format dialog box displayed by clicking the **Take In** button, specify the message format you want to use. For details on the settings you need to enter, see *Variable settings (sample for reading XML data)*.

For CSAFF\_DB\_RequestMessage

Click the ... button and select the `csaff_sample_read_xml_dt.xsd` file.

6. Click **Add**.

The variable you added appears in the list of variables.

7. Repeat steps 2 to 6 for each variable.

8. Click **OK**.

The variable settings are saved and the List Of Variables And Correlation Sets dialog box closes.

Variable settings (sample for reading XML data)

The tables below show the settings for the CSAFF\_DB\_ResponseMessage, CSAFF\_RequestMessage, and CSAFF\_ResponseMessage variables. You do not need to set items that do not appear in this table.

Table D–7: Settings for CSAFF\_DB\_ResponseMessage (sample for reading XML data)

Category	Item	Value to set
<b>Service/Reception</b>	<b>Service name</b>	CSAFF_SAMPLE_DB
	<b>Reception name</b>	--
<b>Target for take in</b>	<b>Operation name</b>	CSAFF_SAMPLE_DB
	<b>Message type</b>	<b>Response message (Body)</b>
--	<b>Message format</b>	csaff_sample_sqlformat

Legend:

--: Not applicable.

Table D–8: Settings for CSAFF\_RequestMessage (sample for reading XML data)

Category	Item	Value to set
<b>Service/Reception</b>	<b>Service name</b>	CSAFF_SAMPLE_READ_XML
	<b>Reception name</b>	--
<b>Target for take in</b>	<b>Operation name</b>	CSAFF_SAMPLE_READ_XML
	<b>Message type</b>	<b>Request message (Body)</b>
--	<b>Message format</b>	adpff_read

Legend:

--: Not applicable.

Table D–9: Settings for CSAFF\_ResponseMessage (sample for reading XML data)

Category	Item	Value to set
<b>Service/Reception</b>	<b>Service name</b>	CSAFF_SAMPLE_READ_XML
	<b>Reception name</b>	--
<b>Target for take in</b>	<b>Operation name</b>	CSAFF_SAMPLE_READ_XML
	<b>Message type</b>	<b>Response message (Body)</b>
--	<b>Message format</b>	csaff_sample_read_xml

## D. Example for setting up file adapter

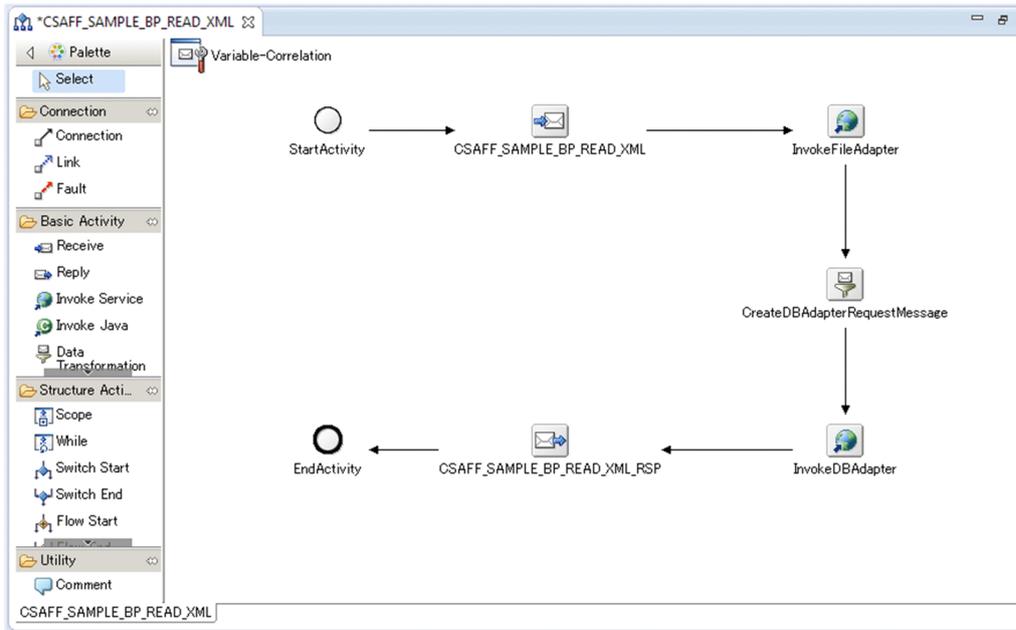
Legend:

--: Not applicable.

### (c) Setting up activities

The following describes how to arrange and define the activities in the figure below, define the data transformations, and connect the activities.

Figure D–3: Overall view of business process definition (sample for reading XML data)



#### ■ Arranging and defining activities

In the sample for reading XML data, set up the activities shown in the following table.

Table D–10: List of activities (sample for reading XML data)

Activity	Activity name	Description
Reception	CSAFF_SAMPLE_BP_READ_XML	Receives messages into the business process.
Reply	CSAFF_SAMPLE_BP_READ_XML_RSP	Issues a response from the business process.
Invoke Service	InvokeFileAadapter	Invokes the file adapter.
Invoke Service	InvokeDBAdapter	Invokes the DB adapter.
Data transformation	CreateDBAdapterRequestMessage	Transforms a response message from the file adapter to a request message for the DB adapter.

To position and set up an activity:

1. In the palette, click the basic or structure activity you want to place on the canvas.  
The activity you clicked is selected.
2. Click the appropriate location on the canvas.  
The selected activity is placed on the canvas. You can reposition the activity by dragging it with your mouse.
3. Double-click the activity on the canvas.  
A dialog box appears in which you can enter settings for the activity you double-clicked.

4. Enter the necessary information in the dialog box.  
For details on the information you need to set, see *Activity settings (sample for reading XML data)*.
5. Click **OK**.  
The dialog box for the activity closes.
6. Repeat steps 1 to 5 for each activity.

#### Activity settings (sample for reading XML data)

The tables below show the settings for the business process activities used in the sample for reading XML data. You do not need to set items that do not appear in this table.

Table D–11: Settings for CSAFF\_SAMPLE\_BP\_READ\_XML

Item	Value to set
<b>Activity name</b>	CSAFF_SAMPLE_BP_READ_XML
<b>Operation name</b>	CSAFF_SAMPLE_BP_READ_XML
<b>Body allocated variable</b>	CSAFF_RequestMessage
<b>Communication model</b>	<b>Sync</b>
<b>Instance generation</b>	<b>Yes</b>

Table D–12: Settings for CSAFF\_SAMPLE\_BP\_READ\_XML\_RSP

Item	Value to set
<b>Activity name</b>	CSAFF_SAMPLE_BP_READ_XML_RSP
<b>Operation name</b>	CSAFF_SAMPLE_BP_READ_XML
<b>Body allocated variable</b>	CSAFF_DB_ResponseMessage

Table D–13: Settings for InvokeFileAdapter (sample for reading XML data)

Item	Value to set
<b>Activity name</b>	InvokeFileAdapter
<b>Service name</b>	CSAFF_SAMPLE_READ_XML
<b>Operation name</b>	CSAFF_SAMPLE_READ_XML
<b>Communication model</b>	<b>Sync</b>
<b>Request message</b>	<b>Body allocated variable</b> CSAFF_RequestMessage
<b>Response message</b>	<b>Body allocated variable</b> CSAFF_ResponseMessage

Table D–14: Settings for InvokeDBAdapter (sample for reading XML data)

Item	Value to set
<b>Activity name</b>	InvokeDBAdapter
<b>Service name</b>	CSAFF_SAMPLE_DB
<b>Operation name</b>	CSAFF_SAMPLE_DB
<b>Communication model</b>	<b>Sync</b>
<b>Request message</b>	<b>Body allocated variable</b> CSAFF_DB_RequestMessage
<b>Response message</b>	<b>Body allocated variable</b> CSAFF_DB_ResponseMessage

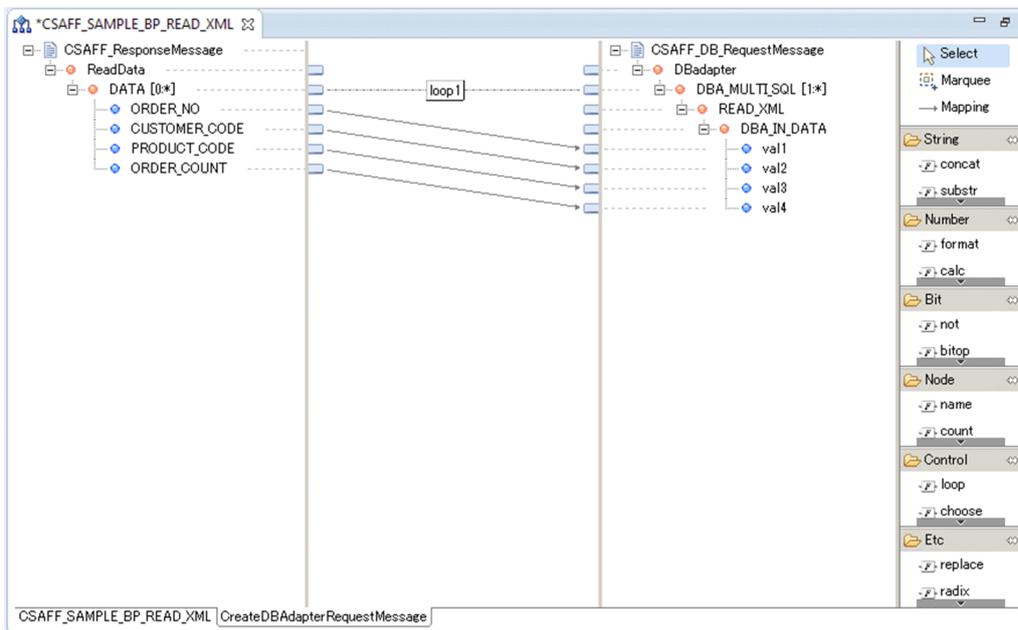
Table D–15: Settings for CreateDBAdapterRequestMessage (sample for reading XML data)

Item	Value to set
Activity name	CreateDBAdapterRequestMessage
Source Variables	CSAFF_ResponseMessage
Destination Variable	CSAFF_DB_RequestMessage
Data TransDefn File	SAMPLE_READ_XML_DT

■ Setting the data transformation definition for the CreateDBAdapterRequestMessage activity

Set the data transformation definition for the CreateDBAdapterRequestMessage activity. The following figure shows the data transformation definition you need to set:

Figure D–4: Data conversion definition window for CreateDBAdapterRequestMessage activity (sample for reading XML data)



To set the data transformation definition for the CreateDBAdapterRequestMessage activity:

1. Display the data transformation (mapping) definition screen from the CreateDBAdapterRequestMessage activity. After the data transformation (mapping) definition screen is displayed, the Select Root Element dialog box appears.
2. Select the root elements for all logical schema names. The **OK** button becomes available.
3. Click **OK**. The schemas of the selected root elements appear in tree format in the transformation-source schema tree viewer and the transformation-destination schema tree viewer.
4. Perform mapping. Mapping involves the following tasks:
  - Placing and setting functions
 The following table shows the function settings:

Table D–16: Function settings (sample for reading XML data)

Function	Function name	Value to set	
loop	loop1	<b>Select Node</b>	DATA
		<b>Set Node Condition</b>	Do not set

- Setting inter-node mapping

The following table shows the inter-node mapping settings:

Table D–17: Mapping settings (function for reading XML data)

Source node	Function	Target node
--	loop1	DBA_MULTI_SQL
ORDER_NO	--	val1
CUSTOMER_CODE	--	val2
PRODUCT_CODE	--	val3
ORDER_COUNT	--	val4

Legend:

--: Not applicable.

5. In the data transformation (mapping) definition screen, right-click an appropriate location in the transformation-source schema tree viewer, transformation-destination schema tree viewer, or mapping viewer, and select **Validate**.

Validation takes place. You can ignore any warning messages that appear.

#### ■ Connecting activities

In the sample business process for reading XML data, connect the activities as shown in the following table.

Table D–18: Activity connections (sample for reading XML data)

Connection-source activity	Connection-target activity
Start	CSAFF_SAMPLE_BP_READ_XML
CSAFF_SAMPLE_BP_READ_XML	InvokeFileAadapter
InvokeFileAadapter	CreateDBAdapterRequestMessage
CreatedDBAdapterRequestMessage	InvokeDBAdapter
InvokedDBAdapter	CSAFF_SAMPLE_BP_READ_XML_RSP
CSAFF_SAMPLE_BP_READ_XML_RSP	Finish

To connect the activities:

1. On the palette, click **Connection**.

Connection mode is selected.

When you align your cursor with an activity that can be specified as a connection source, the background color of the activity changes.

2. On the canvas, click the activity that you want to serve as the connection source.

The activity is set as the connection source.

If you then align your cursor with another activity, the background color of the activity changes if it can be specified as a connection target.

3. On the canvas, click the activity that you want to serve as the connection target.

The activity you selected as the connection source is connected to the activity you selected as the connection target.

#### (d) Saving and validating the business process

To save and validate the business process:

1. Review the definition contents, and then save the business process definition by selecting **File** and then **Save** from the Eclipse menu.  
In properties view, set CSAFF\_B1 as the service ID of the business process.
2. Validate the business process settings.  
Select **Verify** from the pop-up list displayed when you right-click the Service Definition List in the tree view. The results of the validation process appear in Console View.

### (3) Distributing the file adapter and business process deployment definitions

Deploy the file adapter and business process and start them on the server.

To distribute the file adapter and business process deployment definitions:

1. Right-click the Service Definition List in the tree view, and select **Deploy all services to server and start**.  
If the file adapter or business process fails to start, a dialog box appears describing the nature of the error. Look up the message ID displayed in the error details and take the appropriate action.  
If you are not logged in, the Account Verification window appears. In this case, perform step 2.
2. Enter `admin` in the **User ID** field and the **Password** field, and then click **OK**.  
A message indicating that account verification is in progress appears, followed by a message indicating whether verification was successful.

### (4) Creating tables

Create the table the sample business process will access in the database.

Open HiRDB SQL Executer and execute the following SQL statement:

```
CREATE TABLE CSAFF_SAMPLE_RX  
(ORDER_NO INT,  
CUSTOMER_CODE CHAR(5),  
PRODUCT_CODE CHAR(4),  
ORDER_COUNT INT);
```

#### ! Important note

Use ADMIN as the user ID.

### (5) Creating request messages

The following describes how to create a request message by editing the sample request message used when executing the business process.

1. Open the sample request message file in a text editor.  
The sample request message file is *service-platform-installation-directory\CSC\custom-adapter\File\sample\read\_xml\read\_xml\_message.xml*.
2. Add the part in italics below:  

```
<?xml version="1.0" encoding="UTF-8"?>  
<adpff:ReadRequest xmlns:adpff="http://  
FlatFiles.cstmadv.csc.soft.Hitachi.co.jp/ReadRequest">  
  <fileheader>  
    <filename>service-platform-installation-directory\CSC\custom-adapter  
File\sample\read_xml\read_xml_data.xml</filename>  
    <filemode>all</filemode>  
  </fileheader>  
</adpff:ReadRequest>
```
3. Save the file.

## (6) Executing the business process

To execute the sample, start the business process from the service requester.

The flow of service requester processing is as follows:

1. Service requester execution  
The request message of the business process is used as the request message of the file adapter without transformation.
2. File adapter execution  
The file specified in the request message is read.
3. DB adapter message creation  
The response message from the file adapter is transformed to a request message for the DB adapter.
4. DB adapter execution  
The file specified in the request message is registered in the database.
5. Service requester termination  
The response message from the DB adapter is used as the response message of the business process without transformation.

To execute the service requester:

1. Navigate to *service-platform-installation-directory*\CSC\custom-adapter\File\sample\bin.
2. Execute `request.bat`.  
Execute `request.bat` in the following format:  

```
request read_xml service-platform-installation-directory\CSC\custom-adapter
\File\sample\read_xml\read_xml_message.xml
```

---

### Reference note

You can check the contents of the table by starting HiRDB SQL Executer and executing the following SQL statement (using ADMIN as the user ID):

```
SELECT * FROM CSAFF_SAMPLE_RX;
```

---

## D.4 Sample for reading CSV data

In the sample for reading CSV data, a file adapter is used to read data from a file in CSV format, and the data is then stored in the database through a DB adapter. This sample business process selectively acquires data for products whose product code is 1001 in the input file, and stores the data in the database. The process of finding the appropriate product codes is realized by the data transformation functionality.

The following table lists the items that are input and output:

Table D–19: Input and output items (sample for reading CSV data)

Input or output item	Content
Input file of file adapter	<i>service-platform-installation-directory</i> \CSC\custom-adapter\File\sample\read_csv\read_csv_data.csv
Output table of DB adapter	ADMIN.CSAFF_SAMPLE_RC <sup>#</sup>

#

Do not change the schema name.

### (1) Defining the file adapter

#### (a) Adding and validating the file adapter

To add and validate a file adapter:

D. Example for setting up file adapter

1. From the Eclipse menu, select **Window, Show View**, and then **Other**.  
The Show View dialog box appears.
2. Select **HCSC-Definer** and then **HCSCTE View**, and click **OK**.  
The Service Definition List appears in the tree view.
3. Right-click the Service Definition List in the tree view, and select **Add Service Adapter**.  
The Service adapter definition addition wizard appears.
4. From the **Service component type:** drop-down list, select **File adapter**.
5. Click **Next**.  
The Service adapter definition addition (File Adapter) wizard appears.
6. Enter the service name.  
Enter `CSAFF_SAMPLE_READ_CSV` as the service name.
7. Click **Finish**.  
The necessary files are created, and stored in the repository. The Service adapter definition (standard) window appears.
8. Enter settings in the Service adapter definition (standard) and Service adapter definition (details) windows.  
For details on the settings you need to enter, see *File adapter settings (sample for reading CSV data)*.
9. Review the definition contents, and then save file adapter settings by selecting **File** and then **Save** from the Eclipse menu.
10. Verify the file adapter settings.  
Select **Verify** from the pop-up list displayed when you right-click the Service Definition List in the tree view. The results of the validation process appear in Console View.

File adapter settings (sample for reading CSV data)

The table below shows the items you need to set in the Service adapter definition (standard) window when defining the file adapter for the sample for reading CSV data. You do not need to set items that do not appear in this table.

Table D–20: Settings in Service adapter definition (standard) window (sample for reading CSV data)

Category	Item		Value to set	Requires setting
<b>Service component control information</b>	<b>Service name</b>		<code>CSAFF_SAMPLE_READ_CSV</code>	Y
	<b>Service ID</b>		<code>CSAFF_RC</code>	Y
	<b>Service type</b>		File adapter	N
	<b>Maximum instances</b>		0	Y
	<b>Operation</b>		<b>Pattern:</b> Binary file reading <b>Operation name:</b> <code>CSAFF_SAMPLE_READ_CSV</code>	Y
<b>Operation information</b>	<b>Operation name</b>		<code>CSAFF_SAMPLE_READ_CSV</code>	N
	<b>Communication model</b>		<b>Sync</b>	Y
<b>Request message</b>	<b>Standard</b>	<b>Use check box</b>	Do not use (leave the check box cleared)	Y
	<b>Service component</b>	<b>Format ID</b>	<code>SAMPLE_READ_CSV_Request</code>	Y
<b>Message format</b>		<code>adpff_read.xsd</code>	N	
<b>Response message</b>	<b>Standard</b>	<b>Use check box</b>	Select this check box.	Y
	<b>Format ID</b>	<code>SAMPLE_READ_CSV_Standard</code>	Y	

Category	Item		Value to set	Requires setting
Response message	Standard	Message format	csaff_sample_read_csv_standard.xsd	Y
	Service component	Format ID	SAMPLE_READ_CSV_Response	Y
		Message format	csaff_sample_read_csv.fdx	Y
	Data TransDefn File		SAMPLE_READ_CSV_DT	Y

Legend:

Y: Must be set.

N: Check the content that is already displayed.

The table below shows the items you need to set in the Service adapter definition (details) window when defining the file adapter for the sample for reading CSV data. You do not need to set items that do not appear in this table.

Table D–21: Settings in Service adapter definition (details) window (sample for reading CSV data)

Category	Item	Value to set	Requires setting
Service adapter control information	Service adapter (EJB-JAR file)	cscmsg_adpejb.jar	N
	Utility class (JAR file)	adpffpc.jar	N

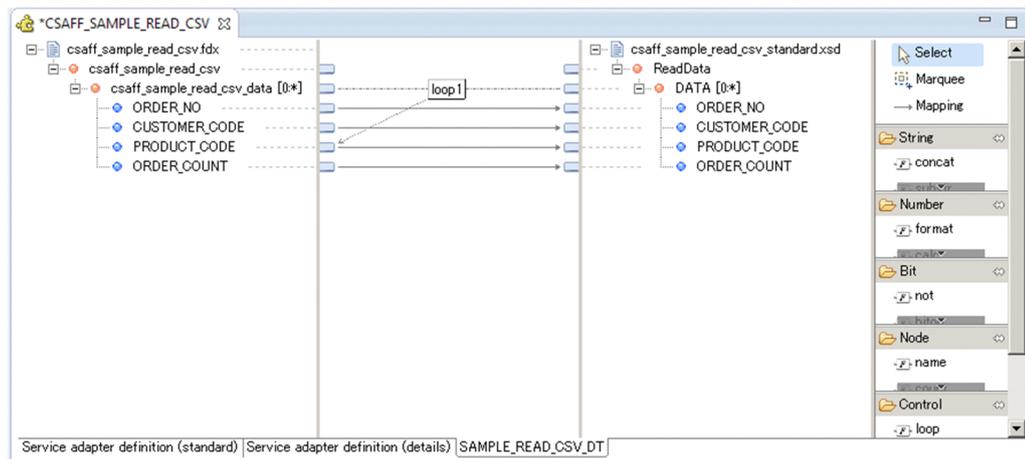
Legend:

N: Check the content that is already displayed.

### (b) Defining data transformation

Set the data transformation definition. The following figure shows the data transformation definition you need to set:

Figure D–5: Data-conversion definition screen for file adapter (sample for reading CSV data)



To define this data transformation:

1. Display the data transformation (mapping) definition screen in the Adapter Definition screen.  
After displaying the data transformation (mapping) definition screen, the Select Root Element dialog box appears.
2. Select the root elements for all logical schema names.  
The **OK** button becomes available.
3. Click **OK**.  
The schemas of the selected root elements appear in tree format in the transformation-source schema tree viewer and the transformation-destination schema tree viewer.

#### D. Example for setting up file adapter

##### 4. Perform mapping.

Mapping involves the following tasks:

- Placing and setting functions

The following table shows the function settings:

Table D–22: Function settings (sample for reading CSV data - File adapter)

Function	Function name	Value to set	
loop	loop1	<b>Select Node</b>	csaff_sample_read_csv_data
		<b>Set Node Condition</b>	See table below

Table D–23: Settings for loop1

Category		Value to set
<b>Node</b>		csaff_sample_read_csv_data
<b>Condition</b>	<b>Condition</b>	<b>Condition format</b>
	<b>Left side</b>	<b>Node or function</b> (PRODUCT_CODE)
	<b>Operation</b>	=
	<b>Right side</b>	<b>Value:</b> 1001

- Setting inter-node mapping

The following table shows the inter-node mapping settings:

Table D–24: Mapping settings (sample for reading CSV data - File adapter)

Source node	Function	Target node
--	loop1	DATA
ORDER_NO	--	ORDER_NO
CUSTOMER_CODE	--	CUSTOMER_CODE
PRODUCT_CODE	--	PRODUCT_CODE
ORDER_COUNT	--	ORDER_COUNT

Legend:

--: Not applicable.

5. In the data transformation (mapping) definition screen, right-click an appropriate location in the transformation-source schema tree viewer, transformation-destination schema tree viewer, or mapping viewer, and select **Validate**.

Validation takes place.

#### (c) Creating the file adapter runtime-environment property file

To create the file adapter runtime-environment property file:

1. Open the sample file adapter runtime-environment property file in a text editor.

The sample file adapter runtime-environment property file is *service-platform-installation-directory\CSC\custom-adapter\File\sample\read\_csv\CSAFF\_RC.properties*.

2. Add the following content to the file:

```
fileaccess.path1 = service-platform-installation-directory\CSC\custom-adapter\File\sample\read_csv
```

#### ! Important note

Use \\ as the delimiting characters between directories.

3. Save the file in the following directory: *service-platform-installation-directory*\CSC\custom-adapter\File\config

## (2) Defining the business process

The task of defining a business process involves adding a business process, defining the contents of the business process, and then validating the business process.

### (a) Adding a business process

To add a business process:

1. From the Eclipse menu, select **Window, Show View**, and then **Other**.  
The Show View dialog box appears.
2. Select **HCSC-Definer** and then **HCSCTE View**, and click **OK**.  
The Service Definition List appears in the tree view.
3. Right-click the Service Definition List in the tree view, and select **Add Business Process**.  
A dialog box appears in which you can add a business process definition.
4. Enter the name of the business process, and select whether to make its status persistent (whether to save the execution status of the business process in the database).  
Enter `CSAFF_SAMPLE_BP_READ_CSV` as the business process name.  
Select *yes* for **Status Persistence**.

---

#### Reference note

Do not use a BPEL file.

---

5. Click **Finish**.  
The necessary files are created, and stored in the repository. The Define Business Process window appears.

### (b) Setting variables

Set the following variables:

- `CSAFF_DB_RequestMessage`
- `CSAFF_DB_ResponseMessage`
- `CSAFF_RequestMessage`
- `CSAFF_ResponseMessage`

---

#### Reference note

The sample business process does not use a correlation set.

---

To set the variables:

1. Double-click the Variable-Correlation icon on the canvas of the Define Business Process window.  
The List Of Variables And Correlation Sets dialog box appears.
2. Select **Variable List** from the list.
3. Enter the variable name.
4. From the **Type:** drop-down list, select **XML**.
5. Perform whichever of the following operations is relevant for the variable you are defining:
  - For `CSAFF_DB_ResponseMessage`, `CSAFF_RequestMessage`, and `CSAFF_ResponseMessage`  
In the Take In Message Format dialog box displayed by clicking the **Take In** button, specify the message format you want to use. For details on the settings you need to enter, see *Variable settings (sample for reading CSV data)*.
  - For `CSAFF_DB_RequestMessage`  
Click the **...** button and select the `csaff_sample_read_csv_dt.xsd` file.

D. Example for setting up file adapter

6. Click **Add**.

The variable you added appears in the list of variables.

7. Repeat steps 2 to 6 for each variable.

8. Click **OK**.

The variable settings are saved and the List Of Variables And Correlation Sets dialog box closes.

Variable settings (sample for reading CSV data)

The tables below show the settings for the CSAFF\_DB\_ResponseMessage, CSAFF\_RequestMessage, and CSAFF\_ResponseMessage variables. You do not need to set items that do not appear in this table.

Table D–25: Settings for CSAFF\_DB\_ResponseMessage (sample for reading CSV data)

Category	Item	Value to set
<b>Service/Reception</b>	<b>Service name</b>	CSAFF_SAMPLE_DB
	<b>Reception name</b>	--
<b>Target for take in</b>	<b>Operation name</b>	CSAFF_SAMPLE_DB
	<b>Message type</b>	<b>Response message (Body)</b>
--	<b>Message format</b>	csaff_sample_sqlformat

Legend:  
--: Not applicable.

Table D–26: Settings for CSAFF\_RequestMessage (sample for reading CSV data)

Category	Item	Value to set
<b>Service/Reception</b>	<b>Service name</b>	CSAFF_SAMPLE_READ_CSV
	<b>Reception name</b>	--
<b>Target for take in</b>	<b>Operation name</b>	CSAFF_SAMPLE_READ_CSV
	<b>Message type</b>	<b>Request message (Body)</b>
--	<b>Message format</b>	adpff_read

Legend:  
--: Not applicable.

Table D–27: Settings for CSAFF\_ResponseMessage (sample for reading CSV data)

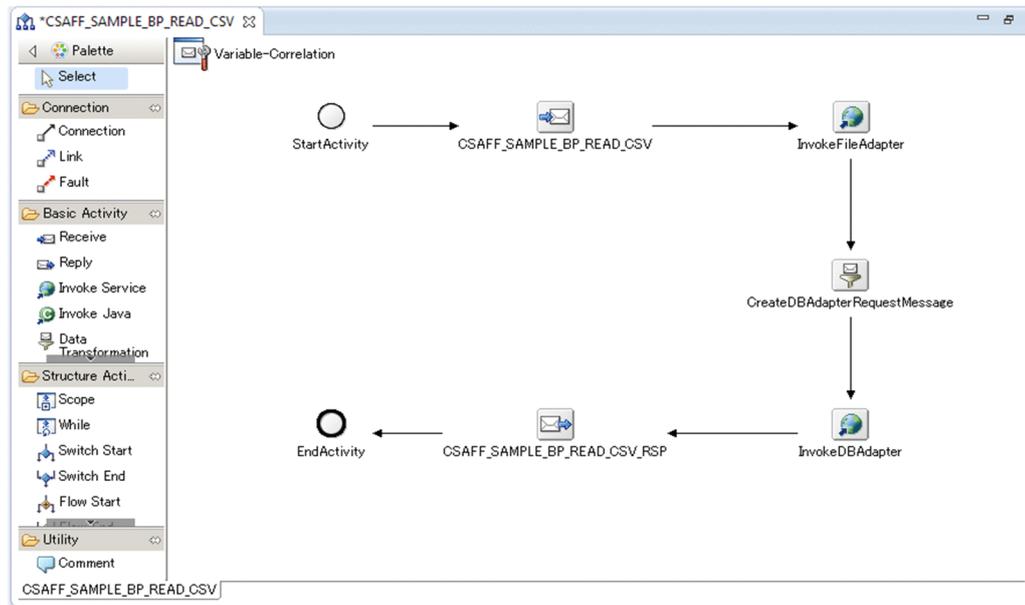
Category	Item	Value to set
<b>Service/Reception</b>	<b>Service name</b>	CSAFF_SAMPLE_READ_CSV
	<b>Reception name</b>	--
<b>Target for take in</b>	<b>Operation name</b>	CSAFF_SAMPLE_READ_CSV
	<b>Message type</b>	<b>Response message (Body)</b>
--	<b>Message format</b>	csaff_sample_standard

Legend:  
--: Not applicable.

(c) Setting up activities

The following describes how to arrange and define the activities in the following figure, define the data transformations, and connect the activities.

Figure D–6: Overall view of business process definition (sample for reading CSV data)



#### ■ Arranging and defining activities

In the sample for reading CSV data, set up the activities shown in the following table.

Table D–28: List of activities (sample for reading CSV data)

Activity	Activity name	Description
Reception	CSAFF_SAMPLE_BP_READ_CSV	Receives messages into the business process.
Response	CSAFF_SAMPLE_BP_READ_CSV_RSP	Issues a response from the business process.
Service invocation	InvokeFileAadapter	Invokes the file adapter.
Service invocation	InvokeDBAdapter	Invokes the DB adapter.
Data transformation	CreateDBAdapterRequestMessage	Transforms a response message from the file adapter to a request message for the DB adapter.

To position and set up an activity:

- In the palette, click the basic or structure activity you want to place on the canvas.  
The activity you clicked is selected.
- Click the appropriate location on the canvas.  
The selected activity is placed on the canvas. You can reposition the activity by dragging it with your mouse.
- Double-click the activity on the canvas.  
A dialog box appears in which you can enter settings for the activity you double-clicked.
- Enter the necessary information in the dialog box.  
For details on the information you need to set, see *Activity settings (sample for reading CSV data)*.
- Click **OK**.  
The dialog box for the activity closes.
- Repeat steps 1 to 5 for each activity.

D. Example for setting up file adapter

Activity settings (sample for reading CSV data)

The tables below show the settings for the business process activities used in the sample for reading CSV data. You do not need to set items that do not appear in this table.

Table D–29: Settings for CSAFF\_SAMPLE\_BP\_READ\_CSV

Item	Value to set
<b>Activity name</b>	CSAFF_SAMPLE_BP_READ_CSV
<b>Operation name</b>	CSAFF_SAMPLE_BP_READ_CSV
<b>Body allocated variable</b>	CSAFF_RequestMessage
<b>Communication model</b>	<b>Sync</b>
<b>Instance generation</b>	<b>Yes</b>

Table D–30: Settings for CSAFF\_SAMPLE\_BP\_READ\_CSV\_RSP

Item	Value to set
<b>Activity name</b>	CSAFF_SAMPLE_BP_READ_CSV_RSP
<b>Operation name</b>	CSAFF_SAMPLE_BP_READ_CSV
<b>Body allocated variable</b>	CSAFF_DB_ResponseMessage

Table D–31: Settings for InvokeFileAadapter (sample for reading CSV data)

Item	Value to set
<b>Activity name</b>	InvokeFileAadapter
<b>Service name</b>	CSAFF_SAMPLE_READ_CSV
<b>Operation name</b>	CSAFF_SAMPLE_READ_CSV
<b>Communication model</b>	<b>Sync</b>
<b>Request message</b>	<b>Body allocated variable</b> CSAFF_RequestMessage
<b>Response message</b>	<b>Body allocated variable</b> CSAFF_ResponseMessage

Table D–32: Settings for InvokeDBAdapter (sample for reading CSV data)

Item	Value to set
<b>Activity name</b>	InvokeDBAdapter
<b>Service name</b>	CSAFF_SAMPLE_DB
<b>Operation name</b>	CSAFF_SAMPLE_DB
<b>Communication model</b>	<b>Sync</b>
<b>Request message</b>	<b>Body allocated variable</b> CSAFF_DB_RequestMessage
<b>Response message</b>	<b>Body allocated variable</b> CSAFF_DB_ResponseMessage

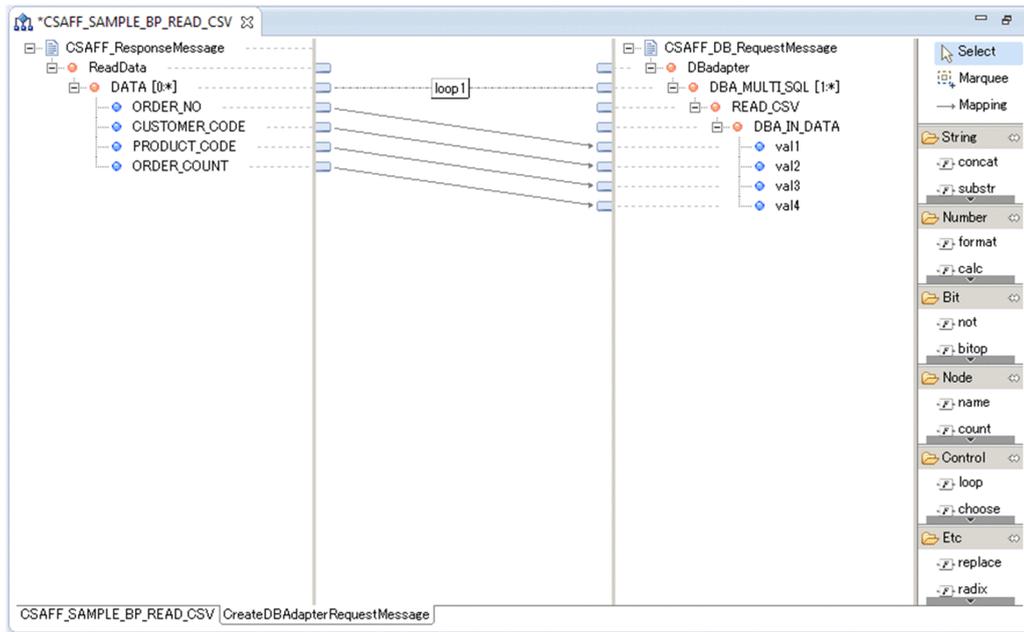
Table D–33: Settings for CreateDBAdapterRequestMessage (sample for reading CSV data)

Item	Value to set
<b>Activity name</b>	CreateDBAdapterRequestMessage
<b>Source Variables</b>	CSAFF_ResponseMessage
<b>Destination Variable</b>	CSAFF_DB_RequestMessage
<b>Data TransDefn File</b>	SAMPLE_READ_CSV_DT

■ Setting the data transformation definition for the CreateDBAdapterRequestMessage activity

Set the data transformation definition for the CreateDBAdapterRequestMessage activity. The following figure shows the data transformation definition you need to set:

Figure D–7: Data conversion definition window for CreateDBAdapterRequestMessage activity (sample for reading CSV data)



To set the data transformation definition for the CreateDBAdapterRequestMessage activity:

1. Display the data transformation (mapping) definition screen from the CreateDBAdapterRequestMessage activity. After displaying the data transformation (mapping) definition screen, the Select Root Element dialog box appears.
2. Select the root elements for all logical schema names. The **OK** button becomes available.
3. Click **OK**. The schemas of the selected root elements appear in tree format in the transformation-source schema tree viewer and the transformation-destination schema tree viewer.
4. Perform mapping. Mapping involves the following tasks:
  - Placing and setting functions

The following table shows the function settings:

Table D–34: Function settings (sample for reading CSV data)

Function	Function name	Value to set	
loop	loop1	<b>Select Node</b>	DATA
		<b>Set Node Condition</b>	Do not set

- Setting inter-node mapping

The following table shows the inter-node mapping settings:

Table D–35: Mapping settings (sample for reading CSV data - Business process)

Source node	Function	Target node
--	loop1	DBA_MULTI_SQL
ORDER_NO	--	val1

#### D. Example for setting up file adapter

Source node	Function	Target node
CUSTOMER_CODE	--	val2
PRODUCT_CODE	--	val3
ORDER_COUNT	--	val4

Legend:

--: Not applicable.

- In the data transformation (mapping) definition screen, right-click an appropriate location in the transformation-source schema tree viewer, transformation-destination schema tree viewer, or mapping viewer, and select **Validate**.

Validation takes place. You can ignore any warning messages that appear.

#### ■ Connecting activities

In the sample business process for reading CSV data, connect the activities as shown in the following table.

Table D–36: Activity connections (sample for reading CSV data)

Connection-source activity	Connection-target activity
Start	CSAFF_SAMPLE_BP_READ_CSV
CSAFF_SAMPLE_BP_READ_CSV	InvokeFileAadapter
InvokeFileAadapter	CreateDBAdapterRequestMessage
CreateDBAdapterRequestMessage	InvokeDBAdapter
InvokeDBAdapter	CSAFF_SAMPLE_BP_READ_CSV_RSP
CSAFF_SAMPLE_BP_READ_CSV_RSP	Finish

To connect the activities:

- On the palette, click **Connection**.  
Connection mode is selected.  
When you align your cursor with an activity that can be specified as a connection source, the background color of the activity changes.
- On the canvas, click the activity that you want to serve as the connection source.  
The activity is set as the connection source.  
If you then align your cursor with another activity, the background color of the activity changes if it can be specified as a connection target.
- On the canvas, click the activity that you want to serve as the connection target.  
The activity you selected as the connection source is connected to the activity you selected as the connection target.

#### (d) Saving and validating the business process

To save and validate the business process:

- Review the definition contents, and then save the business process definition by selecting **File** and then **Save** from the Eclipse menu.  
In properties view, set CSAFF\_B2 as the service ID of the business process.
- Validate the business process settings.  
Select **Verify** from the pop-up list displayed when you right-click the Service Definition List in the tree view. The results of the validation process appear in Console View.

### (3) Distributing the file adapter and business process deployment definitions

Deploy the file adapter and business process and start them on the server.

To distribute the file adapter and business process deployment definitions:

1. Right-click the Service Definition List in the tree view, and select **Deploy all services to server and start**.  
If the file adapter or business process fails to start, a dialog box appears describing the nature of the error. Look up the message ID displayed in the error details and take the appropriate action.  
If you are not logged in, the Account Verification window appears. In this case, perform step 2.
2. Enter `admin` in the **User ID** field and the **Password** field, and then click **OK**.  
A message indicating that account verification is in progress appears, followed by a message indicating whether it was successful.

### (4) Creating tables

Create the table the sample business process will access in the database.

Open HiRDB SQL Executer and execute the following SQL statement:

```
CREATE TABLE CSAFF_SAMPLE_RC
(ORDER_NO INT,
CUSTOMER_CODE CHAR(5),
PRODUCT_CODE CHAR(4),
ORDER_COUNT INT);
```

#### ! Important note

Use ADMIN as the user ID.

### (5) Creating request messages

The following describes how to create a request message by editing the sample request message used when executing the business process.

1. Open the sample request message file in a text editor.  
The sample request message file is `service-platform-installation-directory\CSC\custom-adapter\File\sample\read_csv\read_csv_message.xml`.
2. Add the part in italics below:

```
<?xml version="1.0" encoding="UTF-8"?>
<adpff:ReadRequest xmlns:adpff="http://
FlatFiles.cstmadp.csc.soft.Hitachi.co.jp/ReadRequest">
  <fileheader>
    <filename>service-platform-installation-directory\CSC\custom-adapter
\File\sample\read_csv\read_csv_data.csv</filename>
    <filemode>all</filemode>
  </fileheader>
</adpff:ReadRequest>
```
3. Save the file.

### (6) Executing the business process

To execute the sample, start the business process from the service requester.

The flow of service requester processing is as follows:

1. Service requester execution  
The request message of the business process is used as the request message of the file adapter without transformation.
2. File adapter execution  
The file specified in the request message is read.
3. DB adapter message creation

## D. Example for setting up file adapter

The response message from the file adapter is transformed to a request message for the DB adapter.

### 4. DB adapter execution

The file specified in the request message is registered in the database.

### 5. Service requester termination

The response message from the DB adapter is used as the response message of the business process without transformation.

To execute the service requester:

1. Navigate to *service-platform-installation-directory*\CSC\custom-adapter\File\sample\bin.

2. Execute `request.bat`.

Execute `request.bat` in the following format:

```
request read csv service-platform-installation-directory\CSC\custom-adapter
\File\sample\read_csv\read_csv_message.xml
```

### Reference note

You can check the contents of the table by starting HiRDB SQL Executer and executing the following SQL statement (using ADMIN as the user ID):

```
SELECT * FROM CSAFF_SAMPLE_RC;
```

## D.5 Sample for writing XML data

The sample for writing XML data uses a file adapter to write data retrieved by a DB adapter to a file in XML format.

The following table lists the items that are input and output:

Table D–37: Input and output items (sample for writing XML data)

Input or output item	Content
Output file of file adapter	<i>service-platform-installation-directory</i> \CSC\custom-adapter\File\sample\write_xml\write_xml_data.xml
Input table of DB adapter	ADMIN.CSAFF_SAMPLE_WX#

#

Do not change the schema name.

### (1) Defining the file adapter

#### (a) Adding and validating the file adapter

To add and validate a file adapter:

1. From the Eclipse menu, select **Window**, **Show View**, and then **Other**.  
The Show View dialog box appears.
2. Select **HCSC-Definer** and then **HCSCTE View**, and click **OK**.  
The Service Definition List appears in the tree view.
3. Right-click the Service Definition List in the tree view, and select **Add Service Adapter**.  
The Service adapter definition addition wizard appears.
4. From the **Service component type:** drop-down list, select **File adapter**.
5. Click **Next**.  
The Service adapter definition addition (File Adapter) wizard appears.
6. Enter the service name.  
Enter `CSAFF_SAMPLE_WRITE_XML` as the service name.

7. Click **Finish**.

The necessary files are created, and stored in the repository. The Service adapter definition (standard) window appears.

## 8. Enter settings in the Service adapter definition (standard) and Service adapter definition (details) windows.

For details on the settings you need to enter, see *File adapter settings (sample for writing XML data)*.

9. Review the definition contents, and then save the service adapter definition by selecting **File** and then **Save** from the Eclipse menu.

## 10. Verify the file adapter settings.

Select **Verify** from the pop-up list displayed when you right-click the Service Definition List in the tree view. The results of the validation process appear in Console View.

## File adapter settings (sample for writing XML data)

The table below shows the items you need to set in the Service adapter definition (standard) window when defining the file adapter for the sample for writing XML data. You do not need to set items that do not appear in this table.

Table D–38: Settings in Service adapter definition (standard) window (sample for writing XML data)

Category	Item		Value to set	Requires setting
<b>Service component control information</b>	<b>Service name</b>		CSAFF_SAMPLE_WRITE_XML	Y
	<b>Service ID</b>		CSAFF_WX	Y
	<b>Service type</b>		File adapter	N
	<b>Maximum instances</b>		0	Y
	<b>Operation</b>		<b>Pattern:</b> XML file writing <b>Operation name:</b> CSAFF_SAMPLE_WRITE_XML	Y
<b>Operation information</b>	<b>Operation name</b>		CSAFF_SAMPLE_WRITE_XML	N
	<b>Communication model</b>		Sync	Y
<b>Request message</b>	<b>Standard</b>	Use check box	Do not use (leave the check box cleared)	Y
	<b>Service component</b>	<b>Format ID</b>	SAMPLE_WRITE_XML_Request	Y
		<b>Message format</b>	csaff_sample_write_xml_dt.xsd	Y
<b>Response message</b>	<b>Standard</b>	Use check box	Do not use (leave the check box cleared)	Y
	<b>Service component</b>	<b>Format ID</b>	SAMPLE_WRITE_XML_Response	Y
		<b>Message format</b>	adpff_result.xsd	N

Legend:

Y: Must be set.

N: Check the content that is already displayed.

The table below shows the items you need to set in the Service adapter definition (details) window when defining the file adapter for the sample for writing XML data. You do not need to set items that do not appear in this table.

Table D–39: Settings in Service adapter definition (details) window (sample for writing XML data)

Category	Item	Value to set	Requires setting
<b>Service adapter control information</b>	<b>Service adapter (EJB-JAR file)</b>	cscmsg_adpejb.jar	N

## D. Example for setting up file adapter

Category	Item	Value to set	Requires setting
<b>Service adapter control information</b>	<b>Utility class (JAR file)</b>	adpffpc.jar	N

Legend:

N: Check the content that is already displayed.

### (b) Creating the file adapter runtime-environment property file

To create the file adapter runtime-environment property file:

1. Open the sample file adapter runtime-environment property file in a text editor.

The sample file adapter runtime-environment property file is *service-platform-installation-directory\CSC\custom-adapter\File\sample\write\_xml\CSAFF\_WX.properties*.

2. Add the following content to the file:

```
fileaccess.path1 = service-platform-installation-directory\CSC\custom-adapter\File\sample\write_xml
```

#### ! Important note

Use `\\` as the delimiting characters between directories.

3. Save the file in the following directory: *service-platform-installation-directory\CSC\custom-adapter\File\config*

## (2) Defining the business process

The task of defining a business process involves adding a business process, defining the contents of the business process, and then validating the business process.

### (a) Adding a business process

To add a business process:

1. From the Eclipse menu, select **Window, Show View, and then Other**.  
The Show View dialog box appears.
2. Select **HCSC-Definer** and then **HCSCTE View**, and click **OK**.  
The Service Definition List appears in the tree view.
3. Right-click the Service Definition List in the tree view, and select **Add Business Process**.  
A dialog box appears in which you can add a business process definition.
4. Enter the name of the business process, and select whether to make its status persistent (whether to save the execution status of the business process in the database).  
Enter `CSAFF_SAMPLE_BP_WRITE_XML` as the business process name.  
Select **yes** for **Status Persistence**.

#### Reference note

Do not use a BPEL file.

5. Click **Finish**.

The necessary files are created, and stored in the repository. The Define Business Process window appears.

### (b) Setting variables

Set the following variables:

- `CSAFF_BP_RequestMessage`
- `CSAFF_DB_RequestMessage`
- `CSAFF_DB_ResponseMessage`

- CSAFF\_RequestMessage
- CSAFF\_ResponseMessage

---

**Reference note**

The sample business process does not use a correlation set.

---

To set the variables:

1. On the canvas of the Define Business Process window, double-click the Variable-Correlation icon.  
The List Of Variables And Correlation Sets dialog box appears.
2. Select **Variable List** from the list.
3. Enter the variable name.
4. From the **Type:** drop-down list, select **XML**.
5. Perform whichever of the following operations is relevant for the variable you are defining:
  - For CSAFF\_DB\_ResponseMessage, CSAFF\_RequestMessage, and CSAFF\_ResponseMessage  
In the Take In Message Format dialog box displayed by clicking the **Take In** button, specify the message format you want to use. For details on the settings you need to enter, see *Variable settings (sample for writing XML data)*.
  - For CSAFF\_BP\_RequestMessage  
Click the ... button and select the `csaff_sample_write_xml.xsd` file.
  - For CSAFF\_DB\_RequestMessage  
Click the ... button and select the `csaff_sample_write_xml_db.xsd` file.
6. Click **Add**.  
The variable you added appears in the list of variables.
7. Repeat steps 2 to 6 for each variable.
8. Click **OK**.  
The variable settings are saved and the List Of Variables And Correlation Sets dialog box closes.

**Variable settings (sample for writing XML data)**

The tables below show the settings for the CSAFF\_DB\_ResponseMessage, CSAFF\_RequestMessage, and CSAFF\_ResponseMessage variables. You do not need to set items that do not appear in this table.

Table D–40: Settings for CSAFF\_DB\_ResponseMessage (sample for writing XML data)

Category	Item	Value to set
<b>Service/Reception</b>	<b>Service name</b>	CSAFF_SAMPLE_DB
	<b>Reception name</b>	--
<b>Target for take in</b>	<b>Operation name</b>	CSAFF_SAMPLE_DB
	<b>Message type</b>	<b>Response message (Body)</b>
--	<b>Message format</b>	csaff_sample_sqlformat

Legend:

--: Not applicable.

Table D–41: Settings for CSAFF\_RequestMessage (sample for writing XML data)

Category	Item	Value to set
<b>Service/Reception</b>	<b>Service name</b>	CSAFF_SAMPLE_WRITE_XML
	<b>Reception name</b>	--
<b>Target for take in</b>	<b>Operation name</b>	CSAFF_SAMPLE_WRITE_XML

D. Example for setting up file adapter

Category	Item	Value to set
<b>Target for take in</b>	<b>Message type</b>	<b>Request message (Body)</b>
--	<b>Message format</b>	csaff_sample_write_xml_dt

Legend:  
 --: Not applicable.

Table D–42: Settings for CSAFF\_ResponseMessage (sample for writing XML data)

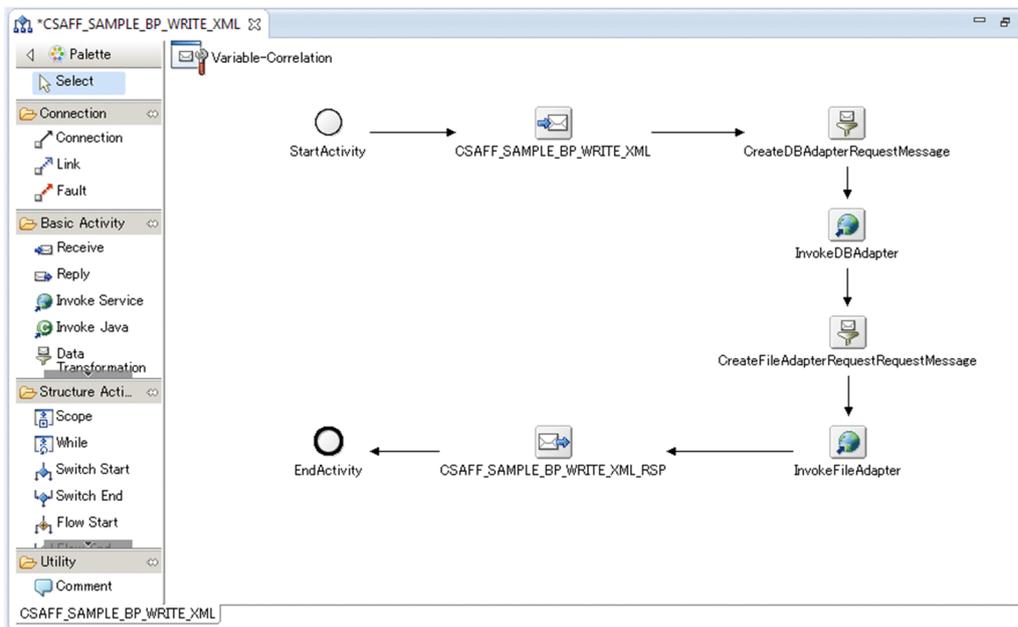
Category	Item	Value to set
<b>Service/Reception</b>	<b>Service name</b>	CSAFF_SAMPLE_WRITE_XML
	<b>Reception name</b>	--
<b>Target for take in</b>	<b>Operation name</b>	CSAFF_SAMPLE_WRITE_XML
	<b>Message type</b>	<b>Response message (Body)</b>
--	<b>Message format</b>	adpff_result

Legend:  
 --: Not applicable.

(c) Setting up activities

The following describes how to arrange and define the activities in the figure below, define the data transformations, and connect the activities.

Figure D–8: Overall view of business process definition (sample for writing XML data)



■ Arranging and defining activities

In the sample for writing XML data, set up the activities shown in the following table.

Table D–43: List of activities (sample for writing XML data)

Activity	Activity name	Description
Reception	CSAFF_SAMPLE_BP_WRITE_XML	Receives messages into the business process.
Reply	CSAFF_SAMPLE_BP_WRITE_XML_RSP	Issues a response from the business process.
Invoke Service	InvokeFileAadapter	Invokes the file adapter.
Invoke Service	InvokeDBAdapter	Invokes the DB adapter.
Data transformation	CreateFileAdapterRequestMessage	Transforms a response message from the DB adapter to a request message for the file adapter.
Data transformation	CreateDBAdapterRequestMessage	Transforms the request message of the business process to a request message for the DB adapter.

To position and set up an activity:

- In the palette, click the basic or structure activity you want to place on the canvas.  
The activity you clicked is selected.
- Click the appropriate location on the canvas.  
The selected activity is placed on the canvas. You can reposition the activity by dragging it with your mouse.
- Double-click the activity on the canvas.  
A dialog box appears in which you can enter settings for the activity you double-clicked.
- Enter the necessary information in the dialog box.  
For details on the information you need to set, see *Activity settings (sample for writing XML data)*.
- Click **OK**.  
The dialog box for the activity closes.
- Repeat steps 1 to 5 for each activity.

Activity settings (sample for writing XML data)

The tables below show the settings for the business process activities used in the sample for writing XML data. You do not need to set items that do not appear in this table.

Table D–44: Settings for CSAFF\_SAMPLE\_BP\_WRITE\_XML

Item	Value to set
<b>Activity name</b>	CSAFF_SAMPLE_BP_WRITE_XML
<b>Operation name</b>	CSAFF_SAMPLE_BP_WRITE_XML
<b>Body allocated variable</b>	CSAFF_BP_RequestMessage
<b>Communication model</b>	<b>Sync</b>
<b>Instance generation</b>	<b>Yes</b>

Table D–45: Settings for CSAFF\_SAMPLE\_BP\_WRITE\_XML\_RSP

Item	Value to set
<b>Activity name</b>	CSAFF_SAMPLE_BP_WRITE_XML_RSP
<b>Operation name</b>	CSAFF_SAMPLE_BP_WRITE_XML
<b>Body allocated variable</b>	CSAFF_ResponseMessage

Table D–46: Settings for InvokeFileAadapter (sample for writing XML data)

Item		Value to set
<b>Activity name</b>		InvokeFileAadapter
<b>Service name</b>		CSAFF_SAMPLE_WRITE_XML
<b>Operation name</b>		CSAFF_SAMPLE_WRITE_XML
<b>Communication model</b>		<b>Sync</b>
<b>Request message</b>	<b>Body allocated variable</b>	CSAFF_RequestMessage
<b>Response message</b>	<b>Body allocated variable</b>	CSAFF_ResponseMessage

Table D–47: Settings for InvokeDBAdapter (sample for writing XML data)

Item		Value to set
<b>Activity name</b>		InvokeDBAdapter
<b>Service name</b>		CSAFF_SAMPLE_DB
<b>Operation name</b>		CSAFF_SAMPLE_DB
<b>Communication model</b>		<b>Sync</b>
<b>Request message</b>	<b>Body allocated variable</b>	CSAFF_DB_RequestMessage
<b>Response message</b>	<b>Body allocated variable</b>	CSAFF_DB_ResponseMessage

Table D–48: Settings for CreateFileAdapterRequestMessage (sample for writing XML data)

Item		Value to set
<b>Activity name</b>		CreateFileAdapterRequestMessage
<b>Source Variables</b>		CSAFF_BP_RequestMessage
		CSAFF_DB_ResponseMessage
<b>Destination Variable</b>		CSAFF_RequestMessage
Data TransDefn File		SAMPLE_WRITE_XML_DT2

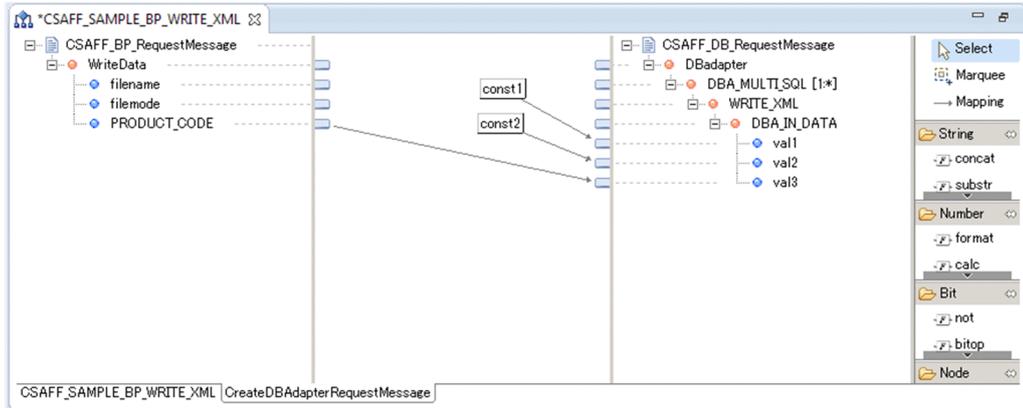
Table D–49: Settings for CreateDBAdapterRequestMessage (sample for writing XML data)

Item		Value to set
<b>Activity name</b>		CreateDBAdapterRequestMessage
<b>Source Variables</b>		CSAFF_BP_RequestMessage
<b>Destination Variable</b>		CSAFF_DB_RequestMessage
Data TransDefn File		SAMPLE_WRITE_XML_DT1

■ Setting the data transformation definition for the CreateDBAdapterRequestMessage activity

Set the data transformation definition for the CreateDBAdapterRequestMessage activity. The following figure shows the data transformation definition you need to set:

Figure D–9: Data conversion definition window for CreateDBAdapterRequestMessage activity (sample for writing XML data)



To set the data transformation definition for the CreateDBAdapterRequestMessage activity:

1. Display the data transformation (mapping) definition screen from the CreateDBAdapterRequestMessage activity.  
After displaying the data transformation (mapping) definition screen, the Select Root Element dialog box appears.
2. Select the root elements for all logical schema names.  
The **OK** button becomes available.
3. Click **OK**.

The schemas of the selected root elements appear in tree format in the transformation-source schema tree viewer and the transformation-destination schema tree viewer.

4. Perform mapping.

Mapping involves the following tasks:

- Placing and setting functions  
The following table shows the function settings:

Table D–50: Function settings (sample for writing XML data - CreateDBAdapterRequestMessage activity)

Function	Function name	Value to set	
const	const1	<b>Constant type</b>	String
		<b>Value</b>	PRODUCT_CODE
const	const2	<b>Constant type</b>	String
		<b>Value</b>	=

- Setting inter-node mapping

The following table shows the inter-node mapping settings:

Table D–51: Mapping settings (sample for writing XML data - CreateDBAdapterRequestMessage activity)

Source node	Function	Target node
--	const1	val1
--	const2	val2
PRODUCT_CODE	--	val3

Legend:

--: Not applicable.

D. Example for setting up file adapter

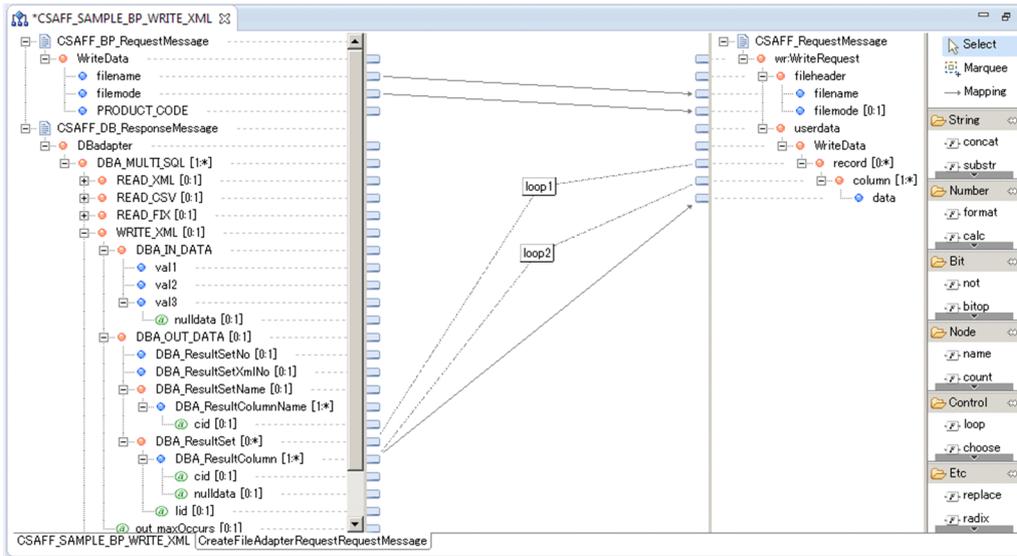
5. In the data transformation (mapping) definition screen, right-click an appropriate location in the transformation-source schema tree viewer, transformation-destination schema tree viewer, or mapping viewer, and select **Validate**.

Validation takes place.

■ Setting the data transformation definition for the CreateFileAdapterRequestMessage activity

Set the data transformation definition for the CreateFileAdapterRequestMessage activity. The following figure shows the data transformation definition you need to set:

Figure D–10: Data conversion definition window for CreateFileAdapterRequestMessage activity (sample for writing XML data)



To set the data transformation definition for the CreateFileAdapterRequestMessage activity:

1. Display the data transformation (mapping) definition screen from the CreateFileAdapterRequestMessage activity. After displaying the data transformation (mapping) definition screen, the Select Root Element dialog box appears.
2. Select the root elements for all logical schema names. The **OK** button becomes available.
3. Click **OK**. The schemas of the selected root elements appear in tree format in the transformation-source schema tree viewer and the transformation-destination schema tree viewer.

4. Perform mapping.

Mapping involves the following tasks:

- Placing and setting functions

The following table shows the function settings:

Table D–52: Function settings (sample for writing XML data - CreateFileAdapterRequestMessage activity)

Function	Function name	Value to set	
loop	loop1	<b>Select Node</b>	DBA_ResultSet#
		<b>Set Node Condition</b>	Do not set
loop	loop2	<b>Select Node</b>	DBA_ResultSetColumn#
		<b>Set Node Condition</b>	Do not set

#  
DBA\_ResultSet and DBA\_ResultColumn use the information under the WRITE\_XML node.

- Setting inter-node mapping

The following table shows the inter-node mapping settings:

Table D–53: Mapping settings (sample for writing XML data - CreateFileAdapterRequestMessage activity)

Source node	Function	Target node
--	loop1	record
--	loop2	column
filename	--	filename
filemode	--	filemode
DBA_ResultColumn#	--	data

Legend:

--: Not applicable.

#

DBA\_ResultColumn uses information under the WRITE\_XML node.

5. In the data transformation (mapping) definition screen, right-click an appropriate location in the transformation-source schema tree viewer, transformation-destination schema tree viewer, or mapping viewer, and select **Validate**.

Validation takes place. You can ignore any warning messages that appear.

#### ■ Connecting activities

In the sample for writing XML data, connect the activities as shown in the following table.

Table D–54: Activity connections (sample for writing XML data)

Connection-source activity	Connection-target activity
Start	CSAFF_SAMPLE_BP_WRITE_XML
CSAFF_SAMPLE_BP_WRITE_XML	CreateDBAdapterRequestMessage
CreateDBAdapterRequestMessage	InvokeDBAdapter
InvokeDBAdapter	CreateFileAdapterRequestMessage
CreateFileAdapterRequestMessage	InvokeFileAadapter
InvokeFileAadapter	CSAFF_SAMPLE_BP_WRITE_XML_RSP
CSAFF_SAMPLE_BP_WRITE_XML_RSP	Finish

To connect the activities:

1. On the palette, click **Connection**.  
Connection mode is selected.  
When you align your cursor with an activity that can be specified as a connection source, the background color of the activity changes.
2. On the canvas, click the activity that you want to serve as the connection source.  
The activity is set as the connection source.  
If you then align your cursor with another activity, the background color of the activity changes if it can be specified as a connection target.
3. On the canvas, click the activity that you want to serve as the connection target.

## D. Example for setting up file adapter

The activity you selected as the connection source is connected to the activity you selected as the connection target.

### (d) Saving and validating the business process

To save and validate the business process:

1. Review the definition contents, and then save file adapter settings by selecting **File** and then **Save** from the Eclipse menu.  
In properties view, set CSAFF\_B4 as the service ID of the business process.
2. Validate the business process settings.  
Select **Verify** from the pop-up list displayed when you right-click the Service Definition List in the tree view. The results of the validation process appear in Console View.

### (3) Distributing the file adapter and business process deployment definitions

Deploy the file adapter and business process and start them on the server.

To distribute the file adapter and business process deployment definitions:

1. Right-click the Service Definition List in the tree view, and select **Deploy all services to server and start**.  
If the file adapter or business process fails to start, a dialog box appears describing the nature of the error. Look up the message ID displayed in the error details and take the appropriate action.  
If you are not logged in, the Account Verification window appears. In this case, perform step 2.
2. Enter `admin` in the **User ID** field and the **Password** field, and then click **OK**.  
A message indicating that account verification is in progress appears, followed by a message indicating whether it was successful.

### (4) Creating tables

Create the table the sample business process will access in the database.

Open HiRDB SQL Executer and execute the following SQL statement:

```
CREATE TABLE CSAFF_SAMPLE_WX
(ORDER_NO INT,
 CUSTOMER_CODE CHAR(5),
 PRODUCT_CODE CHAR(4),
 ORDER_COUNT INT);
INSERT INTO CSAFF_SAMPLE_WX VALUES (1001, 'AA001', '1001', 5);
INSERT INTO CSAFF_SAMPLE_WX VALUES (1002, 'AB002', '1001', 1);
INSERT INTO CSAFF_SAMPLE_WX VALUES (1003, 'AA001', '1102', 3);
INSERT INTO CSAFF_SAMPLE_WX VALUES (1004, 'XA005', '1103', 1);
INSERT INTO CSAFF_SAMPLE_WX VALUES (1005, 'AA001', '1105', 1);
```

#### ! Important note

Use ADMIN as the user ID.

### (5) Creating request messages

The following describes how to create a request message by editing the sample request message used when executing the business process.

1. Open the sample request message file in a text editor.  
The sample request message file is `service-platform-installation-directory\CSC\custom-adapter\File\sample\write_xml\write_xml_message.xml`.

2. Add the part in italics below:

```
<?xml version="1.0" encoding="UTF-8"?>
<WriteData>
  <filename>service-platform-installation-directory\CSC\custom-adapter\File
\sample\write_xml\write_xml_data.xml</filename>
  <filemode>new</filemode>
```

```
<PRODUCT_CODE>1001</PRODUCT_CODE>
</WriteData>
```

3. Save the file.

## (6) Executing the business process

To execute the sample, start the business process from the service requester.

The flow of service requester processing is as follows:

1. Service requester execution  
The request message of the business process is used as the request message of the file adapter without transformation.
2. DB adapter execution  
Data that matches the conditions specified in the request message is acquired from the database.
3. Create file adapter message  
The response message from the DB adapter is transformed to a request message for the file adapter.
4. File adapter execution  
The file specified in the request message is output.
5. Service requester termination  
The response message from the file adapter is used without transformation as the response message of the business process.

To execute the service requester:

1. Navigate to *service-platform-installation-directory*\CSC\custom-adapter\File\sample\bin.

2. Execute *request.bat*.

Execute *request.bat* in the following format:

```
request write_xml service-platform-installation-directory\CSC\custom-adapter
\File\sample\write_xml\write_xml_message.xml
```

---

### Reference note

The data is output to the following file:

```
service-platform-installation-directory\CSC\custom-adapter\File\sample\write_xml
\write_xml_data.xml
```

---

## D.6 Sample for writing CSV data

The sample for writing CSV data uses a file adapter to write data retrieved by a DB adapter to a file in CSV format.

The following table lists the items that are input and output:

Table D-55: Input and output items (sample for writing CSV data)

Input or output item	Content
Output file of file adapter	<i>service-platform-installation-directory</i> \CSC\custom-adapter\File\sample\write_csv\write_csv_data.csv
Input table of DB adapter	ADMIN.CSAFF_SAMPLE_WC#

#

Do not change the schema name.

## (1) Defining the file adapter

### (a) Adding and validating the file adapter

To add and validate a file adapter:

1. From the Eclipse menu, select **Window, Show View**, and then **Other**.  
The Show View dialog box appears.
2. Select **HCSC-Definer** and then **HCSCTE View**, and click **OK**.  
The Service Definition List appears in the tree view.
3. Right-click the Service Definition List in the tree view, and select **Add Service Adapter**.  
The Service adapter definition addition wizard appears.
4. From the **Service component type:** drop-down list, select **File adapter**.
5. Click **Next**.  
The Service adapter definition addition (File Adapter) wizard appears.
6. Enter the service name.  
Enter `CSAFF_SAMPLE_WRITE_CSV` as the service name.
7. Click **Finish**.  
The necessary files are created, and stored in the repository. The Service adapter definition (standard) window appears.
8. Enter settings in the Service adapter definition (standard) and Service adapter definition (details) windows.  
For details on the settings you need to enter, see *File adapter settings (sample for writing CSV data)*.
9. Review the definition contents, and then save the service adapter definition by selecting **File** and then **Save** from the Eclipse menu.
10. Verify the file adapter settings.  
Select **Verify** from the pop-up list displayed when you right-click the Service Definition List in the tree view. The results appear in the Console View.

#### File adapter settings (sample for writing CSV data)

The table below shows the items you need to set in the Service adapter definition (standard) window when defining the file adapter for the sample for writing CSV data. You do not need to set items that do not appear in this table.

Table D–56: Settings in Service adapter definition (standard) window (sample for writing CSV data)

Category	Item	Value to set	Requires setting	
<b>Service component control information</b>	<b>Service name</b>	<code>CSAFF_SAMPLE_WRITE_CSV</code>	Y	
	<b>Service ID</b>	<code>CSAFF_WC</code>	Y	
	<b>Service type</b>	File adapter	N	
	<b>Address</b>	--	--	
	<b>Maximum instances</b>	0	Y	
	<b>Operation</b>	<b>Pattern:</b> Binary file writing <b>Operation name:</b> <code>CSAFF_SAMPLE_WRITE_CSV</code>	Y	
<b>Operation information</b>	<b>Operation name</b>	<code>CSAFF_SAMPLE_WRITE_CSV</code>	N	
	<b>Communication model</b>	<b>Sync</b>	Y	
<b>Request message</b>	<b>Standard</b>	<b>Use check box</b>	Use (select the check box)	Y
		<b>Format ID</b>	<code>SAMPLE_WRITE_CSV_Standard</code>	Y

Category	Item		Value to set	Requires setting
<b>Request message</b>	<b>Standard</b>	<b>Message format</b>	csaff_sample_write_csv_standard.xsd	Y
	<b>Service component</b>	<b>Format ID</b>	SAMPLE_WRITE_CSV_Request	Y
		<b>Message format</b>	csaff_sample_write_csv.fdx	Y
	<b>Data TransDefn File</b>		SAMPLE_WRITE_CSV_DT	Y
<b>Response message</b>	<b>Standard</b>	<b>Use check box</b>	Do not use (leave the check box cleared)	Y
	<b>Service component</b>	<b>Format ID</b>	SAMPLE_WRITE_CSV_Response	Y
		<b>Message format</b>	adpff_result.xsd	N

Legend:

Y: Must be set.

N: Check the content that is already displayed.

--: Do not set.

The table below shows the items you need to set in the Service adapter definition (details) window when defining the file adapter for the sample for writing CSV data. You do not need to set items that do not appear in this table.

Table D–57: Settings in Service adapter definition (details) window (sample for writing CSV data)

Category	Item	Value to set	Requires setting
<b>Service adapter control information</b>	<b>Service adapter (EJB-JAR file)</b>	cscmsg_adpejb.jar	N
	<b>Utility class (JAR file)</b>	adpffpc.jar	N

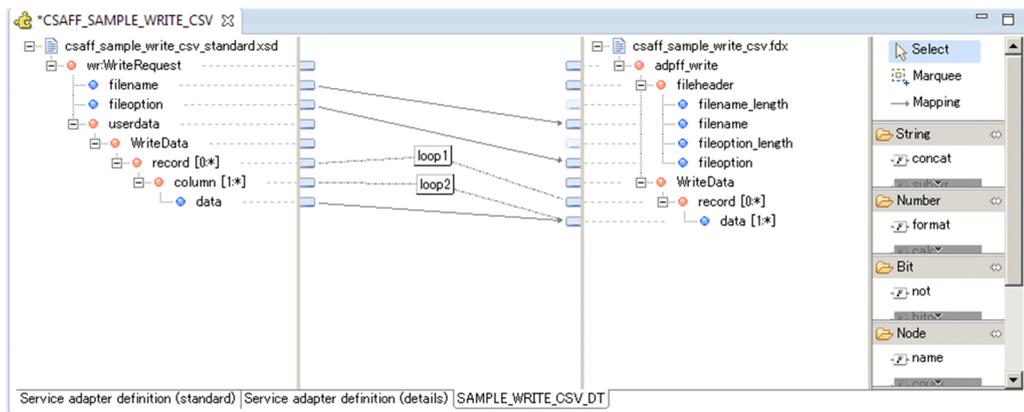
Legend:

N: Check the content that is already displayed.

(b) Defining data transformation

Set the data transformation definition. The following figure shows the data transformation definition you need to set:

Figure D–11: Data-conversion definition screen for file adapter (sample for writing CSV data)



To define this data transformation:

1. Display the data transformation (mapping) definition screen from the Adapter Definition screen.  
After displaying the data transformation (mapping) definition screen, the Select Root Element dialog box appears.

#### D. Example for setting up file adapter

2. Select the root elements for all logical schema names.

The **OK** button becomes available.

3. Click **OK**.

The schemas of the selected root elements appear in tree format in the transformation-source schema tree viewer and the transformation-destination schema tree viewer.

4. Perform mapping.

Mapping involves the following tasks:

- Placing and setting functions

The following table shows the function settings:

Table D–58: Function settings (sample for writing CSV data - File adapter)

Function	Function name	Value to set	
loop	loop1	<b>Select Node</b>	record
		<b>Set Node Condition</b>	Do not set
loop	loop2	<b>Select Node</b>	column
		<b>Set Node Condition</b>	Do not set

- Setting inter-node mapping

The following table shows the inter-node mapping settings:

Table D–59: Mapping settings (sample for writing CSV data - File adapter)

Source node	Function	Target node
--	loop1	record
--	loop2	data
filename	--	filename
fileoption	--	fileoption
data	--	data

Legend:

--: Not applicable.

5. In the data transformation (mapping) definition screen, right-click an appropriate location in the transformation-source schema tree viewer, transformation-destination schema tree viewer, or mapping viewer, and select **Validate**.

Validation takes place. You can ignore any warning messages that appear.

#### (c) Creating the file adapter runtime-environment property file

To create the file adapter runtime-environment property file:

1. Open the sample file adapter runtime-environment property file in a text editor.

The sample file adapter runtime-environment property file is *service-platform-installation-directory\CSC\custom-adapter\File\sample\write\_csv\CSAFF\_WC.properties*.

2. Add the following content to the file:

```
fileaccess.path1 = service-platform-installation-directory\CSC\custom-adapter\File\sample\write_csv
character.code = MS932
codetable.path = storage-path-for-code-conversion-table-(user-mapping-file)
```

The storage path for user mapping files is as follows:

```
service-platform-installation-directory\CSC\lib\external\table
```

3. Save the file.

The storage path for the file is as follows:

`service-platform-installation-directory\CSC\custom-adapter\File\config`

## (2) Defining the business process

The task of defining a business process involves adding a business process, defining the contents of the business process, and then validating the business process.

### (a) Adding a business process

To add a business process:

1. From the Eclipse menu, select **Window, Show View**, and then **Other**.  
The Show View dialog box appears.
2. Select **HCSC-Definer** and then **HCSCTE View**, and click **OK**.  
The Service Definition List appears in the tree view.
3. Right-click the Service Definition List in the tree view, and select **Add Business Process**.  
A dialog box appears in which you can add a business process definition.
4. Enter the name of the business process, and select whether to make its status persistent (whether to save the execution status of the business process in the database).  
Enter `CSAFF_SAMPLE_BP_WRITE_CSV` as the business process name.  
Select *yes* for **Status Persistence**.

---

#### Reference note

Do not use a BPEL file.

---

5. Click **Finish**.

The necessary files are created, and stored in the repository. The Define Business Process window appears.

### (b) Setting variables

Set the following variables:

- `CSAFF_BP_RequestMessage`
- `CSAFF_DB_RequestMessage`
- `CSAFF_DB_ResponseMessage`
- `CSAFF_RequestMessage`
- `CSAFF_ResponseMessage`

---

#### Reference note

The sample business process does not use a correlation set.

---

To set the variables:

1. On the canvas of the Define Business Process window, double-click the Variable-Correlation icon.  
The List Of Variables And Correlation Sets dialog box appears.
2. Select **Variable List** from the list.
3. Enter the variable name.
4. From the **Type:** drop-down list, select **XML**.
5. Perform whichever of the following operations is relevant for the variable you are defining:
  - For `CSAFF_DB_ResponseMessage`, `CSAFF_RequestMessage`, and `CSAFF_ResponseMessage`  
In the Take In Message Format dialog box displayed by clicking the **Take In** button, specify the message format you want to use. For details on the settings you need to enter, see *Variable settings (sample for writing CSV data)*.
  - For `CSAFF_BP_RequestMessage`  
Click the **...** button and select the `csaff_sample_write_csv.xsd` file.

D. Example for setting up file adapter

For CSAFF\_DB\_RequestMessage

Click the ... button and select the csaff\_sample\_write\_csv\_db.xsd file.

6. Click **Add**.

The variable you added appears in the list of variables.

7. Repeat steps 2 to 6 for each variable.

8. Click **OK**.

The variable settings are saved and the List Of Variables And Correlation Sets dialog box closes.

Variable settings (sample for writing CSV data)

The tables below show the settings for the CSAFF\_DB\_ResponseMessage, CSAFF\_RequestMessage, and CSAFF\_ResponseMessage variables. You do not need to set items that do not appear in this table.

Table D–60: Settings for CSAFF\_DB\_ResponseMessage (sample for writing CSV data)

Category	Item	Value to set
<b>Service/Reception</b>	<b>Service name</b>	CSAFF_SAMPLE_DB
	<b>Reception name</b>	--
<b>Target for take in</b>	<b>Operation name</b>	CSAFF_SAMPLE_DB
	<b>Message type</b>	<b>Response message (Body)</b>
--	<b>Message format</b>	csaff_sample_sqlformat

Legend:

--: Not applicable.

Table D–61: Settings for CSAFF\_RequestMessage (sample for writing CSV data)

Category	Item	Value to set
<b>Service/Reception</b>	<b>Service name</b>	CSAFF_SAMPLE_WRITE_CSV
	<b>Reception name</b>	--
<b>Target for take in</b>	<b>Operation name</b>	CSAFF_SAMPLE_WRITE_CSV
	<b>Message type</b>	<b>Request message (Body)</b>
--	<b>Message format</b>	csaff_sample_write_csv_standard

Legend:

--: Not applicable.

Table D–62: Settings for CSAFF\_ResponseMessage (sample for writing CSV data)

Category	Item	Value to set
<b>Service/Reception</b>	<b>Service name</b>	CSAFF_SAMPLE_WRITE_CSV
	<b>Reception name</b>	--
<b>Target for take in</b>	<b>Operation name</b>	CSAFF_SAMPLE_WRITE_CSV
	<b>Message type</b>	<b>Response message (Body)</b>
--	<b>Message format</b>	adpff_result

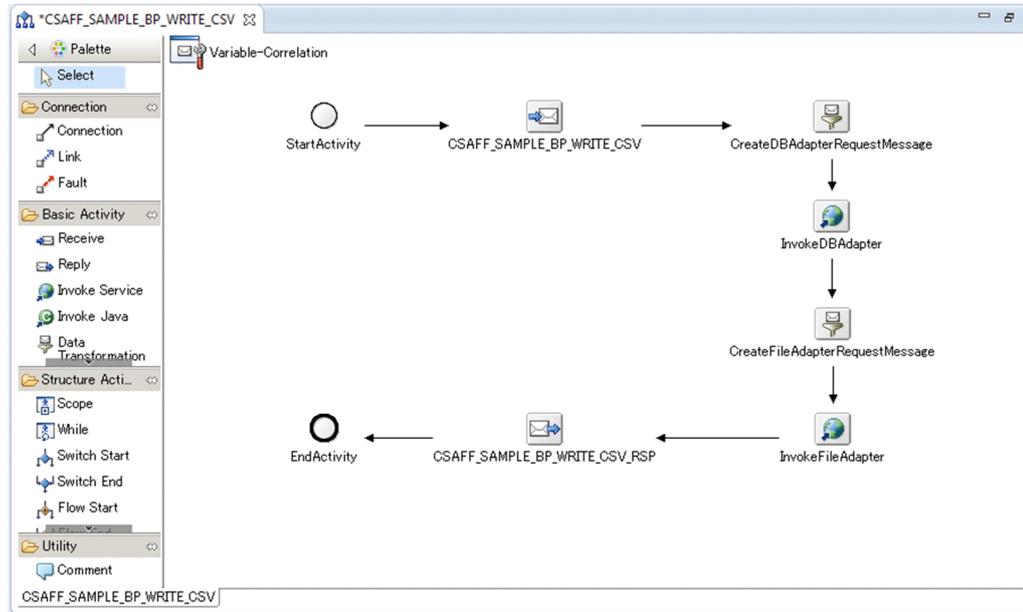
Legend:

--: Not applicable.

## (c) Setting up activities

The following describes how to arrange and define the activities in the figure below, define the data transformations, and connect the activities.

Figure D–12: Overall view of business process definition (sample for writing CSV data)



#### ■ Arranging and defining activities

In the sample for writing CSV data, set up the activities shown in the following table.

Table D–63: List of activities (sample for writing CSV data)

Activity	Activity name	Description
Reception	CSAFF_SAMPLE_BP_WRITE_CSV	Receives messages into the business process.
Reply	CSAFF_SAMPLE_BP_WRITE_CSV_RSP	Issues a response from the business process.
Invoke Service	InvokeFileAadapter	Invokes the file adapter.
Invoke Service	InvokeDBAdapter	Invokes the DB adapter.
Data transformation	CreateFileAdapterRequestMessage	Transforms a response message from the DB adapter to a request message for the file adapter.
Data transformation	CreateDBAdapterRequestMessage	Transforms the request message of the business process to a request message for the DB adapter.

To position and set up an activity:

1. In the palette, click the basic or structure activity you want to place on the canvas.  
The activity you clicked is selected.
2. Click the appropriate location on the canvas.  
The selected activity is placed on the canvas. You can reposition the activity by dragging it with your mouse.
3. Double-click the activity on the canvas.  
A dialog box appears in which you can enter settings for the activity you double-clicked.
4. Enter the necessary information in the dialog box.

D. Example for setting up file adapter

For details on the information you need to set, see *Activity settings (sample for writing CSV data)*.

5. Click **OK**.

The dialog box for the activity closes.

6. Repeat steps 1 to 5 for each activity.

Activity settings (sample for writing CSV data)

The tables below show the settings for the business process activities used in the sample for writing CSV data. You do not need to set items that do not appear in this table.

Table D–64: Settings for CSAFF\_SAMPLE\_BP\_WRITE\_CSV

Item	Value to set
<b>Activity name</b>	CSAFF_SAMPLE_BP_WRITE_CSV
<b>Operation name</b>	CSAFF_SAMPLE_BP_WRITE_CSV
<b>Body allocated variable</b>	CSAFF_BP_RequestMessage
<b>Communication model</b>	<b>Sync</b>
<b>Instance generation</b>	<b>Yes</b>

Table D–65: Settings for CSAFF\_SAMPLE\_BP\_WRITE\_CSV\_RSP

Item	Value to set
<b>Activity name</b>	CSAFF_SAMPLE_BP_WRITE_CSV_RSP
<b>Operation name</b>	CSAFF_SAMPLE_BP_WRITE_CSV
<b>Body allocated variable</b>	CSAFF_ResponseMessage

Table D–66: Settings for InvokeFileAadapter (sample for writing CSV data)

Item	Value to set
<b>Activity name</b>	InvokeFileAadapter
<b>Service name</b>	CSAFF_SAMPLE_WRITE_CSV
<b>Operation name</b>	CSAFF_SAMPLE_WRITE_CSV
<b>Communication model</b>	<b>Sync</b>
<b>Request message</b>	<b>Body allocated variable</b> CSAFF_RequestMessage
<b>Response message</b>	<b>Body allocated variable</b> CSAFF_ResponseMessage

Table D–67: Settings for InvokeDBAdapter (sample for writing CSV data)

Item	Value to set
<b>Activity name</b>	InvokeDBAdapter
<b>Service name</b>	CSAFF_SAMPLE_DB
<b>Operation name</b>	CSAFF_SAMPLE_DB
<b>Communication model</b>	<b>Sync</b>
<b>Request message</b>	<b>Body allocated variable</b> CSAFF_DB_RequestMessage
<b>Response message</b>	<b>Body allocated variable</b> CSAFF_DB_ResponseMessage

Table D–68: Settings for CreateFileAdapterRequestMessage (sample for writing CSV data)

Item	Value to set
<b>Activity name</b>	CreateFileAdapterRequestMessage

Item	Value to set
<b>Source Variables</b>	CSAFF_BP_RequestMessage
	CSAFF_DB_ResponseMessage
<b>Destination Variable</b>	CSAFF_RequestMessage
Data TransDefn File	SAMPLE_WRITE_CSV_DT2

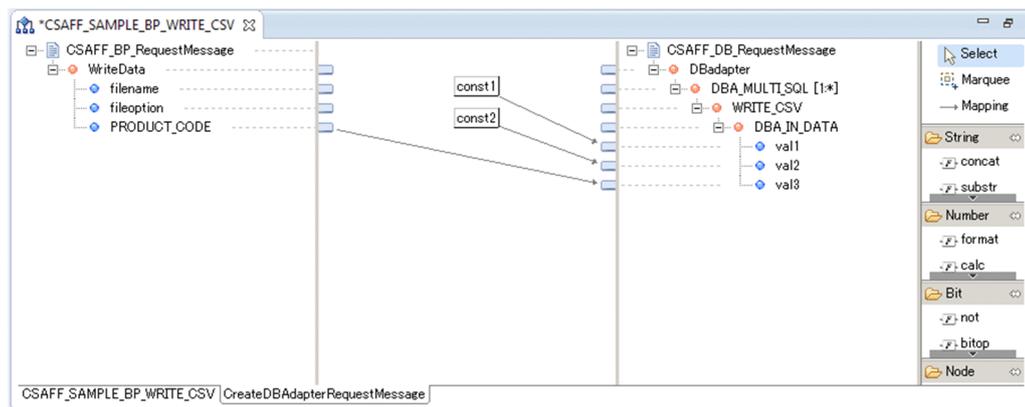
Table D–69: Settings for CreateDBAdapterRequestMessage (sample for writing CSV data)

Item	Value to set
<b>Activity name</b>	CreateDBAdapterRequestMessage
<b>Source Variables</b>	CSAFF_BP_RequestMessage
<b>Destination Variable</b>	CSAFF_DB_RequestMessage
Data TransDefn File	SAMPLE_WRITE_CSV_DT1

#### ■ Setting the data transformation definition for the CreateDBAdapterRequestMessage activity

Set the data transformation definition for the CreateDBAdapterRequestMessage activity. The following figure shows the data transformation definition you need to set:

Figure D–13: Data conversion definition window for CreateDBAdapterRequestMessage activity (sample for writing CSV data)



To set the data transformation definition for the CreateDBAdapterRequestMessage activity:

1. Display the data transformation (mapping) definition screen from the CreateDBAdapterRequestMessage activity.  
After displaying the data transformation (mapping) definition screen, the Select Root Element dialog box appears.
2. Select the root elements for all logical schema names.  
The **OK** button becomes available.
3. Click **OK**.  
The schemas of the selected root elements appear in tree format in the transformation-source schema tree viewer and the transformation-destination schema tree viewer.
4. Perform mapping.  
Mapping involves the following tasks:
  - Placing and setting functions  
The following table shows the function settings:

D. Example for setting up file adapter

Table D-70: Function settings (sample for writing CSV data - CreateDBAdapterRequestMessage activity)

Function	Function name	Value to set	
const	const1	Constant type	String
		Value	PRODUCT_CODE
const	const2	Constant type	String
		Value	=

- Setting inter-node mapping

The following table shows the inter-node mapping settings:

Table D-71: Mapping settings (sample for writing CSV data - CreateDBAdapterRequestMessage activity)

Source node	Function	Target node
--	const1	val1
--	const2	val2
PRODUCT_CODE	--	val3

Legend:

--: Not applicable.

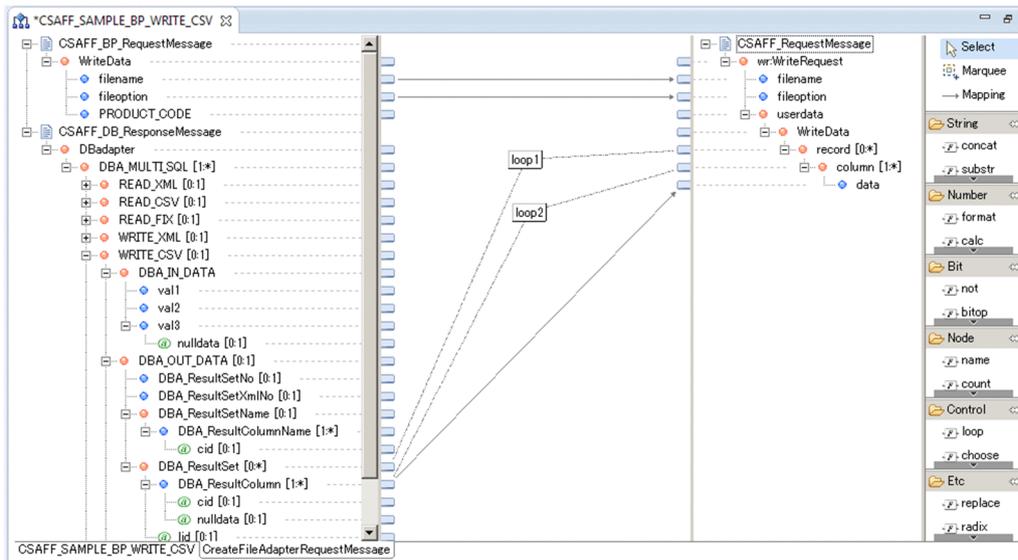
5. In the data transformation (mapping) definition screen, right-click an appropriate location in the transformation-source schema tree viewer, transformation-destination schema tree viewer, or mapping viewer, and select **Validate**.

Validation takes place.

■ Setting the data transformation definition for the CreateFileAdapterRequestMessage activity

Set the data transformation definition for the CreateFileAdapterRequestMessage activity. The following figure shows the data transformation definition you need to set:

Figure D-14: Data conversion definition window for CreateFileAdapterRequestMessage activity (sample for writing CSV data)



To set the data transformation definition for the CreateFileAdapterRequestMessage activity:

1. Display the data transformation (mapping) definition screen from the CreateFileAdapterRequestMessage activity.  
After displaying the data transformation (mapping) definition screen, the Select Root Element dialog box appears.
2. Select the root elements for all logical schema names.  
The **OK** button becomes available.

3. Click **OK**.

The schemas of the selected root elements appear in tree format in the transformation-source schema tree viewer and the transformation-destination schema tree viewer.

4. Perform mapping.

Mapping involves the following tasks:

- Placing and setting functions

The following table shows the function settings:

Table D–72: Function settings (sample for writing CSV data - CreateFileAdapterRequestMessage activity)

Function	Function name	Value to set	
loop	loop1	<b>Select Node</b>	DBA_ResultSet <sup>#</sup>
		<b>Set Node Condition</b>	Do not set
loop	loop2	<b>Select Node</b>	DBA_ResultColumn <sup>#</sup>
		<b>Set Node Condition</b>	Do not set

#

DBA\_ResultSet and DBA\_ResultColumn use the information under the WRITE\_CSV node.

- Setting inter-node mapping

The following table shows the inter-node mapping settings:

Table D–73: Mapping settings (sample for writing CSV data - CreateFileAdapterRequestMessage activity)

Source node	Function	Target node
--	loop1	record
--	loop2	column
filename	--	filename
fileoption	--	fileoption
DBA_ResultColumn <sup>#</sup>	--	data

Legend:

--: Not applicable.

#

DBA\_ResultColumn uses information under the WRITE\_CSV node.

5. In the data transformation (mapping) definition screen, right-click an appropriate location in the transformation-source schema tree viewer, transformation-destination schema tree viewer, or mapping viewer, and select **Validate**.

Validation takes place.

#### ■ Connecting activities

In the sample for writing CSV data, connect the activities as shown in the following table.

#### D. Example for setting up file adapter

Table D–74: Activity connections (sample for writing CSV data)

Connection-source activity	Connection-target activity
Start	CSAFF_SAMPLE_BP_WRITE_CSV
CSAFF_SAMPLE_BP_WRITE_CSV	CreateDBAdapterRequestMessage
CreateDBAdapterRequestMessage	InvokeDBAdapter
InvokeDBAdapter	CreateFileAdapterRequestMessage
CreateFileAdapterRequestMessage	InvokeFileAadapter
InvokeFileAadapter	CSAFF_SAMPLE_BP_WRITE_CSV_RSP
CSAFF_SAMPLE_BP_WRITE_CSV_RSP	Finish

To connect the activities:

1. On the palette, click **Connection**.  
Connection mode is selected.  
When you align your cursor with an activity that can be specified as a connection source, the background color of the activity changes.
2. On the canvas, click the activity that you want to serve as the connection source.  
The activity is set as the connection source.  
If you then align your cursor with another activity, the background color of the activity changes if it can be specified as a connection target.
3. On the canvas, click the activity that you want to serve as the connection target.  
The activity you selected as the connection source is connected to the activity you selected as the connection target.

#### (d) Saving and validating the business process

To save and validate the business process:

1. Review the definition contents, and then save file adapter settings by selecting **File** and then **Save** from the Eclipse menu.  
In properties view, set CSAFF\_B5 as the service ID of the business process.
2. Validate the business process settings.  
Select **Verify** from the pop-up list displayed when you right-click the Service Definition List in the tree view. The results appear in Console View.

### (3) Distributing the file adapter and business process deployment definitions

Deploy the file adapter and business process and start them on the server.

To distribute the file adapter and business process deployment definitions:

1. Right-click the Service Definition List in the tree view, and select **Deploy all services to server and start**.  
If the file adapter or business process fails to start, a dialog box appears describing the nature of the error. Look up the message ID displayed in the error details and take the appropriate action.  
If you are not logged in, the Account Verification window appears. In this case, perform step 2.
2. Enter `admin` in the **User ID** field and the **Password** field, and then click **OK**.  
A message indicating that account verification is in progress appears, followed by a message indicating whether it was successful.

### (4) Creating tables

Create the table the sample business process will access in the database.

Open HiRDB SQL Executer and execute the following SQL statement:

```

CREATE TABLE CSAFF_SAMPLE_WC
(ORDER_NO INT,
 CUSTOMER_CODE CHAR(5),
 PRODUCT_CODE CHAR(4),
 ORDER_COUNT INT);
INSERT INTO CSAFF_SAMPLE_WC VALUES (1001, 'AA001', '1001', 5);
INSERT INTO CSAFF_SAMPLE_WC VALUES (1002, 'AB002', '1001', 1);
INSERT INTO CSAFF_SAMPLE_WC VALUES (1003, 'AA001', '1102', 3);
INSERT INTO CSAFF_SAMPLE_WC VALUES (1004, 'XA005', '1103', 1);
INSERT INTO CSAFF_SAMPLE_WC VALUES (1005, 'AA001', '1105', 1);

```

### ! Important note

Use ADMIN as the user ID.

## (5) Creating request messages

The following describes how to create a request message by editing the sample request message used when executing the business process.

1. Open the sample request message file in a text editor.

The sample request message file is *service-platform-installation-directory\CSC\custom-adapter\File\sample\write\_csv\write\_csv\_message.csv*.

2. Add the part in italics below:

```

<?xml version="1.0" encoding="UTF-8"?>
<WriteData>
  <filename>service-platform-installation-directory\CSC\custom-adapter\File
\sample\write_csv\write_csv_data.csv</filename>
  <fileoption>mode=new</fileoption>
  <PRODUCT_CODE>1001</PRODUCT_CODE>
</WriteData>

```

3. Save the file.

## (6) Executing the business process

To execute the sample, start the business process from the service requester.

The flow of service requester processing is as follows:

1. Service requester execution

The request message of the business process is used as the request message of the file adapter without transformation.

2. DB adapter execution

Data that matches the conditions specified in the request message is acquired from the database.

3. Create file adapter message

The response message from the DB adapter is transformed to a request message for the file adapter.

4. File adapter execution

The file specified in the request message is output.

5. Service requester termination

The response message from the file adapter is used without transformation as the response message of the business process.

To execute the service requester:

1. Navigate to *service-platform-installation-directory\CSC\custom-adapter\File\sample\bin*.

2. Execute *request.bat*.

Execute *request.bat* in the following format:

```

request write_csv service-platform-installation-directory\CSC\custom-adapter
\File\sample\write_csv\write_csv_message.xml

```

#### D. Example for setting up file adapter

---

##### Reference note

The data is output to the following file:

```
service-platform-installation-directory\CSC\custom-adapter\File\sample\write_csv  
\write_csv_data.csv
```

---

## E. Example for defining file operations adapter

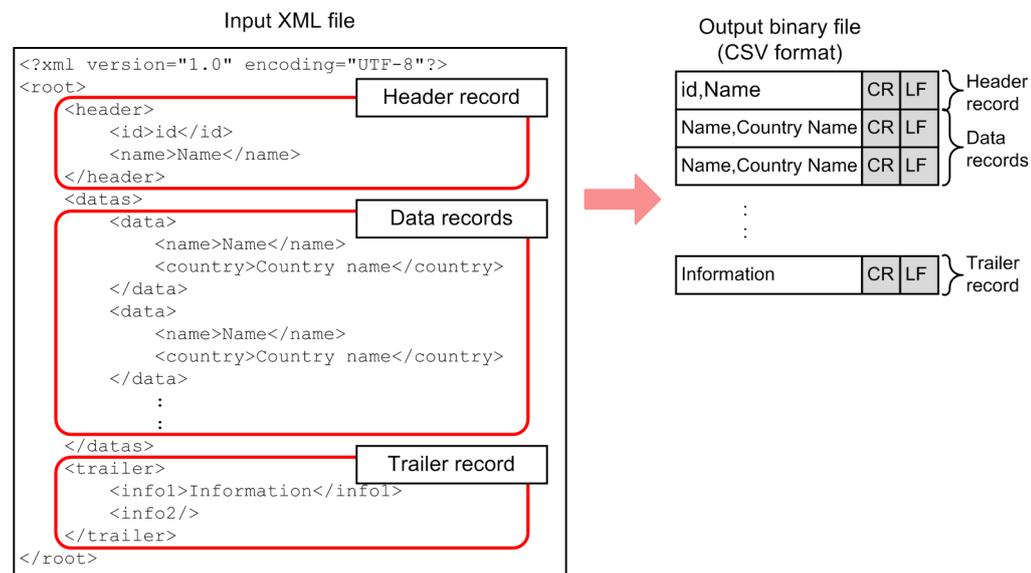
This appendix provides examples of defining message formats and self-defined files when using file transformation operations in a file operations adapter.

### E.1 When converting from the XML format to the binary format with the split processing method

This section provides an example of the definitions you need to create when using the split processing method to transform messages from XML to binary format.

The following figure shows an example of an input XML file and output binary file used in this example:

Figure E-1: Example of input XML file and output binary file



This example uses header records and trailer records. The split processing method applies at the level of individual records.

The following table lists the files required for this transformation:

Table E-1: Files required for transformation (when transforming from XML to binary format)

#	File type	Format
1	Input XML schema	XSD
2	Binary format definition file for output header records	FDX
3	Binary format definition file for output data records	FDX
4	Binary format definition file for output trailer records	FDX
5	Data transformation definition file for header records	XSL
6	Data transformation definition file for data records	XSL
7	Data transformation definition file for trailer records	XSL
8	File operation adapter definition file	Property file

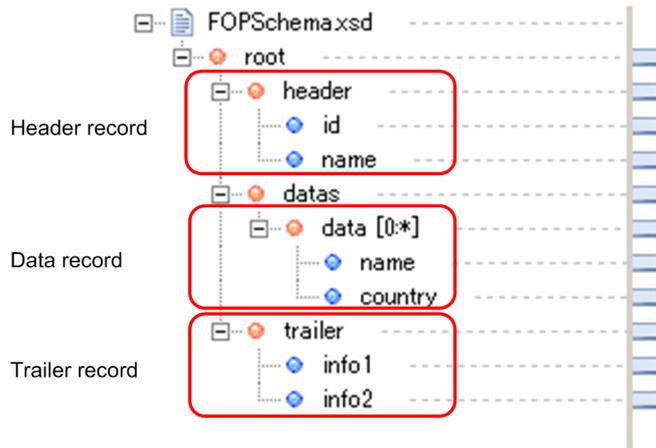
When outputting a file in binary format, you need to create binary format definition files (#2, #3, and #4) and data transformation definition files (#5, #6, and #7) for each record type.

E. Example for defining file operations adapter

For details on how to create binary format definition files, see *4.4 Creating Message Formats (Binary Format Definition File)* in the manual *Service Platform Basic Development Guide*. For details on how to create data transformation definition files, see the description for displaying from the Eclipse menu in *6.3.1 Procedure for Defining Data Transformation* in the manual *Service Platform Basic Development Guide*.

1. Input XML schema (XSD file)

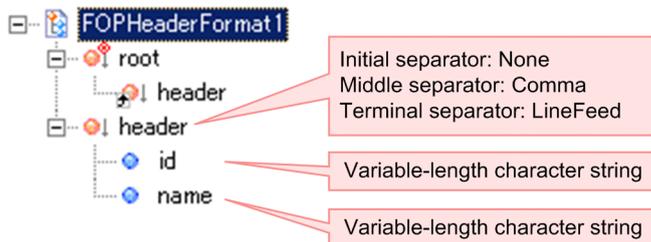
Use an application such as the WST (Web Standard Tools) provided by Eclipse to create an input XML schema with the content below. Specify `FOPSchema.xsd` as the file name of the input XML schema.



The figure shows the input XML schema loaded as the transformation-source schema when creating a mapping definition file.

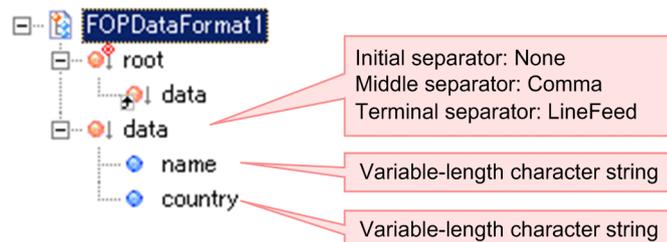
2. Binary format definition file for output header records (FDX file)

Create the binary format definition file as shown below. Specify `FOPHeaderFormat1.fdx` as the file name of the binary format definition file.



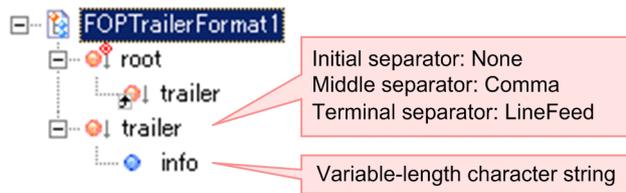
3. Binary format definition file for output data records (FDX file)

Create the binary format definition file as shown below. Specify `FOPDataFormat1.fdx` as the file name of the binary format definition file.



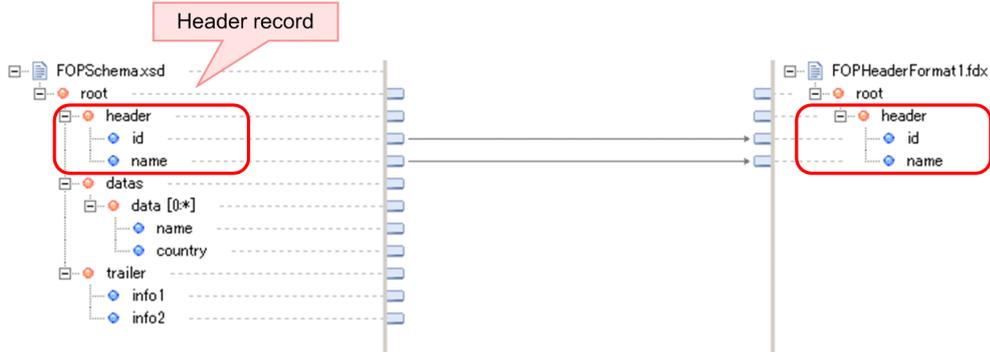
4. Binary format definition file for output trailer records (FDX file)

Create the binary format definition file as shown below. Specify `FOPTrailerFormat1.fdx` as the file name of the binary format definition file.



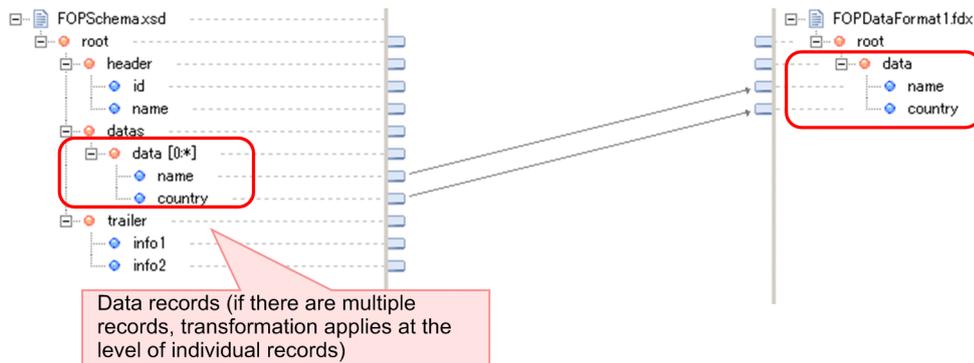
5. Data transformation definition file for header records (XSL file)

Create the data transformation definition file as shown below. Specify `FOPHeader1.xsl` as the file name of the XSL file. When creating a new mapping definition file, specify `FOPSchema.xsd` as the transformation-source schema and `FOPHeaderFormat1.fdx` as the transformation-target schema.



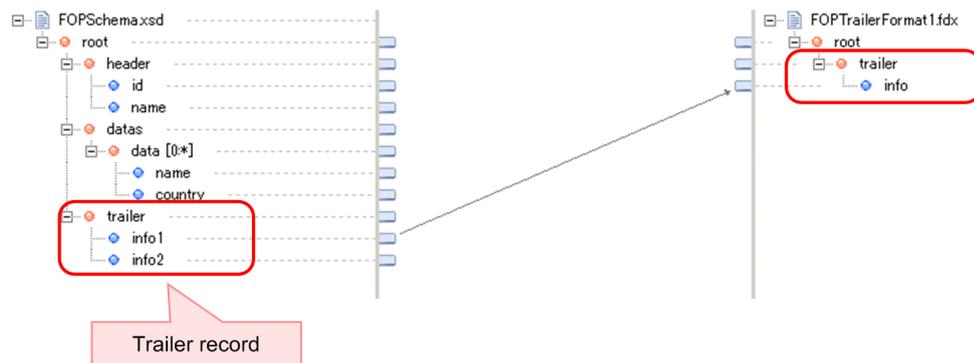
6. Data transformation definition file for data records (XSL file)

Create the data transformation definition file as shown below. Specify `FOPData1.xsl` as the file name of the XSL file. When creating a new mapping definition file, specify `FOPSchema.xsd` as the transformation-source schema and `FOPDataFormat1.fdx` as the transformation-target schema.



7. Data transformation definition file for trailer records (XSL file)

Create the data transformation definition file as shown below. Specify `FOPTrailer1.xsl` as the file name of the XSL file. When creating a new mapping definition file, specify `FOPSchema.xsd` as the transformation-source schema and `FOPTrailerFormat1.fdx` as the transformation-target schema.



## 8. File operation adapter definition file

In the Service adapter definition (details) screen, select `cscFileOperation.properties` in the **Self-defined file** area, and click Edit. Then, define the file as follows:

```
#Specify use of the split processing method
csc.adapter.fileOperation.transform.splitLoad = ON

#Specify whether to use header and trailer records
csc.adapter.fileOperation.transform.headerRecord = ON
csc.adapter.fileOperation.transform.trailerRecord = ON

#Define the input XML file
csc.adapter.fileOperation.transform.input = xml
csc.adapter.fileOperation.transform.header.inElement = /root/header
csc.adapter.fileOperation.transform.data.inElement = /root/datas/data
csc.adapter.fileOperation.transform.trailer.inElement = /root/trailer

#Define the output binary file
csc.adapter.fileOperation.transform.output = non-xml
csc.adapter.fileOperation.transform.header.outFormat = FOPHeaderFormat1.fdx
csc.adapter.fileOperation.transform.data.outFormat = FOPDataFormat1.fdx
csc.adapter.fileOperation.transform.trailer.outFormat = FOPTrailerFormat1.fdx

#Define the XSL file for each record type
csc.adapter.fileOperation.transform.header.styleSheet = FOPHeader1.xsl
csc.adapter.fileOperation.transform.data.styleSheet = FOPData1.xsl
csc.adapter.fileOperation.transform.trailer.styleSheet = FOPTrailer1.xsl
```

For details on file operation adapter definition files, see *File operations adapter definition file* in the manual *Service Platform Reference Guide*.

## 9. Service adapter definition

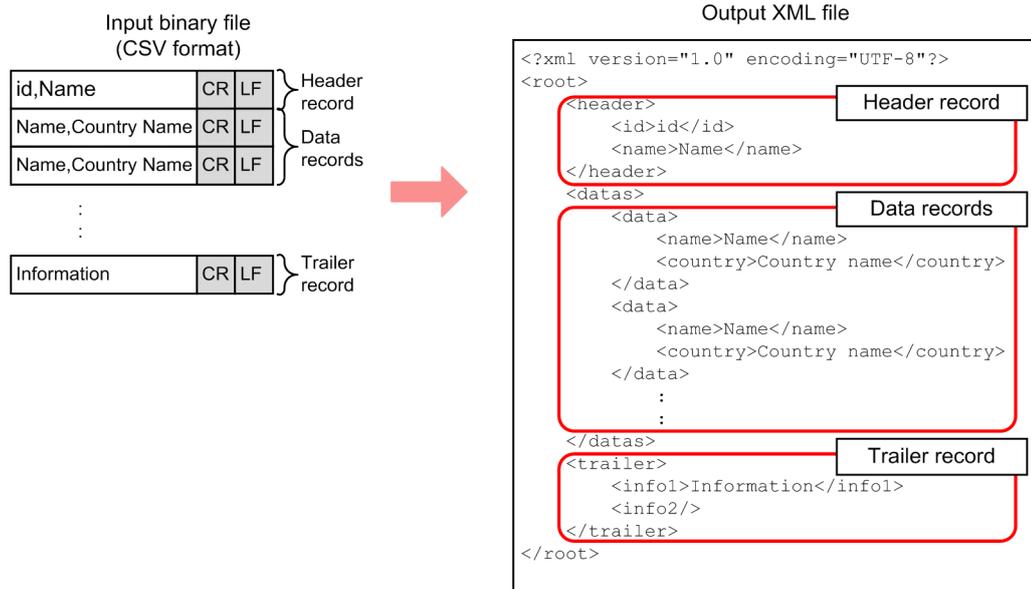
In the Service adapter definition (details) screen, add the following files as self-defined files:

- FOPHeaderFormat1.fdx
- FOPDataFormat1.fdx
- FOPTrailerFormat1.fdx
- FOPHeader1.xsl
- FOPData1.xsl
- FOPTrailer1.xsl

## E.2 When converting from the binary format to the XML format with the split processing method

This section provides an example of the definitions you need to create when using the split processing method to transform messages from binary to XML format.

Figure E-2: Example of input binary file and output XML file



This example uses header records and trailer records. Split processing applies at the level of individual records.

The following table lists the files required for this transformation:

Table E-2: Files required for transformation (when transforming from binary to XML format)

#	File type	Format
1	Binary format definition file for input header records	FDX
2	Binary format definition file for input data records	FDX
3	Binary format definition file for input trailer records	FDX
4	Output XML schema	XSD
5	Data transformation definition file for header record	XSL
6	Data transformation definition file for data records	XSL
7	Data transformation definition file for trailer records	XSL
8	XML file for output template	XML
9	File operation adapter definition file	Property file

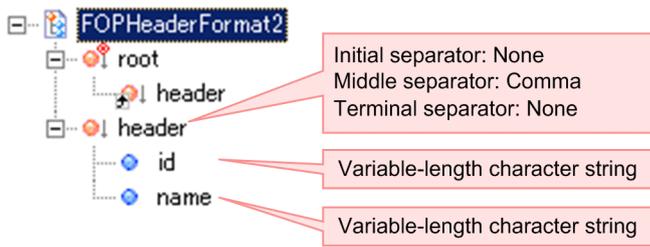
When inputting a file in binary format, you need to create binary format definition files (#1, #2, and #3) for each record type. Note that because line feed codes are deleted when the file is input, you do not need to specify a terminal separator. You also need to create data transformation definition files (#5, #6, and #7) for each record type.

For details on how to create binary format definition files, see *4.4 Creating Message Formats (Binary Format Definition File)* in the manual *Service Platform Basic Development Guide*. For details on how to create data transformation definition files, see the description for displaying from the Eclipse menu in *6.3.1 Procedure for Defining Data Transformation* in the manual *Service Platform Basic Development Guide*.

#### 1. Binary format definition file for input header records (FDX file)

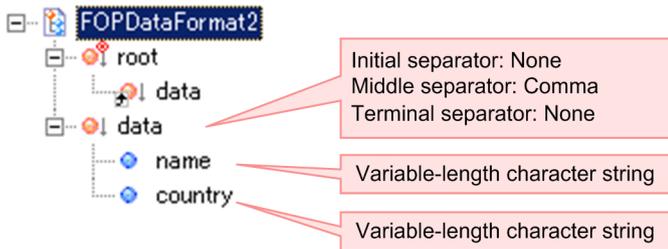
Create the binary format definition file as shown below. Specify `FOPHeaderFormat2.fdx` as the file name of the binary format definition file.

E. Example for defining file operations adapter



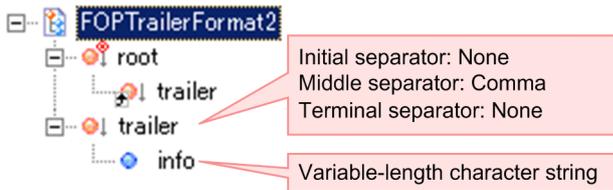
2. Binary format definition file for input data records (FDX file)

Create the binary format definition file as shown below. Specify FOPDataFormat2.fdx as the file name of the binary format definition file.



3. Binary format definition file for input trailer records (FDX file)

Create the binary format definition file as shown below. Specify FOPTrailerFormat2.fdx as the file name of the binary format definition file.

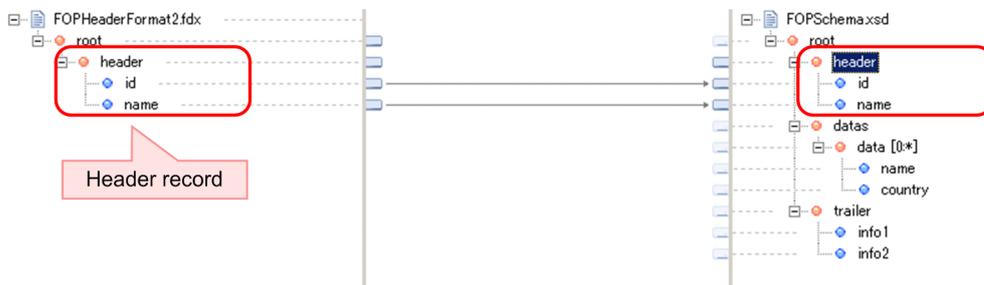


4. Output XML schema

The file names and contents are the same as for the input XML schema in E.1 When converting from the XML format to the binary format with the split processing method.

5. Data transformation definition file for header records (XSL file)

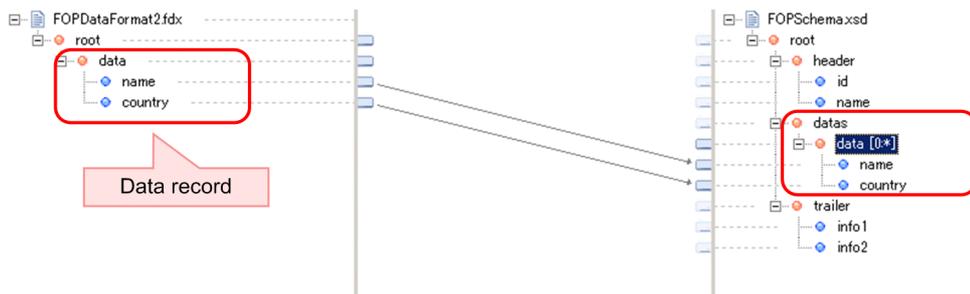
Create the data transformation definition file as shown below. Specify FOPHeader2.xsl as the file name of the XSL file. Specify FOPHeaderFormat2.fdx as the transformation-source schema and FOPSchema.xsd as the transformation-target schema.



In this example, the only nodes to be mapped are the header elements. For details on how to limit which nodes are subject to mapping, see 6.4.8 Restricting mapping range in the manual Service Platform Basic Development Guide.

6. Data transformation definition file for data records (XSL file)

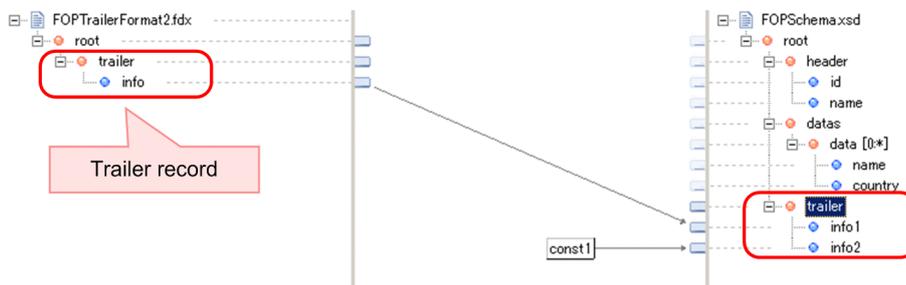
Create the data transformation definition file as shown below. Specify FOPData2.xsl as the file name of the XSL file. Specify FOPDataFormat2.fdx as the transformation-source schema, and FOPSchema.xsd as the transformation-target schema.



In this example, the only nodes to be mapped are the data elements.

7. Data transformation definition file for trailer records (XSL file)

Create the data transformation definition file as shown below. Specify FOPTrailer2.xsl as the file name of the XSL file. Specify FOPTrailerFormat2.fdx as the transformation-source schema, and FOPSchema.xsd as the transformation-target schema.



In this example, a set constant function is used to restrict mapping to the info1 element only.

8. XML file for output template

Use an application such as the WST (Web Standard Tools) provided by Eclipse to create an output template XML file. Specify xml\_output\_template.xml as the file name.

The records in the output template XML file below are replaced with transformed records when the file is output. Create the output template XML file according to the output XML schema of the records. Transformed records cannot be output unless the file conforms to the output XML schema.

```

<?xml version="1.0" encoding="UTF-8"?>
<root>
  <header/> Header record
  <datas>
    <data/> Data record
  </datas>
  <trailer/> Trailer record
</root>

```

9. File operation adapter definition file

In the Service adapter definition (details) screen, select cscFileOperation.properties in the Self-defined file area, and click Edit. Then, define the file as follows:

## E. Example for defining file operations adapter

```
#Specify use of the split processing method
csc.adapter.fileOperation.transform.splitLoad = ON

#Specify the line feed separator method
csc.adapter.fileOperation.transform.format = separate

#Specify whether to use header and trailer records
csc.adapter.fileOperation.transform.headerRecord = ON
csc.adapter.fileOperation.transform.trailerRecord = ON

#Define the input binary file
csc.adapter.fileOperation.transform.input = non-xml
csc.adapter.fileOperation.transform.header.inFormat = FOPHeaderFormat2.fdx
csc.adapter.fileOperation.transform.data.inFormat = FOPDataFormat2.fdx
csc.adapter.fileOperation.transform.trailer.inFormat = FOPTrailerFormat2.fdx

#Define the output XML file
csc.adapter.fileOperation.transform.output = xml
csc.adapter.fileOperation.transform.outputTemplateXmlFile = xml_output_template.xml
csc.adapter.fileOperation.transform.header.outElement = /root/header
csc.adapter.fileOperation.transform.data.outElement = /root/datas/data
csc.adapter.fileOperation.transform.trailer.outElement = /root/trailer

#Define the XSL file for each record type
csc.adapter.fileOperation.transform.header.styleSheet = FOPHeader2.xsl
csc.adapter.fileOperation.transform.data.styleSheet = FOPData2.xsl
csc.adapter.fileOperation.transform.trailer.styleSheet = FOPTrailer2.xsl
```

### 10. Service adapter definition

In the Service adapter definition (details) screen, add the following files as self-defined files:

- FOPHeaderFormat2.fdx
- FOPDataFormat2.fdx
- FOPTrailerFormat2.fdx
- FOPHeader2.xsl
- FOPData2.xsl
- FOPTrailer2.xsl
- xml\_output\_template.xml

## F. Example for setting up business processes using HTTP reception

This appendix provides an example of setting up an HTTP reception and business process, using a sample that returns distance traveled as a response when parameters (kilometers per hour and time elapsed) are passed to the HTTP reception.

Examples are given for standard mode and pass-through mode.

### F.1 Setup example of HTTP reception (in standard mode)

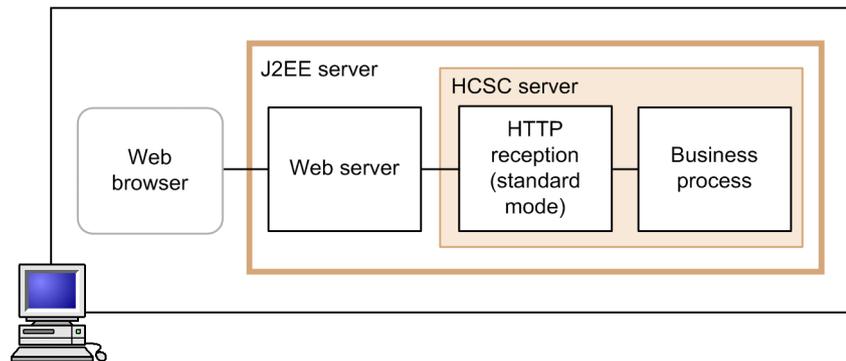
This section describes an example of setting up an HTTP reception for a business process executed in standard mode.

#### (1) Overview of setup example

This subsection describes the system configuration and flow of processing in this setup example.

The following figure shows an example of the system configuration:

Figure F–1: Configuration of setup example (standard mode)



In this example, the operator uses a Web browser installed on the machine where the HCSC server environment is set up. This example uses a Web server incorporated into the J2EE server. The HTTP reception uses standard mode.

The flow of this setup example is as follows:

1. The operator uses a `GET` method to invoke the HTTP reception from the Web browser.  
At this point, he or she sets kilometers per hour and time as parameters of the query string of the HTTP request.
2. The data transformation definitions in the business process are used to calculate the distance traveled, and the result is set in the body of the response message.
3. The headers specified in the HTTP Response Header definition file are set and the response is returned to the Web browser.

Header allocated variables are not used in this example. The setup procedure is described below.

#### (2) Creating the message formats for the HTTP reception

Create the message formats used in the HTTP reception. The message format templates are stored in the following folder:

- Location of template files

---

```
service-platform-installation-directory\CSC\custom-reception\http\schema
```

---

Copy the entire folder to another location before editing the files it contains. Use the copied files as the basis when creating the message formats.

(a) Creating the request message (body) format

Create the message format to set in body allocated variables in the HTTP reception during reception. Create the message format by editing the template.

Edit the file `urecp_http_body_detail_request.xsd`.

Note: Do not edit `urecp_http_body_request.xsd`.

In this example, you add the elements shown in italics which represent kilometers per hour and hours traveled.

- `urecp_http_body_detail_request.xsd`

---

```
<?xml version="1.0" encoding="UTF-8" ?>
<xsd:schema elementFormDefault="qualified"
targetNamespace="http://www.hitachi.co.jp/soft/xml/cosminexus/csc/reception/http/request"
xmlns:hrc="http://www.hitachi.co.jp/soft/xml/cosminexus/csc/reception/http/request"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:complexType name="http-body-requestType">
    <xsd:sequence>
      <!-- User Customize -->
      <xsd:element name="kilometers_per_hour" type="xsd:positiveInteger"/>
      <xsd:element name="hours" type="xsd:positiveInteger"/>
    </xsd:sequence>
  </xsd:complexType>
</xsd:schema>
```

---

HTTP reception header allocated variables are not used in this example.

(b) Creating the response message (body) format

Create the message format to set in body allocated variables in the HTTP reception when providing a response.

The message format below describes processing that returns the distance traveled. Create this format, and save it in the copied folder. The file name is `urecp_http_body_response.xsd`. Note that there is no template file for the response message (body) format.

- `urecp_http_body_response.xsd`

---

```
<?xml version="1.0" encoding="UTF-8" ?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  <xsd:element name="HTTPResponseBody">
    <xsd:complexType>
      <xsd:sequence>
        <xsd:element name="distance" type="xsd:positiveInteger"/>
      </xsd:sequence>
    </xsd:complexType>
  </xsd:element>
</xsd:schema>
```

---

(3) Adding the HTTP reception

To add an HTTP reception:

1. Right-click the Service Definition List in the tree view, and select **Add Business Process**.  
A business process is added. You do not define the contents of the business process at this stage.
2. In the Service Definition List in the tree view, right-click the business process you added, and select **Add User Defined Reception**.  
The Reception Type Selection wizard appears.
3. Select **HTTP Reception** as the reception type, and then click **Next**.
4. Specify a name for the reception in **Reception name**.
5. Click **Finish**.  
The HTTP reception is added to the business process, and the User-defined Reception Definition window appears.

#### (4) Defining the HTTP reception

The following shows the information you need to set in the User-defined Reception Definition window. For details on the User-defined Reception Definition window, see *1.2.6 User-Defined Reception Definition Window* in the manual *Service Platform Reference Guide*.

##### (a) Settings in the User Definition Reception window (basic) window

Enter the settings as shown in the table below.

Table F–1: Settings in the User Definition Reception window (basic) window

Category	Item	Value to set	
<b>User defined reception information</b>	<b>Reception name</b>	HTTP Reception	
	<b>Reception ID</b>	rcpl	
	<b>Reception type</b>	Custom reception	
	<b>Default operation name</b>	calculateDistance	
	<b>Operation</b>	calculateDistance	
<b>Operation information</b>	<b>Operation name</b>	calculateDistance	
	<b>Communication model</b>	<b>Sync</b>	
<b>Request message (Body)</b>	<b>Use any type</b> check box	Do not use (leave the check box cleared)	
	<b>Reception</b>	<b>Message format</b>	urecp_http_body_request.xsd
	<b>Service component</b>	<b>Use check box</b>	Do not use (leave the check box cleared)
		<b>Message format</b>	--
<b>Data-conversion definition</b>		--	
<b>Response message (Body)</b>	<b>Use any type</b> check box	Do not use (leave the check box cleared)	
	<b>Reception</b>	<b>Message format</b>	urecp_http_body_response.xsd (the file you created in (2) <i>Creating the message formats for the HTTP reception</i> )
	<b>Service component</b>	<b>Use check box</b>	Do not use (leave the check box cleared)
		<b>Message format</b>	--
	<b>Data-conversion definition</b>		--

Legend:

--: Do not set.

##### (b) Editing the HTTP Response Header definition file

The following describes how to edit the HTTP Response Header definition file. Note that because header allocated variables are not used in the business process in this example, a description of the properties is omitted.

1. In the **Self-defined file** area of the User Definition Reception window (detail), select `csurecphttp_header.properties` and then click **Edit**.  
An editor appears in which you can edit the self-defined file.
2. Amend the content of the HTTP Response Header definition file as follows:

---

```
Connection=close
extension-header=extension-header1
```

---

Note: `extension-header` is set to allow for arbitrary extension headers.

3. From the Eclipse menu, select **File** and then **Save**.  
The HTTP Response Header definition file is saved.

(c) Editing the HTTP reception definition file

The following describes how to edit the HTTP reception definition file.

1. In the **Self-defined file** area of the User Definition Reception window (detail), select `cscurecphhttp.properties` and then click **Edit**.

An editor appears in which you can edit the self-defined file.

2. Amend the content of the HTTP reception definition file as follows:

```
#urecp-http.context-root=
#urecp-http.max-threads=10
#urecp-http.exclusive-threads=0
#urecp-http.queue-size=8192
#urecp-http.pooled-instance.minimum=0
#urecp-http.pooled-instance.maximum=0
#urecp-http.ejb-transaction-timeout=0
#httprecp.switchover.pass-through.mode=false
httprecp.http.charset=UTF-8
httprecp.response.header.filename=cscurecphhttp_header.properties
#httprecp.system-exception.status-code=500
#httprecp.response.generate.content-length=true
```

Because the context root (`urecp-http.context-root`) is omitted in this example, the reception ID of the HTTP reception is used as the default value.

3. From the Eclipse menu, select **File** and then **Save**.

The HTTP reception definition file is saved.

(5) Defining the business process

The following describes how to define the business process.

(a) Creating the business process

In this example, you create a business process in which a request message passed to an HTTP reception is transformed by a data transformation activity to generate a response message.

Arrange and connect the activities as shown in the following figure:

Figure F–2: Example of business process creation



For details on how to arrange and connect activities, see 5.4 *Deploying and Linking Activities* in the manual *Service Platform Basic Development Guide*.

(b) Defining variables

Define the variables used in the business process.

The following table shows the variables to define:

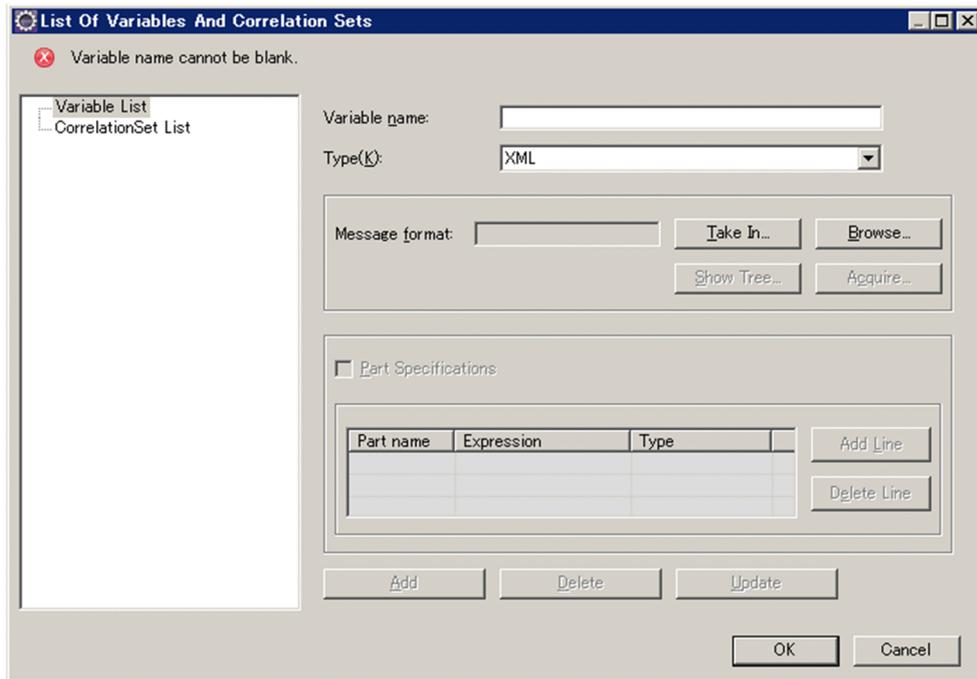
Variable name	Description
<code>request_body</code>	The request message (body) variable for the HTTP reception
<code>response_body</code>	The response message (body) variable for the HTTP reception

Correlation sets are not used in this example.

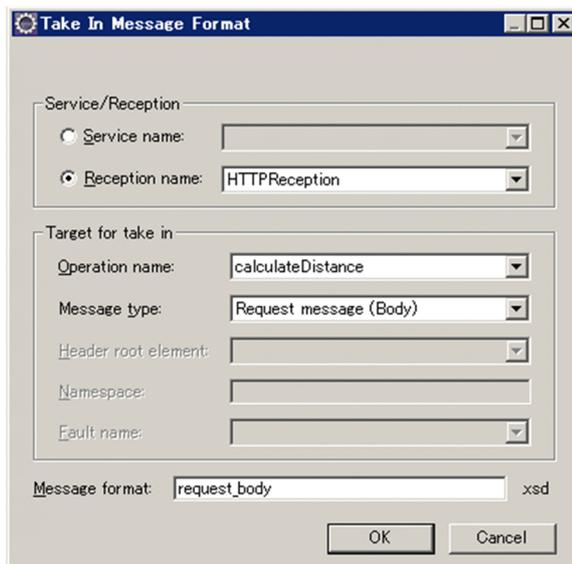
The definition of these variables takes place in the Take In Message Format dialog box. To define these variables:

1. Double-click the Variable-Correlation icon on the canvas of the Define Business Process window.

The List Of Variables And Correlation Sets dialog box appears.



2. In **Variable name**, enter `request_body` or `response_body`.
3. From the **Type**: drop-down list, select **XML**.
4. In the **Message format** area, click the **Take In** button.  
The Take In Message Format dialog box appears.
5. Set the following variables:



- `request_body`

Category	Item	Value to set
Service/Reception	Service name	--
	Reception name	HTTP Reception
Target for take in	Operation name	calculateDistance

F. Example for setting up business processes using HTTP reception

Category	Item	Value to set
Target for take in	Message type	Request message (Body)
	Fault name	--
--	Message format	request_body

- response\_body

Category	Item	Value to set
Service/Reception	Service name	--
	Reception name	HTTP Reception
Target for take in	Operation name	calculateDistance
	Message type	Response message (Body)
	Fault name	--
--	Message format	response_body

Legend:

--: Do not set. Alternatively, there is no corresponding category.

6. Click **OK**, and in the List Of Variables And Correlation Sets dialog box, click **Add**.

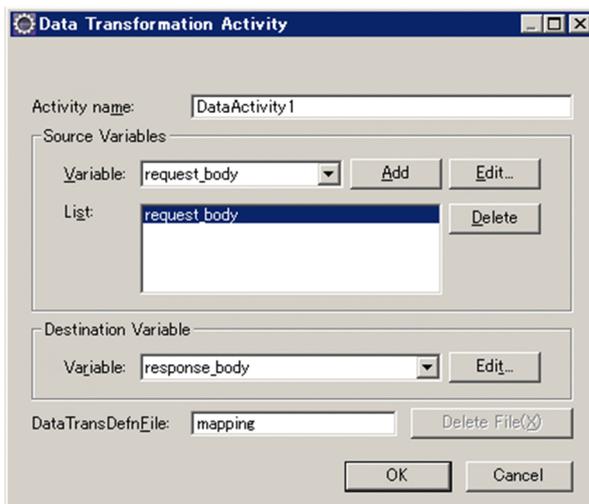
The variable you defined is added.

(c) Defining data transformation

In the data transformation, a perform node operation function (`calc1`) determines the product of the two parameters in the request message, and maps the resulting value to the response message.

To define this data transformation:

1. Double-click the data transformation activity on the canvas of the Define Business Process window.  
The Data Transformation Activity dialog box appears.
2. In the **Source Variables** area, specify `request_body`.
3. In the **Destination Variable** area, specify `response_body`.
4. In the **Data TransDefn File** field, specify the name you want to give the data transformation definition.



5. Click **OK**.

6. Double-click the data transformation activity.

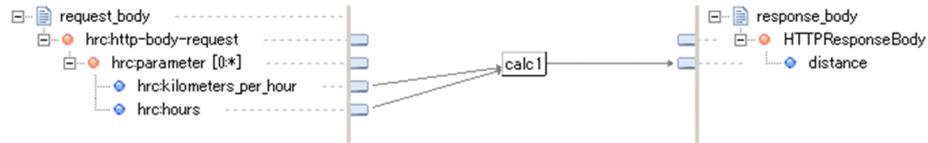
The Select Root Element dialog box appears.

7. Click **OK**.

The Data-conversion definition screen appears.

8. Select the perform node operation function (`calc`) from the palette, and place it in the mapping viewer.

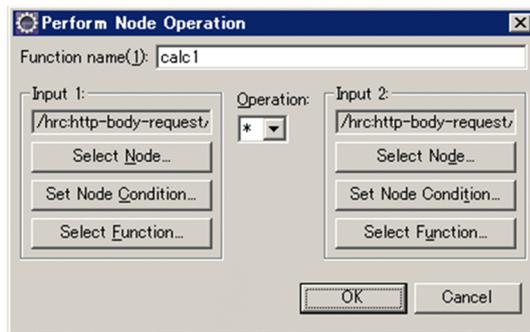
9. Set the mapping lines as shown in the following figure.



10. Double-click the perform node operation function.

The Perform Node Operation dialog box appears.

11. From the **Operation** drop-down list, select the operator you want to use (\* in this example).



12. Click **OK**.

This completes the process of defining the data transformation.

(d) Setting up receive and reply activities

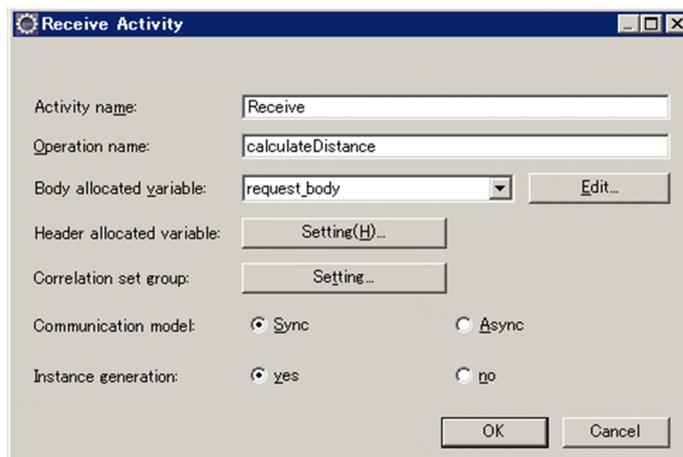
The following describes how to set up the receive and reply activities.

■ Example of setting receive activity

1. Double-click the receive activity on the canvas of the Define Business Process window.

The Receive Activity dialog box appears.

2. Enter the settings as shown below.



Item	Value to set
Activity name	Receive

F. Example for setting up business processes using HTTP reception

Item	Value to set
<b>Operation name</b>	calculateDistance
<b>Body allocated variable</b>	request_body
<b>Header allocated variable</b>	--
<b>Correlation set group</b>	--
<b>Communication model</b>	<b>Sync</b>
<b>Instance generation</b>	<b>Yes</b>

Legend:

--: Do not set.

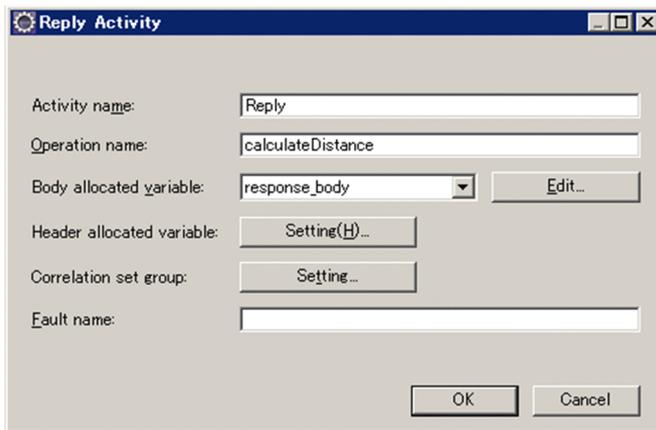
3. Click **OK**.

■ Example of setting reply activity

1. Double-click the reply activity on the canvas of the Define Business Process window.

The Reply Activity dialog box appears.

2. Enter the settings as shown below.



Item	Value to set
<b>Activity name</b>	<b>Reply</b>
<b>Operation name</b>	calculateDistance
<b>Body allocated variable</b>	response_body
<b>Header allocated variable</b>	--
<b>Correlation set group</b>	--
<b>Fault name</b>	--

Legend:

--: Do not set.

3. Click **OK**.

(6) Invoking the HTTP reception

The following describes how to execute the business process you created.

## (a) Preparation

1. From the Eclipse menu, select **File** and then **Save All** to save the HCSC component you defined.
2. From the Windows Start menu, select **Cosminexus** and then **Start Test Server**.
3. Right-click the Service Definition List in the tree view, and select **Deploy all services to server and start**.  
The repository that includes the business process and the HTTP reception is deployed to the execution environment.

## (b) Invoking the HTTP reception

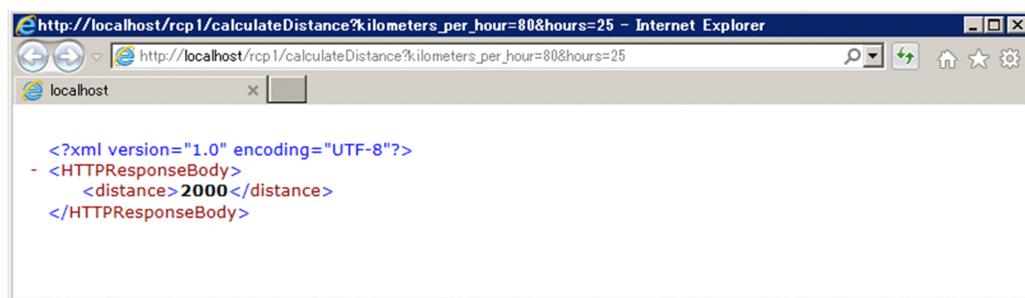
Open your Web browser, and specify the URL below in the address bar. You can change the values in the query string as needed.

---

```
http://localhost/rcp1/calculateDistance?kilometers_per_hour=80&hours=25
```

---

The execution results are displayed in the Web browser, as shown in the following figure:



## F.2 Setup example of HTTP reception (in pass-through mode)

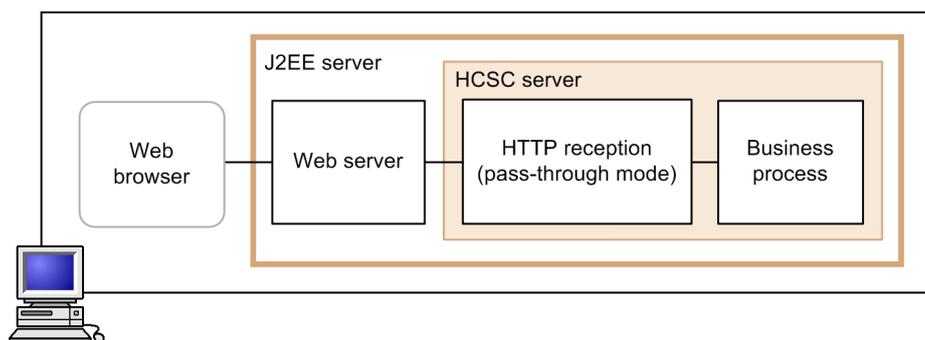
This section describes an example of setting up an HTTP reception for a business process executed in pass-through mode.

### (1) Overview of setup example

This subsection describes the system configuration and flow of processing in this setup example.

The following figure shows an example of the system configuration:

Figure F-3: System configuration in setup example (pass-through mode)



In pass-through mode, the business process is invoked by directly specifying the request message (XML) of the HTTP reception in a Web browser.

### (2) Creating the message format of the HTTP reception

Use the same procedure as in *F.1(2) Creating the message formats for the HTTP reception*.

### (3) Adding an HTTP reception

Use the same procedure as in *F.1(3) Adding the HTTP reception*.

### (4) Defining the HTTP reception

#### (a) Settings in the User Definition Reception window (basic)

Use the same settings as in *F.1(4)(a) Settings in the User Definition Reception window (basic) window*.

#### (b) Editing the HTTP Response Header definition file

Use the same procedure as in *F.1(4)(b) Editing the HTTP Response Header definition file*.

#### (c) Editing the HTTP reception definition file

Use the same procedure as in *F.1(4)(c) Editing the HTTP reception definition file*. However, for pass-through mode, specify `true` for the `httprecp.switchover.pass-through.mode` property. The following is an example of the information you need to set:

---

```
#urecp-http.context-root=
#urecp-http.max-threads=10
#urecp-http.exclusive-threads=0
#urecp-http.queue-size=8192
#urecp-http.pooled-instance.minimum=0
#urecp-http.pooled-instance.maximum=0
#urecp-http.ejb-transaction-timeout=0
httprecp.switchover.pass-through.mode=true
httprecp.http.charset=UTF-8
httprecp.response.header.filename=cscurecphhttp_header.properties
#httprecp.system-exception.status-code=500
#httprecp.response.generate.content-length=true
```

---

### (5) Defining the business process

Use the same procedure as in *F.1(5) Defining the business process*.

### (6) Invoking the HTTP reception

In pass-through mode, you directly specify the request message of the HTTP reception in your Web browser. In this example, the HTTP reception is invoked using the following HTML (POST method):

1. Create the following HTML file and display it in your Web browser:

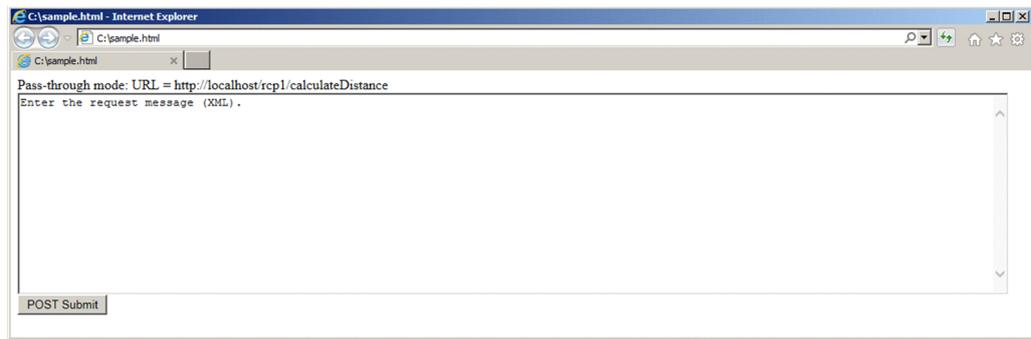
- `sample.html`

---

```
<html>
  <head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
  </head>
  <body>
    <form method="POST" action="http://localhost/rcpl/calculateDistance">
      <label>Pass-through mode: URL = http://localhost/rcpl/calculateDistance</label><br>
      <textarea name=msg rows=15 cols=140 wrap=soft>Enter the request message (XML). </
textarea><br>
      <input type="submit" value="POST Submit" />
    </form>
  </body>
</html>
```

---

- Page displayed in Web browser:



2. Enter the request message for the HTTP reception in the input area.

Because this example uses the same business process as the example for standard mode, specify the request message you defined for that example in *F.1(2) Creating the message formats for the HTTP reception*.

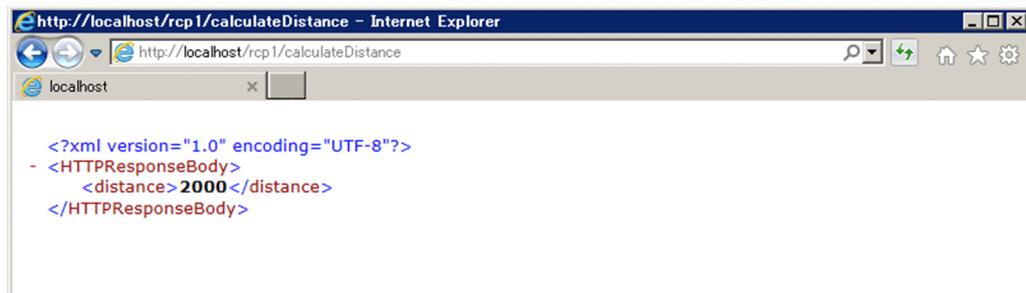
---

```
<?xml version="1.0" encoding="UTF-8" ?>
<hrc:http-body-request
  xmlns:hrc="http://www.hitachi.co.jp/soft/xml/cosminexus/csc/reception/http/request"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.hitachi.co.jp/soft/xml/cosminexus/csc/reception/http/
request_urecp_http_body_request.xsd ">
  <hrc:parameter>
    <hrc:kilometers_per_hour>80</hrc:kilometers_per_hour>
    <hrc:hours>25</hrc:hours>
  </hrc:parameter>
</hrc:http-body-request>
```

---

3. Click the **Execute by POST** button.

The results appear in the Web browser. The results are as follows:



## G. Example for setting up business processes using HTTP reception and HTTP adapter

This appendix describes how to set up business processes that use an HTTP reception and an HTTP adapter, using the example of a system that performs the following processing:

- Downloading a file from an HTTP server to an HTTP client
- Dynamically changing the connection destination of an HTTP adapter for individual requests

### ! Important note

The instructions in this appendix assume that the business process will be executed in an execution environment (test environment) in the development environment.

### G.1 Setup example 1 (downloading the file from the HTTP server to the HTTP client)

#### (1) Overview of setup example

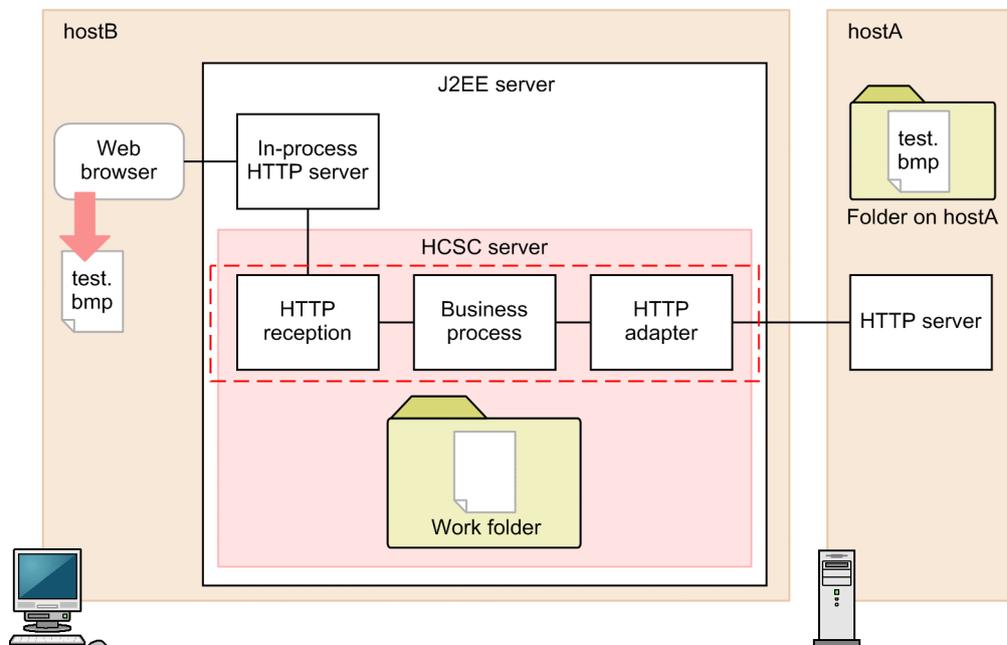
In this setup example, you will define a business process that performs the following process:

1. Access the HTTP server `hostA` from a HTTP adapter, and download a file (`test.bmp`) on the host to the work folder.
2. Download `test.bmp` to the HTTP client from the work folder via a HTTP reception.

#### (a) Environment configuration

The following figure shows an example of the environment created in this setup example:

Figure G–1: Environment configuration created in setup example 1



This setup example describes the procedure for defining the elements in the box with the dashed border in the figure.

A Web browser serves as the HTTP client.

The work folder will be used to store the received file.

**! Important note**

The HTTP server used in this setup example must have been set up in advance. It needs to be configured such that the HTTP client can download `test.bmp` by accessing `http://hostA/test.bmp`.

The HTTP server must be configured to append a Content-Disposition header to the response header returned to the HTTP adapter. This allows the HTTP adapter to recognize that the file data has been downloaded.

**(b) Flow of setup procedure**

The setup procedure consists of the following steps:

1. Setting up the execution environment (test environment)
2. Setting up the development environment
3. Setting the definition files
4. Defining the HTTP adapter
5. Adding a business process
6. Adding an HTTP reception
7. Defining the business process
8. Defining the data transformation
9. Preparing to execute the business process
10. Executing the business process

Each step is described in detail below.

**(2) Setting up the execution environment (test environment)**

Use the HCSC Easy Setup function to create the J2EE server and HCSC server.

1. Create the test environment by following the instructions in *2.4.2 Executing HCSC easy setup functionality* in the *Service Platform Basic Development Guide*.  
You must select **SOAP1.1/1.2 combined mode** under **HCSC server** on the **Main** tab. Selecting **SOAP1.1/1.2 combined mode** is a prerequisite for using the HTTP adapter.  
Whether you select **Model with DB/without RM**, **Model without DB/RM**, or **Model with DB/RM** on the **Main** tab has no bearing on whether the HTTP adapter can be executed.
2. Set the option definition file for the J2EE server (`usrconf.cfg`) by following the instructions in *3.3.13(1) Setting up the option definition file for J2EE servers*.
3. Stop the J2EE server and HCSC server in the test environment by selecting **Programs**, **Cosminexus**, and then **Stop Test Server** from the **Start** menu.
4. Start the J2EE server and HCSC server again in the test environment by selecting **Programs**, **Cosminexus**, and then **Start Test Server**.

**(3) Setting up the development environment**

Set up the development environment by following the procedure below.

1. First, create the HCSCTE project by following the procedures in *(1) Creating HCSCTE projects* to *(3) Importing the system configuration definition into the development environment* in *3.5.7 Deploying definitions to the HCSC server* in the *Service Platform First Step Guide*. Then, export the system configuration definition from the test environment and import it to the development environment.

**(4) Setting the definition files**

Set the definition files required for the HTTP reception and HTTP adapter to operate in the test environment.

G. Example for setting up business processes using HTTP reception and HTTP adapter

(a) Definition files for HTTP reception

The following table shows the definition file types used by an HTTP reception, and whether each file type is used in this setup example.

Type of definition file	Description	Used in this setup example
HTTP Response Header definition file	Sets the HTTP Response Header information used as the default values for header variables when the definition of the HTTP reception omits the response message format.	Not used.
HTTP reception definition file	Sets information about the operation of the HTTP reception.	Used.

Because this setup example uses a work folder to store the received file, set the following properties in the HTTP reception definition file:

Property name	Value to set	Description
<code>httprecp.switchover.file-transfer.mode</code>	<code>true</code>	Set <code>true</code> for this property to generate the work folder.
<code>httprecp.response.ignore-bodymsg</code>	<code>true</code>	Set <code>true</code> for this property so that the response of the HTTP reception is the file in the work folder, not the body of the response message.

You do not need to set values for properties that are not shown in the table.

For details on each property, see *HTTP reception definition file* in the manual *Service Platform Reference Guide*.

Editing the HTTP reception definition file is performed a part of (6) *Adding a business process*.

(b) Definition files for HTTP adapter

The following table shows the definition file types used by an HTTP adapter, and whether each file type is used in this setup example.

Type of definition file	Description	Used in this setup example
HTTP-adapter definition file	Sets operation information for the HTTP adapter that is not environment-dependent.	Not used (default values are used for all properties).
HTTP-adapter runtime-environment property file	Sets operation information for the HTTP adapter that is environment-dependent.	Used.
HTTP-adapter runtime-environment common property file	Sets operation information for the HTTP adapter that is environment-dependent. Use this type of file to change the definitions of multiple adapters as a batch.	Not used.

Although a HTTP adapter would usually require system properties to be set in addition to the properties in these definition files, use of these system properties is not part of this setup example.

The following table shows the properties to set in the HTTP-adapter runtime-environment property file:

Property name	Value to set	Description
<code>adphttp.request.method</code>	<code>GET</code>	Because the file download request is implemented as a <code>GET</code> method, set <code>GET</code> as the value of this property.
<code>adphttp.request.uri-scheme-authority</code>	Set a value appropriate to the environment in which the HTTP adapter operates.	Specify the URI scheme and authority. The connection target you specify must have been prepared in advance. In this setup example, specify <code>http://hostA</code> .

Property name	Value to set	Description
<code>adphttp.request.uri-path</code>	Set a value appropriate to the environment in which the HTTP adapter operates.	Specify the URI path. In this setup example, set <code>/test.bmp</code> as the path used to download the <code>test.bmp</code> file.
<code>adphttp.request.output-folder-name</code>	-- (do not set)	Because the file downloaded in this setup example is output to the work folder, you do not need to set a value for this parameter.

You do not need to set values for properties that are not shown in the table.

For details on each property, see *HTTP-adapter runtime-environment property file* in the manual *Service Platform Reference Guide*.

To set the HTTP-adapter runtime-environment property file:

1. Copy the template file (*service-platform-installation-directory*\CSC\custom-adapter\HTTP\config\templates\serviceid.properties) to the following folder:  
*service-platform-installation-directory*\CSC\custom-adapter\HTTP\config
2. Open the copied file in a text editor or other tool, and edit its contents as follows:

```
adphttp.request.method=GET
adphttp.request.uri-scheme-authority=http://hostA
adphttp.request.uri-path=/test.bmp
#adphttp.request.uri-query.<INDEX>=<QUERY-NAME>=<QUERY-VALUE>
#adphttp.request.header.authorization.type=none
#adphttp.request.header.authorization=
#adphttp.request.header.content-type.charset=
#adphttp.request.header.content-type=
#adphttp.request.header.userdef.<INDEX>=<FIELD-NAME>:<FIELD-VALUE>
#adphttp.request.part.message.binding=none
#adphttp.request.part.file.1.input-folder-name=
#adphttp.request.part.file.1.local-file-name=
#adphttp.request.output-folder-name=
#adphttp.config.trace.path=
#adphttp.config.messagelog.level=10
#adphttp.config.methodtrace.level=3
#adphttp.config.methodtrace.filenum=8
#adphttp.config.methodtrace.filesize=2097152
#adphttp.config.exptrace.filenum=8
#adphttp.config.exptrace.filesize=2097152
```

3. Change the file name of the copied file to `adphttp.properties`.

`adphttp` is the service ID of the HTTP adapter. This service ID is used to define the HTTP adapter in the development environment.

### (c) Setting the HCSC server runtime definition file

Because this setup example uses the root of the default work folder, there is no need to set an HCSC server runtime definition file.

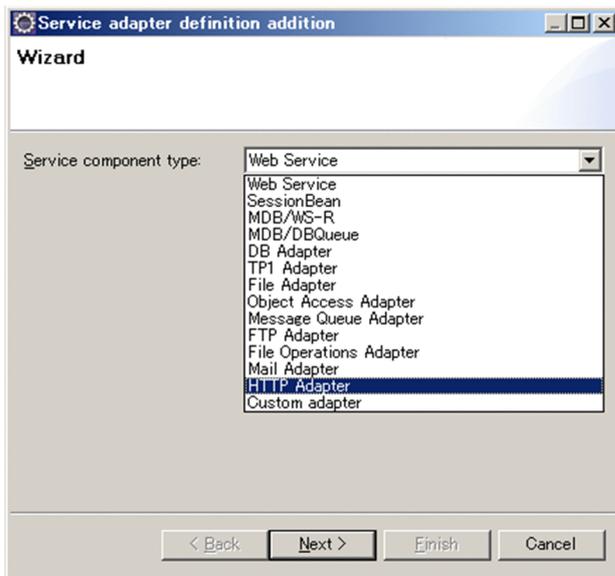
For details on the work folder, see *2.13.6 Managing HTTP request files* in the manual *Service Platform Overview*.

## (5) Defining the HTTP adapter

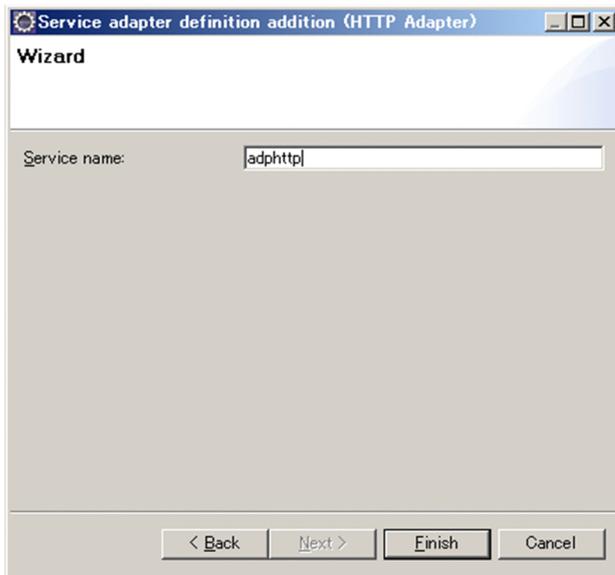
Define the HTTP adapter in the development environment. To define the HTTP adapter:

1. From the Eclipse menu, select **Window, Show View**, and then **Other**.  
The Show View dialog box appears.
2. Select **HCSC-Definer** and then **HCSCTE View**, and click **OK**.  
The Service Definition List appears in the tree view.
3. Right-click the Service Definition List in the tree view, and select **Add Service Adapter**.  
A dialog box appears in which you can set the type of service to implement from the service adapter you are adding.

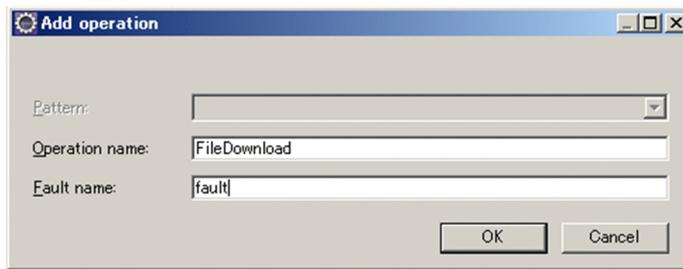
G. Example for setting up business processes using HTTP reception and HTTP adapter



4. From the **Service component type:** drop-down list, select **HTTP Adapter** and then click **Next**.  
A dialog box appears in which you can enter the information needed to add the HTTP adapter.
5. Enter the service name.  
In this example, use `adphttp` as the service name.



6. Click **Finish**.  
The Service adapter definition (standard) screen appears.
7. Click the **Add** button beside **Operation**, and enter the operation name and fault name.  
As the operation name, specify `FileDownload`, and as the fault name, specify `fault`.



8. In the **Add operation** dialog box, click **OK**.

9. In the **Request message** area, click the **Browse** button beside the **Message format** field in the **Service component** area, and specify the following format definition file:

```
service-platform-installation-directory\CSC\custom-adapter\HTTP\schema
\adphttp_body_empty.fdx
```

Note:

Because the HTTP adapter in this setup example only receives a file, you can use the empty message format definition files provided by Service Platform to define the bodies of the request and response messages.

10. In the **Request message** area, specify a value in the **Format ID** field in the **Service component** area.

In this example, use `format_req` as the format ID.

11. In the **Response message** area, click the **Browse** button beside the **Message format** field in the **Service component** area, and specify the following format definition file:

```
service-platform-installation-directory\CSC\custom-adapter\HTTP\schema
\adphttp_body_empty.fdx
```

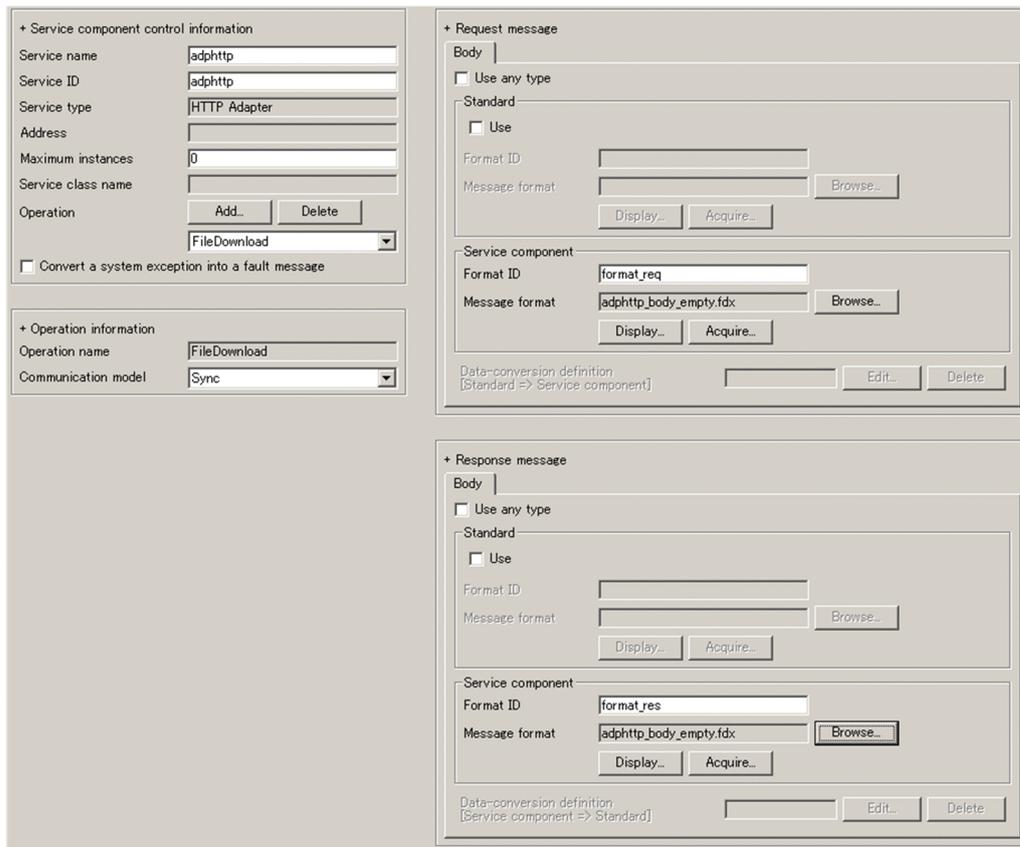
12. In the **Response message** area, specify a value in the **Format ID** field in the **Service component** area.

In this example, use `format_res` as the format ID.

13. In the **Service component control information** area, specify `adphttp` in the **Service ID** field.

The following figure shows an example of the Service adapter definition (standard) tab with the above information filled in:

G. Example for setting up business processes using HTTP reception and HTTP adapter



Do not change the option selected in the **Communication model** drop-down list from **Sync**. Note that you do not need to enter settings on the Service adapter definition (details) tab in this example. The default settings are used as-is.

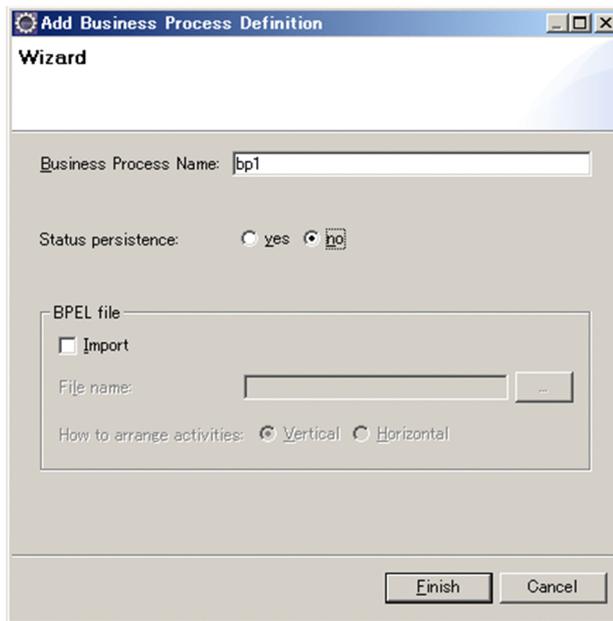
14. From the **File** menu, select **Save**.  
The HTTP adapter definition is saved.

(6) Adding a business process

Add a business process in the development environment. To add a business process:

1. Right-click the Service Definition List in the tree view, and select **Add Business Process**.  
A dialog box appears in which you can add a business process definition.
2. Enter the settings as shown below.

Item	Setting
<b>Business process name</b>	Enter bp1.
<b>Status Persistence</b>	Select no.
<b>BPEL file</b>	Leave the <b>Import</b> check box cleared.



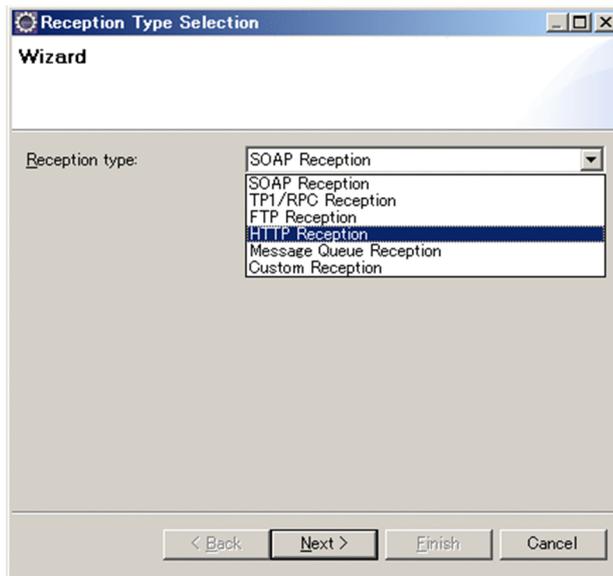
3. Click **Finish**.

The business process is added, and the Define Business Process window appears.

### (7) Adding an HTTP reception

Add the HTTP reception that will receive the requests from the HTTP client. To add an HTTP reception:

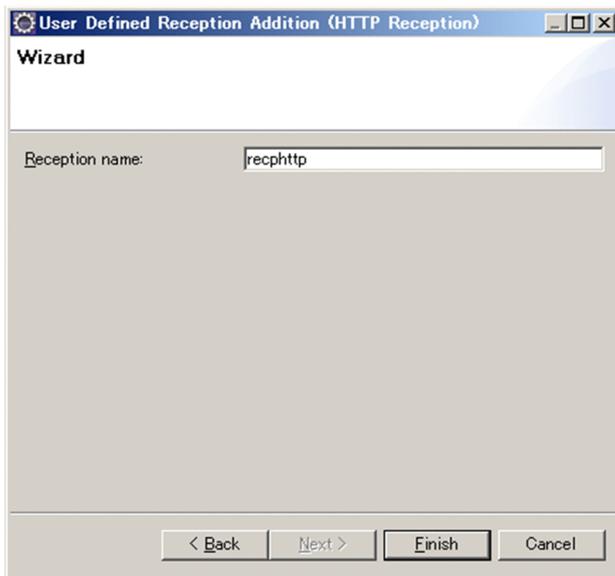
1. Right-click the business process (bp1) in the tree view, and select **Add User Defined Reception**.  
The Reception Type Selection wizard appears.
2. From the **Reception type:** drop-down list, select **HTTP Reception**, and then click **Next**.



A dialog box appears in which you can enter the information required to add the HTTP reception.

3. Specify a reception name in the **Reception name** field.  
In this example, use `recphttp` as the reception name.

G. Example for setting up business processes using HTTP reception and HTTP adapter

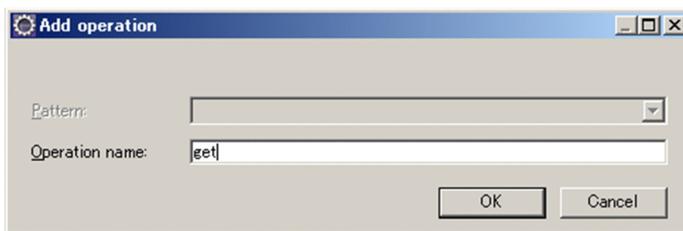


4. Click **Finish**.

The User-defined Reception Definition screen appears with the **User defined reception (standard)** tab displayed.

5. Click the **Add** button beside **Operation**, and enter the operation name.

In this example, use `get` as the operation name.



6. In the **Add operation** dialog box, click **OK**.

7. In the **Request message** area, click the **Browse** button beside the **Message format** field in the **Reception** area, and specify the following format definition file:

```
service-platform-installation-directory\CSC\custom-reception\http\schema  
\urecp_http_body_request.xsd
```

8. In the **Response message** area, click the **Browse** button beside the **Message format** field in the **Reception** area, and specify the following format definition file:

```
service-platform-installation-directory\CSC\custom-reception\http\schema  
\urecp_http_dummy_body_response.xsd
```

Do not change the option selected in the **Communication model** drop-down list from **Sync**.

The following figure shows an example of the User defined reception (standard) tab with this information entered:

9. Click the **User defined reception (details)** tab.

The User defined reception (details) tab appears.

10. In the **Self-defined file** area, select `cscurecphttp.properties` and click **Edit**.

An editor appears in which you can edit the self-defined file.

11. Amend the content of the HTTP reception definition file as follows:

---

```
#urecp-http.context-root=
#urecp-http.max-threads=10
#urecp-http.exclusive-threads=0
#urecp-http.queue-size=8192
#urecp-http.pooled-instance.minimum=0
#urecp-http.pooled-instance.maximum=0
#urecp-http.ejb-transaction-timeout=0
#httprecp.switchover.pass-through.mode=false
httprecp.switchover.file-transfer.mode=true
#httprecp.http.charset=UTF-8
httprecp.response.ignore-bodymsg=true
#httprecp.response.header.filename=
#httprecp.system-exception.status-code=500
#httprecp.response.generate.content-length=true
#httprecp.file-trans.temp-file.partsize-threshold=0
#httprecp.file-trans.maxsize.part=2147483647
#httprecp.file-trans.maxsize.request=2147483647
#httprecp.work-dir.auto-delete=true
#httprecp.response.download.disposition-type=attachment
```

---

12. From the **File** menu, select **Save All**.

The definition of the HTTP reception and the HTTP reception definition file are saved.

(8) Defining the business process

Define the flow of the business process.

(a) Defining variables

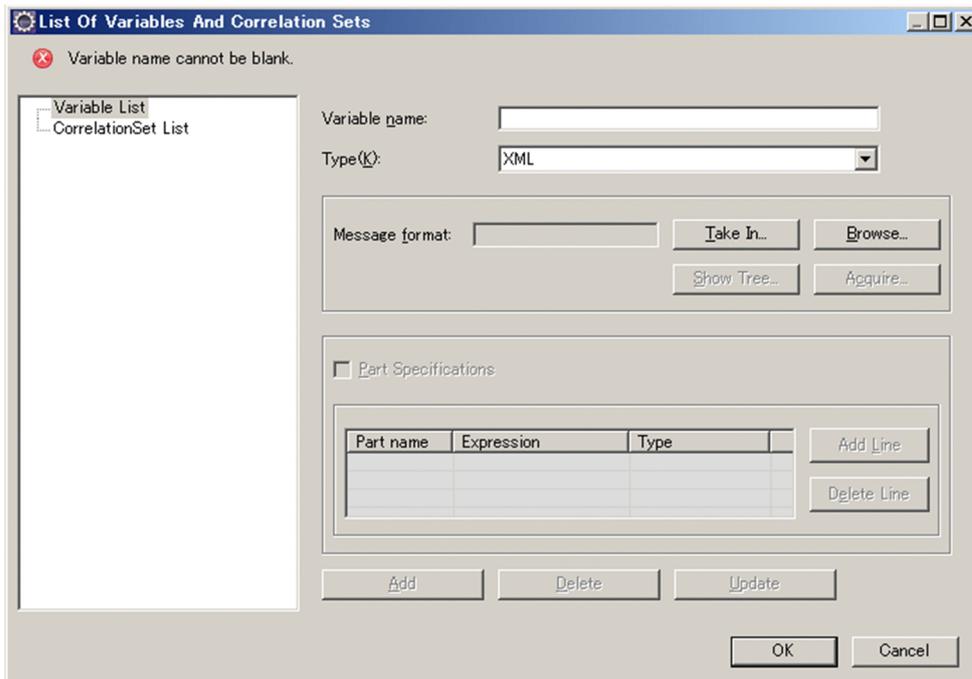
Individually define each variable used in the business process. The following table lists the variables you need to define in this setup example:

Table G–1: List of variables to define

#	Variable name	Type	Purpose
1	recphttp_head_request	XML	Variable for request message (header) of HTTP reception
2	recphttp_head_response	XML	Variable for response message (header) of HTTP reception
3	adphttp_head_request	XML	Variable for request message (header) of HTTP adapter
4	adphttp_head_response	XML	Variable for response message (header) of HTTP adapter
5	recphttp_body_request	XML	Variable for request message (body) of HTTP reception
6	recphttp_body_response	XML	Variable for response message (body) of HTTP reception
7	adphttp_body_request	non-XML	Variable for request message (body) of HTTP adapter
8	adphttp_body_response	non-XML	Variable for response message (body) of HTTP adapter

To define a variable:

1. Double-click the Variable-Correlation icon on the canvas of the Define Business Process window. The List Of Variables And Correlation Sets dialog box appears.



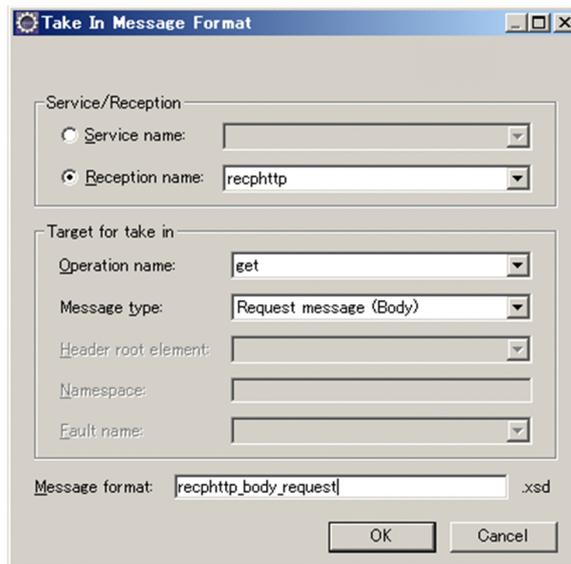
2. Perform the appropriate operations for the variable you are defining.

- For variables #1 to #4:
  - Specify the name and type of a variable from #1 to #4 in Table G-1, and then click **Browse**.
  - A dialog box appears in which you can specify the file to use as the message format. Specify the reference format definition shown in the following table:

#	Variable name	Reference format definition
1	recphttp_head_request	service-platform-installation-directory\CSC\custom-reception\http\schema\urecp_http_header_request.xsd
2	recphttp_head_response	service-platform-installation-directory\CSC\custom-reception\http\schema\urecp_http_header_response.xsd
3	adphttp_head_request	service-platform-installation-directory\CSC\custom-adapter\HTTP\schema\adphttp_header_request1.xsd
4	adphttp_head_response	service-platform-installation-directory\CSC\custom-adapter\HTTP\schema\adphttp_header_response1.xsd

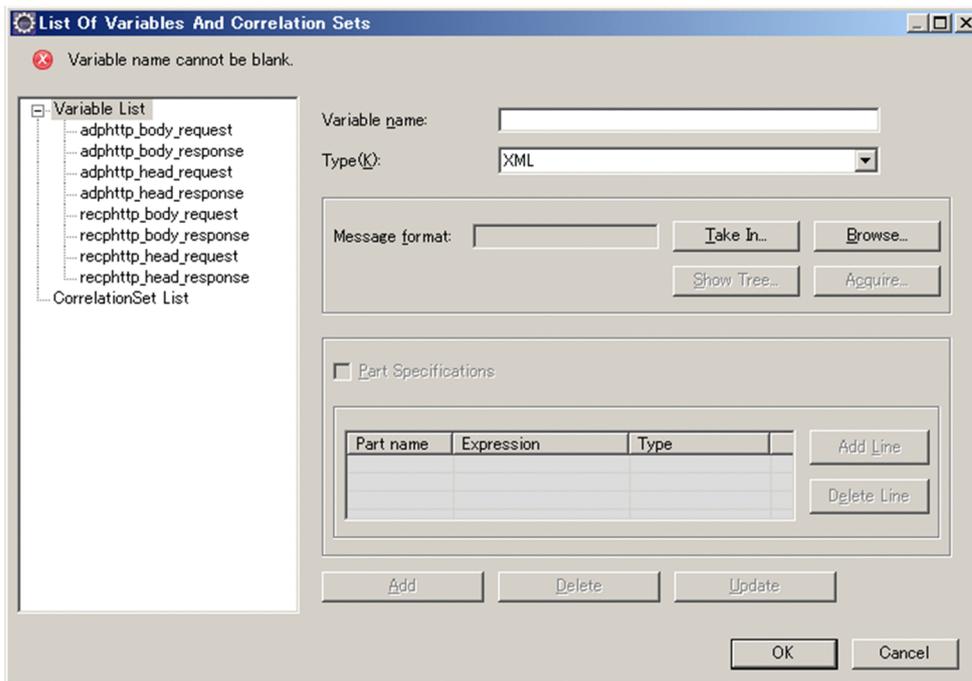
- For variables #5 to #8:  
 In the List Of Variables And Correlation Sets dialog box, specify the name and type of a variable from #5 to #8 in Table G-1, and then click **Take In**.  
 The Take In Message Format dialog box appears. Specify the settings as follows, and then click **OK**.

#	Variable name	Service/Reception	Operation name	Message type	Message format
5	recphttp_body_request	<b>Reception name</b> recphttp	get	<b>Request message (Body)</b>	recphttp_body_request.xsd
6	recphttp_body_response	<b>Reception name</b> recphttp	get	<b>Response message (Body)</b>	recphttp_body_response.xsd
7	adphttp_body_request	<b>Service name</b> adphttp	FileDownload	<b>Request message (Body)</b>	adphttp_body_request.fdx
8	adphttp_body_response	<b>Service name</b> adphttp	FileDownload	<b>Response message (Body)</b>	adphttp_body_response.fdx



- In the List Of Variables And Correlation Sets dialog box, click **Add**.  
 The variable you added appears in the list of variables. Follow these steps again until each of variables #1 to #8 has been defined.

G. Example for setting up business processes using HTTP reception and HTTP adapter



4. Click **OK**.

The List Of Variables And Correlation Sets dialog box closes.

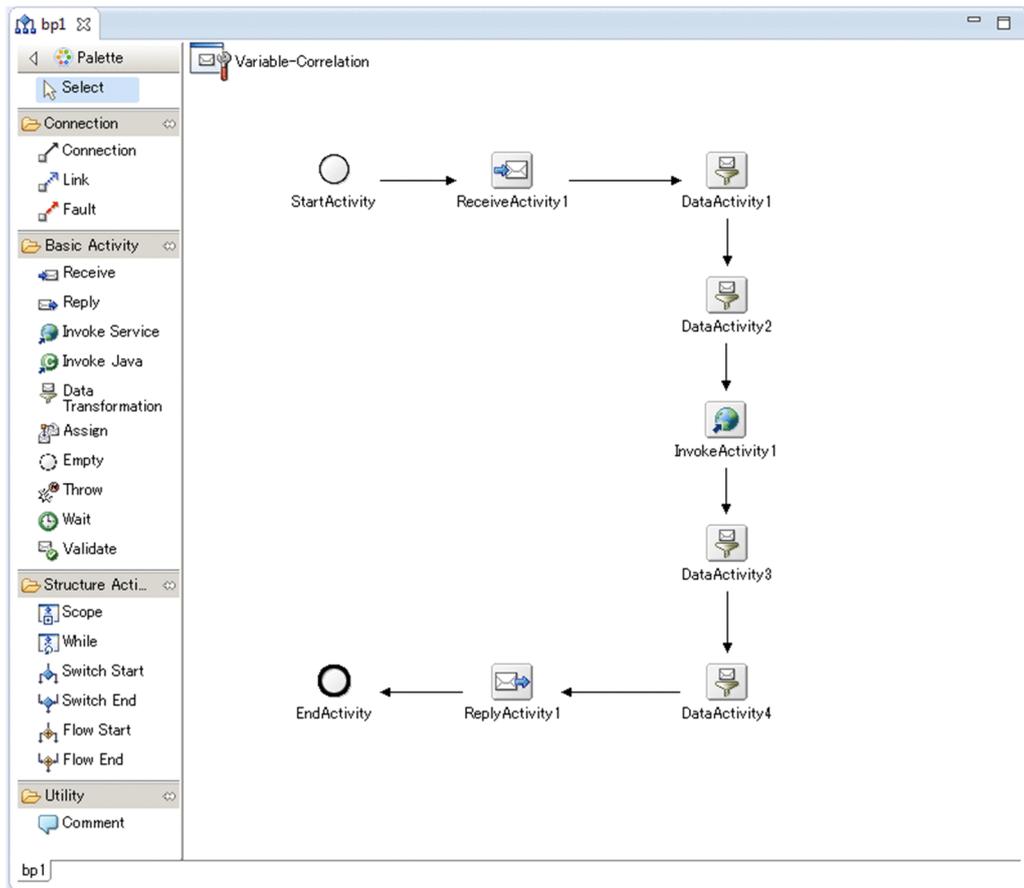
(b) Arranging activities

In the Define Business Process window, arrange the activities in the sequence shown in the following table, and establish the appropriate connections between the activities.

No.	Activity kind	Initial activity name	Description
1	Start	StartActivity	Represents the start of the business process. This activity appears on the canvas by default.
2	Reception	ReceptionActivity1	Defines the interface that receives request messages from the HTTP client.
3	Data transformation	DataActivity1	Defines data transformation processing.
4	Data transformation	DataActivity2	Defines data transformation processing.
5	Service invocation	InvokeActivity1	Defines the transmission of request messages to an existing HCSC component. In this setup example, the component is the HTTP adapter.
6	Data transformation	DataActivity3	Defines data transformation processing.
7	Data transformation	DataActivity4	Defines data transformation processing.
8	Response	ReplyActivity1	Defines the interface that returns a response to request messages received from the HTTP client.
9	Finish	EndActivity	Represents the end of the business process. This activity appears on the canvas by default.

For details on how to arrange activities on the canvas, see 5.4.1 *Deploying Activities* in the manual *Service Platform Basic Development Guide*.

The following figure shows the contents of the Define Business Process window after these activities have been arranged.



Note: You do not need to position the activities exactly as shown in the figure.

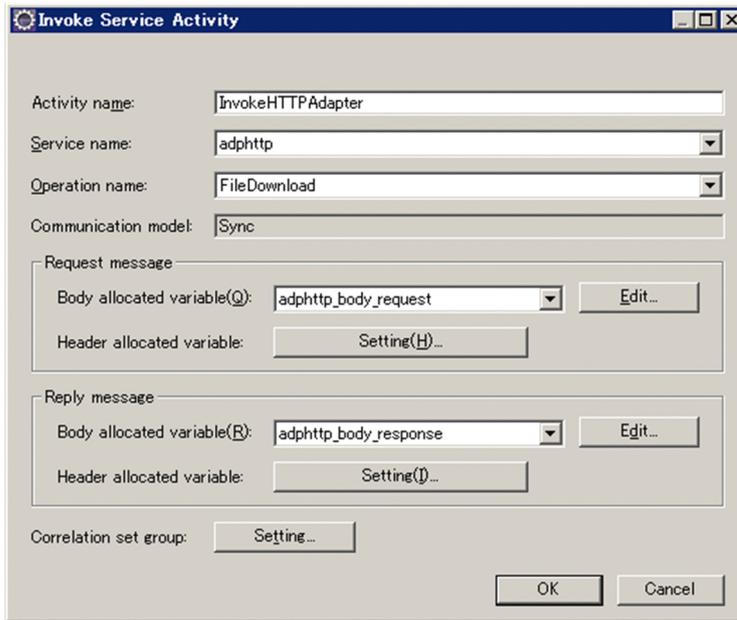
(c) Defining the invoke service activity

To define the invoke service activity that invokes the HTTP adapter:

1. Double-click the invoke service activity (InvokeActivity1).  
The Invoke Service Activity dialog box appears.
2. Enter the settings as shown below.

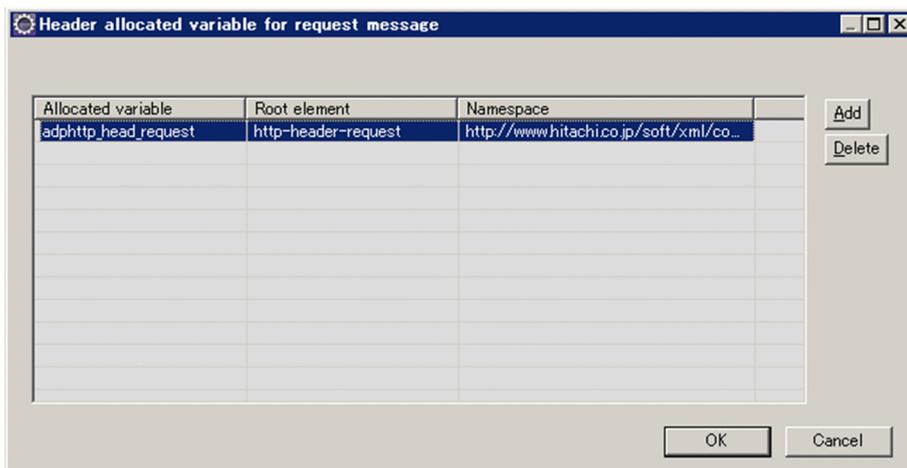
Item		Setting
<b>Activity name</b>		<b>InvokeHTTPAdapter</b>
<b>Service name</b>		adphttp
<b>Operation name</b>		FileDownload
<b>Communication model</b>		<b>Sync</b> (cannot be changed)
<b>Request message</b>	<b>Body allocated variable</b>	adphttp_body_request
<b>Response message</b>	<b>Body allocated variable</b>	adphttp_body_response

G. Example for setting up business processes using HTTP reception and HTTP adapter



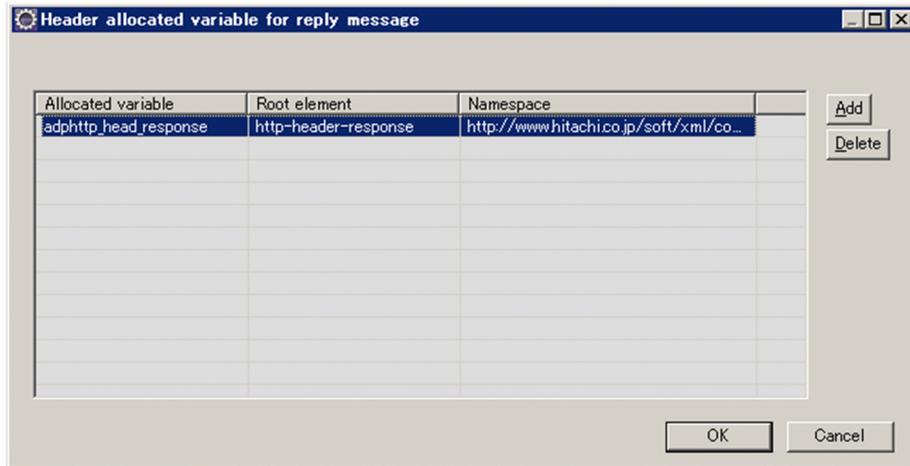
3. In the **Request message** area, click the **Setting** button beside **Header allocated variable**.  
The Header allocated variable for request message dialog box appears.
4. Click **Add**.
5. Enter the following settings in the Header allocated variable for request message dialog box:

Item	Setting
<b>Allocated variable</b>	adphttp_head_request
<b>Root element</b>	http-header-request
<b>Namespace</b>	http://www.hitachi.co.jp/soft/xml/cosminexus/csc/adapter/http/header_request



6. Click **OK** in the Header allocated variable for request message dialog box.  
The Header allocated variable for request message dialog box closes.
7. In the **Response message** area, click the **Setting** button beside **Header allocated variable**.  
The Header allocated variable for reply message dialog box appears.
8. Click **Add**.
9. Enter the following settings in the Header allocated variable for reply message dialog box:

Item	Setting
<b>Allocated variable</b>	adphttp_head_response
<b>Root element</b>	http-header-response
<b>Namespace</b>	http://www.hitachi.co.jp/soft/xml/cosminexus/csc/adapter/http/header_response



10. Click **OK** in the Header allocated variable for reply message dialog box.  
The Header allocated variable for reply message dialog box closes.
11. Click **OK** in the Invoke service activity dialog box.  
The settings are saved and the Invoke service activity dialog box closes.

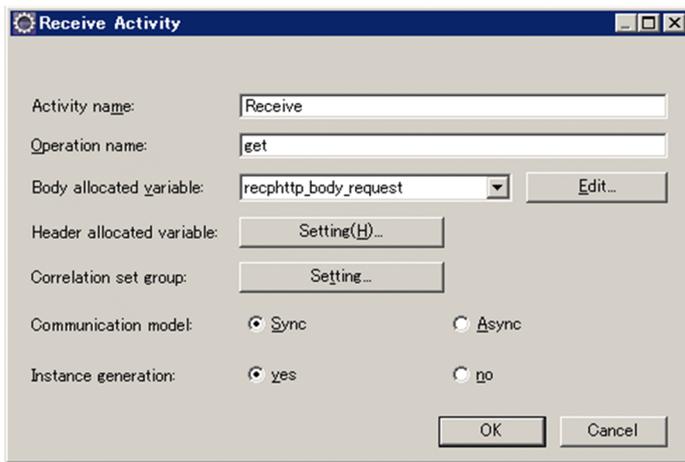
(d) Defining the reception activity

Define the reception activity by following the procedure below.

1. Double-click the reception activity (ReceptionActivity1).  
The Receive Activity dialog box appears.
2. Enter the settings as shown below.

Item	Setting
<b>Activity name</b>	<b>Receive</b>
<b>Operation name</b>	get
<b>Body allocated variable</b>	recphttp_body_request
<b>Communication model</b>	<b>Sync</b>
<b>Instance generation</b>	yes

G. Example for setting up business processes using HTTP reception and HTTP adapter



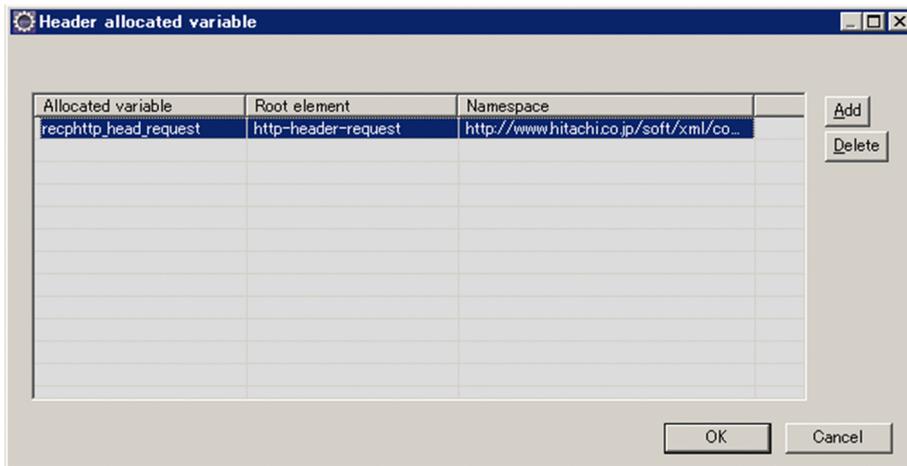
3. Click the **Setting** button beside **Header allocated variable**.

The Header allocated variable dialog box appears.

4. Click **Add**.

5. Enter the following settings in the Header allocated variable dialog box:

Item	Setting
<b>Allocated variable</b>	recphttp_head_request
<b>Root element</b>	http-header-request
<b>Namespace</b>	http://www.hitachi.co.jp/soft/xml/cosminexus/csc/reception/http/request



6. Click **OK** in the Header allocated variable dialog box.

The Header allocated variable dialog box closes.

7. Click **OK** in the Receive Activity dialog box.

The settings are saved and the Receive Activity dialog box closes.

(e) Defining the reply activity

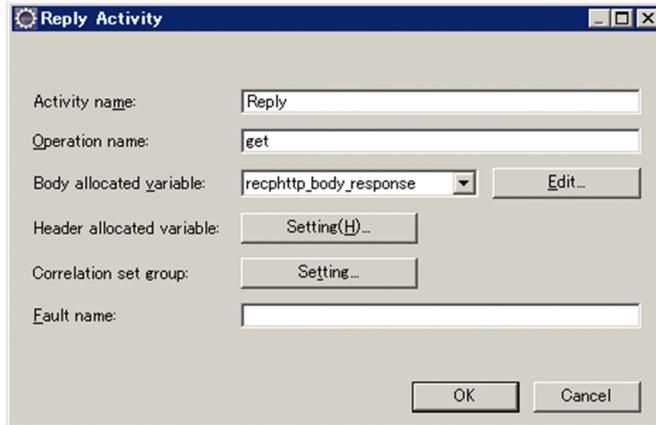
Define the reply activity by following the procedure below.

1. Double-click the reply activity (ReplyActivity1).

The Reply Activity dialog box appears.

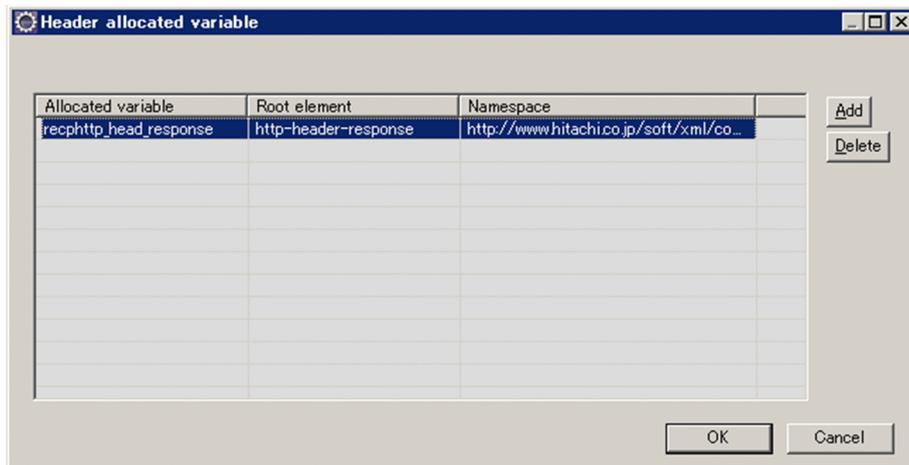
2. Enter the settings as shown below.

Item	Setting
<b>Activity name</b>	<b>Reply</b>
<b>Operation name</b>	get
<b>Body allocated variable</b>	recphttp_body_response
<b>Fault name</b>	-- (do not set)



- Click the **Setting** button beside **Header allocated variable**.  
The Header allocated variable dialog box appears.
- Click **Add**.
- Enter the following settings in the Header allocated variable dialog box:

Item	Setting
<b>Allocated variable</b>	recphttp_head_response
<b>Root element</b>	http-header-response
<b>Namespace</b>	http://www.hitachi.co.jp/soft/xml/cosminexus/csc/reception/http/response

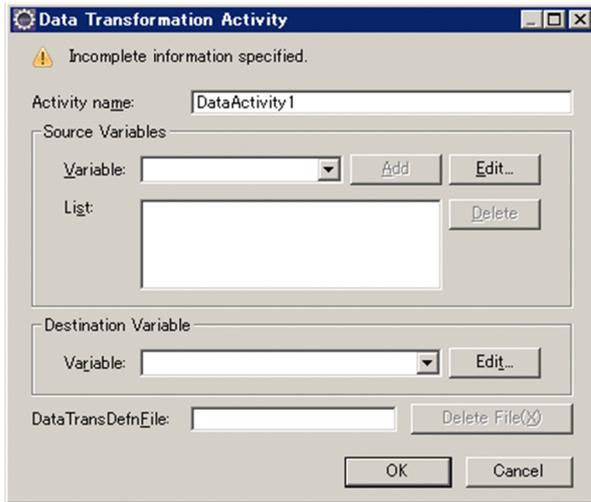


- Click **OK** in the Header allocated variable dialog box.  
The Header allocated variable dialog box closes.
- Click **OK** in the reply activity dialog box.  
The settings are saved and the reply activity dialog box closes.

(f) Defining the data transformation activities

Define the various data transformation activities by following the procedure below.

1. Double-click a data transformation activity (one of DataActivity1 to DataActivity4).  
The Data Transformation Activity dialog box appears.



2. Enter the appropriate settings for each data transformation as shown in the following table.

Activity	Item		Setting
DataActivity1	<b>Activity name</b>		CreateHTTPAdapterBodyRequestMessage
	<b>Source Variables</b>	<b>List<sup>#</sup></b>	recphttp_body_request
	<b>Destination Variable</b>	<b>Variable</b>	adphttp_body_request
	<b>Data TransDefn File</b>		CreateHTTPAdapterBodyRequestMessage
DataActivity2	<b>Activity name</b>		CreateHTTPAdapterHeaderRequestMessage
	<b>Source Variables</b>	<b>List<sup>#</sup></b>	recphttp_head_request
	<b>Destination Variable</b>	<b>Variable</b>	adphttp_head_request
	<b>Data TransDefn File</b>		CreateHTTPAdapterHeaderRequestMessage
DataActivity3	<b>Activity name</b>		CreateHTTPReceptionBodyResponseMessage
	<b>Source Variables</b>	<b>List<sup>#</sup></b>	adphttp_body_response
	<b>Destination Variable</b>	<b>Variable</b>	recphttp_body_response
	<b>Data TransDefn File</b>		CreateHTTPReceptionBodyResponseMessage
DataActivity4	<b>Activity name</b>		CreateHTTPReceptionHeaderResponseMessage
	<b>Source Variables</b>	<b>List<sup>#</sup></b>	adphttp_head_response
	<b>Destination Variable</b>	<b>Variable</b>	recphttp_head_response
	<b>Data TransDefn File</b>		CreateHTTPReceptionHeaderResponseMessage

#

The variable is added to the list when you select it from the **Variable** drop-down list in the **Source Variables** area.

- Click **OK** in the Data Transformation Activity dialog box.

The settings are saved and the Data Transformation Activity dialog box closes. Use this procedure to define all four data transformations.

## (9) Defining the data transformations

Generate the mapping definitions for the data transformation activities you defined in (8)(f) *Defining the data transformation activities*.

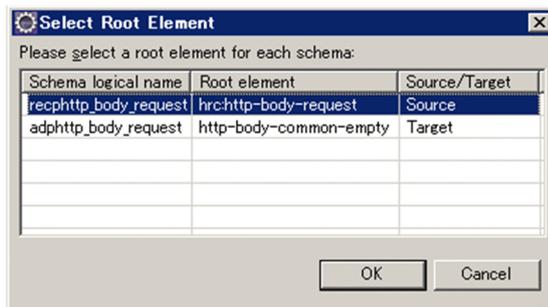
For details on how to work with mapping definitions, see 6.4 *Mapping* in the manual *Service Platform Basic Development Guide*.

The mapping definitions of the data transformation activities are shown below.

### (a) CreateHTTPAdapterBodyRequestMessage

- Double-click the data transformation activity (CreateHTTPAdapterBodyRequestMessage) in the Define Business Process window.

The Select Root Element dialog box appears.



- Click **OK**.

The mapping viewer appears.



You do not need to define mapping for this activity. Because this setup example executes processing that sends a file, use an empty request message (body) for the HTTP adapter.

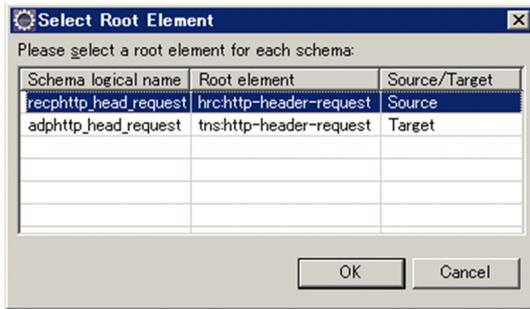
### (b) CreateHTTPAdapterHeaderRequestMessage

Set up mapping so that the file downloaded by the HTTP adapter is stored in the work folder. Because the method and URI of the request are defined in the HTTP-adapter runtime-environment property file, you do not need to map those of the request message.

- Double-click the data transformation activity (CreateHTTPAdapterHeaderRequestMessage) in the Define Business Process window.

The Select Root Element dialog box appears.

G. Example for setting up business processes using HTTP reception and HTTP adapter



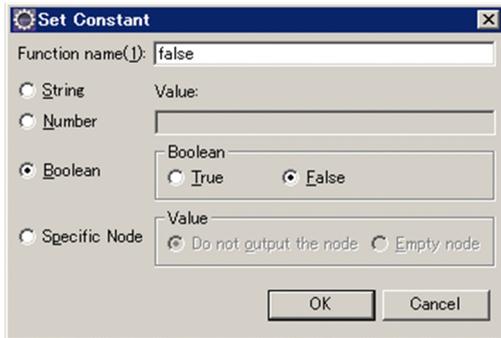
2. Click **OK**.

The mapping viewer appears.

3. Place the `const` function on the canvas, and then double-click it.

The Set Constant dialog box appears.

4. Change the function name to `false`, and select **Boolean** and **False**.



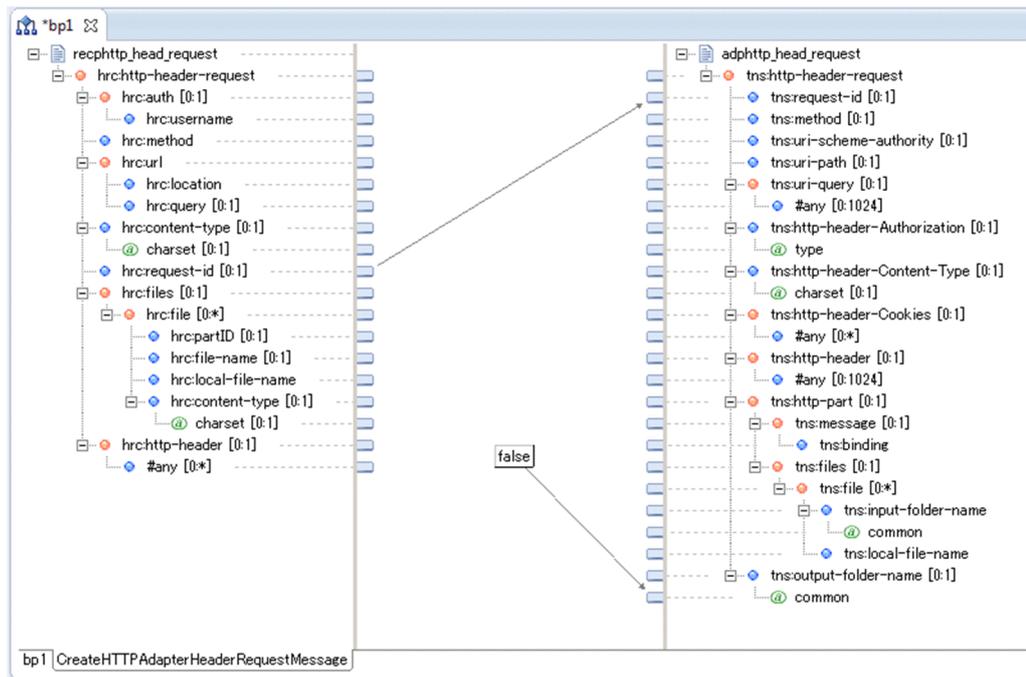
5. Click **OK**.

The Set Constant dialog box closes.

6. Set up mapping with reference to the table below.

The mapping sources and targets are represented as paths from the root element.

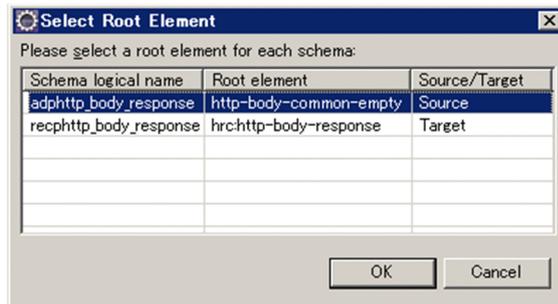
Mapping source	Mapping target
<code>/hrc:http-header-request/hrc:request-id</code>	<code>/tns:http-header-request/tns:request-id</code>
<code>const function (false)</code>	<code>/tns:http-header-request/tns:output-folder-name/@common</code>



(c) CreateHTTPReceptionBodyResponseMessage

1. Double-click the data transformation activity (CreateHTTPReceptionBodyResponseMessage) in the Define Business Process window.

The Select Root Element dialog box appears.



2. Click **OK**.

The mapping viewer appears.



You do not need to define mapping for this activity. In this setup example, because the HTTP reception executes processing that downloads a file when responding, use an empty response message (body) for the HTTP reception.

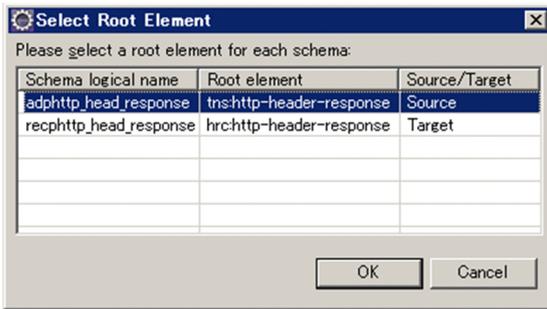
(d) CreateHTTPReceptionHeaderResponseMessage

Map the file information that the HTTP reception returns to the client.

1. Double-click the data transformation activity (CreateHTTPReceptionHeaderResponseMessage) in the Define Business Process window.

The Select Root Element dialog box appears.

G. Example for setting up business processes using HTTP reception and HTTP adapter



2. Click **OK**.

The mapping viewer appears.

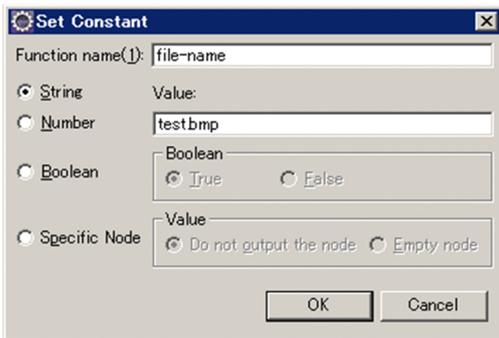
3. Place the `const` function on the canvas, and then double-click it.

The Set Constant dialog box appears.

4. Change the function name to `file-name`.

5. Select **String**, and as the value, specify a character string to use as the file name of the downloaded file.

In this setup example, specify `test.bmp`.



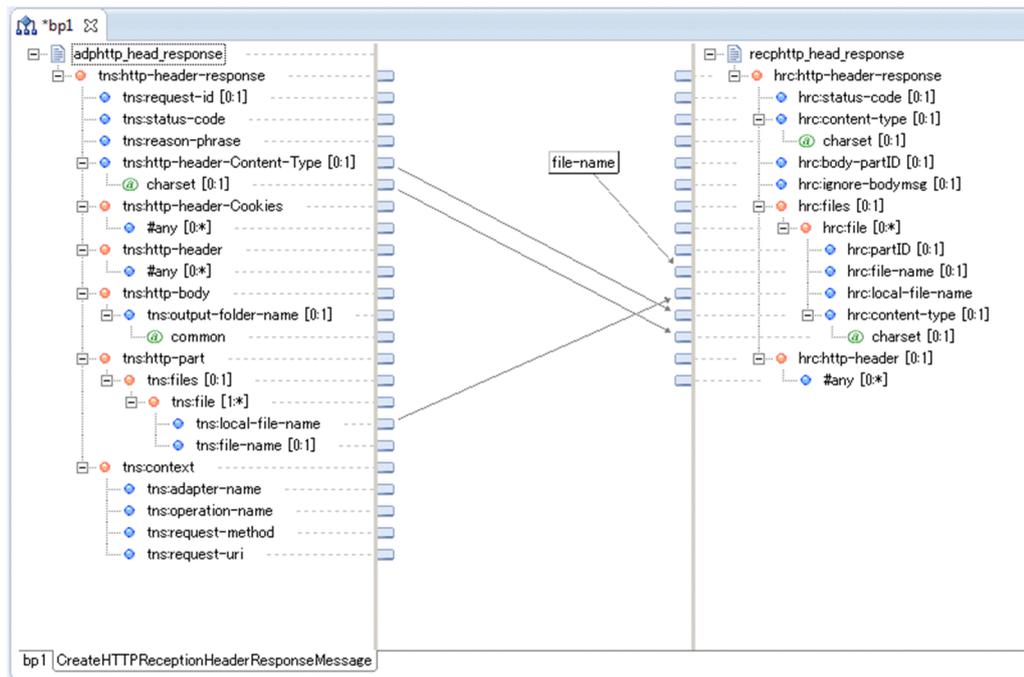
6. Click **OK**.

The Set Constant dialog box closes.

7. Set up mapping with reference to the table below.

The paths of the mapping sources and targets are from the root element.

Mapping source	Mapping target
const function (file-name)	/hrc:http-header-response/hrc:files/hrc:file/hrc:file-name
/tns:http-header-response/tns:http-part/tns:files/tns:file/tns:local-file-name	/hrc:http-header-response/hrc:files/hrc:file/hrc:local-file-name
/tns:http-header-response/tns:http-header-Content-Type	/hrc:http-header-response/hrc:files/hrc:file/hrc:content-type
/tns:http-header-response/tns:http-header-Content-Type/@charset	/hrc:http-header-response/hrc:files/hrc:file/hrc:content-type/@charset



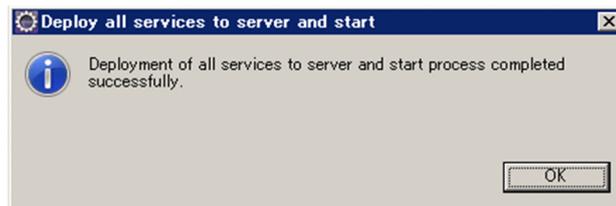
## (10) Preparing to execute the business process

Save the HCSC component you defined, and define its deployment. To save and deploy the HCSC component:

1. From the **File** menu, select **Save All**.  
The definitions of the HTTP adapter, HTTP reception, and business process are saved.
2. Right-click the Service Definition List in the tree view, and select **Deploy all services to server and start**.  
If you are not logged in, the Account Verification window appears. In this case, perform step 3.
3. Enter **admin** in the **User ID** field and the **Password** field, and then click **OK**.

A message indicating that account verification is in progress appears, followed by a message indicating whether it was successful.

If the following dialog box appears, the deployment process has been successful.



If the process has failed, read the error message and make sure that the information you have defined is correct.

## (11) Executing the business process

1. Open your Web browser, and access the following URL:

`http://hostB/rcp1/get`

As *hostB*, specify a value that will resolve to the IPv4-format IP address of host B. The elements following the host name are the reception ID and the name of the reception operation.

If the business process was executed correctly, the dialog box appears, and you can download the file from the HTTP reception.

The file name of the downloaded file is the name specified in the `const` function in the mapping definition.

## G.2 Setup example 2 (dynamically changing the connection destination of the HTTP adapter for each request)

### (1) Overview of setup example

This setup example involves modifying the definitions you created in *G.1 Setup example 1 (downloading the file from the HTTP server to the HTTP client)* to define a business process that dynamically changes the connection destination path of the HTTP adapter according to the nature of the request.

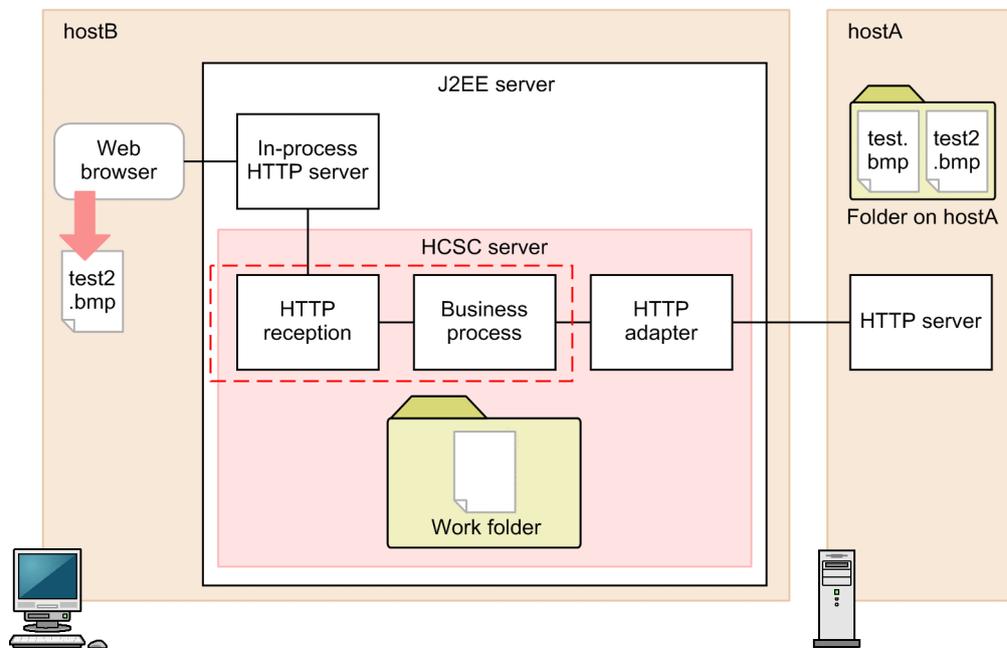
A query passed to the HTTP reception provides the input data used to dynamically change the connection destination.

You must have completed the steps in *G.1 Setup example 1 (downloading the file from the HTTP server to the HTTP client)* from beginning to end before performing the steps in this setup example.

#### (a) Environment configuration

The following figure shows an example of the environment created in this setup example:

Figure G–2: Environment configuration created in setup example 1



This setup example describes the procedure for changing the definition of the elements in the box with the dashed border in the figure.

As in setup example 1, a requirement for this setup example is that the HTTP servers to which the HTTP adapter will connect are set up in advance. In setup example 2, the HTTP server must be configured so that the HTTP adapter can download test2.bmp by accessing `http://hostA:80/test2.bmp`.

#### (b) Flow of setup procedure

The setup procedure consists of the following steps:

1. Creating the message format
2. Changing the variable definition
3. Changing the data transformation definition
4. Preparing to execute the business process

## 5. Executing the business process

Each step is described in detail below.

### (2) Creating the message format

Create the message format to use as the request message (body) of the HTTP reception. The query information passed from the HTTP client is stored in the message format you created.

Use the template to create the message format. To create the message format using the template:

1. Copy the following folder containing the message format template to a location of your choice:  
*service-platform-installation-directory\CSC\custom-reception\http\schema*
2. Open the `urecp_http_body_detail_request.xsd` file in the folder in a text editor.
3. Add the element shown in italics which represents the file name.

---

```
<?xml version="1.0" encoding="UTF-8" ?>
<xsd:schema elementFormDefault="qualified"
targetNamespace="http://www.hitachi.co.jp/soft/xml/cosminexus/csc/reception/http/request"
xmlns:hrc="http://www.hitachi.co.jp/soft/xml/cosminexus/csc/reception/http/request"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<xsd:complexType name="http-body-request">
  <xsd:sequence>
    <!-- User Customize -->
    <xsd:element name="filename" type="xsd:string"/>
  </xsd:sequence>
</xsd:complexType>
</xsd:schema>
```

---

4. Save the file.

Note: Do not edit `urecp_http_body_request.xsd`.

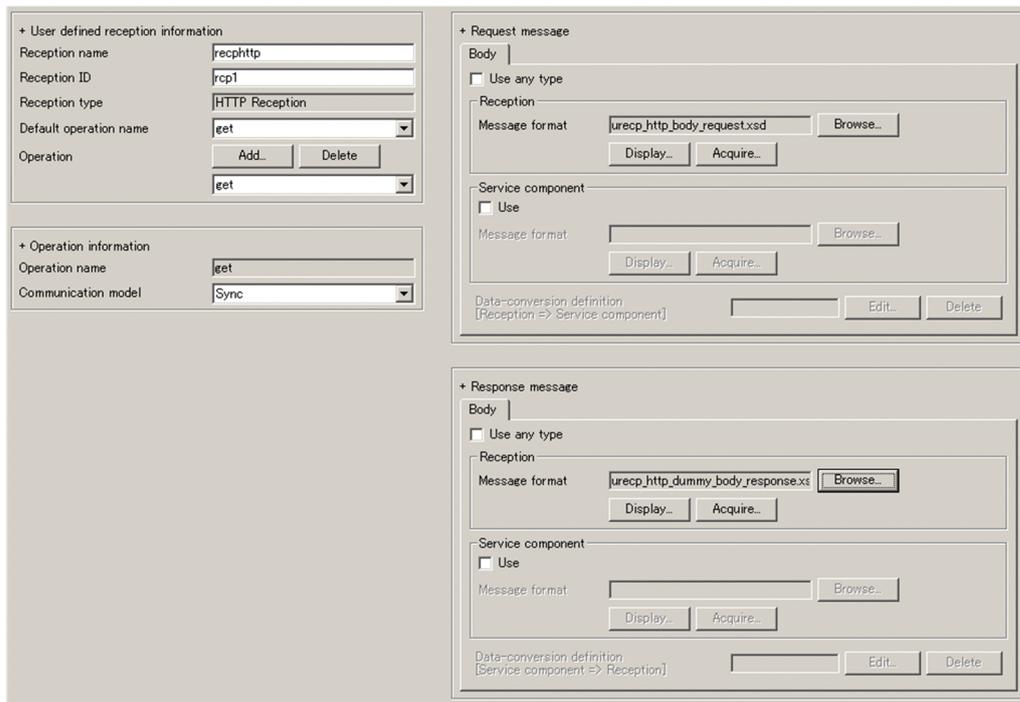
### (3) Changing the variable definition

#### (a) HTTP reception

Apply the message format you created in (2) to the HTTP reception definition.

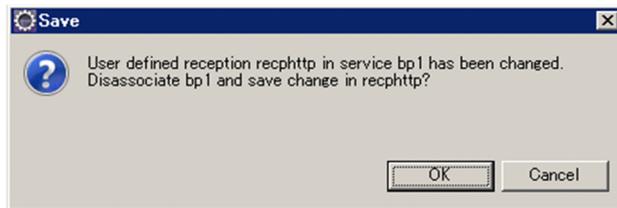
1. In the tree in the service definition list, double-click the HTTP reception (`recphhttp`).  
The User-defined Reception Definition window for the HTTP reception appears.
2. In the **Request message** area, click the **Browse** button beside the **Message format** field in the **Reception** area, and specify the following format definition file:  
*copy-destination-folder-in(2)\schema\urecp\_http\_body\_request.xsd*

G. Example for setting up business processes using HTTP reception and HTTP adapter



3. Save the HTTP reception by selecting **Save** from the **File** menu.

If the following confirmation dialog box appears when you attempt to save the HTTP reception, click **OK**:

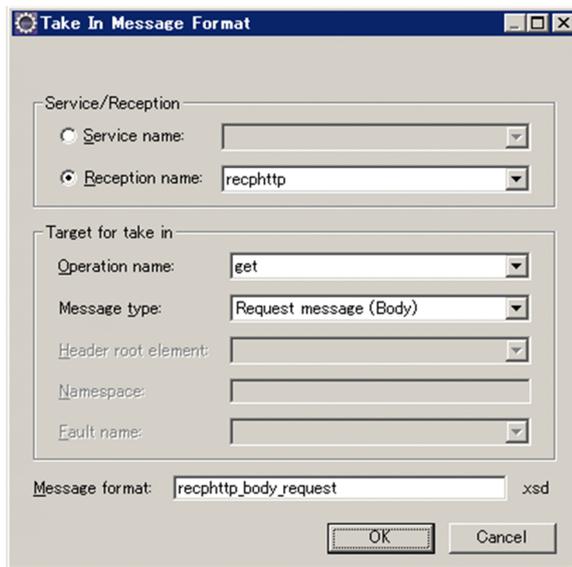


(b) Business process

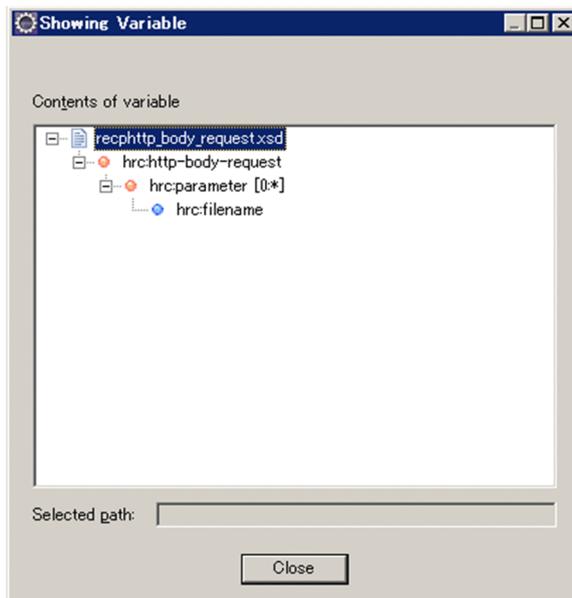
Update the body allocated variable `recphttp_body_request` for the HTTP reception request message to reflect the message format you created in (2).

1. In the service definition list in the tree view, double-click the business process (bp1).  
The Define Business Process window appears.
2. On the canvas of the Define Business Process window, double-click the Variable-Correlation icon.  
The List Of Variables And Correlation Sets dialog box appears.
3. In the variable list tree, select `recphttp_body_request` and click **Take In**.  
The Take In Message Format dialog box appears.
4. Specify the following information, and then click **OK**:

Variable name	Service/Reception	Operation name	Message type	Message format
<code>recphttp_body_request</code>	<b>Reception name</b> <code>recphttp</code>	<code>get</code>	<b>Request message (Body)</b>	<code>recphttp_body_request.xsd</code>



5. In the List Of Variables And Correlation Sets dialog box, click **Update**.  
The variable is updated.
6. In the List Of Variables And Correlation Sets dialog box, click **Show Tree**.  
The Showing Variable dialog box appears.



Make sure that the variable has been updated, and close the dialog box.

7. In the List Of Variables And Correlation Sets dialog box, click **OK**.  
The List Of Variables And Correlation Sets dialog box closes.

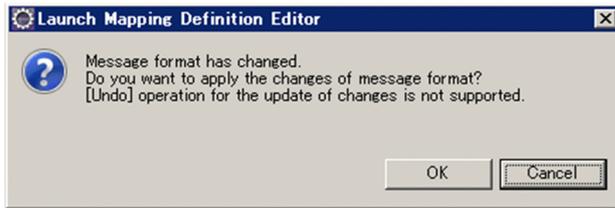
#### (4) Changing the data transformation definition

Use the request message (body) variable for the HTTP reception to change the mapping definition.

##### (a) Mapping to HTTP adapter body request message

1. Double-click the data transformation activity (CreateHTTPAdapterBodyRequestMessage) in the Define Business Process window.
2. A dialog box appears in which you can confirm the launch of the mapping definition editor.

G. Example for setting up business processes using HTTP reception and HTTP adapter



3. Click **OK**.

The Mapping Definition window appears.



The filename element has been added as a mapping source. As in setup example 1, you do not need to configure mapping for this element.

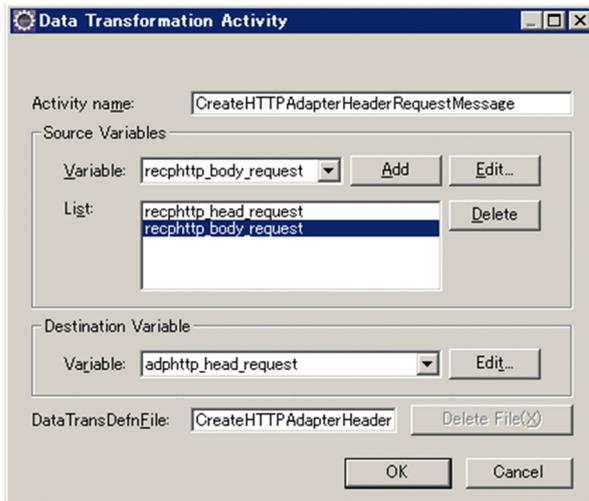
(b) Mapping to HTTP adapter header request message

1. Right-click the data transformation activity (CreateHTTPAdapterHeaderRequestMessage) in the Define Business Process window, and then click **Setting**.

The Data Transformation Activity dialog box appears.

2. Select recphttp\_body\_request from the **Variable** drop-down list in the **Source Variables** area, and then click **Add**.

The recphttp\_body\_request variable is added to the list.



3. Click **OK**.

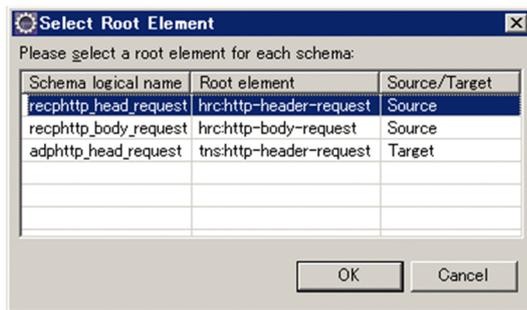
The Data Transformation Activity dialog box closes.

4. Double-click the data transformation activity (CreateHTTPAdapterHeaderRequestMessage).

A dialog box appears in which you can confirm the launch of the mapping definition editor.

5. Click **OK**.

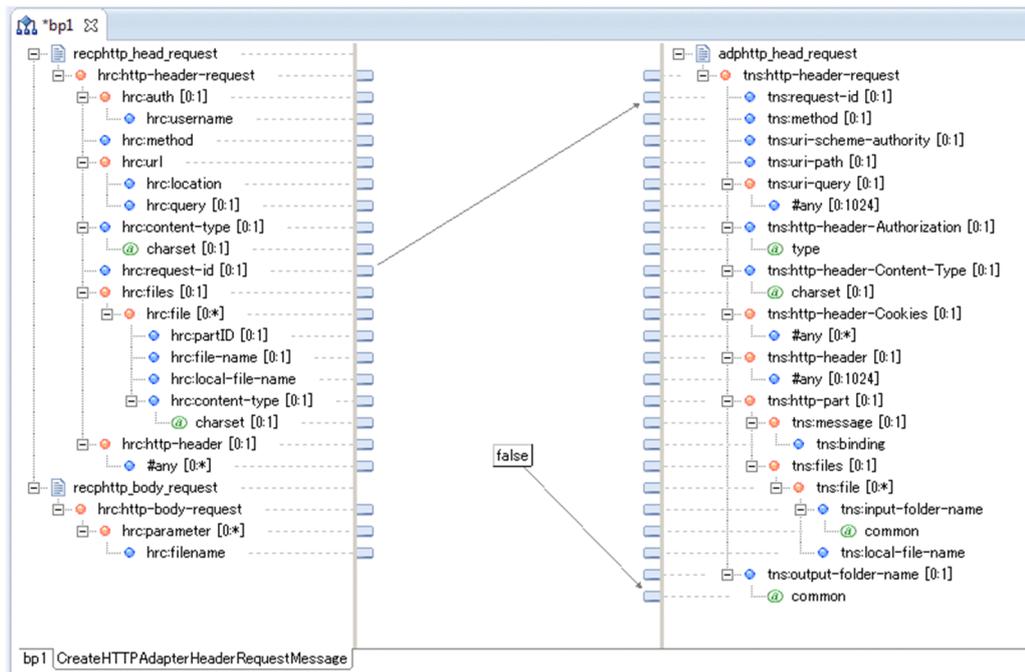
The Select Root Element dialog box appears.



6. Click **OK**.

The Mapping Definition window appears.

The recphttp\_body\_request variable now appears as a mapping source.

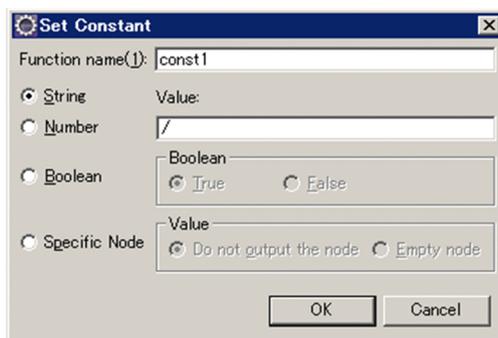


You can then use the filename element of recphttp\_body\_request to define mapping that changes the name of the file acquired by the HTTP adapter from the value set in the HTTP-adapter runtime-environment property file.

7. Place the const function on the canvas, and then double-click it.

The Set Constant dialog box appears.

8. Select **String**, and in **Value**, specify /.



9. Click **OK**.

G. Example for setting up business processes using HTTP reception and HTTP adapter

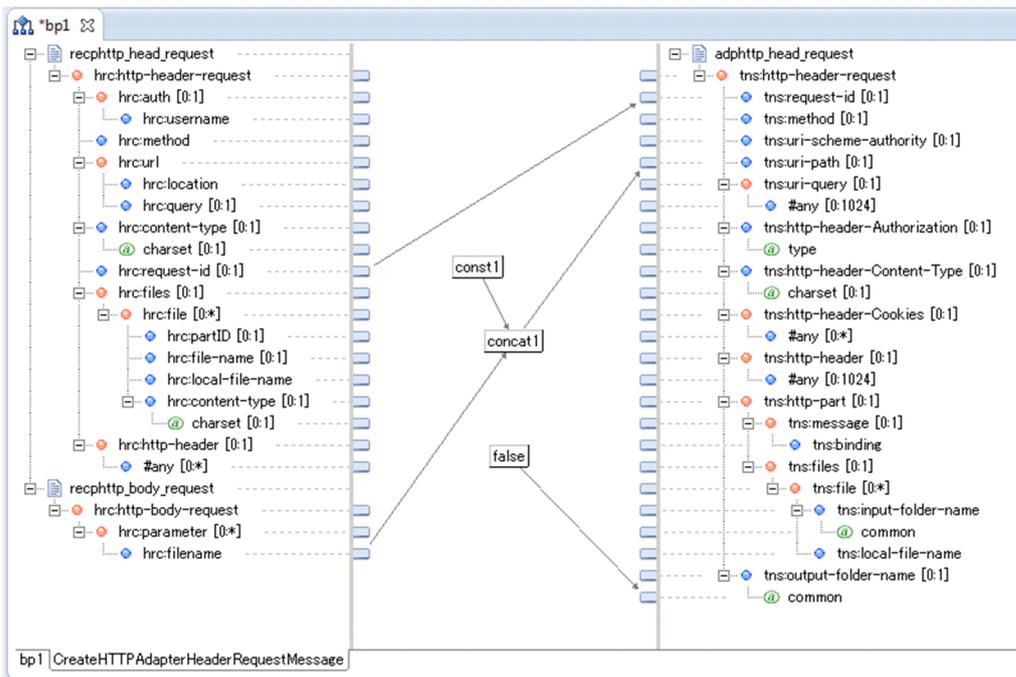
The Set Constant dialog box closes.

10. Place the `concat` function on the canvas.

11. Set up mapping with reference to the table below.

Take care to ensure that the elements are mapped in the correct sequence. The paths of the mapping sources and targets are paths from the root element.

No.	Mapping source	Mapping target
1	const function	concat function
2	/hrc:http-body-request/hrc:parameter/ hrc:filename	concat function
3	concat function	/tns:http-header-request/tns:uri-path

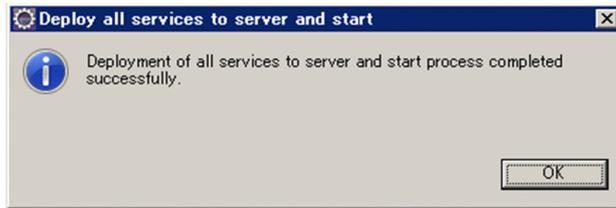


(5) Preparing to execute the business process

Define deployment of the HCSC component you defined. To deploy the HCSC component:

1. From the **File** menu, select **Save All**.  
The definitions of the HTTP adapter, HTTP reception, and business process are saved.
2. Right-click the Service Definition List in the tree view, and select **Deploy all services to server and start**.  
If you are not logged in, the Account Verification window appears. In this case, perform step 3.
3. Enter `admin` in the **User ID** field and the **Password** field, and then click **OK**.  
A message indicating that account verification is in progress appears, followed by a message indicating whether it was successful.

If the following dialog box appears, the deployment process was successful.



If the process failed, read the error message and make sure that the information you have defined is correct.

## (6) Executing the business process

1. Open your Web browser, and access the following URL:

```
http://hostB/rcpl/get?filename=test2.bmp
```

As *hostB*, specify a value that will resolve to the IPv4-format IP address of host B. The elements following the host name are the reception ID and the name of the reception operation. In the query, specify the *filename* parameter you defined.

If the business process was executed correctly, you can download the file from the HTTP reception. In this case, the dialog box appears in your Web browser:

Note:

Because the file name of the downloaded file is set statically in *G.1(9)(c)*

*CreateHTTPReceptionBodyResponseMessage*, the name of the downloaded file will be `test.bmp` as in setup example 1.

Based on the execution results of setup example 2, confirm that the connection-destination path set in the HTTP-adapter runtime-environment property file has been replaced with the request message (header) of the HTTP adapter.

---

### Reference note

If you want to make sure that different files are being downloaded, download the file of setup example 1 by accessing the following URL with the parameter changed:

```
http://hostB/rcpl/get?filename=test.bmp
```

---

## H. Security settings required in FTP linkage

This appendix describes the security settings required when using FTP linkage.

### H.1 Secure connection using FTPS (FTP adapter)

In an FTP adapter, the use of FTPS (File Transfer Protocol over SSL/TLS) is supported for connections to FTP servers. Use of the FTPS protocol ensures that the user name and password required for authentication are encrypted, making it less likely for this information to be intercepted by third parties.

#### (1) Settings related to FTPS

The following describes the settings that relate to the FTPS protocol. You can apply these settings to an FTP adapter by specifying them in a request message or in the FTP-adapter runtime-environment property file. If the same setting is specified in a request message and the FTP-adapter runtime-environment property file, the setting in the request message has priority.

Table H-1: FTP-related settings

Item	Description
Whether to use FTPS	Set whether to use standard FTP or FTPS for connections with the FTP server. By default, the system uses standard FTP.
Protocol used for secure communication	Select one of the following as the protocol for secure communication: <ul style="list-style-type: none"> <li>• SSL</li> <li>• TLS</li> </ul>
Whether to use Implicit mode for FTPS connections	Select one of the following: <ul style="list-style-type: none"> <li>• Implicit mode Communication is encrypted as soon as the connection is established with the FTP server.</li> <li>• Explicit mode After connecting to the FTP server, communication is encrypted from the point when an AUTH command is executed on the FTP client.</li> </ul> <p>In each mode, a different port number is used for the control connection on the FTP server. For this reason, you need to change the control connection port number of the connection destination FTP server when you change the mode.</p>
Whether to encrypt communication over data connections	Set whether to encrypt communication over data connections when using FTPS for connections with the FTP server.
Whether to perform server authentication	Select one of the following: <ul style="list-style-type: none"> <li>• Perform server authentication The system checks whether the public key certificate of the connection destination FTP server can be trusted, and only proceeds if the certificate can be trusted.</li> <li>• Do not perform server authentication The system performs the requested processing without checking whether the public key certificate of the connection destination FTP server can be trusted.</li> </ul>

For details on the items that can be set in a request message, see *3.3.10(1)(a) Request message format for FTP adapters*. For details on the items you can set in the FTP-adapter runtime-environment property file, see *FTP Adapter Execution Environment Property file* in the manual *Service Platform Reference Guide*.

#### ! Important note

The FTP adapter cannot connect to an FTPS server that requires client authentication.

## (2) Settings related to keystores

When FTPS is used, if you use an FTP adapter to authenticate servers, a keystore (truststore) that contains trusted server certificates is required.

The truststore must contain one of the following certificates:

- Certificate for the public key of the connection destination FTP server
- If the certificate for the public key of the connection destination FTP server is issued by a CA, the certificate for the CA that issued the certificate
- If the above CA is an intermediate CA, the certificate for a CA that corresponds to any of the CAs between the intermediate CA and the root CA

For details on how to import certificates to a keystore, see the section related to *keytool* in the JDK documentation.

The following describes how to specify the settings related to keystore paths and passwords.

### Keystore path

When authenticating a server, the FTP adapter searches keystores in the following order, and then authenticates the server by using the truststore that is found first:

1. If the `javax.net.ssl.trustStore` property is specified in `usrconf.properties` (user property file for J2EE servers), the keystore that exists in the path specified in this system property is used.
2. If a file exists in the following path, the file is used as a keystore:  
`service-platform-installation-directory\jdk\jre\lib\security\jssecacerts`
3. If a file exists in the following path, the file is used as a keystore:  
`service-platform-installation-directory\jdk\jre\lib\security\cacerts`

If a keystore is not found in the above locations, the server is determined to be untrusted.

### Keystore password

If the keystore password is specified in the `javax.net.ssl.trustStorePassword` property of `usrconf.properties` (user property file for J2EE servers), the integrity of the keystore data is checked before the keystore is used.

If the `javax.net.ssl.trustStorePassword` property is not specified, the keystore data is not verified.

For details on `usrconf.properties` (user property file for J2EE servers), see *2.4 usrconf.properties (User property file for J2EE servers)* in the manual *uCosminexus Application Server Definition Reference Guide*.

## H.2 Encryption and authentication by secure protocols (FTP inbound adapter)

You can use FTPS to encrypt the data sent and received by the FTP inbound adapter, including user names and passwords. Note that the FTP inbound adapter only supports server authentication.

The following describes the FTPS-related protocols and commands supported by the FTP inbound adapter, and the settings that relate to FTPS.

### (1) Supported FTPS-related protocols and commands

The FTP inbound adapter supports the protocols in the following table:

Table H–2: Supported FTPS-related protocols and versions

Protocol	Version
SSL	3.0
TLS	1.0

The FTP inbound adapter also supports the following FTPS-related commands:

- AUTH

- PBSZ
- PROT

## (2) Configuring the FTP inbound adapter

You can enter FTPS-related settings in the attribute file for the FTP inbound adapter.

For details on how to set the attribute file for the FTP inbound adapter and the items you can define in the file, see the description of FTP inbound adapter configuration in 3.2.3 *Setting up the FTP inbound adapter* in the manual *Service Platform Setup and Operation Guide*.

## (3) Configuring the keystore

The FTP inbound adapter only supports keystores in JKS format. Use the `keytool` command supplied with the JDK to create keystores and keys in JKS format. For details on the `keytool` command, see the supplied documentation.

When you use a keystore that contains multiple keys, the same password must be set for every key, including those that are not used by the FTP inbound adapter.

Any upper-case characters in the `alias` option of the `keytool` command are saved as lower-case characters when the keystore is created. For this reason, do not use upper-case in the `nioListener_ftps_keystore_keyAlias` attribute of the attribute file of the FTP inbound adapter. Specify the alias name as it appears in the output of the `keytool` command executed with the `list` option specified.

## H.3 Definition items in the property file at the time of setup (FTP inbound adapter)

The following describes the FTPS-related settings in the definition items in the attribute file used during setup of the FTP inbound adapter. For details on the other definition items, see 3.2.3 *Setting up the FTP inbound adapter* in the manual *Service Platform Setup and Operation Guide*.

Table H-3: List of definition items in attribute file (FTPS-related settings)

Definition item	Definition name	Permitted values	Default value	Remarks
Whether to enable FTPS	<code>server_ftps_enable</code>	true or false	false	If you specify true, the FTPS-related settings take effect. If you specify false, all FTPS-related settings are ignored.
Implicit mode for control connection	<code>nioListener_ftps_implicitMode</code>	true or false	false	If you specify true, the control connection operates in Implicit mode. If you specify false, it operates in Explicit mode.
Keystore file path <sup>#</sup>	<code>nioListener_ftps_keystore_file</code>	Any character string	Empty string	Specify the file by absolute path.
Keystore password	<code>nioListener_ftps_keystore_password</code>	A character string of 6 or more characters	Empty string	--
Key password	<code>nioListener_ftps_keystore_keyPassword</code>	A character string of 6 or more characters	Empty string	If you omit this item, the keystore password is assumed to be the key password.
Key alias	<code>nioListener_ftps_key</code>	Any character string	Empty string	If you omit this item, the FTP inbound adapter uses the first key in the keystore (when sorted in ascending

Definition item	Definition name	Permitted values	Default value	Remarks
Key alias	store_keyAlias	Any character string	Empty string	order of the byte array of the key certificate fingerprint).
Encryption of data connection	nioListener_ftps_dataConnection_secure	true or false	true	This setting applies if you do not specify the <code>PROT</code> command. If the <code>PROT</code> command is executed successfully, the value specified in the <code>PROT</code> command applies.  If you specify <code>true</code> , the data connection is encrypted.  It is not encrypted if you specify <code>false</code> .

## Legend:

--: Not applicable.

#

An error occurs if the path you specify contains the characters `//`, `./`, or `../`. Note that in Windows, this value is not case sensitive.

## I. Security settings required in the HTTP adapter

This section describes the security settings required in the HTTP adapter.

### (1) Secure connection using HTTPS

The HTTP adapter supports the connection with the server through HTTPS. By using the connection through HTTPS, you encrypt the user name and the password required for authentication thereby reducing the risk of interception by a third party.

#### (a) HTTPS related settings

The following table lists the items that are set as HTTPS related information:

Table I-1: HTTPS related settings

Items	Settings
Use of HTTPS	Specifies "https" in the schema of URI when using HTTPS.
Security communication protocol to be used when using HTTPS	The most appropriate protocol is set automatically from among TLS1.0, SSL3.0 and other protocols that can be used.
Server authentication	<p>You can set whether to perform server authentication with the <code>adphttp.protocol.https-server-authentication</code> property of the HTTP-adapter definition file.</p> <ul style="list-style-type: none"> <li>When you perform server authentication (default): Checks if the public key certificate of the connection destination HTTP server can be trusted, and executes the process only if the certificate can be trusted.</li> <li>When you do not perform server authentication: Executes the process without checking whether the public key certificate of the connection destination HTTP server can be trusted.</li> </ul>
Host name validation	<p>You can set whether or not to perform the host name validation with the <code>adphttp.protocol.https-hostname-verification</code> property of the HTTP-adapter definition file.</p> <ul style="list-style-type: none"> <li>When you perform host name validation(default): Checks if the host name in the certificate matches with the host name of the destination URI, and then executes process.</li> <li>When you do not perform host name validation: Executes the process without checking if the host name in the certificate matches with the host name of the destination URI.</li> </ul>

For details on the HTTP-adapter definition file, see *HTTP-adapter definition file* in the manual *Service Platform Reference Guide*.

#### ! Important note

In the HTTP adapter, you cannot connect to the HTTPS server that requires client authentication.

#### (b) Key store related settings

When you use HTTPS or when you perform server authentication in the HTTP adapter, a key store that stores the certificate that you can trust (trust store) is required.

Any of the following certificates must be stored in the trust store:

- Public key certificate of the connection destination HTTP server
- If the above mentioned certificate is issued by CA, certificate of CA that issued the above mentioned certificate
- If the above mentioned CA is intermediate CA, then CA certificate corresponding to any CA from the intermediate CA up to the root CA
- For the method to import certificates to the key store, see the "keytool" related part in the JDK document.

The method to set the keystore path and password is as follows:

### Keystore path

When you perform server authentication, the HTTP adapter searches the keystore in the following order, and performs server authentication by using the trust store which is detected first:

1. If the `javax.net.ssl.trustStore` property of `usrconf.properties` (user property file of the J2EE server) is specified, uses the key store that exists in the path specified in this system property.
2. If a file exists in the following path, uses this file as the keystore:  
<Installation directory of service platform>\jdk\jre\lib\security\jssecacerts
3. If a file exists in the following path, uses this file as the keystore.  
<Installation directory of service platform >\jdk\jre\lib\security\cacerts

If none of the above mentioned points are applicable, then the server is determined as a server that cannot be trusted.

### Keystore password

If the password of the keystore is specified in `javax.net.ssl.trustStorePassword` property of `usrconf.properties` (user property file of the J2EE server), before using key store, an investigation is conducted to check if the key store data is complete.

If the `javax.net.ssl.trustStorePassword` property is not specified, the investigation for the keystore data is not conducted.

For details on `usrconf.properties` (user property file for the J2EE server), see *2.4 usrconf.properties (user property file for J2EE server)* in the manual *Application Server Definition Reference Guide*.

---

## J. Glossary

### **Terminology used in this manual**

For the terms used in the manual, see *Application Server and BPM/ESB Platform Terminology Guide*.

---

# Index

## A

---

about sample 352  
acquiring failure information (custom reception) 287  
adapter context interface (custom adapter development framework APIs) 298  
adapters  
    defining 77  
    deleting 255  
    displaying validation contents 254  
    editing 253  
    list of settings in definition 227  
    saving 252  
    that cannot be deleted 255  
    using already defined adapter to add 86  
    validating 254  
    validation method 254  
adding a new MDB (DB queue) adapter 82  
adding a new MDB (WS-R) adapter 82  
adding FTP reception 28  
adding HTTP reception 39  
adding Message Queue reception 47  
adding new file adapter 83  
adding new file operation adapter 84  
adding new FTP adapter 84  
adding new HTTP adapter 85  
adding new mail adapter 85  
adding new Message Queue adapter 84  
adding new Object Access adapter 83  
adding new Session Bean adapter 82  
adding new SOAP adapter 81  
adding new TP1 adapter 83  
adding service adapter 81  
adding SOAP reception 15  
adding TP1/RPC reception 18  
API of custom adapter development framework 295  
APIs of custom reception framework 264

## C

---

changing information of user-defined reception 75  
connection definition identifier 153  
creating and editing file operations adapter definition file 187  
creating and editing HTTP adapter definition file 216  
creating binary format definition file 130  
creating definition file of FTP adapter 177  
creating definition file of FTP reception 27  
creating definition file of HTTP reception 38  
creating definition file of Message Queue reception 46  
creating JAR file (library JAR file) 224  
creating JAR file (protocol converter JAR file) 223  
creating library JAR file 224  
creating message format (for DB adapter message) 107  
creating message format (for file adapter) 123  
creating message format (for file operations adapters) 178  
creating message format (for FTP adapters) 153

creating message format (for mail adapter) 189  
creating message format definition file (for Message Queue adapter) 139  
creating message format of FTP reception 20  
creating message format of HTTP reception 32  
creating message format of Message Queue reception 44  
creating message formats (for HTTP adapters) 205  
creating Message Queue adapter runtime-environment property file 153  
creating protocol converter JAR file 223  
creating service requester (for DB adapter) 114  
creating SQL operation definition file 98  
creating XML format definition file 127  
creating XML format definition file for a DB adapter 107  
CSCMsgCustomAdapterConstant class 309  
CSCMsgCustomAdapterContext interface 298  
CSCMsgCustomProtocolConverter interface 295  
CSCMsgMessageConstant interface 300  
CSCMsgRequestMessage interface 301  
CSCMsgResponseMessage interface 305  
custom adapter  
    adding a new 86  
custom adapter definition file (definition files used during General custom adapter development) 317  
custom adapter definition screen 324  
custom adapter development framework action definition file 311  
custom adapter development framework action definition file (custom adapter definition) 220  
custom adapter property file (custom adapter definition) 220  
custom adapter property file (definition files used during General custom adapter development) 311  
custom reception 258  
custom reception definition 274  
custom reception development 260  
custom reception operations 277  
custom reception overview 258  
custom reception tuning 282

## D

---

database adapter  
    adding a new 83  
database adapters  
    defining 77, 98  
DB adapter settings 354  
defining  
    custom adapter 218  
defining a SOAP reception 10  
defining content of service adapter 87  
defining data transformation (file adapter) 137  
defining DB adapter 334  
defining DB adapter using DB adapter definition support function 334  
defining file adapters 123  
defining file operations adapters 178  
defining FTP adapters 153

defining FTP reception 19  
 defining HTTP adapters 205  
 defining HTTP reception 31  
 defining mail adapter 189  
 defining Message Queue adapters 139  
 defining Message Queue reception 43  
 defining an Object Access adapter 139  
 defining SOAP adapters 87  
 defining SOAP reception after defining business process 13  
 defining SOAP reception before defining business process 10  
 defining TP1/RPC reception 18  
 defining a TP1 adapter 123  
 definition file  
     custom adapter development framework 311  
 definition items in property file at time of setup 454

## E

---

editing definition of TP1/RPC reception 18  
 editing HITACHI Application Integrated Property File (for Message Queue adapter) 151  
 editing Message Queue adapter communication-configuration definition file 151  
 editing Message Queue adapter environment- definition file 150  
 encryption and authentication by secure protocols 453  
 environment settings when using custom adapter 330  
 example for defining file operations adapter 401  
 example for setting up business processes using HTTP reception 409  
 example for setting up business processes using HTTP reception and HTTP adapter 420  
 example for setting up file adapter 352  
 example of creating an XML format definition file for a DB adapter 108  
 example of creating SQL operation definition file 106  
 example of defining mail adapter 197  
 example of dynamically changing connection-destination information of service adapter 65  
 exception classes (custom adapter development framework APIs) 310  
 exporting and importing at time of distributed development 350

## F

---

flow of defining FTP reception 19  
 flow of defining HTTP reception 31  
 flow of defining Message Queue reception 43  
 flow of defining TP1/RPC reception 18  
 for Message Queue adapter 238  
 FTP command permission list definition file (for FTP reception) 28  
 FTP execution permission list definition file 28  
 FTP reception config file 28  
 FTP reception definition file 28

## H

---

HITACHI Application Integrated Property File 111

HITACHI Application Integrated Property File for custom adapter (custom adapter definition) 220  
 HITACHI Application Integrated Property File for custom adapter (definition files used during General custom adapter development) 312

## L

---

list command option definition file 28

## M

---

message examples 133  
 message interfaces (custom adapter development framework APIs) 300  
 method of notifying error in custom reception 283

## N

---

notes on using mail adapter 189

## O

---

operation in Service Adapter Settings window (custom adapter) 225  
 operation in Service Adapter Settings window (DB adapter) 110  
 operation in Service Adapter Settings window (for mail adapter) 197  
 operation in Service Adapter Settings window (Session Bean) 91  
 operations to be performed on service adapter settings screen (for file operations adapters) 187  
 operations to be performed on service adapter settings screen (for FTP adapters) 178  
 operations to be performed on service adapter settings screen (for HTTP adapters) 215  
 operations to be performed on service adapter settings screen (for Message Queue adapter) 149  
 operations to be performed on service adapter settings screen (for file adapter) 137  
 operations to be performed in service adapter settings screen (Web service) 88  
 overview of DB adapter definition support function 334  
 overview of development of receptions and service adapters 1  
 overview of FTP receptions 9  
 overview of HTTP receptions 9  
 overview of Message Queue receptions 9  
 overview of SOAP receptions 8  
 overview of TP1/RPC receptions 9

## P

---

points to be considered when using file transformation operation 188  
 positioning of receptions and service adapters in whole system 2  
 protocol converter interfaces (custom adapter development framework APIs) 295

## R

---

- receptions that can be used 4
- replacing EAR file 114
- request message at the time of writing file (XML format) 133
- request message at time of reading file 133
- request message at time of writing file (binary format) 135
- request message format (DB adapter) 114
- request message format for file operations adapters 178
- request message format for FTP adapters 154
- request message format of FTP reception 20
- response message format (DB adapter) 118
- response message format for file operations adapters 184
- response message format for FTP adapters 172
- response message format of FTP reception 24

## S

---

- sample for reading CSV data 365
- sample for reading XML data 356
- sample for writing CSV data 387
- sample for writing XML data 376
- sample program 317
- saving user-defined reception 72
- secure connection using FTPS 452
- security settings required in HTTP adapter 456
- self-defined file for protocol converter 220
- service adapter
  - defining 77
  - defining (MDB (DB queue)) 97
  - defining (MDB (WS-R)) 94
  - defining (SessionBean) 90
- Service adapter definition (details) screen 329
- Service adapter definition (standard) screen 325
- Service Adapter Definition screen
  - operations in (MDB (DB queue)) 97
  - operations in (MDB (WS-R)) 95
- service adapters that can be used 5
- setting business processes (for HTTP adapters) 217
- setting client definition file (Session Bean) 93
- settings in service adapter settings screen (for SessionBean) 229
- settings in service adapter settings screen (for SOAP adapter) 227
- settings related to FTPS 452
- settings related to keystores 452, 453
- settings to be performed on service adapter settings screen (for custom adapters) 250
- settings to be performed on service adapter settings screen (for DB adapters) 234
- settings to be performed on service adapter settings screen (for file adapters) 236
- settings to be performed on service adapter settings screen (for file operations adapters) 242
- settings to be performed on service adapter settings screen (for FTP adapters) 241
- settings to be performed on service adapter settings screen (for HTTP adapters) 249
- settings to be performed on service adapter settings screen (for mail adapters) 248

- settings to be performed on service adapter settings screen (for MDB(DB queue) adapters) 233
- settings to be performed on service adapter settings screen (for MDB (WS-R) adapters) 231
- setting up activity (HTTP reception) 40
- setting up option definition file for J2EE servers (for HTTP adapters) 205
- setup example of HTTP reception (in pass-through mode) 417
- setup example of HTTP reception (in standard mode) 409
- SQL operation definition file 98
- structure of SQL operation definition file 99

## T

---

- troubleshooting exceptions in operations performed from the converter 332
- troubleshooting General custom adapter deployment 330
- troubleshooting General custom adapter startup 331
- types of message format (for fileadapter) 126

## U

---

- urecp\_http\_body\_request.xsd (request message format (for body variable)) 34
- urecp\_http\_dummy\_body\_response.xsd (response message format (for body variable)) 36
- urecp\_http\_header\_request.xsd (request message format (for header variable)) 33
- urecp\_http\_header\_response.xsd (response message format (for header variable)) 35
- user-defined reception 8
  - checking 71
  - defining 7
  - deleting 76
  - validating 73
  - validation contents 73
  - validation method 73

## W

---

- workflow for defining (service adapter) 78
- work flows for defining SOAP reception 10
- WSDL
  - creating 49
  - example of business process used for creating 49
  - example of creating 51