HITACHI
Inspire the Next

uCosminexus Application Server

Overview

3020-3-Y01-30(E)

■ Relevant program products

Applicable OS: Windows Server 2008 x86, Windows Server 2008 x64#, Windows Server 2008 R2#, Windows Server 2012#, Windows Server 2012 R2#

P-2443-7S94 uCosminexus Service Platform version 09-60

Applicable OS: Windows Vista, Windows 7 x86, Windows 7 x64#, Windows 8 x86, Windows 8 x64#, Windows 8.1 x86, Windows 8.1 x64#

P-2443-7T94 uCosminexus Service Architect version 09-60

Applicable OS: Windows Server 2008 x64, Windows Server 2008 R2, Windows Server 2012, Windows Server 2012 R2

P-2943-7S94 uCosminexus Service Platform version 09-60

Applicable OS: Windows 7 x64, Windows 8 x64, Windows 8.1 x64

P-2943-7T94 uCosminexus Service Architect version 09-60

Applicable OS: AIX V6.1, AIX V7.1

P-1M43-7S91 uCosminexus Service Platform(64) version 09-60

Applicable OS: Red Hat Enterprise Linux 5 (AMD/Intel 64), Red Hat Enterprise Linux 6 (AMD/Intel 64)

P-9W43-7S91 uCosminexus Service Platform version 09-60

Applicable OS: Windows Server 2008 x86, Windows Server 2008 x64#, Windows Server 2008 R2#, Windows Server 2012#, Windows XP, Windows Vista, Windows 7 x86, Windows 7 x64#, Windows 8 x86, Windows 8 x64#

P-2443-7H94 uCosminexus Client 09-50

Applicable OS: Windows Server 2008 x64, Windows Server 2008 R2, Windows Server 2012

P-2943-7K97 uCosminexus Application Server 09-50

Applicable OS: Windows 7 x64, Windows 8 x64

P-2943-7F97 uCosminexus Developer 09-50

Applicable OS: AIX V6.1, AIX V7.1

P-1M43-7K91 uCosminexus Application Server (64) 09-50

Applicable OS: HP-UX 11i V2 (IPF), HP-UX 11i V3 (IPF)

P-1J43-7K91 uCosminexus Application Server 09-50

P-1J43-7S91 uCosminexus Service Platform 09-50

Applicable OS: Red Hat Enterprise Linux 5(AMD/Intel 64), Red Hat Enterprise Linux 6 (AMD/Intel 64)

P-9W43-7K92 uCosminexus Application Server 09-50

#: Available only in WOW64 (Windows On Windows 64) environments.

This manual can be used for the products other than the products shown above. For details, see *Release Notes*.

Hitachi trace common library is installed with this product.

■ Export restrictions

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

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

■ Trademarks

BIG-IP, 3-DNS, iControl Services Manager, FirePass, and F5 are either trademarks or registered trademarks of F5 Networks, Inc.

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

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

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

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

Ethernet is a registered trademark of Xerox Corp.

F5, F5 Networks, BIG-IP and iControl are either trademarks or registered trademarks of F5 Networks, Inc in the United States and/or other countries.

Firefox is a registered trademark of the Mozilla Foundation.

HP is a trademark of Hewlett-Packard Development Company, L.P. in the U.S. and other countries.

HP-UX is a product name of Hewlett-Packard Development Company, L.P. in the U.S. and other countries.

IBM, AIX and WebSphere are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide.

IBM, HACMP are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide.

IBM, SecureWay are trademarks of International Business Machines Corporation, registered in many jurisdictions worldwide.

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

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

Itanium is a trademark of Intel Corporation in the United States and other countries.

Linux(R) is the registered trademark of Linus Torvalds in the U.S. and other countries.

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

Microsoft, and Hyper-V are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

Microsoft .NET is software for connecting people, information, systems, and devices.

Microsoft Access is a registered trademark of Microsoft Corporation in the U.S. and other countries.

Microsoft Office and Excel are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

Microsoft Office and Visio are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

Mozilla is a trademark of the Mozilla Foundation in the U.S and other countries.

Netscape is a trademark of AOL Inc.in the U.S. and other countries.

OMG, CORBA, IIOP, 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 and/or its affiliates.

RC2 is a registered trademark or trademark of EMC Corporation in the United States and/or other countries.

RC4 is a registered trademark or trademark of EMC Corporation in the United States and/or other countries.

Red Hat is a trademark or a registered trademark of Red Hat Inc. in the United States and other countries.

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

SAP and R/3 and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP AG in Germany and other countries.

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

All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc., in the United States and other countries. The products bearing SPARC trademarks are based upon an architecture developed by Sun Microsystems, Inc.

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

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

VMware is a registered trademark or trademark of VMware, Inc. in the United States and/or other jurisdictions.

VMware and VMware vSphere ESX are registered trademarks or trademarks of VMware, Inc. in the United States and/or other jurisdictions.

VMware and VMware vSphere are registered trademarks or trademarks of VMware, Inc. in the United States and/or other jurisdictions.

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

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

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

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

X/Open is a registered trademark of The Open Group in the U.K. and other countries.

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

Struts is a frame to build a Web application of Servlets and JSPs released by Apache Struts Project managed by the Apache Software Foundation.

The SOAP application development support function and the SOAP communication platform use WSDL4J distributed based on Common Public License Version 1.0.

This product installs RSA BSAFE(R) software of EMC Corporation.

Other company and product names mentioned in this document may be the trademarks of their respective owners.

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

This product includes software developed by Ben Laurie for use in the Apache-SSL HTTP server project.

Portions of this software were developed at the National Center for Supercomputing Applications (NCSA) at the University of Illinois at Urbana-Champaign.

This product includes software developed by the University of California, Berkeley and its contributors.

This software contains code derived from the RSA Data Security Inc. MD5 Message-Digest Algorithm, including various modifications by Spyglass Inc., Carnegie Mellon University, and Bell Communications Research, Inc (Bellcore).

Regular expression support is provided by the PCRE library package, which is open source software, written by Philip Hazel, and copyright by the University of Cambridge, England. The original software is available from ftp://ftp.csx.cam.ac.uk/pub/software/programming/pcre/

This product includes software developed by Ralf S. Engelschall <rse@engelschall.com> for use in the mod_ssl project (http://www.modssl.org/).

Throughout this document Hitachi has attempted to distinguish trademarks from descriptive terms by writing the name with the capitalization used by the manufacturer, or by writing the name with initial capital letters. Hitachi cannot attest to the accuracy of this information. Use of a trademark in this document should not be regarded as affecting the validity of the trademark.

HITACHI
Inspire the Next

Hitachi, Ltd.

Java™ COMPATIBLE ENTERPRISE EDITION

Java™ COMPATIBLE

■ Acknowledgement

Reliable Messaging includes the technology development results of the 'Business Grid Computing Project' carried out from the year 2003 for three years by the Ministry of Economy, Trade and Industry.

■ Microsoft product name abbreviations

This manual uses the following abbreviations for Microsoft product names.

Full name or meaning	Abbreviation
Microsoft(R) Windows(R) Internet Explorer(R) 6	Internet Explorer 6
Microsoft(R) Windows(R) Internet Explorer(R) 7	Internet Explorer 7
Microsoft(R) Windows(R) Internet Explorer(R) 8	Internet Explorer 8

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

Note that Windows 32 bit and Windows 64 bit are sometimes respectively referred to as Windows x86 and Windows x64.

#: These products are abbreviated with a generic name as Windows Server 2008 Enterprise.

In the text describing Window menus, the Application Server and BPM & ESB platform manuals assume that you are using the following OSs:

For the computer on which the execution environment of Application Server and Service Platform is configured

Windows Server 2008

For the computer on which the development environment of Application Server and Service Platform is configured

Windows 7

The **Start** menu is not displayed in the case of Windows 8 or Windows Server 2012. Therefore, select the menu from the **Start** window that you can display from the bottom left part of the window.

■ Restrictions

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

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

Printed in Japan.

■ Issued

Oct. 2015: 3020-3-Y01-30(E)

■ Copyright

Summary of amendments

The following table lists changes in this manual (3020-3-Y01-30(E)) and product changes related to this manual:

Changes	Location
The manual system of uCosminexus Service Platform and uCosminexus Service Architect have been added.	*chapter 7* to *chapter 10*

In addition to the above changes, minor editorial corrections have been made.

Note that some information formerly placed in the Preface has been moved to the appendix *Reference Material for This Manual*.

Preface

This manual provides an overview of Cosminexus Application Server and the BPM/ESB base. This manual explains the product components of Application Server and the BPM/ESB base, and the functionality of these products. It also explains the organization of the manuals for Application Server and the BPM/ESB base.

For Application Server and the BPM/ESB base, the following program products are used to build and operate the systems and to develop applications:

- uCosminexus Application Server
- uCosminexus Client
- uCosminexus Developer
- uCosminexus Service Architect
- uCosminexus Service Platform

This chapter describes the terms and abbreviations commonly used in the manuals for these program products.

Note that "Cosminexus" and "uCosminexus" might be abbreviated in this document.

■ Related manuals

The following table shows the titles of related manuals, and related manuals used in this manual.

Application Server and BPM/ESB base related manuals

Formal name	Reference number
uCosminexus Application Server First Step Guide	*3020-3-Y00(E)*
uCosminexus Application Server Overview	*3020-3-Y01(E)*
uCosminexus Application Server System Setup and Operation Guide	*3020-3-Y02(E)*
uCosminexus Application Server Virtual System Setup and Operation Guide	*3020-3-Y03(E)*
uCosminexus Application Server System Design Guide	*3020-3-Y04(E)*
uCosminexus Application Server Web Container Functionality Guide	*3020-3-Y05(E)*
uCosminexus Application Server EJB Container Functionality Guide	*3020-3-Y06(E)*
uCosminexus Application Server Common Container Functionality Guide	*3020-3-Y07(E)*
uCosminexus Application Server Expansion Guide	*3020-3-Y08(E)*
uCosminexus Application Server Security Management Guide	*3020-3-Y09(E)*
uCosminexus Application Server Operation, Monitoring, and Linkage Guide	*3020-3-Y10(E)*
uCosminexus Application Server Maintenance and Migration Guide	*3020-3-Y11(E)*
uCosminexus Application Server Compatibility Guide	*3020-3-Y12(E)*
uCosminexus Application Server Application Setup Guide	*3020-3-Y13(E)*
uCosminexus Application Server Management Portal Operation Guide	*3020-3-Y14(E)*
uCosminexus Application Server Command Reference Guide	*3020-3-Y15(E)*
uCosminexus Application Server Definition Reference Guide	*3020-3-Y16(E)*
uCosminexus Application Server Application and Resource Definition Reference Guide	*3020-3-Y17(E)*
uCosminexus Application Server HTTP Server User Guide	*3020-3-Y18(E)*
uCosminexus Application Server Cosminexus Reliable Messaging	*3020-3-Y19(E)*

Formal name	Reference number
uCosminexus Application Server Application Development Guide	*3020-3-Y20(E)*
uCosminexus Application Server API Reference Guide	*3020-3-Y21(E)*
uCosminexus Application Server XML Processor User Guide	*3020-3-Y22(E)*
uCosminexus Application Server Web Service Development Guide	*3020-3-Y23(E)*
uCosminexus Application Server Web Service Security Users Guide	*3020-3-Y24(E)*
uCosminexus Application Server SOAP Application Development Guide	*3020-3-Y25(E)*
uCosminexus Application Server XML Security - Core User Guide	*3020-3-Y26(E)*
uCosminexus Application Server Messages	*3020-3-Y27(E)*
TPBroker Version 5 Transactional distribution object infrastructure TPBroker Users Guide	*3020-3-Y28(E)*
TPBroker Version 5 Transactional distribution object infrastructure TPBroker Operation Guide	*3020-3-Y29(E)*
VisiBroker Version 5 Borland(R) Enterprise Server VisiBroker(R) Developers Guide	*3020-3-Y30(E)*
VisiBroker Version 5 Borland(R) Enterprise Server VisiBroker(R) Programmers Reference	*3020-3-Y31(E)*
uCosminexus Application Server Audit Log Messages	*3020-3-Y32(E)*
uCosminexus Service Platform First Step Guide	*3020-3-Y41(E)*
uCosminexus Service Platform Overview	*3020-3-Y42(E)*
uCosminexus Service Platform Basic Development Guide	*3020-3-Y43(E)*
uCosminexus Service Platform Reception and Adapter Definition Guide	*3020-3-Y44(E)*
uCosminexus Service Platform Setup and Operation Guide	*3020-3-Y45(E)*
uCosminexus Service Platform Reference Guide	*3020-3-Y46(E)*
uCosminexus Service Platform Messages	*3020-3-Y47(E)*
uCosminexus Application Server and BPM/ESB Platform Terminology Guide	*3020-3-Y99(E)*

For the correspondence between the products and manuals, see *3.4 Relationship of products and manuals*.

■ Generic names for a folder and a path

For content common among Windows, HP-UX, AIX, Linux, and Solaris, this manual uses the word *directory* to refer to a Windows *folder*, and uses a slash ("/") to refer to a backslash ("\"). For Windows, replace the word *directory* with *folder*, and "/" with "\".

■ Abbreviations for other products and functions

The following table shows the abbreviations used in this document for other products and functions than those from Microsoft.

Abbreviations		Product name and function name
ACOS	AX2000	AX2000
	AX2500	AX2500
	BS320	BS320 Load Balancer Blade
AMD-V		AMD Virtualization
Application Development Plug-in		Cosminexus Application Development Plug-in

Abbreviations		Product name and function name
Application Server		uCosminexus Application Server
		uCosminexus Application Server(64)
BIG-IP	BIG-IP v9	BIG-IP software version 9.1.0 or later
	BIG-IP v10.1	BIG-IP software version 10.1.0 or later
	BIG-IP v10.2	BIG-IP software version 10.2.0 or later
	BIG-IP v11	BIG-IP software version 11.0.0 or later
BJEX or Batch Job Execution Server		Batch Job Execution System - Base uCosminexus Batch Job Execution Server
BPM/ESB platform		Cosminexus business process management/enterprise service bus
CJMSP blocker		Cosminexus JMS provider blocker function
CJMSP resource adapter		Cosminexus JMS provider resource adapter
CJMS provider		Cosminexus JMS provider
CJPA provider		Cosminexus JPA provider
Client		uCosminexus Client
Common Library		Cosminexus Common Library
Component Container		Cosminexus Component Container
Component Container - Client		Cosminexus Component Container - Client
Component Container - Redirector		Cosminexus Component Container - Redirector
Component Library		Cosminexus Component Library
CTM or Component Transaction Monitor		Cosminexus Component Transaction Monitor
DABroker Library		Cosminexus DABroker Library
DABroker Library-Database Connection Server		Cosminexus DABroker Library-Database Connection Server
DB Connector		Cosminexus DB Connector
DB Connector for Reliable Messaging		DB Connector for Cosminexus Reliable Messaging
DCCM3		VOS1 DCCM3
		VOS3 XDM/DCCM3
Developer		uCosminexus Developer
Developer's Kit for Java		Cosminexus Developer's Kit for JavaTM
Driver		Cosminexus Driver
EADs or Elastic Application Data store for Application Server		uCosminexus Elastic Application Data store for Application Server
EADs client or Elastic Application Data store Client for Application Server		uCosminexus Elastic Application Data store Client for Application Server
Eclipse		Eclipse 3.6.1
FireFox		FireFox(R)
HCSC		Hitachi Cosminexus Service Coordinator

Abbreviations		Product name and function name
HCSC-Business Process, business process infrastructure or BP infrastructure		Hitachi Cosminexus Service Coordinator - Business Process
HCSC-Data Transform or data transformation infrastructure		Hitachi Cosminexus Service Coordinator - Data Transform
HCSC-Manager or HCSC-MNG		Hitachi Cosminexus Service Coordinator - Manager
HCSC-Messaging, HCSC-MSG, or messaging infrastructure		Hitachi Cosminexus Service Coordinator - Messaging
HCSCTE		Hitachi Cosminexus Service Coordinator Tools for Eclipse
HiRDB or HiRDB server	HiRDB Embedded Server, or embedded database	HiRDB Embedded Server Version 8
	HiRDB Server	HiRDB Server Version 9
	HiRDB Server with Additional Function	HiRDB Server with Additional Function Version 9
	HiRDB/Parallel Server	HiRDB/Parallel Server Version 7
		HiRDB/Parallel Server Version 8
	HiRDB/Single Server	HiRDB/Single Server Version 7
		HiRDB/Single Server Version 8
HiRDB Run Time or HiRDB Client		HiRDB/Run Time Version 7
		HiRDB/Run Time Version 8
		HiRDB/Run Time Version 9
		HiRDB/Developer's Kit Version 8
		HiRDB/Developer's Kit Version 9
		HiRDB Developer's Suite Version 9
HiRDB Type4 JDBC Driver		HiRDB Type4 JDBC driver
HTTP Server		Cosminexus HTTP Server
Intel VT		Intel(R) Virtualization Technology
IPF		Itanium(R) Processor Family
JAX-WS functionality		Cosminexus JAX-WS
JP1/AJS	JP1/AJS - Agent	Job Management Partner 1/Automatic Job Management System 3 - Agent
	JP1/AJS - Manager	Job Management Partner 1/Automatic Job Management System 3 - Manager
	JP1/AJS - View	Job Management Partner 1/Automatic Job Management System 3 - View
JP1/Audit Management - Manager		Job Management Partner 1/Audit Management - Manager
		Job Management Partner 1/NETM/Audit - Manager
JP1/Cm2	JP1/Cm2/ESA	Job Management Partner 1/Cm2/Extensible SNMP Agent
JP1/ESP		Job Management Partner 1/Extensible Service Probe
JP1/File Transmission Server/FTP		Job Management Partner 1 Version 9 JP1/File Transmission Server/FTP

Abbreviations			Product name and function name
JP1/IM	JP1/IM - Manager		Job Management Partner 1/Integrated Management - Manager
	JP1/IM - View		Job Management Partner 1/Integrated Management - View
JP1/ITRM	JP1/ITRM - Manager		Job Management Partner 1/IT Resource Management - Manager
JP1/NETM/Audit			Job Management Partner 1/NETM/Audit - Manager
JP1/PFM			Job Management Partner 1/Performance Management
JP1/PFM	JP1/PFM - Agent	JP1/PFM - Agent for Cosminexus	Job Management Partner 1/Performance Management - Agent Option for uCosminexus Application Server
		JP1/PFM - Agent for Virtual Machine	Job Management Partner 1/Performance Management - Agent Option for Virtual Machine
	JP1/PFM - Base		Job Management Partner 1/Performance Management - Base
	JP1/PFM - Manager		Job Management Partner 1/Performance Management - Manager
	JP1/PFM - Web Console		Job Management Partner 1/Performance Management - Web Console
JP1/SC/DPM			Job Management Partner 1/ServerConductor/Deployment Manager Standard Edition
Loadflowbal			HA8000-ie/Loadflowbal
Management Server			Cosminexus Management Server
Manager			Cosminexus Manager
MyEclipse			MyEclipse for Cosminexus
Oracle, or ORACLE	Oracle 11g		Oracle Database 11g
			Oracle Database 11g R2(TM)
	Oracle 12c		Oracle Database 12c Enterprise Edition
PP installer			Hitachi PP installer
PRF or Performance Tracer			Cosminexus Performance Tracer
Process Modeler			Process Modeler 5 for Microsoft(R) Visio Professional Edition
RM or Reliable Messaging			Cosminexus Reliable Messaging
SAP R/3			SAP R/3(R)
Server Plug-in			Cosminexus Server Plug-in
Service Adapter Architect for Flat Files			uCosminexus Service Adapter Architect for Flat Files
Service Adapter Architect for FTP			uCosminexus Service Adapter Architect for FTP
Service Adapter Architect for Message Queue			uCosminexus Service Adapter Architect for Message Queue
Service Adapter Architect for Object Access			uCosminexus Service Adapter Architect for Object Access
Service Adapter Architect for TP1			uCosminexus Service Adapter Architect for TP1
Service Architect			uCosminexus Service Architect

Abbreviations			Product name and function name
Service Adapter for Flat Files			uCosminexus Service Adapter for Flat Files
Service Adapter for FTP			uCosminexus Service Adapter for FTP
Service Adapter for Message Queue			uCosminexus Service Adapter for Message Queue
Service Adapter for Object Access			uCosminexus Service Adapter for Object Access
Service Adapter for TP1			uCosminexus Service Adapter for TP1
Service Coordinator Interactive Workflow			uCosminexus Service Coordinator Interactive Workflow
Service Coordinator-Manager			Cosminexus Service Coordinator-Manager
Service Coordinator or CSC			Cosminexus Service Coordinator
Service Development Plug-in			Cosminexus Service Development Plug-in
Service Platform			uCosminexus Service Platform
			uCosminexus Service Platform(64)
Service Platform - Base			uCosminexus Service Platform - Base
Smart Composer			Cosminexus Smart Composer
SOAP communication infrastructure			Cosminexus SOAP communication infrastructure
Struts			Jakarta Struts 1.1
TMS-4V/SP			Transaction Management System-4V/System Product
TMS-4V/SP/Server			Transaction Management System-4V/System Product/ Server
TP1 Connector			uCosminexus TP1 Connector
TP1/Base			uCosminexus TP1/Server Base
TP1/Client	TP1/Client/J		uCosminexus TP1/Client/J
	TP1/Client/P		uCosminexus TP1/Client/P
	TP1/Client/W		uCosminexus TP1/Client/W
TP1/COBOL adapter			TP1/COBOL adapter for Cosminexus
TP1/EE			TP1/Server Base Enterprise Option
TP1/Message Queue - Access			uCosminexus TP1/Message Queue - Access
TP1/Web			uCosminexus TP1/Web
TPBroker			Cosminexus TPBroker
TPBroker for C++			Cosminexus TPBroker for C++
TPBroker for Java			Cosminexus TPBroker for Java
UNIX	AIX		AIX V6.1
			AIX V7.1
	HP-UX or HP-UX (IPF)		HP-UX 11i V2 (IPF)
			HP-UX 11i V3 (IPF)
	Linux	Linux (IPF)	Red Hat Enterprise Linux(R) 5 Advanced Platform (Intel Itanium)
			Red Hat Enterprise Linux(R) 5 (Intel Itanium)

Abbreviations			Product name and function name
UNIX	Linux	Linux (x86/ AMD64 & Intel EM64T)	Red Hat Enterprise Linux(R) 5 Advanced Platform (AMD/ Intel 64)
			Red Hat Enterprise Linux(R) 5 (AMD/Intel 64)
			Red Hat Enterprise Linux(R) 6 Advanced Platform (x86)
			Red Hat Enterprise Linux(R) 6 (AMD/Intel 64)
			Red Hat Enterprise Linux(R) 6 (x86)
			Red Hat Enterprise Linux(R) 6 Advanced Platform (AMD/ Intel 64)
			Red Hat Enterprise Linux(R) Server 6 (32-bit x86)
			Red Hat Enterprise Linux(R) Server 6 (64-bit x86_64)
	Solaris		Solaris 10
			Solaris 10 (SPARC)
			Solaris 10 (x64)
			Solaris 9
			Solaris 9 (SPARC)
Virtual Server Manager			Cosminexus Virtual Server Manager
VMware	VMware ESX		VMware(R) ESX
	VMware Tools		VMware(R) Tools
	VMware vCenter Server		VMware(R) vCenter Server
	VMware vSphere Client		VMware(R) vSphere Client
Web Redirector			uCosminexus Web Redirector
Web Server			Hitachi Web Server
Web Services			Cosminexus Web Services
Web Services - Base			Cosminexus Web Services - Base
Web Services - Security			Cosminexus Web Services - Security
Web service security functionality			Cosminexus Web service security functionality
XDM/RD E2			Extensible Data Manager/Relational Database Extended Version 2
XML Processor			Cosminexus XML Processor
XML Security			Cosminexus XML Security
XML Security - Core			Cosminexus XML Security - Core
Adapter command			Cosminexus adapter command
JPA function of Application Server			Cosminexus JPA
Installer			Hitachi integrated installer
Common modules			Cosminexus common modules
Class wise statistics			Hitachi class wise statistics
Code conversion - Development Kit			uCosminexus Hitachi code conversion - Development Kit

Abbreviations	Product name and function name
Code conversion - Server Runtime	uCosminexus Hitachi code conversion - Server Runtime
Product information file	Cosminexus product information file
JavaVM or JavaVM of product	Hitachi JavaVM
Trace common library	Hitachi trace common library
Version information	Hitachi version information
Batch library	Cosminexus batch library
Standard login module	Cosminexus standard login module
Monitor startup command	Cosminexus monitor startup command
XML Processor	Cosminexus XML Processor

In this document, JP1 related products are abbreviated with a generic name JP1.

As a generic name, application server is used for both Application Server and Developer.

For each version of Linux, the following abbreviations might be used.

Abbreviation	OS
Red Hat Enterprise Linux 5	Red Hat Enterprise Linux(R) 5 Advanced Platform (Intel Itanium)
	Red Hat Enterprise Linux(R) 5 (Intel Itanium)
	Red Hat Enterprise Linux(R) 5 Advanced Platform (AMD/Intel 64)
	Red Hat Enterprise Linux(R) 5 (AMD/Intel 64)
Red Hat Enterprise Linux Server 6	Red Hat Enterprise Linux(R) Server 6 (32-bit x86)
	Red Hat Enterprise Linux(R) Server 6 (64-bit x86_64)

■ Java related abbreviations

Abbreviation	Java term
AWT	Abstract Window Toolkit
BMP	Bean-Managed Persistence
BMT	Bean-Managed Transaction
BOM	Byte Order Mark
CDI	Contexts and Dependency Injection
Connector 1.0	J2EE™ Connector Architecture 1.0
Connector 1.5	J2EE™ Connector Architecture 1.5
DI	Dependency Injection
EAR	Enterprise Archive
EJB or Enterprise JavaBeans	Enterprise JavaBeans™
EJB QL	EJB™ Query Language
J2EE or Java 2 Platform, Enterprise Edition	J2EE™
	Java™ 2 Platform, Enterprise Edition

Abbreviation	Java term
J2SE	Java™ 2 Platform, Standard Edition
JAAS	Java™ Authentication and Authorization Service
JAR	Java™ Archive
Java	Java™
Java 2 Runtime Environment, Standard Edition	Java™ 2 Runtime Environment, Standard Edition
Java 2 SDK, or Java 2 SDK, Standard Edition	Java™ 2 Software Development Kit, Standard Edition
JavaAPI	Java™ Application Programming Interface
JavaBeans	JavaBeans™
Java EE, or Java Platform, Enterprise Edition	Java™ Platform, Enterprise Edition
Java HotSpot Client VM	Java HotSpot™ Client Virtual Machine
Java HotSpot Server VM	Java HotSpot™ Server Virtual Machine
JavaMail	JavaMail™
Java SE	Java™ Platform, Standard Edition
JavaVM or JVM	Java™ Virtual Machine
JAX-WS	Java™ API for XML-Based Web Services
JAXB	Java™ Architecture for XML Binding
JAXB or The Java Architecture for XML Binding	The Java™ Architecture for XML Binding
JAXP or Java API for XML Processing	Java™ API for XML Processing
JCA	J2EE™ Connector Architecture
JCE	Java™ Cryptography Extension
JDBC	Java™ Database Connectivity
	JDBC™
JDK	Java™ Development Kit
	JDK™
JMS	Java™ Message Service
JMX	Java™ Management Extensions
JNDI	Java Naming and Directory Interface™
JNI	Java™ Native Interface
JPA	Java™ Persistence API
JPQL	Java™ Persistence Query Language
JSF	JavaServer™ Faces Reference Implementation (RI) Version: 1.1_01 FCS
JSP	JavaServer Pages™
	JSP™

Abbreviation	Java term
JSTL	JavaServer Pages™ Standard Tag Library
JTA	Java™ Transaction API
JTS	Java™ Transaction Service
POJO	Plain Old Java™ Object
SAAJ	SOAP with Attachments API for Java™
Servlet or servlet	Java™ Servlet
SMAP	Source Map
WAR	Web Archive

■ Abbreviations for general terms

The following table shows abbreviations used in this document for general terms.

Abbreviation	Term
ACL	Access Control List
ACOS	Advanced Core Operating System
AES	Advanced Encryption Standard
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
BLOB	Binary Large Object
BP	Business Process
BPEL	Business Process Execution Language
BPM	Business Process Management
BPMN	Business Process Modeling Notation
C14N	Canonicalization
CA	Certificate Authority
CMP	Container-Managed Persistence
CMR	Container-Managed Relationship
CMT	Container-Managed Transaction
CoC	Convention over Configuration
CORBA	Common Object Request Broker Architecture
CPU	Central Processing Unit
CR	Carriage Return
CRL	Certificate Revocation List
CSR	Certificate Signing Request
CSS	Cascading Style Sheets
CSV	Comma Separated Value

Abbreviation	Term
CUI	Character User Interface
CVS	Concurrent Versions System
DAO	Data Access Object
DB	Database
DBMS	Database Management System
DD	Deployment Descriptor
DDL	Data Definition Language
DES	Data Encryption Standard
DII	Dynamic Invocation Interface
DIT	Directory Information Tree
DLL	Dynamic Link Library
DMZ	Demilitarized Zone
DN	Distinguished Name
DNS	Domain Name System
DOM	Document Object Model
DoS	Denial of Service
DSA	Digital Signature Algorithm
DTD	Document Type Definition
DTO	Data Transfer Object
EIS	Enterprise Information System
EL	Expression Language
EOD	Ease of Development
ERP	Enterprise Resource Planning
ESB	Enterprise Service Bus
ETL	Extract Transform Loading
EUC	Extended UNIX Code
FAQ	Frequently Asked Questions
FF	Form Feed
FIFO	First-In First-Out
FK	Foreign Key
FLOPS	Floating point number Operations Per Second
FQDN	Fully Qualified Domain Name
FTP	File Transfer Protocol
GC	Garbage Collection
GIF	Graphic Interchange Format
GMT	Greenwich Mean Time

Abbreviation	Term
GPKI	Government Public Key Infrastructure
GUI	Graphical User Interface
HA	High Availability
HMAC	Hash based MAC
HNTRLib	Hitachi Network Objectplaza Trace Library
HTML	Hyper Text Markup Language
HTTP	Hyper Text Transfer Protocol
HTTPS	Hyper Text Transfer Protocol Security
I/O	Input/Output
IANA	Internet Assigned Numbers Authority
ID	Identifier
IDE	Integrated Development Environment
IIOP	Internet Inter-Orb Protocol
IIS	Internet Information Services
IP	Internet Protocol
IPF	Itanium(R) Processor Family
IPv6	Internet Protocol Version 6
ISAPI	Internet Server Application Programming Interface
iSCSI	Internet Small Computer System Interface
ISO	International Organization for Standardization
IT	Information Technology
IV	Initialization Vector
JIS	Japanese Industrial Standards
JSSE	Java Secure Socket Extension
JST	Japan Standard Time
JVMDI	Java Virtual Machine Debug Interface
JVMPI	Java Virtual Machine Profiler Interface
JVMTI	Java Virtual Machine Tool Interface
LAN	Local Area Network
LB	Load Balancer
LDAP	Lightweight Directory Access Protocol
LDIF	LDAP Data Interchange Format
LF	Line Feed
LGPKI	Local Government Public Key Infrastructure
MAC	Message Authentication Code
MB/s	Megabyte per Second

Abbreviation	Term
MBean	Managed Bean
Mbit/s	Megabit per Second
MDA	Model Driven Architecture
MDB	Message-Driven Bean
MHP	Message Handling Program
MIB	Management Information Base
MIME	Multipurpose Internet Mail Extensions
MIPS	Million Instructions Per Second
MTU	Maximum Transmission Unit
MVC	Model View Controller
NIC	Network Interface Card
NTP	Network Time Protocol
OAEP	Optimal Asymmetric Encryption Padding
OASIS	Organization for the Advancement of Structured Information Standards
OID	Object Identifier
OLTP	On-Line Transaction Processing
OMG	Object Management Group
ORB	Object Request Broker
OS	Operating System
OTM	Object Transaction Monitor
OTS	Object Transaction Service
PIM	Platform Independent Model
PK	Primary Key
PKI	Public Key Infrastructure
POA	Portable Object Adapter
POP3	Post Office Protocol - Version 3
PSM	Platform Specific Model
PTP	Point-to-Point
QName	Qualified Name
QoS	Quality of Service
RAC	Real Application Clusters
RAR	Resource Adapter Archive
	Roshal Archive
RD orRDB	Relational Database
REST	Representational State Transfer
RFC	Request For Comments

Abbreviation	Term
RMD	Reliable Messaging Destination
RMI	Remote Method Invocation
RMS	Reliable Messaging Source
RPC	Remote Procedure Call
RSA	Rivest, Shamir and Adleman
SaaS	Software as a Service
SAN	Storage Area Network
SAS	Serial Attached SCSI
SAX	Simple API for XML
SAX1	Simple API for XML 1.0
SAX2	Simple API for XML 2.0
SDK	Software Development Kit
SEI	Service Endpoint Interface
SFO	Session Fail Over
SHA	Secure Hash Algorithm
SMTP	Simple Mail Transfer Protocol
SMTPS	SMTP over SSL
SNMP	Simple Network Management Protocol
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SpecInt	Standard Performance Evaluation Corporation Integer benchmark
SPI	Service Provider Interface
SPP	Service Providing Program
ssh	Secure Shell
SSL	Secure Sockets Layer
SUP	Service Using Program
TCP	Transmission Control Protocol
TCS	Transaction Context Server
TLD	Tag Library Descriptor
TLS	Transport Layer Security
TrAX	Transformation API for XML
TSC	Time Stamp Counter
TSV	Tab Separated Values
UAC	User Account Control
UAP	User Application Program
UCS	Universal multi-octet coded Character Set

Abbreviation	Term
UDDI	Universal Description, Discovery and Integration
UML	Unified Modeling Language
UNC	Universal Naming Convention
UOC	User Own Coding
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
URN	Uniform Resource Name
UTC	Universal Time Coordinated
UTF	UCS Transformation Format
UTF-8	8-bit UCS Transformation Format
VM	Virtual Machine
W3C	World Wide Web Consortium
WFA	Work-Flow Architecture
WS	Web Service
WSDL	Web Services Description Language
WSDL4J	Web Services Description Language for Java Toolkit
WS-I	Web Services Interoperability
WS-R	Web Services Reliability
WST	Web Standard Tools
WTP	Eclipse Web Tools Platform
XML	Extensible Markup Language
XPath	XML Path Language
XSL	Extensible Stylesheet Language
XSLT	XSL Transformations

■ Conventions: Fonts and symbols

Font and symbol conventions are classified as:

- General font conventions
- Conventions in syntax explanations

These conventions are described below.

General font conventions

The following table lists the general font conventions:

Font	Convention
Bold	Bold type indicates text on a window, other than the window title. Such text includes menus, menu options, buttons, radio box options, or explanatory labels. For example, bold is used in sentences such as the following: • From the **File** menu, choose **Open**.

Font	Convention
Bold	• Click the **Cancel** button. • In the **Enter name** entry box, type your name.
Italics	Italics are used to indicate a placeholder for some actual text provided by the user or system. Italics are also used for emphasis. For example: • Write the command as follows: `copy` *source-file target-file* • Do *not* delete the configuration file.
`Code font`	A code font indicates text that the user enters without change, or text (such as messages) output by the system. For example: • At the prompt, enter `dir`. • Use the `send` command to send mail. • The following message is displayed: `The password is incorrect.`

Examples of coding and messages appear as follows (although there may be some exceptions, such as when coding is included in a diagram):

```
MakeDatabase
...
StoreDatabase temp DB32
```

In examples of coding, an ellipsis (...) indicates that one or more lines of coding are not shown for purposes of brevity.

■ Conventions: Version numbers

The version numbers of Hitachi program products are usually written as two sets of two digits each, separated by a hyphen. For example:

- Version 1.00 (or 1.0) is written as 01-00.
- Version 2.05 is written as 02-05.
- Version 2.50 (or 2.5) is written as 02-50.
- Version 12.25 is written as 12-25.

The version number might be shown on the spine of a manual as *Ver. 2.00,* but the same version number would be written in the program as *02-00*.

■ Conventions: KB, MB, GB, and TB

This manual uses the following conventions:

- 1 KB (kilobyte) is 1,024 bytes.
- 1 MB (megabyte) is $1{,}024^2$ bytes.
- 1 GB (gigabyte) is $1{,}024^3$ bytes.
- 1 TB (terabyte) is $1{,}024^4$ bytes.

■ Non-supported functionality

Some functionality described in this manual is not supported. Non-supported functionality includes:

- Audit log functionality
- Business Process Management and Enterprise Service Bus
- Compatibility functionality

- Cosminexus Component Transaction Monitor
- Cosminexus DABroker Library
- Cosminexus Reliable Messaging
- Cosminexus TPBroker and VisiBroker
- Cosminexus Web Service - Security
- Cosminexus XML Security - Core functionality
- JP1 linkage functionality
- Management portal functionality
- Migration functionality
- Session failover functionality
- SOAP applications complying with specifications other than JAX-WS 2.1
- uCosminexus OpenTP1 linkage functionality
- Virtualized system functionality
- XML Processor high-speed parse support functionality

■ Non-supported compatibility functionality

"Compatibility functionality" in the above list refers to the following functionality:

- Basic mode
- Check of JSP source compliance (cjjsp2java) with JSP1.1 and JSP1.2 specifications
- Database connection using Cosminexus DABroker Library
- EJB client application log subdirectory exclusive mode
- J2EE application test functionality
- Memory session failover functionality
- Servlet engine mode
- Simple Web server functionality
- Switching multiple existing execution environments
- Using EJB 2.1 and Servlet 2.4 annotation

Contents

Part 1: Overview

1 Application Server and BPM/ESB as a Cloud Service Platform

This chapter gives an overview of Application Server and the BPM/ESB base. Application Server and the BPM/ESB base are products that serve as a platform to execute cloud services.

Application Server is located at the core of a business system and serves as a base for executing applications. With Application Server, you can build and operate an execution environment that is compliant with *Java EE 6* technology standards.

Also, the BPM/ESB base is a product that is a base for promptly responding to the optimization and changes to a business system according to business needs, by using BPEL-compliant business process management and the enterprise service path function. This product can integrate a variety of systems as services, by using an adapter.

1.1 Role of Application Server and the BPM/ESB base in the cloud Service Platform

Application Server and the BPM/ESB base are products that create and execute efficient services as a cloud Service Platform.

The quick creation and execution of services required by the ever-changing business environment is essential on a system that provides cloud services. Application Server and the BPM/ESB base serve as the base to create and execute the services.

Application Server is a base to execute the online processes and batch processes required for efficiently executing the created services as per the required format. The *BPM/ESB base* executes a control to combine multiple services according to the business requirements and execute these services as a single new service.

This section explains the position of Application Server and the BPM/ESB base on a cloud Service Platform.

1.1.1 Role of Application Server

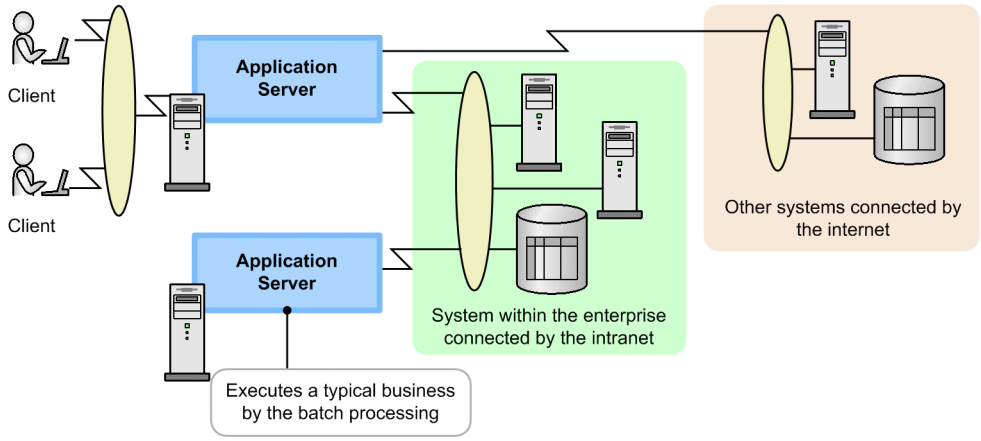
The ever-changing information-based society requires an infrastructure that can flexibly support the market needs for expanding business. To implement such an infrastructure, it is necessary to build a highly extensible information system, and the merits of this information system play a major role in determining the success of a business. Concurrently, business lifecycles are getting shorter in accordance with the speed of social changes. Therefore, it is now an essential requirement of an information system to be able to flexibly support the changes in business and to reuse the existing resources without any wastage.

Application Server realizes a system that satisfies these requirements.

Application Server is a base for an information system having high reliability and expandability that can flexibly respond to the drastic changes in a business environment and business strategies. You can use Application Server to build an execution environment for applications that are compliant with Java EE technology standards, and can efficiently develop and operate the systems with the high flexibility and reliability that are features of Java EE.

The following figure shows an overview of Application Server.

Figure 1–1: Overview of Application Server



Application Server is a base for executing business based on requests sent online from clients and a platform for executing daily business operations by using batch processing.

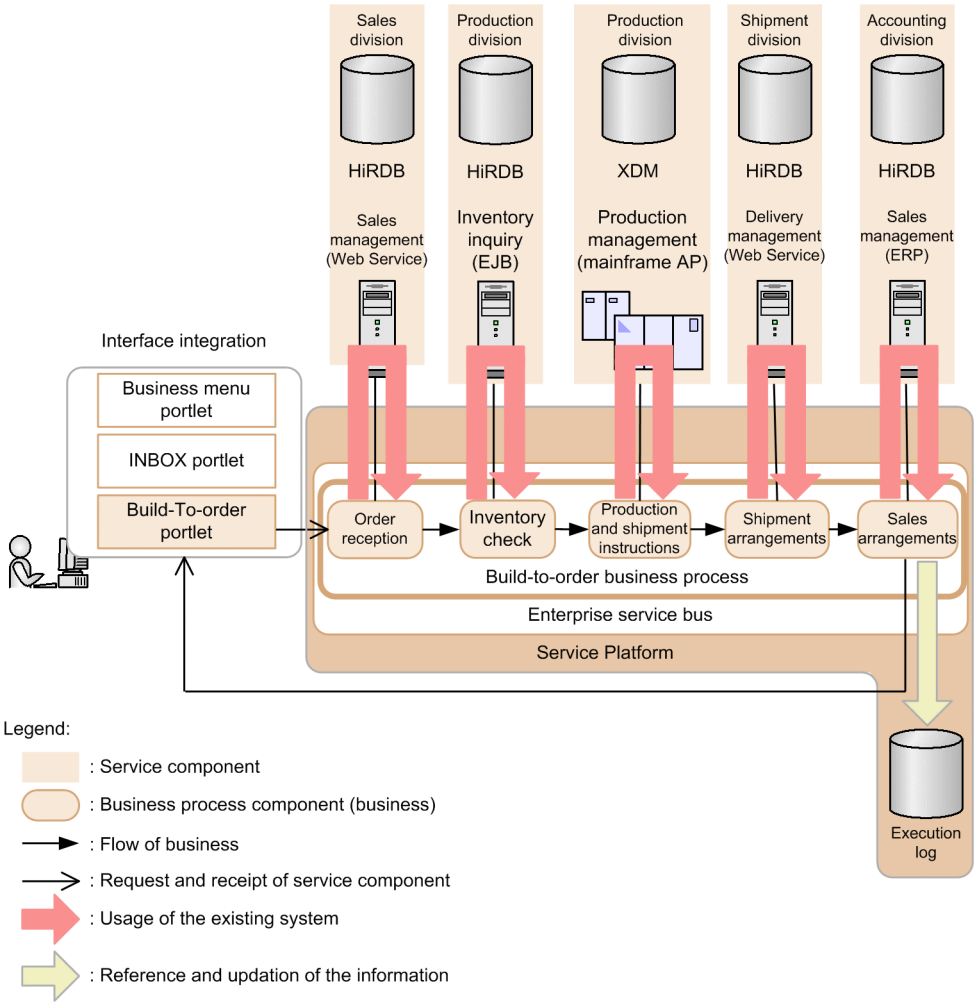
In addition to Java, Application Server is also compliant with other business standards such as CORBA. This enables smooth integration with other systems and also the porting from other systems.

1.1.2 What is the BPM/ESB base

The BPM/ESB base provides the functionality for implementing a system that applies SOA (Service Oriented Architecture) on the Service Platform. This section describes the objectives and details of the Service Platform implemented by the BPM/ESB base.

The *Service Platform* is a platform for the development and operation of an SOA applied system. You can build and execute right from a business process up to the connection with a service, in a unified development and operating environment. As a result, you can quickly build and execute a new system with the combination of services by deriving the advantages of an SOA. At the center is process integration that integrates services and processes. The Service Platform enables the process integration. The following figure shows an example of a system that uses the Service Platform.

Figure 1–2: Example of a system that uses the Service Platform



The Figure 1-2 shows an example of using SOA in the production and distribution system. The service is automatically called in accordance with the business flow. The configuration shown in Figure 1-2 has the following advantages:

- The business operations can be easily opened step by step.
- A business process corresponding to the actual business can be executed.
- A speedy delivery is possible with a proper understanding of the stock status and the production progress status.
- The lead time can be curtailed.

The execution and operation environment of the Service Platform have functionality for the service integration in addition to the functionality of the execution environment of Application Server. This functionality has the enterprise

service bus functionality, which is the core functionality of SOA. You can build a system that quickly adopts the changes in the policies to be executed by freely combining the services. You can also build a highly reliable service by freely combining services that are separated from the existing system or services that are externally provided.

The development environment of the Service Platform has a functionality to execute the system integration in addition to the functionality of the development environment of Application Server. You can use definition tools such as the process definition, data conversion definition, and service adapter definition that are required for the process integration, as a Plug-in of Eclipse. You can integrate a process with a series of operations on Eclipse starting from the business process up to the service connection.

Among the SOA applied system development techniques, the development environment of the The Service Platform supports all the tasks from the detailed design up to the implementation and testing of the business processes that include interfaces. You can also design and implement the components, by using the functionality of the Service Platform.

As a result of this, it is possible to integrate a service that is already operating in the execution environment of an application or in an environment other than the Service Platform and to provide this service as a new service to the user.

1.2 Relationship of Application Server with BPM/ESB base

This section describes the relationship of Application Server with the BPM/ESB base.

Application Server consists of the application development environment to develop an application for implementing the services depending on the business contents and the application execution environment that provides services to the users by executing the developed application.

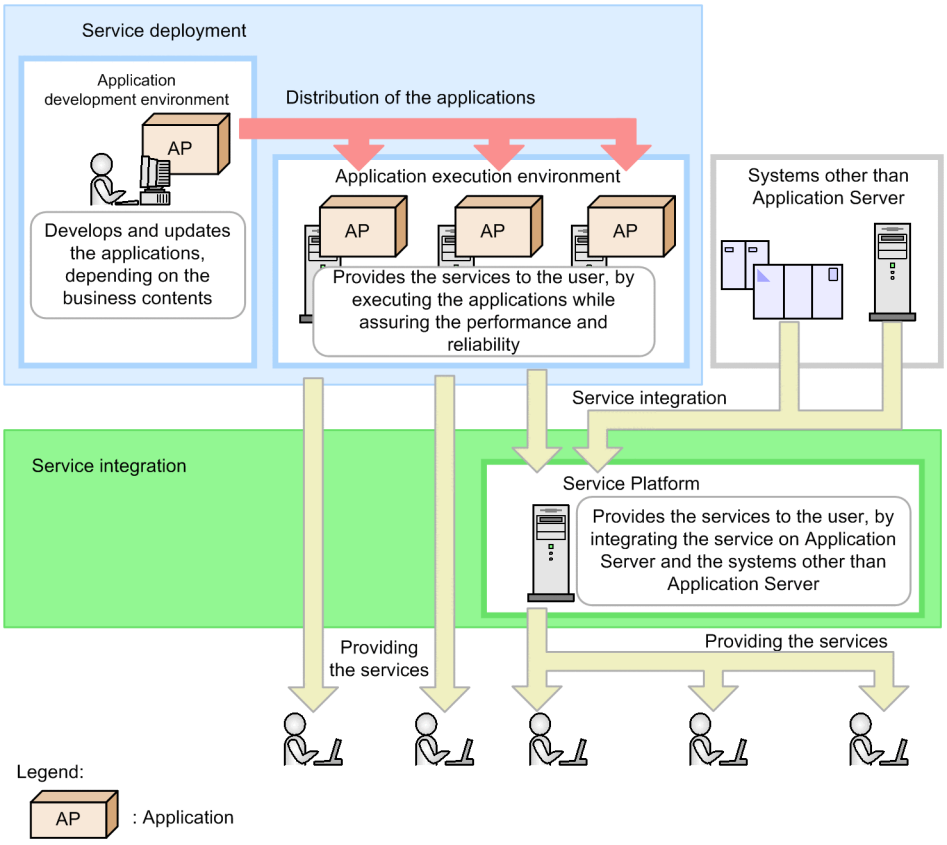
The Service Platform that serves as the BPM/ESB base consists of functionality for the BPEL-compliant business process management and functionality for the enterprise service bus that implements the service integration.

The ESB uses an application required for business, as a service. The ESB combines the existing services with the newly created services for promptly providing a new service. The business process defines the execution sequence when multiple services are combined.

You can implement a one-stop-service with the combination of new applications and existing applications by using the above functionalities.

The following figure shows an example of combining Application Server and the Service Platform.

Figure 1–3: Example of combining Application Server and the Service Platform



The *application execution environment* provides services to the user, by promptly processing the requests from the user while securing performance and reliability. This environment efficiently executes batch jobs. The application execution environment is an execution platform with high availability and fault-tolerance for executing applications with various requirements, such as services that must be operated round the clock, critical operations that cannot be stopped, and key operations executed as batch jobs. You can also use the application execution environment to implement the system for security and internal control. You can also execute one application in multiple execution environments to secure availability.

You can use the *application development environment* to develop a new application or to change the existing application when business contents are changed, or when there is a change in the organization structure. You can

perform seamless and efficient operations ranging from the window designing to the business-processing implementation for integrating with back-end systems by using the application development environment.

You can use the *Service Platform* for re-using the existing applications or services provided by other systems, when business contents are changed or the organization structure is changed. The Service Platform can integrate the services provided by the application execution environment and the services provided by other than Application Server, and can provide these integrated services to the user as a new service. You can use the Service Platform as a platform to provide the existing services provided by various applications and systems, with an integrated interface.

2 Product Configuration of Application Server and the BPM/ESB Base

This chapter describes the product configuration of Application Server and the BPM/ESB base. Application Server and the BPM/ESB base consist of the products for building an execution environment and development environment, and also optional products that you might require in the execution and development environments. Furthermore, optional products consist of multiple software, called *component software*.

2.1 Classification and features of products

Application Server products can be classified according to their usage as follows. The following table describes the product classifications.

Table 2–1: Classification of Application Server products

Classification	Products
Products for building the Application Server execution environment	• *Application Server*
Products for building the Application Server development environment	• *Developer*
Products for building the BPM/ESB base execution environment	• *Service Platform*
Products for building the BPM/ESB base development environment	• *Service Architect*
Optional products	• *Client*

This section explains the features of each product.

2.1.1 Products for building an Application Server execution environment

Application Server is the product for building the execution environment. Application Server builds an execution environment to execute the Java EE-compliant J2EE applications on the server machine. You can also execute an application corresponding to a Web service and an application corresponding to an asynchronous communication with messages. You can use a Web browser or an EJB client application on a client machine.

Note that with the execution environment, you can also use optional products depending on the system configuration.

2.1.2 Products for building the Application Server development environment

Developer is the product for building the development environment. Developer builds a development environment for J2EE applications to be executed in the Application Server execution environment. You can perform operations such as assembling, debugging, deploying, and testing, depending on the J2EE applications developed with IDE.

Furthermore, when you use Eclipse to develop J2EE applications, you can use WTP (Eclipse Web Tools Project). WTP is an integrated development environment in which the plug-ins, used for the application development on Eclipse, provided by eclipse.org, are compiled. If you use WTP, you can execute a series of development-related operations ranging up to coding, testing, and debugging operations.

Note that you can use Developer only in the Windows environment.

2.1.3 Product for building the BPM/ESB base execution environment

Service Platform is the product for building the BPM/ESB base execution environment.

This product operates on a system with SOA. This product sets up the contents developed by using Service Architect and delivers the system information required for the development to Service Architect.

For details on operations by using Service Platform, see the *BPM/ESB Service Platform System Setup and Operation Guide*.

2.1.4 Product for building the BPM/ESB base development environment

Service Architect is the product for building the BPM/ESB base development environment.

You can develop a definition for calling a service required for an SOA applied system, as well as a definition of the business process and data conversion. You can use Service Platform to set up the contents developed by using Service Architect.

For details on system development using Service Architect, see the *BPM/ESB Service Platform Basic Development Guide*, and the *BPM/ESB Service Platform Reception and Adapter Definition Guide*.

2.1.5 Optional products

You can use optional products depending on the system configuration when building the execution environment on Application Server.

You can use optional products to set up a machine with a limited usage in a business system.

Optional products are as follows:

- *Client*

  You can use *Client* for a configuration that directly calls an Enterprise Bean on the execution environment from an application on the client machine. Furthermore, you can use *Client* for a configuration that calls a Web Service in the Web Services execution environment from an application on the client machine.

For details on the OS on which the optional products can be used, see *2.3 (1) Prerequisite OS*.

2.2 Component software

This chapter describes details on the component software of Application Server and the BPM/ESB base. The component software executes Application Server and BPM/ESB base functionality in combination with the functionality of other component software and not independently.

The component software corresponds to the Application Server and BPM/ESB base functionality. The functionality that can be executed differs depending on the type of the component software included in the product.

2.2.1 Relationship of products and component software

This section describes the relationship of the products with the component software included in the products.

The following table describes the component software corresponding to each product:

Table 2–2: Relation of products and the component software

Component software name	Application Server		BPM/ESB base		Optional products
	Application Server	Developer	Service Platform	Service Architect	Client
Application Development Plug-in	--	Y	--	Y	--
Component Container	Y	Y	Y	Y	--
Component Container - Client	--	--	--	--	Y
Component Container - Redirector	Y	Y	Y	Y	--
Component Transaction Monitor	Y	Y	Y	Y	--
Developer's Kit for Java	Y	Y	Y	Y	Y
HTTP Server	Y	Y	Y	Y	--
Performance Tracer	Y	Y	Y	Y	Y
Reliable Messaging	Y	Y	Y	Y	--
Service Coordinator	--	--	Y	Y	--
Service Development Plug-in	--	--	--	Y	--
TPBroker	Y	Y	Y	Y	Y
Web Services - Security	Y	Y	Y	Y	--
XML Processor	Y	Y	Y	Y	Y
HiRDB Embedded Server Version 8	--	Y	--	Y	--

Legend:

Y: Included

--: Not included

2.2.2 Overview of functionality of the component software

This section describes the functional overview of the respective component software.

(1) Application Development Plug-in

Application Development Plug-in is the component software that provides the following functionality to be used in a development environment:

- **Development Environment Instant Setup**
  You can use this function to set up a debug environment of an application with a wizard program (GUI).
- **Eclipse setup functionality**
  This functionality enables you to set up the Eclipse environment required for developing J2EE applications by using Eclipse.

(2) Component Container

Component Container is the component software that provides the following functionality as the core of the application execution platform:

- Functionality for the execution environment (J2EE server) of J2EE applications
- Functionality for the execution environment (batch server) of batch applications
- Functionality for the execution and development environments of Web Services
- Functionality for managing the operations of Application Server

An overview of the respective functionality is as follows:

(a) Functionality for the execution environment (J2EE server) of J2EE applications

This functionality executes the J2EE server that is the framework for executing a server-side business process program (business logic) as a component. It includes functionalities such as the Web container and EJB container. This functionality complies with the specifications included in the Java Platform, Enterprise Edition (Java EE). For details on the corresponding specifications, see *4.6.2 Standard specifications supported by Application Server*.

This functionality also provides the basic functionality related to load balancing distribution, flow control, priority control, and blocking a service of the Enterprise Bean by the CTM functionality.

(b) Functionality for the execution environment (batch server) of batch applications

This functionality executes a batch application on the server. You can execute a Java application in which the contents of a batch process are implemented in Java. The cost of the JavaVM start-up is controlled by operating an application on the server machine. You can execute an efficient database access and full garbage collection control using the connection pool and statement pool.

(c) Functionality for the execution and development environment of Web Services

This functionality serves as an execution environment and development environment for Web Services. The functionality provides the following engines or APIs:

- SOAP Web Services compliant with the JAX-WS specifications, the JAX-WS engine required for executing the client, and APIs and commands required for developing Web Services.
- Servers of RESTful Web Services (Web resource) compliant with the JAX-RS specifications, the JAX-RS engine required for executing the client, and APIs required for developing Web Services.

  Reference note

  You can also use the existing SOAP application development support functionality.

(d) Functionality for managing operations of Application Server

This functionality manages the following operations of Application Server:

- Building and operating Application Servers in a batch
- Setting up applications and resources on a J2EE server
- Integrated user management by seamlessly linking the user management functionality of multiple J2EE applications

- Collecting the log output by all the functionality of Application Server

Furthermore, this functionality also provides the commands for managing the operations of Application Server by linking with other operation management program products such as JP1.

(3) Component Container - Client

Component Container - Client is the component software for building an execution environment for EJB client applications. It is a servlet of Component Container.

(4) Component Container - Redirector

Component Container-Redirector is the component software for linking a Web container to the Web server. By registering the redirector module provided by this component software on a Web server, you can process a particular request among the HTTP requests to the Web server; on a specified Web container. You can also process requests by distributing the requests to multiple Web containers.

(5) Component Transaction Monitor

Component Transaction Monitor is the component software that executes the load distribution and the flow control by scheduling requests from the Enterprise Bean client. You can divide the processes depending on the load status of the J2EE server by managing a queue of every application. You can control the number of requests to be processed at a time and also block a particular J2EE application when switching the business process programs. Thus, it is possible to continue the stable system operation which in turn results in improving the availability and reliability of the business system.

You can also schedule the execution of batch applications.

(6) Developer's Kit for Java

Developer's Kit for Java is the component software that complies with Java Platform, Standard Edition 6. The corresponding JDK version of Oracle is JDK 6. For details on the functionality, commands, and API that can be used with JDK6, see the *JDK 6 documentation* provided by Oracle.

(7) HTTP Server

HTTP Server is a Web server for mission critical domains and this server supports Secure Sockets Layer (SSL) based on *Apache HTTP Server*.

(8) Performance Tracer

Performance Tracer is the component software that outputs the trace information to analyse the bottlenecks in the performance of a process. This component software outputs the trace information for analyzing the performance at every defined point when processing a request in the system built on Application Server. You can investigate the system bottlenecks by analyzing this information. If an error occurs, the location of the error can also be identified.

(9) Reliable Messaging

Reliable Messaging is the component software that provides functionality for a highly reliable messaging base that executes sending and receiving of highly reliable messages asynchronously between business components, between the systems in a company, and between the systems within and outside the company. This software complies with the specifications of WS-Reliability.

(10) Service Coordinator

Service Coordinator is the component software that provides functionality for building and operating a system with SOA. This software includes the following functionality:

- Business process execution functionality (HCSC-Business Process)
- Data conversion functionality (HCSC-Data Transform)

- Functionality for managing the operations of the execution environment (HCSC-Manager)
- Messaging control and service link functionality (HCSC-Messaging)
- Reception and adapter

(11) Service Development Plug-in

Service Development Plug-in is a development environment for Service Coordinator. You can use this plug-in to define the reception, adapter, business processes, and the data conversion required for SOA-applied systems.

(12) TPBroker

TPBroker is the component software that is used for communication between the servlet or JSP and the Enterprise Bean and that is an RMI-IIOP communication base in Java EE and the distributed transaction base. Furthermore, this software provides an execution environment and a development environment for CORBA client applications in the Java EE environment. It also provides functionality for the execution environment of the RMI-IIOP communication platform, and the CORBA client applications to be used in the execution environment of EJB client applications.

(13) Web Services - Security

Web Services-Security is a component software that provides functionality compliant with the security standard *WS-Security* of SOAP Web Services. This component software attaches an XML signature to a SOAP message and encrypts a SOAP message. You can send or receive SOAP messages in a secured manner by using Web Services - Security. You can also decode encrypted SOAP messages by validating the signatures attached to the SOAP messages.

This software also provides functionality that supports the development of an application for executing encryption and decoding of data by generating and validating XML signatures or by XML encryption.

(14) XML Processor

XML Processor provides the functionality for reading, operating, and generating the XML documents that support JAXP/JAXB, which are industry-wide standards.

(15) HiRDB Embedded Server Version 8

HiRDB Embedded Server Version 8 is a database that can be used for testing and debugging when developing an application.

2.3 Operating environment of Application Server

This chapter describes the operating environment of Application Server.

(1) Prerequisite OS

The prerequisite OS for each target product is as follows:

(a) Prerequisite OS of Application Server

The following table describes the prerequisite OS of Application Server.

Table 2–3: Prerequisite OS of Application Server

Prerequisite OS	Version of prerequisite OS
Windows	Windows Server 2012 Windows Server 2008 R2 Windows Server 2008 x86 Windows Server 2008 x64
AIX	AIX V6.1 AIX V7.1
HP-UX	HP-UX 11i V2 (IPF) HP-UX 11i V3 (IPF)
Linux	Red Hat Enterprise Linux 5 Advanced Platform (AMD/Intel 64) Red Hat Enterprise Linux 5 (AMD/Intel 64) Red Hat Enterprise Linux Server 6 (32-bit x86) Red Hat Enterprise Linux Server 6 (64-bit x86_64)

(b) Prerequisite OS of Developer

The following table describes the prerequisite OS of Developer. Note that these products are not supported in UNIX.

Table 2–4: Prerequisite OS of Developer

Prerequisite OS	Version of prerequisite OS
Windows	Windows 8 Windows 7 Windows Vista Windows XP

(c) Prerequisite OS of Client

The following table describes the prerequisite OS of Client. Note that Client is not supported in UNIX.

Table 2–5: Prerequisite OS of Client

Prerequisite OS	Version of prerequisite OS
Windows	Windows 8 Windows 7 Windows Server 2012 Windows Server 2008 R2 Windows Server 2008 x86 Windows Server 2008 x64

Prerequisite OS	Version of prerequisite OS
Windows	Windows Vista Windows XP

(2) Web environment

This section describes the prerequisite Web environment. Note that the Web server is not required when using an in-process HTTP server.

(a) Web server

The following table describes the prerequisite Web server:

Table 2–6: Prerequisite Web server of Application Server

Prerequisite OS	Prerequisite products
Windows	Microsoft IIS HTTP Server#
UNIX	HTTP Server#

#
Component software of Application Server.

(b) Web browser

The following table describes the prerequisite Web browser.

Table 2–7: Prerequisite Web browser of Application Server

Web environment	Prerequisite product
Web browser	A browser that supports HTTP1.0 or HTTP1.1 (Internet Explorer 6.0 or later, Or Mozilla Firefox2 or later)

(3) Language

The prerequisite language is Java.

(4) Database

For details on the prerequisite database, see *6.1 Linking with the database*.

2.4 Operating environment of the BPM/ESB base

This chapter describes the operating environment of Service Platform realized by the BPM/ESB base.

2.4.1 Prerequisite software

This chapter describes the prerequisite software of Service Platform.

(1) Prerequisite OS

This section describes the prerequisite OS for each software product that configures Service Platform.

(a) Prerequisite OS of Service Architect

The following table describes the prerequisite OS of Service Architect:

Table 2–8: Prerequisite OS of Service Architect

Prerequisite OS	Version of prerequisite OS
Windows	Windows XP Windows Vista Windows 7 Windows 8

(b) Prerequisite OS of Service Platform

The following table describes the prerequisite OS of Service Platform:

Table 2–9: Prerequisite OS of Service Platform

Prerequisite OS	Version of prerequisite OS
Windows	Windows Server 2008 x86 Windows Server 2008 x64 Windows Server 2008 R2 Windows Server 2012 Standard Windows Server 2012 Datacenter
AIX	AIX V6.1 AIX V7.1
HP-UX	HP-UX 11i V2 (IPF) HP-UX 11i V3 (IPF)
Linux or Linux (x86/AMD64 & Intel EM64T)	Red Hat Enterprise Linux 5 Advanced Platform (AMD/Intel 64) Red Hat Enterprise Linux 5 (AMD/Intel 64) Red Hat Enterprise Linux Server 6 (32-bit x86) Red Hat Enterprise Linux Server 6 (64-bit x86_64)

(2) Prerequisite database

The following table describes the prerequisite database of Service Platform:

Table 2–10: Prerequisite database of Service Platform

Prerequisite database	Version
HiRDB	HiRDB/Single Server Version 8

Prerequisite database	Version
HiRDB	HiRDB/Parallel Server Version 8 HiRDB Server Version 9
Oracle	Oracle 11g

2.4.2 Related software

This chapter describes the software related to Service Platform.

(1) Windows Server Failover Cluster

You use *Windows Server Failover Cluster* for configuring an HA cluster by combining two HCSC servers in an execution environment. You can use this only in Windows.

For details on the cluster configuration of the HCSC server, see the *1.4.2 Redundant configuration of the HCSC server using cluster software* in the manual *BPM/ESB Service Platform Overview*.

(2) HA monitor

You use the *HA monitor* when configuring the HA cluster by combining two HCSC servers in an execution environment. You can use the HA monitor only in UNIX.

For details on the cluster configuration of HCSC server, see *1.4.2 Redundant configuration of the HCSC server using cluster software* in the manual *BPM/ESB Service Platform Overview*.

(3) TP1/Server Base Enterprise Option

You use *TP1/Server Base Enterprise Option* for communicating with different components by using a database between the execution environment and OpenTP1 of Service Platform.

TP1/Server Base Enterprise Option supports the DB queue protocol. You can link Service Platform and the DB queue reception with a service adapter.

(4) code conversion - Development Kit

You use *Code Conversion Development Kit* in the development environment of Service Platform in the following cases:

- If the user independently executes the external characters mapping
- When customizing the code conversion by using the character code conversion UOC

(5) code conversion - Server Runtime

You use *Code Conversion - Server Runtime* in the execution environment of Service Platform in the following cases:

- If the user independently executes the external characters mapping
- When customizing the code conversion by using the character code conversion UOC

This product is specific to the Windows platform.

(6) code conversion - Runtime

You use *Code Conversion - Runtime* in the execution environment of Service Platform in the following cases:

- If the user independently executes the external character mapping
- When customizing the code conversion by using the character code conversion UOC

This product is specific to the UNIX platform.

(7) JP1 related products

You use *JP1 related products* to efficiently perform operations such as monitoring and detection of an entire business system built with Service Platform.

For details on operations of a system linked with JP1, see Chapter *12.Operation of a system linked with JP1* in the *Application Server Operation, Monitoring, and Linkage Guide*.

3 Manual organization and reading sequence

This chapter explains the organization and the sequence for reading the manuals of Application Server and the BPM/ESB base.

3.1 Manual organization

This section describes the organization of the Application Server and BPM/ESB base manuals. Note that when using the BPM/ESB base, reference the manuals of Application Server if required in addition to the manuals of the BPM/ESB base.

The terms *Application Server* and are omitted from the manual names in the figure.

3.1.1 Application Server manual

The following figure shows the Application Server manual organization.

Figure 3–1: Application Server manual organization (Manuals to be referenced for building and operating the execution environment)

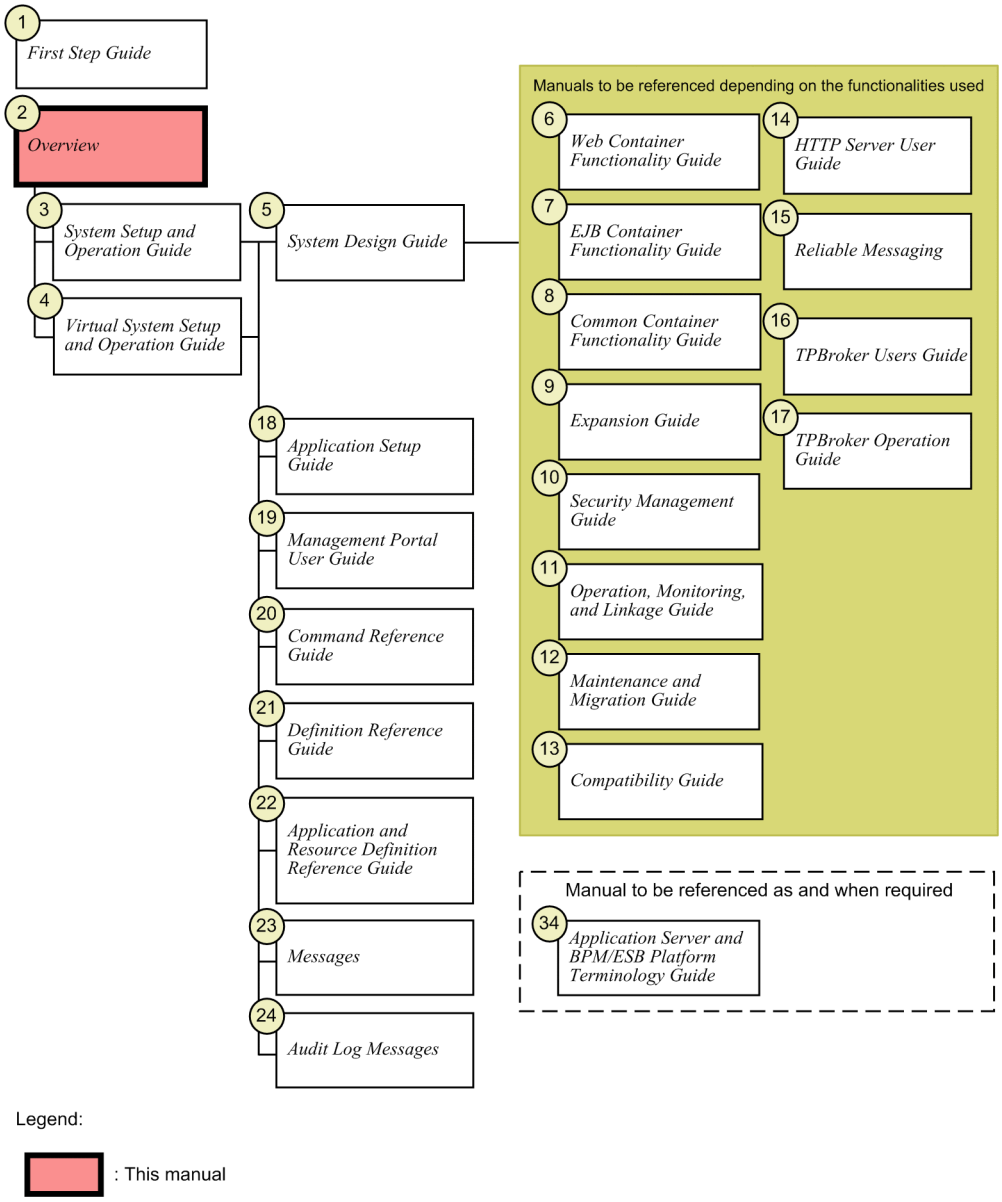
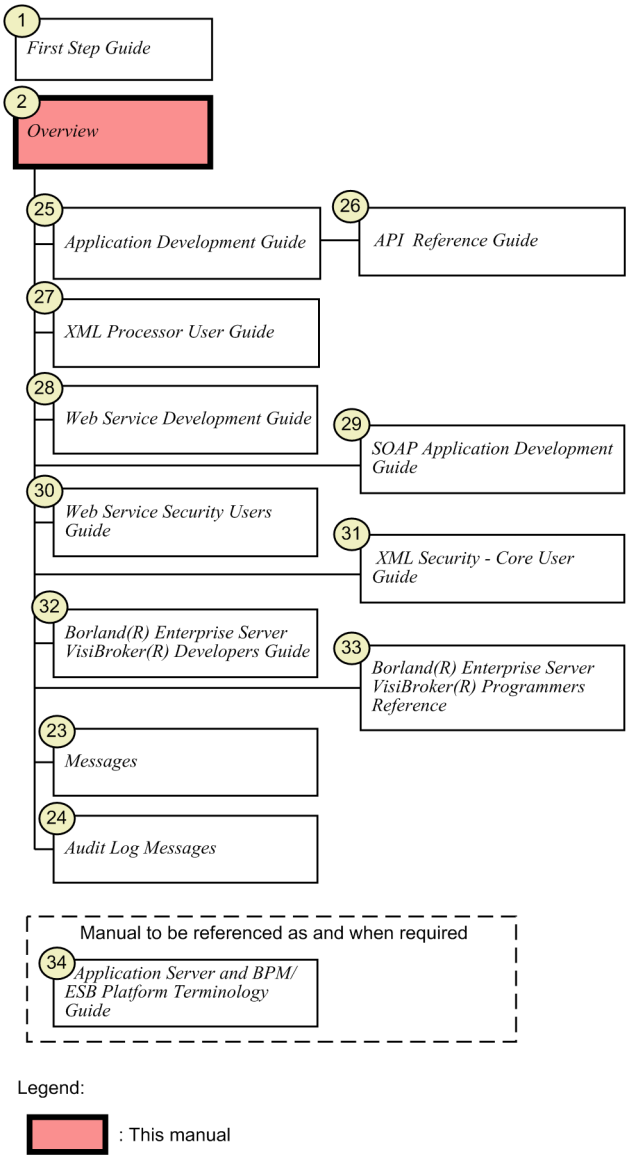


Figure 3–2: Application Server manual organization (Manuals to be referenced for developing an application)



The following table gives an overview of the contents explained in the respective manuals. Note that the item numbers in the table correspond to the item numbers in the figure.

Table 3–1: Overview of the Application Server manuals

No.	Manual name	Contents
1	*Application Server First Step Guide*	This manual describes the procedure to build the development environment and execution environment to operate a sample program.
2	*Application Server Overview*	This manual gives an overview of the products of Application Server.
3	*Application Server System Setup and Operation Guide*	This manual describes the methods to build and operate a system by installing Application Server. Reference this manual first when installing Application Server in the environment for the production operation.
4	*Application Server Virtual System Setup and Operation Guide*	This manual describes the procedure to design, build and operate when building Application Server on a virtual server.

No.	Manual name	Contents
5	*Application Server System Design Guide*	Reference this manual when building and operating a system with a configuration other than the system configuration explained in the *Application Server System Setup and Operation Guide*. This manual describes the guidelines to review the system configuration and the operation methods corresponding to the system objectives. This manual also describes the methods of tuning.
6	*Application Server Web Container Functionality Guide*	This manual describes details on the functions of Application Server including the execution method and the settings required in the execution environment of Application Server. Among the J2EE server functions, this manual also describes the functions of a Web container that is the execution base for Web applications. It also describes a function to be executed by linking with the Web server.
7	*Application Server EJB Container Functionality Guide*	This manual describes details on the functions of Application Server including the execution method and the settings required in the execution environment of Application Server. Among the J2EE server functions, this manual describes the function of an EJB container that is the execution base for an Enterprise Bean.
8	*Application Server Common Container Functionality Guide*	This manual describes details on the functions of Application Server including the execution method and the settings required in the execution environment of Application Server. Among the J2EE server functions, this manual describes the functions that can be used in both, the Web container and EJB container.
9	*Application Server Expansion Guide*	This manual describes details on the functions of Application Server including the execution method and the settings required in the execution environment of Application Server. This manual describes the extended functions of Application Server.
10	*Application Server Security Management Guide*	This manual describes details on the functions of Application Server including the execution method and the settings required in the execution environment of Application Server. This manual describes the functions to ensure the security of a system of Application Server. It also describes the system configuration and the operation method to ensure security.
11	*Application Server Operation, Monitoring, and Linkage Guide*	This manual describes details on the functions of Application Server including the execution method and the settings required in the execution environment of Application Server. This manual describes details on the functions used for system operations, the function to monitor the system, and the function to link with other products.
12	*Application Server Maintenance and Migration Guide*	This manual describes details on the functions of Application Server including the execution method and the settings required in the execution environment of Application Server. This manual describes the functions required for system maintenance in the case of trouble occurrence, and the system transition corresponding to the upgrading of a product version.
13	*Application Server Maintenance and Migration Guide*	This manual describes details on the functions of Application Server including the execution method and the settings required in the execution environment of Application Server. This manual describes the function for compatibility provided by previous versions of Application Server.
14	*Application Server HTTP Server User Guide*	This manual describes details on the setup and management of HTTP Server (Web server).
15	*Reliable Messaging*	This manual explains the method to link an application by an asynchronous communication of messages with Reliable Messaging.
16	*TPBroker Users Guide*	This manual describes the overview, functions, and operation method of TPBroker.

No.	Manual name	Contents
17	*TPBroker Operation Guide*	Among the functions of TPBroker, this manual describes the troubleshooting of the ORB function, extended functions required to operate the ORB functions, methods of linking TPBroker with other products, and the transition in the case of upgrading of version.
18	*Application Server Application Setup Guide*	This manual describes the operations of J2EE applications and resources with the server management command.
19	*Application Server Management Portal User Guide*	This manual describes details on the windows and operations of the management portal.
20	*Application Server Command Reference Guide*	This manual describes the commands to be used when building and operating a system, and developing an application.
21	*Application Server Definition Reference Guide*	This manual describes the method of defining a file used when building and operating a system, and developing an application. This manual describes the files used to define a J2EE server or Management Server.
22	*Application Server Application and Resource Definition Reference Guide*	This manual describes the method of defining a file used when building and operating a system, and developing an application. This manual describes the files used to set the attributes of an application and resources.
23	*Application Server Messages*	This manual describes the messages output when building and operating a system, and when developing an application.
24	*Application Server Audit Log Message*	This manual describes the messages that are used when auditing the system.
25	*Application Server Application Development Guide*	This manual describes the method of developing an application. It also explains the method to set up a development environment.
26	*Application Server API Reference Guide*	This manual describes the API and tags used for developing an application.
27	*Application Server XML Processor User Guide*	This manual describes the functions, creation method, and the method of using the XML purser and the XSLT transformer provided by XML Processor.
28	*Application Server Web Service Development Guide*	This manual describes the method to develop and execute SOAP Web Services according to the JAX-WS specifications and the method to execute RESTful Web Services according to the JAX-RS specifications.
29	*Application Server SOAP Application Development Guide*	This manual describes the method to develop and execute SOAP applications using the SOAP application development support function.
30	*Application Server Web Service Security Users Guide*	This manual describes the Web service security functions.
31	*XML Security - Core User Guide*	This manual describes the function to generate and validate the XML signature data and the function to encrypt and decode the XML.
32	*Borland(R) Enterprise Server VisiBroker(R) Developers Guide*	This manual describes how to use Borland Enterprise Server VisiBroker and its advanced functions.
33	*Borland(R) Enterprise Server VisiBroker(R) Programmers Reference*	This manual describes the class and interface information, program tools, and the command line option provided by Borland Enterprise Server VisiBroker.
34	*Application Server and BPM/ESB Platform Terminology Guide*	This manual describes the terminology used in the manuals for Application Server and BPM/ESB Platform.

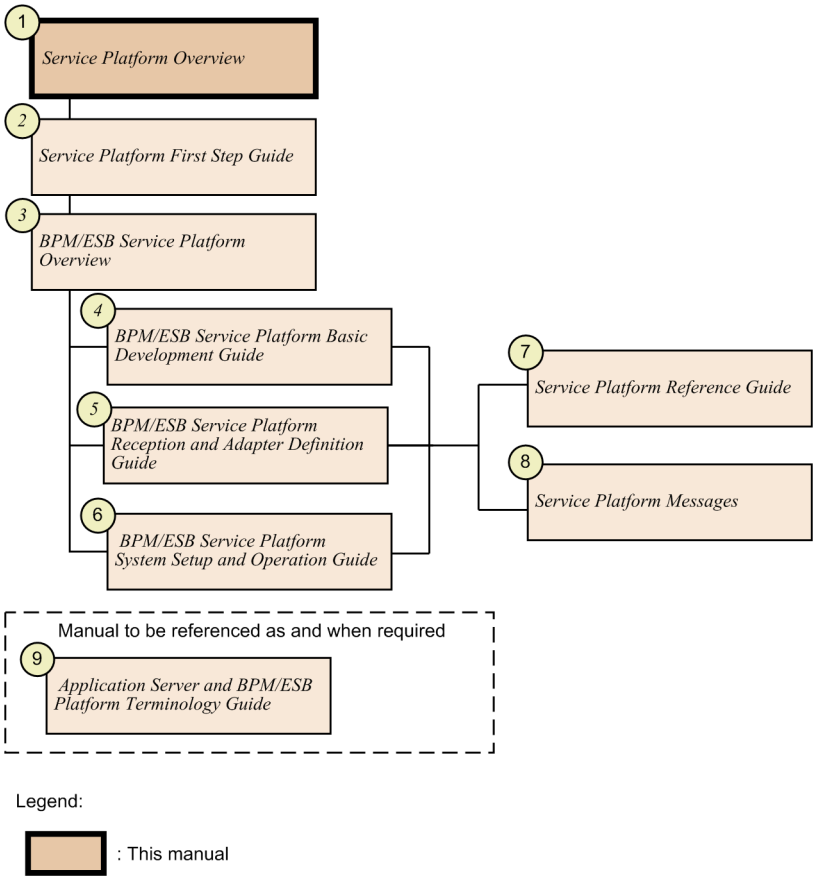
The manual organization has been changed in version 09-50. The main changes are as follows:

- *Appendix A Glossary* in the manual *Application Server Overview* is published separately in the *Application Server and BPM/ESB Platform Terminology Guide*.
- *Part 1 Overview* and *Part 2 Messages* that are output during system setup and operation or during application development in the manual *Application Server Messages* are published separately in the manual *Application Server Messages*. *Part 3 Messages* used when auditing the system is published separately in the manual *Application Server Audit Log Messages*.

3.1.2 BPM/ESB base manual organization

The following figure shows the BPM/ESB base manual organization.

Figure 3–3: BPM/ESB base manual organization



The following table gives an overview of the contents explained in the respective manuals. Note that the item numbers in the table correspond to the item numbers in Figure 3-3.

Table 3–2: Overview of manuals of the BPM/ESB base

No.	Manual name	Contents
1	*Application Server Overview*	This manual gives an overview of the functions to execute a system with SOA, product configuration, and the operation environment.
2	*Service Platform First Step Guide*	Reference this manual to understand the development method of the service integrated environment (Service Platform) supported by SOA. Reference this manual for executing the method of building an environment, the operation and checking method, and the development method with a sample program provided by Service Platform.
3	*BPM/ESB Service Platform Overview*	This manual describes the functionalities that can be used in the service integrated environment (Service Platform) supported by SOA. Reference this manual to learn about the functionality for smooth operations of the system and for improving the performance of the system.
4	*BPM/ESB Service Platform Basic Development Guide*	Reference this manual to develop a service integrated environment (Service Platform). Reference this manual to understand a series of development methods like how to build a development environment, an XML schema, a business process definition,

No.	Manual name	Contents
4	*BPM/ESB Service Platform Basic Development Guide*	a service component definition, a data conversion definition, and a service requester.
5	*BPM/ESB Service Platform Reception and Adapter Definition Guide*	Reference this manual to understand how to develop a reception and adapter, among the service components required for the actual operation of the service integrated environment (Service Platform) supported by SOA.
6	*BPM/ESB Service Platform System Setup and Operation Guide*	Reference this manual to actually build and operate the service integrated environment (Service Platform) supported by SOA. Reference this manual to understand the flow from development to the actual operation, how to build an operating environment and execution environment, the method of system operations, and troubleshooting.
7	*BPM/ESB Service Platform Reference*	This manual describes the windows, commands, and definition files used in the service integrated environment (Service Platform) supported by SOA. Reference this manual to understand the windows contents, commands syntax, and definition file syntax.
8	*BPM/ESB Service Platform Messages*	This manual describes the messages output in the service integrated environment (Service Platform) supported by SOA. Reference this manual to understand the cause of the message and the countermeasures.
9	*Application Server and BPM/ESB Platform Terminology Guide*	This manual describes the terminology used in the manuals of Application Server and BPM/ESB Platform.

(1) About the changes of the manual organization from version 08-70 to 09-00

The BPM/ESB base manual organization has been changed in version 09-00.

The following table describes the relationship of the previous manual organization with the new manual organization for the versions 08-70 and 09-00.

Table 3–3: Table showing the relationship of the previous manual organization with the new manual organization for the versions 08-70 and 09-00

No.	Old (version 08-70 manuals)#	New (version 09-00 manuals)
1	*Service Platform Overview* (3020-3-U51(E))	*Application Server Overview* (3020-3-Y01(E))
2	*Service Platform Development Guide* (3020-3-U52(E))	*BPM/ESB Service Platform Basic Development Guide* (3020-3-Y43(E))
		BPM/ESB Service Platform Reception and Adapter Definition Guide (3020-3-Y44(E))
3	*BPM/ESB Service Platform System Setup and Operation Guide* (3020-3-U53(E))	*BPM/ESB Service Platform System Setup and Operation Guide* (3020-3-Y45(E))
4	*Service Platform Functionality Guide* (3020-3-U54(E))	*BPM/ESB Service Platform Overview* (3020-3-Y42(E))
5	*Service Platform Sample Program Guide* (3020-3-U55(E))	*Service Platform First Step Guide* (3020-3-Y41(E))
6	*BPM/ESB Service Platform Reference* (3020-3-U56(E))	*BPM/ESB Service Platform Reference* (3020-3-Y46(E))

No.	Old (version 08-70 manuals)#	New (version 09-00 manuals)
7	*BPM/ESB Service Platform Messages* (3020-3-U57(E))	*BPM/ESB Service Platform Messages* (3020-3-Y47(E))
8	*Service Platform TP1 Adapter User Guide* (3020-3-U58(E))	--
9	*Service Platform TP1 File Adapter User Guide* (3020-3-P06(E))	--

Legend:

--: Removed

#

The serial data number is omitted. The term *Business Process Management / Enterprise service V8* before the term *Service Platform* is omitted from the full names of the old manuals.

The main changes are as follows:

- Separate edition of manual

  The *Service Platform Development Guide* was divided into two parts.
- Abolishment of manuals

  The contents of the following manuals were integrated with the contents of the other manuals of *BPM/ESB Service Platform* and the following manuals were abolished:
  - *Service Platform TP1 Adapter User Guide*
  - *Service Platform TP1 File Adapter User Guide*
- Manual integration

  The contents of the following manuals were integrated into one manual:
  - *Service Platform Overview*
  - *Application Server Version 8 Overview*
- Change in the manual names

  The names of the following manuals were changed to match the contents:
  - *Service Platform Functionality Guide*
  - *Service Platform Sample Program Guide*

Depending on the changes in the manual organization, the contents were moved between the manuals.

The following table describes the relationship between the previous and new manuals in which the major contents were moved from the manuals of version 08-70 to the manuals of the version 09-00.

- *Service Platform Development Guide*
- *Service Platform TP1 Adapter User Guide*
- *Service Platform TP1 File Adapter User Guide*

(a) Table describing the relationship between the previous manual and the new *Service Platform Development Guide*

Old contents configuration (*Service Platform Development Guide*)	New contents configuration (*BPM/ESB Service Platform Basic Development Guide*)
1st Edition Developing a system using the standard protocol	-- (Edition title is removed)
Chapter 1 Overview of developing a system based on SOA	Chapter 1 Overview of system development based on SOA
Chapter 2 Before developing a system	Chapter 2 Before developing a system

Old contents configuration (*Service Platform Development Guide*)	New contents configuration (*BPM/ESB Service Platform Basic Development Guide*)
Chapter 3 Project management and repository management	Chapter 3 Project management and repository management
Chapter 4 Creating a message format	Chapter 4 Creating a message format
Chapter 5 Defining an adapter	-- (Moved to the *BPM/ESB Service Platform Reception and Adapter Definition Guide*)
Chapter 6 Defining a business process	Chapter 5 Defining a business process
Chapter 7 Defining data conversion	Chapter 6 Defining data conversion
Chapter 8 Defining a user-defined reception	-- (Moved to the *BPM/ESB Service Platform Reception and Adapter Definition Guide*)
Chapter 9 Packaging and deployment definition of HCSC components	Chapter 7 Packaging and deployment definition of HCSC components
Chapter 10 Creating a service requestor	Chapter 8 Creating a service requester
Chapter 11 Debugging a business process	Chapter 9 Debugging a business process
2nd Edition Development and operations of a system linked with OpenTP1	-- (Edition title is removed)
Chapter 12 Overview of TP1/RPC reception	-- (Moved to the *BPM/ESB Service Platform Overview*)
Chapter 13 Defining TP1/RPC reception	-- (Moved to the *BPM/ESB Service Platform Reception and Adapter Definition Guide*)
Chapter 14 Operation of TP1/RPC reception	-- (deleted)
Chapter 15 Tuning of TP1/RPC reception	-- (Moved to the *BPM/ESB Service Platform Overview*)
Chapter 16 Error countermeasures (TP1 linkage system)	-- (Moved to the *BPM/ESB Service Platform System Setup and Operation Guide*)
3rd Edition Development and operation of a system linked with FTP	-- (Edition title is removed)
Chapter 17 Overview of FTP linkage	-- (Moved to the *BPM/ESB Service Platform Overview*)
18.1 Overview of an FTP adapter	-- (Moved to the *BPM/ESB Service Platform Overview*)
18.2 Functionality of an FTP adapter	-- (Moved to the *BPM/ESB Service Platform Overview*)
18.3 Setting up an FTP adapter	-- (Moved to the *BPM/ESB Service Platform Reception and Adapter Definition Guide*)
18.4 Message format of an FTP adapter	-- (Moved to the *BPM/ESB Service Platform Reception and Adapter Definition Guide*)
19.1 Overview of a file operation adapter	-- (Moved to the *BPM/ESB Service Platform Overview*)
19.2 Functionality of a file conversion operation	-- (Moved to the *BPM/ESB Service Platform Overview*")
19.3 Setting up a file operation adapter	-- (Moved to the *BPM/ESB Service Platform Reception and Adapter Definition Guide*)
19.4 Message format of a file operation adapter	-- (Moved to the *BPM/ESB Service Platform Reception and Adapter Definition Guide*)
20.1 Overview of an FTP reception	-- (Moved to the *BPM/ESB Service Platform Overview*)
20.2 Functionality of an FTP reception	-- (Moved to the *BPM/ESB Service Platform Overview*)
20.3 What is an FTP inbound adapter	-- (Moved to the *BPM/ESB Service Platform Overview*)

Old contents configuration (*Service Platform Development Guide*)	New contents configuration (*BPM/ESB Service Platform Basic Development Guide*)
20.4 Setting up an FTP reception	-- (Moved to the *BPM/ESB Service Platform Reception and Adapter Definition Guide*)
20.5 Message format of an FTP reception	-- (Moved to the *BPM/ESB Service Platform Reception and Adapter Definition Guide*)
Chapter 21 Building and operating a system linked with FTP	-- (Moved to the *BPM/ESB Service Platform System Setup and Operation Guide*)
Chapter 22 Error countermeasures (FTP linkage)	-- (Moved to the *BPM/ESB Service Platform System Setup and Operation Guide*)
Chapter 23 Migration from the old version (FTP adapter)	Appendix A.3 Procedure of migration using an FTP adapter in the old version
Appendix	Appendix
Appendix A Migration from the old version	Appendix A Migration from the old version
Appendix B Migration from the evaluated version	Appendix B Migration from the evaluated version
Appendix C Developing a system using an upper design tool	Appendix C Developing a system using an upper design tool
Appendix D Example of developing a system using an upper design tool	Appendix D Example of developing a system using upper design tool
Appendix E BPEL support scope used for linking with an upper design tool	Appendix E BPEL support scope used for linking an upper design tool
--	Appendix F Inheriting HTTP header and Cookie information using service adapter
Appendix F Custom adapter development framework	-- (Moved to the *BPM/ESB Service Platform Reception and Adapter Definition Guide*)
Appendix G Custom reception	-- (Moved to the *BPM/ESB Service Platform Reception and Adapter Definition Guide*)
Appendix H Emulation of service requestor	Appendix G Emulation of service adapter
Appendix I Character code conversion using the character code conversion UOC	Appendix H Character code conversion using the character code conversion UOC
Appendix J Executing command using the external tool execution function	-- (Deleted)
Appendix K Example of converting a data format obtained by a database adapter	Appendix I Example of converting a data format obtained by a database adapter
--	Appendix J Automatic mapping of data obtained by a database adapter
Appendix L Defining a database adapter with the database adapter definition support function	-- (Moved to the *BPM/ESB Service Platform Reception and Adapter Definition Guide*)
Appendix M Customizing WSDL with an external binding file	Appendix K Customizing WSDL with an external binding file
Appendix N Reference information of this manual	-- (Integrated with the manual *Application Server Overview*)
Appendix O Glossary	Appendix L Glossary

Legend:

--: The contents have been deleted.

(b) New location of contents previously described in the *Service Platform TP1 Adapter User Guide*

Moved from (*Service Platform TP1 Adapter User Guide*)	Moved to
Chapter 1 Overview	*BPM/ESB Service Platform Overview*
2.1 RPC Communication	*BPM/ESB Service Platform Overview*
2.2 Monitoring time	*BPM/ESB Service Platform Overview*
2.3 Communicating the error information	*BPM/ESB Service Platform System Setup and Operation Guide*
2.4 Obtaining log	*BPM/ESB Service Platform System Setup and Operation Guide*
Chapter 3 Setup	*BPM/ESB Service Platform Reception and Adapter Definition Guide*
Chapter 4 Operations	-- (Deleted)
Chapter 5 Messages	*BPM/ESB Service Platform Messages*
Chapter 6 Troubleshooting	*BPM/ESB Service Platform System Setup and Operation Guide*
Appendix	-- (Deleted)
Appendix A Table of handling factor elements of the TP1/Client/J exception	*BPM/ESB Service Platform System Setup and Operation Guide*
Appendix B Migration from the old version	*BPM/ESB Service Platform Basic Development Guide*
Appendix C Reference information of this manual	-- (Integrated with the manual *Application Server Overview*)
Appendix D Glossary	*Application Server Overview*

Legend:
--: The contents have been deleted.

(c) New location of contents previously described in the *Service Platform TP1 File Adapter User Guide*

Moved from (*Service Platform TP1 File Adapter User Guide*)	Moved to
Chapter 1 Overview	*BPM/ESB Service Platform Overview*
2.1 File format	*BPM/ESB Service Platform Overview*
2.2 Reading and writing mode	*BPM/ESB Service Platform Overview*
2.3 Exclusion control	*BPM/ESB Service Platform Overview*
2.4 Data conversion	-- (Deleted)
2.5 Obtaining log and trace	*BPM/ESB Service Platform System Setup and Operation Guide*
2.6 Error process	*BPM/ESB Service Platform System Setup and Operation Guide*
Chapter 3 Setup	*BPM/ESB Service Platform Reception and Adapter Definition Guide*
Chapter 4 Operations	-- (Deleted)
Chapter 5 Setting example	*BPM/ESB Service Platform Reception and Adapter Definition Guide*
Chapter 6 Messages	*BPM/ESB Service Platform Messages*
Chapter 7 Troubleshooting	*BPM/ESB Service Platform System Setup and Operation Guide*

Moved from (*Service Platform TP1 File Adapter User Guide*)	Moved to
Appendix A Messages example	*BPM/ESB Service Platform Reception and Adapter Definition Guide*
Appendix B Glossary	*Application Server Overview*

Legend:
--: The contents have been deleted.

(2) About the changes in the manual organization from version 09-00 to 09-50

The manual organization has been changed in version 09-50. The main changes are as follows:

- *Appendix A Glossary* in the manual *Application Server Overview* is published separately in the *Application Server and BPM/ESB Platform Terminology Guide*.

For other changes, see the change history of each manual.

3.2 Sequence for reading the Application Server manuals

This section describes the sequence for reading the Application Server manuals.

This section describes the sequence for reading according to the following objectives:

- To validate and evaluate the operations of Application Server
- To set up and operate a system by installing Application Server in the production environment
- To develop an application operating on Application Server
- To take countermeasures for trouble that occurs
- To check the functions of Application Server

Reference note

This section describes the sequence for reading the main manuals that are referenced. All the manuals of Application Server are not covered. If required, also see other manuals that are not mentioned here based on the reference instructions in the manuals introduced here.

3.2.1 Validating and evaluating operations of Application Server

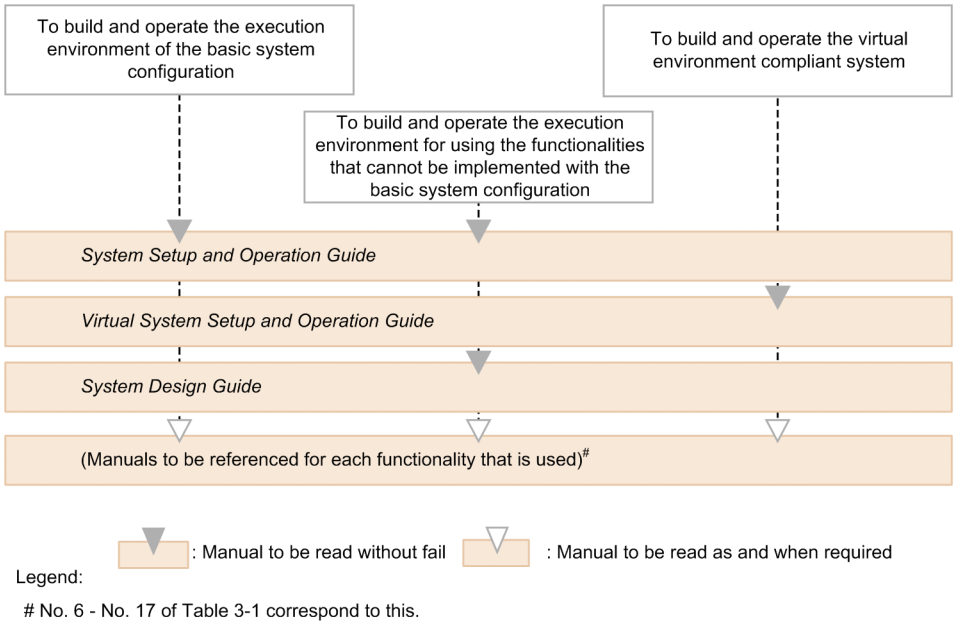
Perform the operations described in the *Application Server First Step Guide* to quickly set up the development environment and execution environment so that you can validate and evaluate the operations of Application Server.

Note that the contents explained in this manual are the procedures to operate a sample program in the test environment and not the procedures to build an actual production environment. To build an actual production environment, see the *Application Server System Setup and Operation Guide*.

3.2.2 Building and operating a system by installing Application Server in the production environment

The following figure shows the manuals to be referenced for building and operating a system by installing Application Server in the production environment.

Figure 3–4: Sequence for reading to build and operate a system by installing Application Server in the production environment



When installing Application Server in the production environment, the manuals to be referenced differ depending on whether you are building a system that is supported for a virtual environment. For building a system for a physical environment and not a virtual environment, first see the *Application Server System Setup and Operation Guide*. This manual describes the procedure for building and operating a system corresponding to the general system configuration when installing Application Server.

However, a part of the system configuration cannot be built and operated based on only the contents described in the *Application Server System Setup and Operation Guide*. In such cases, see the contents of the *Application Server System Setup and Operation Guide* and if required, see the *Application Server System Design Guide*.

For building a system corresponding to the virtual environment, see the *Application Server Virtual System Setup and Operation Guide*.

Note that individual settings are required depending on the functions used for building and operating a system with any method. For details on individual settings, see the manual that explains about the functions used.

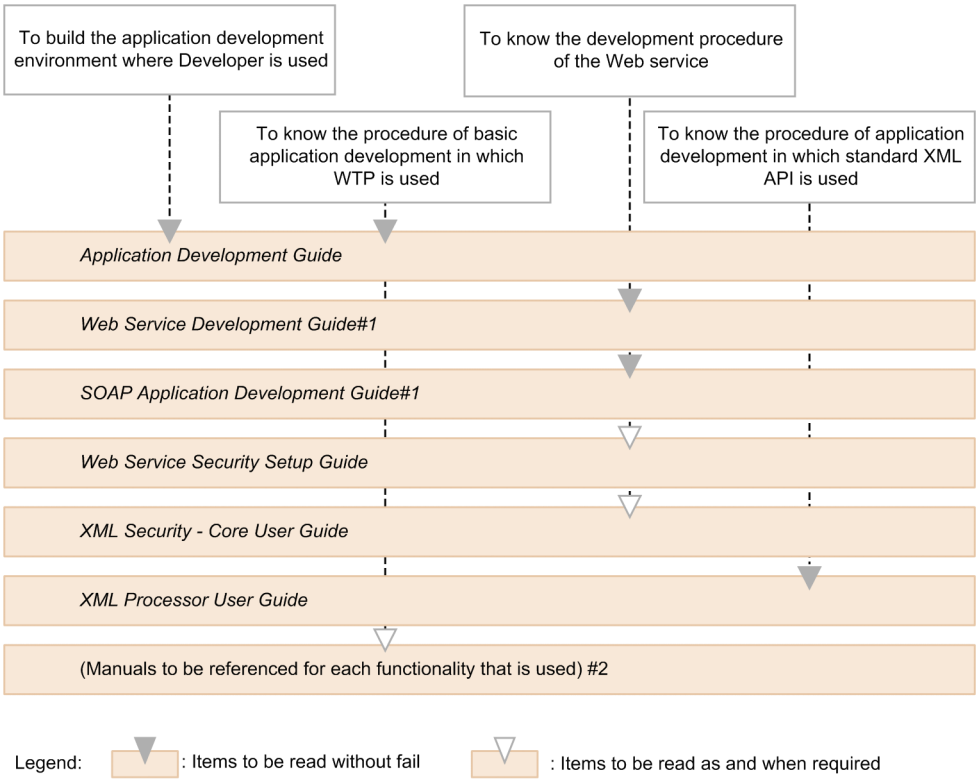
For details on the objectives of a system and the functions used and the corresponding manuals, see section *5. Objective-wise Introduction to Functions of Application Server*. This section describes the functions corresponding to the following objectives and the method of reading the manuals when these functions are used:

- Improving the system performance
- Improving the system availability
- Improving the system reliability
- Operating the system efficiently

3.2.3 Developing an application operating on Application Server

The following figure shows the manuals to be referenced to develop an application operating on Application Server.

Figure 3–5: Sequence for reading to develop an application on Application Server



#1 Select and read either of these manuals depending on the compliant specifications
#2 No. 6 – No. 17 of Table 3-1 are applicable.

Developer provides a function to build an environment for developing an application with WTP. For the procedure of building an environment to develop an application with WTP, see the *Application Server Application Development Guide*. See this manual also for the procedure of developing with WTP.

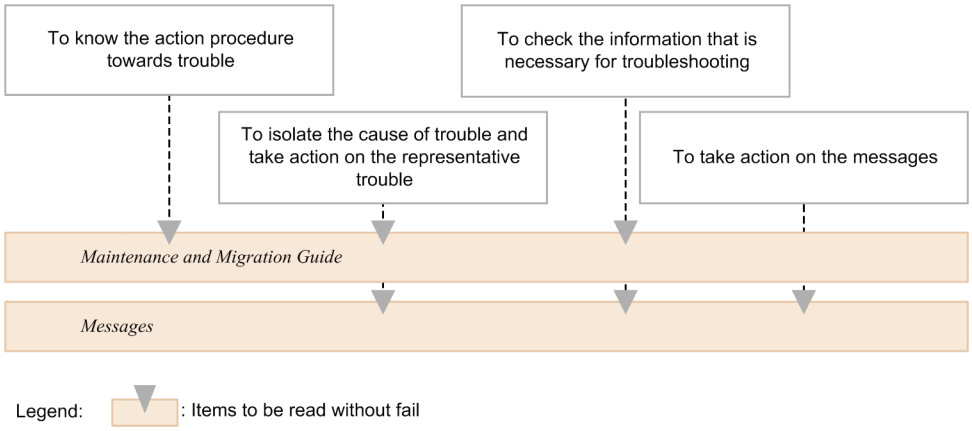
For the procedure of developing an application as a Web service, see the *Application Server Web Service Development Guide* or the *Application Server SOAP Application Development Guide*. For using the functions related to security during the Web service development, see the *Application Server Web Service Security Users Guide* and the *XML Security -Core User Guide* as and when required.

For the development procedure of an application using the standard XML API, see the *Application Server XML Processor User Guide*.

3.2.4 Taking actions on trouble that occurs

The following figure shows the manuals to be referenced when trouble occurs in a system operated with Application Server.

Figure 3–6: Sequence for reading when trouble occurs in Application Server



If trouble occurs in the system, see the *Application Server Maintenance and Migration Guide*. This section describes the procedure for measures against trouble, the method of taking measures against typical trouble, and the information required for troubleshooting.

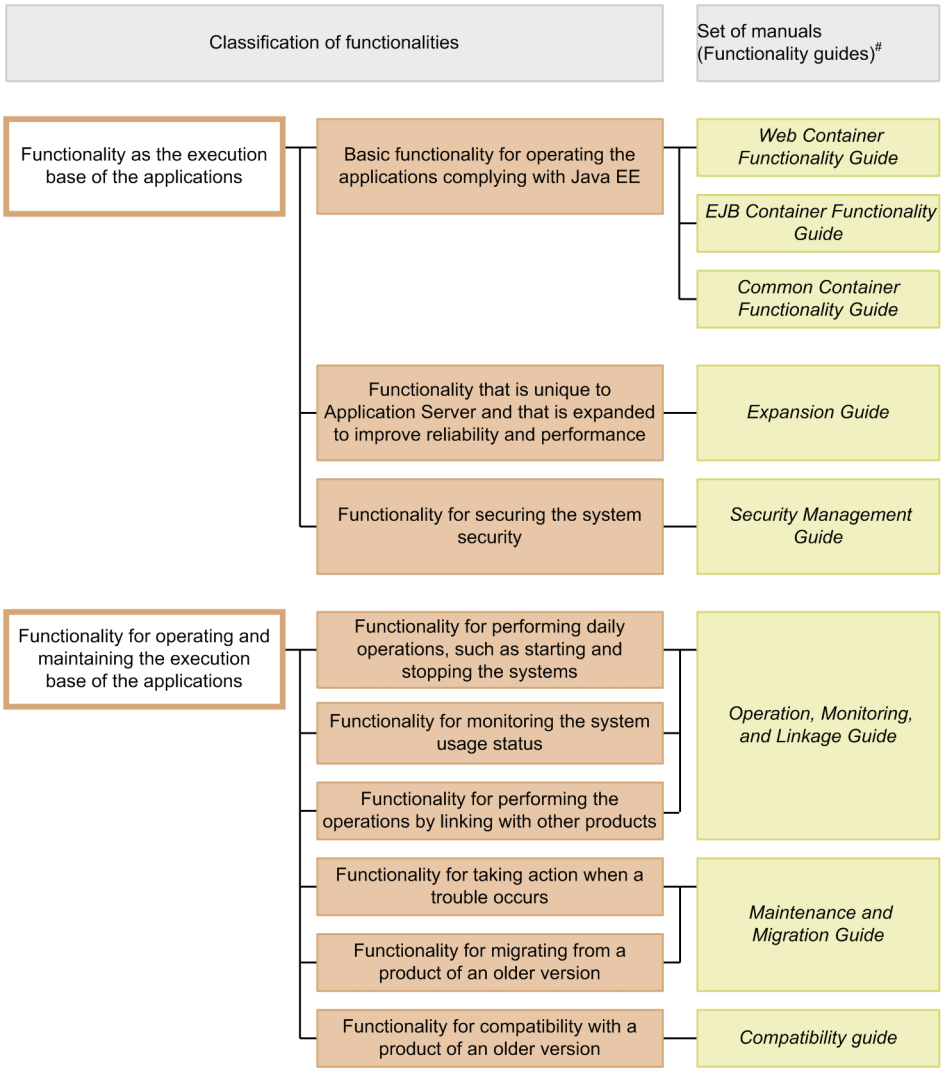
For directly handling the output messages, see the manual *Application Server Messages* based on the prefix, ID, and the text of the output message.

3.2.5 Checking the functions of Application Server

Application Server provides exclusively extended functions in addition to the functions corresponding to the JavaEE standard specifications.

As for the manuals of Application Server, the functions of Application Server are classified as shown in the following table and explained in eight manuals. For details on the functions used, reference these manuals as and when required. There is no sequence for reading these manuals.

Figure 3–7: Classification of the functions of Application Server



#: The term Functionality Guide has been omitted from the manual names.

For details on the functions included in each type, see *Chapter 1* in each manual shown in Figure 3-7.

In addition to the functions shown here, the following functions are explained in separate manuals.

Table 3–4: Functions described in the manuals other than those shown in Figure3-7

No.	Function	Corresponding manual
1	Functions of Web server (HTTP Server) provided by Application Server	*Application Server HTTP Server User Guide*
2	Functions for developing a program to operate an XML document using the standard XML API	*Application Server XML Processor User Guide*
3	Functions for developing and executing SOAP Web Services according to the JAX-WS specifications and for executing RESTful Web Services according to the JAX-RS specifications.	*Application Server Web Service Development Guide*
4	Functions for developing and executing SOAP applications	*Application Server SOAP Application Development Guide*
5	Functions for ensuring the security of SOAP Web Services	*Application Server Web Service Security Users Guide* *XML Security - Core User Guide*

No.	Function	Corresponding manual
6	Functions for executing an asynchronous communication of messages using Reliable Messaging in the resource adapter provided by Application Server	*Reliable Messaging*
7	Functions of TPBroker and VisiBroker	*TPBroker Users Guide* *TPBroker Operation Guide* *Borland(R) Enterprise Server VisiBroker(R) Developers Guide* *Borland(R) Enterprise Server VisiBroker(R) Programmers Reference*

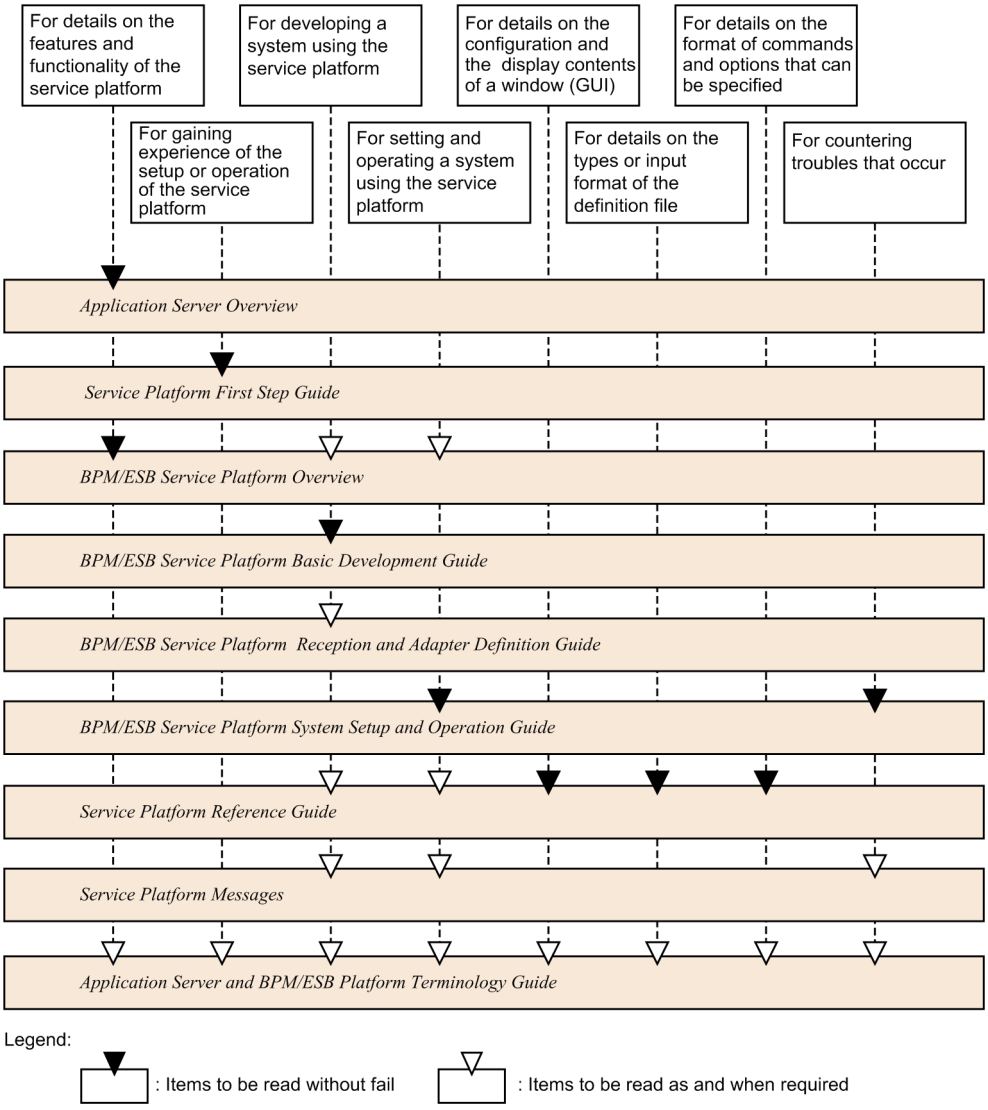
(1) Functions that can be used with Application Server

You can use the following Java EE or J2EE functions with Application Server:

- Servlet 2.2, Servlet 2.3, Servlet 2.4, Servlet 2.5, Servlet 3.0
- JSP 1.2, JSP 2.0, JSP 2.1
  For details, see the manual *Application Server Web Container Functionality Guide*.
- EJB 2.0 (Message-driven Bean, local interfaces, CMP 1.1, CMP 2.0), EJB 2.1, EJB 3.0, EJB 3.1
  For details, see the manual *Application Server EJB Container Functionality Guide*.
- Business interfaces, annotations, DIs
- JDBC 2.0 Core, JDBC 2.0 option package, JDBC 3.0, JDBC 4.0
- JMS 1.0.2, JMS 1.1
- Connector 1.0 (JCA 1.0), Connector 1.5 (JCA 1.5)
- JTA 1.0.1 (local, global)
- JPA 1.0
- JavaMail 1.2, JavaMail 1.3, JavaMail 1.4
  For details, see the *Application Server Common Container Functionality Guide*.
- Timer and Work Manager for Application Servers
  For details, see the *Application Server Expansion Guide*.

3.3 Sequence for reading the BPM/ESB base manuals

Read the BPM/ESB base manuals shown in the manual organization figure in the following order.



This section describes the sequence for reading the main BPM/ESB base manuals.

The sequence for reading the manuals according to the following objectives is explained here:

- Validating and evaluating the operations of Service Platform
- Checking the functions of Service Platform
- Developing a service integration environment (Service Platform) supported by SOA
- Building and operating a service integration environment (Service Platform) supported by SOA
- Taking action against trouble that occurs in a service integration environment (Service Platform) supported by SOA

Reference note

This section describes the sequence for reading the main manuals that are referenced. Reference the manuals that are not mentioned here based on the reference instructions in the manuals introduced in this section.

3.3.1 Validating and evaluating operations of Service Platform

On an actual machine, follow the procedures in the *Service Platform First Step Guide* to perform tasks so that you can validate and evaluate the operations of Service Platform and experience the operations from building an environment up to executing a sample program.

Note that the content of this manual describes the procedure to operate a sample program in the test environment built within the development environment. It does not describe the procedure to build an execution environment used as the actual operating environment. For building an operation environment, see the *BPM/ESB Service Platform System Setup and Operation Guide*.

3.3.2 Checking the functions of Service Platform

The BPM/ESB base provides the functions to execute an SOA-applied system on Service Platform.

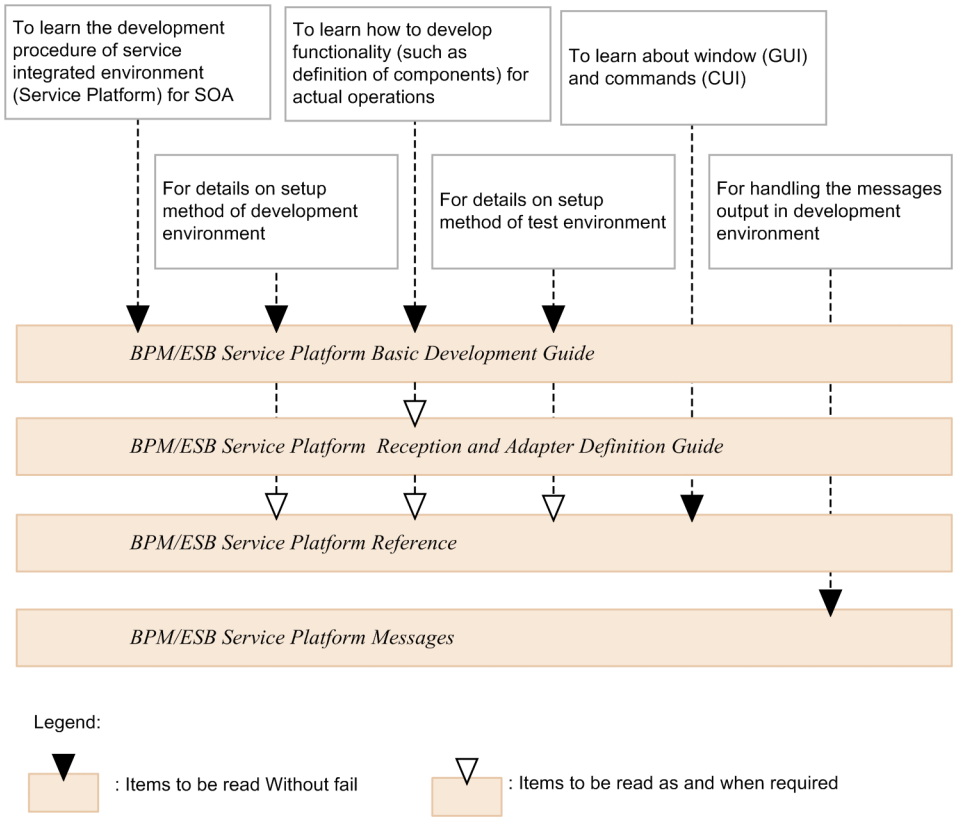
Service Platform configures a system by mutually connecting the development, execution , and operation environment.

For checking the functions provided by the respective environments, see the manual *BPM/ESB Service Platform Overview*. You can reference this manual for the functions used, as and when required.

3.3.3 Developing a service integration environment (Service Platform) supported by

The following figure shows the manuals to be referenced for developing a service integration environment (Service Platform) supported by SOA.

Figure 3–8: Sequence for reading for developing a service integration environment (Service Platform) supported by SOA



For the method of developing an XML schema, business process definition, service component definition, data conversion definition, and service requestor, see the *BPM/ESB Service Platform Basic Development Guide*. See this

manual also for building a development environment, and installation as well as a simple setup for the execution of the test environment.

For the reception and adapter definition from the service components, see the *BPM/ESB Service Platform Reception and Adapter Definition Guide*.

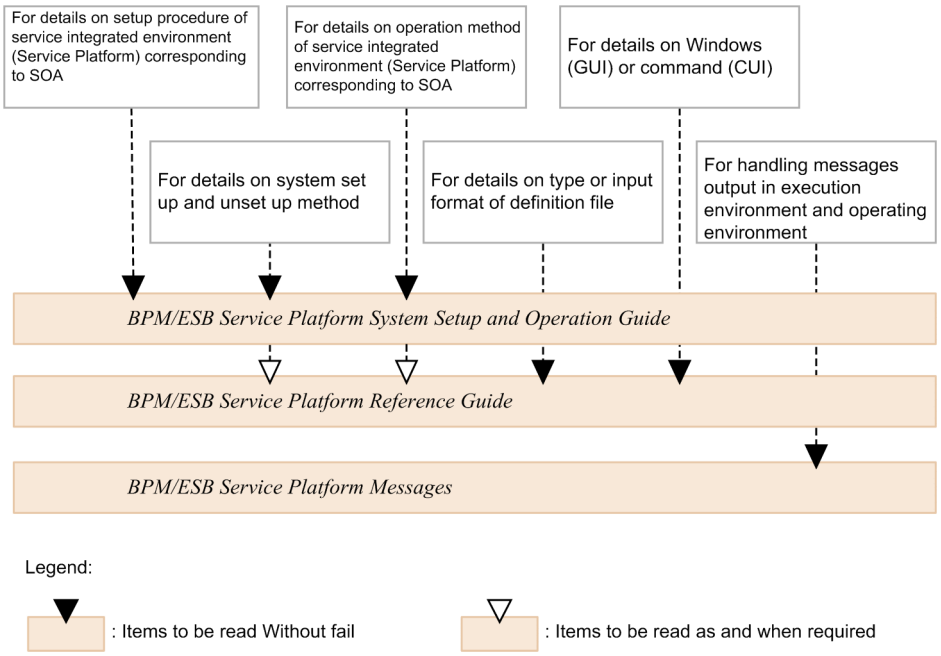
For details on the windows and commands used in the development environment and the definition file required for development, see the manual *BPM/ESB Service Platform Reference*.

For details on the messages output in the development environment, see *BPM/ESB Service Platform Messages*.

3.3.4 Building and operating a service integration environment (Service Platform) supported by SOA

The following figure shows the manuals to be referenced for building and operating a service integration environment (Service Platform) supported by SOA.

Figure 3–9: Sequence for reading for building and operating a service integration environment (Service Platform) supported by SOA



For building and operating a service integration environment (Service Platform) supported by SOA by installing the BPM/ESB base, see the *BPM/ESB Service Platform System Setup and Operation Guide*.

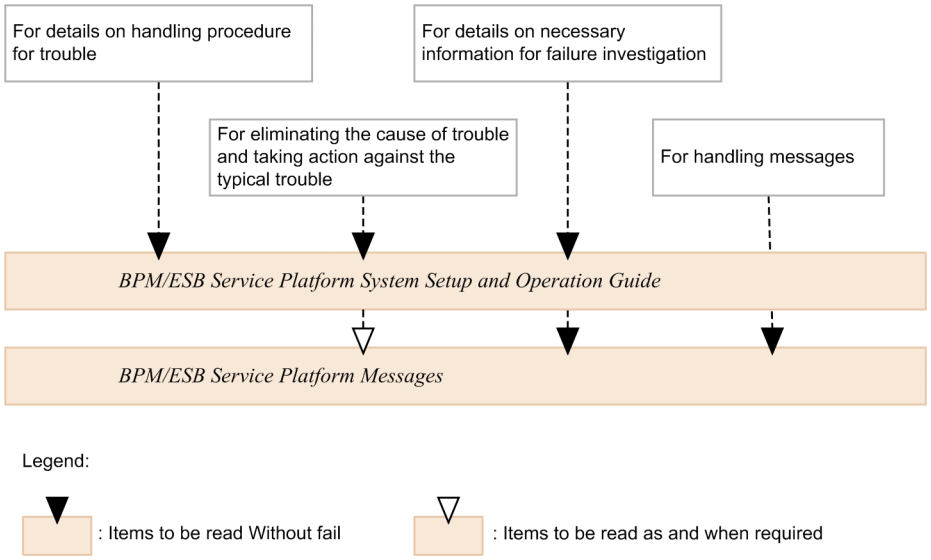
However, it is not possible to build a part of the system based only on the contents described in the *BPM/ESB Service Platform System Setup and Operation Guide.* See the contents of the *Application Server System Setup and Operation Guide* as and when required.

For operating a system after building, see the *BPM/ESB Service Platform System Setup and Operation Guide*.

3.3.5 Taking measures against the troubles that occur in a service integrated environment (Service Platform) supported by SOA

The following figure shows the manuals to be referenced when trouble occurs in a service integration environment (Service Platform) supported by SOA.

The sequence for reading when trouble occurs in the Service Platform



If trouble occurs in the system, first see the *BPM/ESB Service Platform System Setup and Operation Guide*.This manual describes the measures to be taken if trouble occurs, countermeasures for typical trouble, and the information required for troubleshooting.

Also, to directly troubleshoot in response to output messages, based on items such as the prefix, ID, and message text of an output message, reference the description in the manual *BPM/ESB Service Platform Messages*.

3.4 Relationship of products and manuals

This section describes the relationship of the products with each Application Server and BPM/ESB base manual.

Table 3–5: Relationship of products and manuals

No.	Manual name	Product name				
		Application Server	Developer	Service Platform	Service Architect	Client
1	*Application Server First Step Guide*	Y	Y	Y	Y	Y
2	*Application Server Overview*	Y	Y	Y	Y	Y
3	*Application Server System Setup and Operation Guide*	Y	Y	Y	Y	Y
4	*Application Server Virtual System Setup and Operation Guide*	Y	--	Y	--	--
5	*Application Server System Design Guide*	Y	Y	Y	Y	Y
6	*Application Server Web Container Functionality Guide*	Y	Y	Y	Y	Y
7	*Application Server EJB Container Functionality Guide*	Y	Y	Y	Y	Y
8	*Application Server Common Container Functionality Guide*	Y	Y	Y	Y	Y
9	*Application Server Expansion Guide*	Y	Y	Y	Y	Y
10	*Application Server Security Management Guide*	Y	Y	Y	Y	Y
11	*Application Server Operation, Monitoring, and Linkage Guide*	Y	Y	Y	Y	Y
12	*Application Server Maintenance and Migration Guide*	Y	Y	Y	Y	Y
13	*Application Server Maintenance and Migration Guide*	Y	Y	Y	Y	Y
14	*Application Server Application Setup Guide*	Y	Y	Y	Y	Y
15	*Application Server Management Portal User Guide*	Y	Y	Y	Y	Y
16	*Application Server Command Reference Guide*	Y	Y	Y	Y	Y
17	*Application Server Definition Reference Guide*	Y	Y	Y	Y	Y
18	*Application Server Application and Resource Definition Reference Guide*	Y	Y	Y	Y	Y
19	*Application Server HTTP Server User Guide*	Y	Y	Y	Y	--
20	*Reliable Messaging*	Y	Y	Y	Y	--
21	*Application Server Application Development Guide*	--	Y	--	Y	--
22	*Application Server API Reference Guide*	Y	Y	Y	Y	Y
23	*Application Server XML Processor User Guide*	--	Y	Y	Y	--
24	*Application Server Web Service Development Guide*	Y	Y	Y	Y	Y

No.	Manual name	Product name				
		Application Server	Developer	Service Platform	Service Architect	Client
25	*Application Server Web Service Security Users Guide*	Y	Y	Y	Y	--
26	*Application Server SOAP Application Development Guide*	Y	Y	Y	Y	Y
27	*Application Server XML Security - Core User Guide*	Y	Y	Y	Y	--
28	*Application Server Messages*	Y	Y	Y	Y	Y
29	*Application Server Audit log Messages*	Y	Y	Y	Y	Y
30	*TPBroker Users Guide*	Y	Y	Y	Y	--
31	*TPBroker Operation Guide*	Y	Y	Y	Y	Y
32	*Borland(R) Enterprise Server VisiBroker(R) Developers Guide*	Y	Y	Y	Y	--
33	*Borland(R) Enterprise Server VisiBroker(R) Programmers Reference*	Y	Y	Y	Y	Y
34	*Service Platform First Step Guide*	--	--	Y	Y	--
35	*BPM/ESB Service Platform Overview*	--	--	Y	Y	--
36	*BPM/ESB Service Platform Basic Development Guide*	--	--	Y	Y	--
37	*BPM/ESB Service Platform Reception and Adapter Definition Guide*	--	--	Y	Y	--
38	*BPM/ESB Service Platform System Setup and Operation Guide*	--	--	Y	Y	--
39	*BPM/ESB Service Platform Reference*	--	--	Y	Y	--
40	*BPM/ESB Service Platform Messages*	--	--	Y	Y	--
41	*Application Server and BPM/ESB Platform Terminology Guide*	Y	Y	Y	Y	Y

Legend:

Y: Related

--: Not related.

Note:

When reading manuals number 1 through 33, depending on the product in use, you might have to substitute the name of the product in use for a term in the manual. Substitute the name of the product in use as described in the following table. Note that for Developer or Service Architect, you need to substitute the name only when you are using the product in a test environment.

Product name used	Term used in the manual
Developer	Application Server
Service Architect	
Service Platform	

4 Overview of Application Server

This chapter describes an overview of Application Server and the characteristics in each environment.

Application Server is the core of a business system and a base for executing an application. You can build and operate an execution environment that complies with the standard Java EE.

You can also build an environment for efficiently developing an application to be executed in the execution environment.

4.1 Application execution environment and application development environment

This section describes the characteristics of Application Server.

Application Server is a base for executing a service (*Service deployment*). Application Server builds an application execution environment that provides services to a user, while securing performance and reliability. Application Server can also build an environment for developing the applications to be used as services.

The characteristics of the execution environment and development environment of an application are as follows:

Application execution environment

The application execution environment is an environment that is a base for providing services to the user by executing an application.

The following are the two types of businesses that can be executed in the application execution environment:

- **Online business (Online process)**

  In this business type, the requests sent from the user through the internet or intranet can be processed at any time. The online business executes an application developed using the Java EE technology. This environment is called the *J2EE application execution environment* .

- **Batch business (Batch process)**

  In this business type, the routine jobs are processed in a batch at a fixed time. You can execute the batch jobs that were earlier executed on a main frame, by using Java, which is an open environment technology. The application that executes the batch jobs developed by Java is called a batch application. An environment that executes the batch application is called the *Batch application execution environment*.

Application development environment

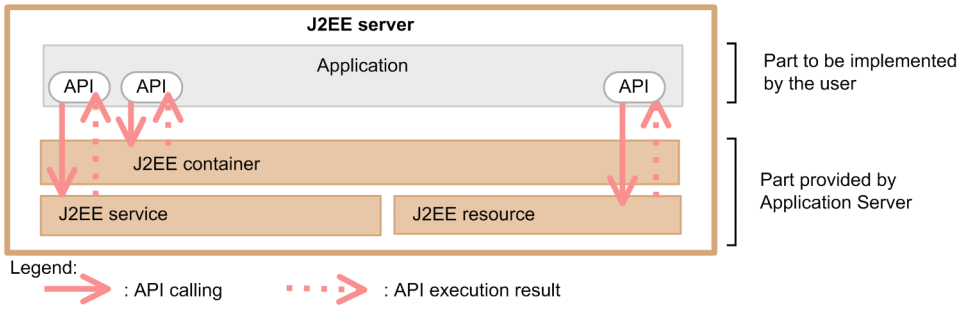
The application development environment is an environment for developing an application to be operated in the application execution environment. You can build and operate a development environment that provides an overall support to an application operating in the execution environment from the development of the application up to the debugging.

Application Server builds the execution environment of an application compliant with Java EE, which is a standard specification. A server process that has functionality to execute an application compliant to Java EE is called a *J2EE server*.

The J2EE server provides functionality required to execute a J2EE application developed by the user by following the specifications described in Java EE, such as the J2EE container, J2EE services, and J2EE resources. For example, the J2EE server uses J2EE services and J2EE resources to provide functions such as the transaction management and security management to execute common processes in multiple jobs. When developing an application, you can execute the common processes in multiple jobs without doing any complicated coding, by calling the API provided by the J2EE container, J2EE services, and J2EE resources in the application.

The following figure shows the relationship of an application and the J2EE server.

Figure 4–1: Relationship of an application and the J2EE server



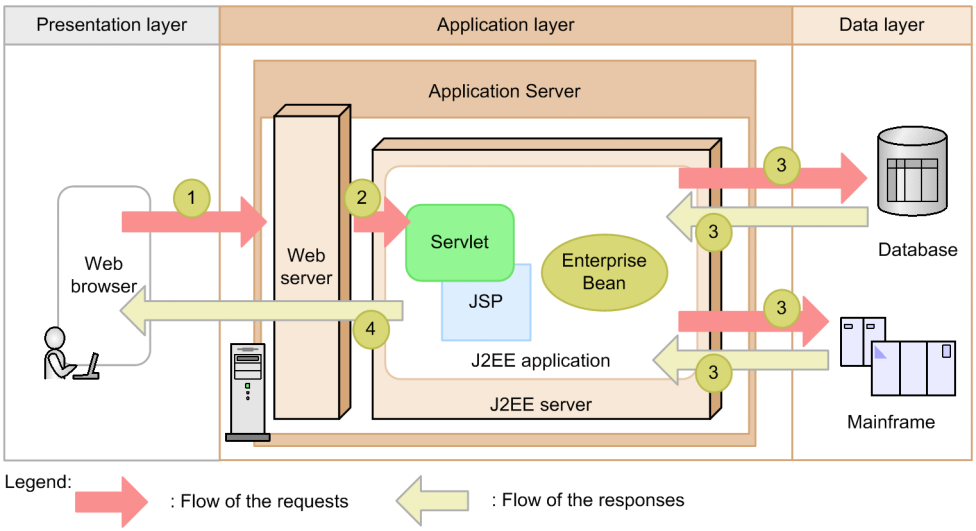
Application Server is an execution environment server base for an application and is centred on the J2EE server. Application Server is located at the center of an information system and it delivers the processes in a business system such as delivering user requests to a database.

The *J2EE application* is executed on Application Server to provide the services corresponding to the requests from the user. The J2EE application that operates on a J2EE server is developed according to the business contents to be executed.

The J2EE application receives requests from the user, executes the process and returns the result to the user. In addition to this, the J2EE application executes the process by exchanging the data with other systems such as the database and the mainframe and obtaining the required information.

The following figure shows the request process flow in the system centred on Application Server.

Figure 4–2: Request process flow in a system focusing on Application Server



The explanation of the flow in the figure is as follows:

1. When a user executes a process on the Web browser, a request is sent from the Web browser to Application Server. The request sent from the Web browser is received by the Web server, which is a part of Application Server.
2. The J2EE application corresponding to the request from the user is executed on Application Server. The J2EE application consists of a program (Servlet or JSP) to receive the requests sent from the Web server and a program (such as Enterprise Bean) to execute the business processes.
3. The J2EE application accesses the database and other systems to process the request from the user, as and when required.
4. When the business process is complete, a response is sent to the Web browser through the Web server. A window to be displayed on the Web browser is generated by a servlet or a JSP in the J2EE application. The generated contents are displayed as the process contents on the Web browser operated by the user.

Application server has safe operability and high fault tolerance and you can build a system that can be executed with excellent performance. In addition to this, you can build and operate an efficient system with functions for the smooth execution of operations.

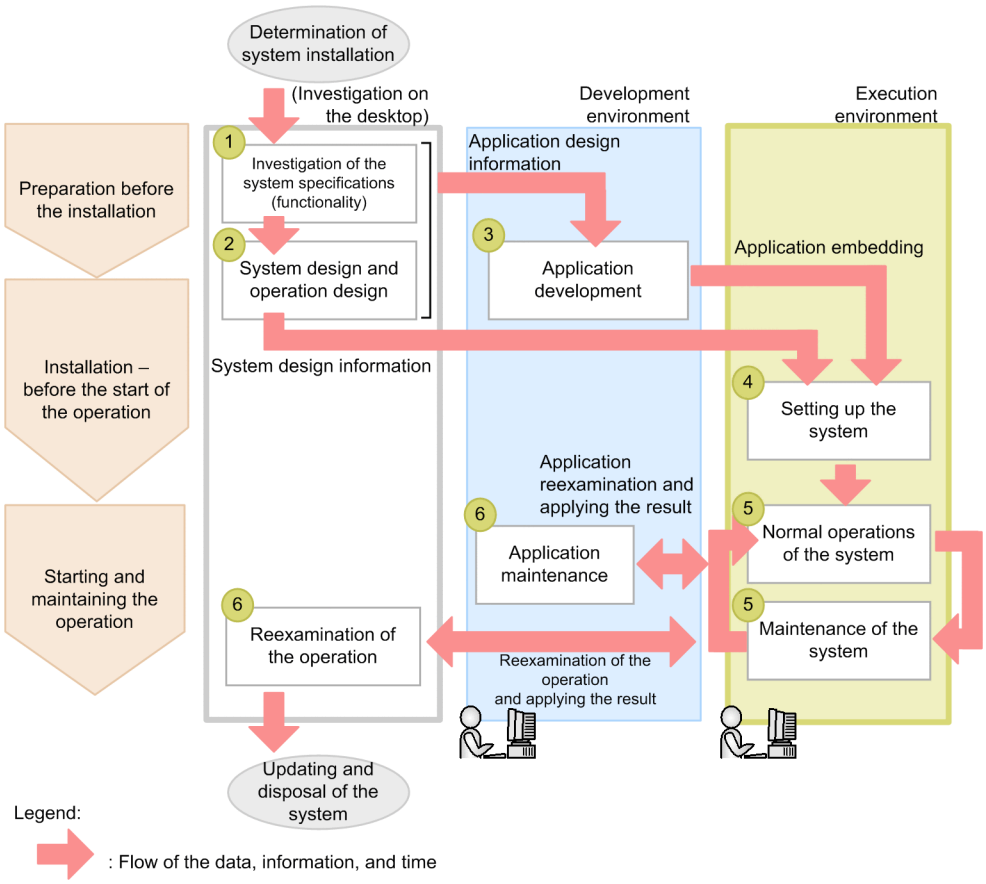
4.2 Life cycle of a system built with Application Server

The life cycle of a system built with Application Server consists of the following steps:

1. Investigating system specifications (functions)
2. System design and operation design
3. Developing an application
4. Building a system
5. Regular operations and maintenance of the system
6. Rechecking the operations and maintaining the application

The following figure shows the life cycle of a system in the development environment and execution environment of an application.

Figure 4–3: Life cycle of a system



The explanation of each step is as follows:

1. Investigating system specifications (functions)

   Investigate and determine the functions of Application Server to be used, depending on the system objectives and business contents.

2. System design and operation design

   Design the system configuration and the operation method depending on the system objectives, the business to be executed, and the method of using the system.

3. Developing an application

   Develop an application that fulfills the business requirements, based on the application design information investigated in steps 1 and 2 above.

4. Building a system

   Build a system based on the system design information investigated in steps 1 and 2 above. In addition to this, install the application developed in step 3 in the system.

5. Regular operations and maintenance of the system

   Execute the application and start the system operations. In system operations, repeat the routine operations and the regular system maintenance.

6. Rechecking the operations and maintaining the application

   Recheck the operation method as needed, depending on the changes in the business contents, or the changes in the system size. Perform the application maintenance, as and when required.

The life cycle continues until the system is updated or discarded.

4.2.1 Investigating the system specifications (functions)

Check and determine the functions to be used in the execution environment, depending on the system objectives and the business contents.

For details on investigating and determining the functions to be used, see section *3.2.5 Checking the functions of Application Server*.

4.2.2 System design and Operation design

Investigate the following points for the system design and operation design of Application Server:

- **How to configure a system/how to secure the system scalability**

  First define whether it is a system that executes online processes with J2EE applications or a system that executes batch processes with batch applications.

  Then, depending on the system size, determine the physical deployment of the Web server, J2EE server, batch server, and each process. For a system that executes online processes, investigate whether it is required to distribute the load using multiple servers. You must also investigate the system scalability. After the actual routine operations are started, it is very important to secure the scalability of the system to change the system size and to locally control the impact of any trouble.

- **Which operation method is to be used/how to secure the availability and reliability**

  When operating a business system, consider the method for most efficient operation while securing the system availability and reliability. Application Server has an operation management function called Management Server that functions for managing and operating multiple machines in a system, in a batch.

  Also consider whether to link to JP1 and cluster software as a method for efficiently operating the entire system.

- **How to ensure system security**

  For a mission-critical business system, it is mandatory to ensure security. You must clarify the viewpoint for building a secure system at the system design stage, and determine the procedure for installation and operation. When building a system that connects to an external network, consider how to build a highly reliable system by properly setting up a firewall, an intrusion detection system, and an SSL accelerator.

- **What kind of tuning is required to improve the performance**

  A rigorous performance design is required according to the system requirements. You can improve the overall system performance by effectively using pool or cache, and setting up a proper timeout.

- **How to perform tuning of JavaVM**

  The processes of a J2EE server operating on Application Server are executed on JavaVM. You can control the frequency of the full garbage collection by properly managing the memory space used by JavaVM and can thus improve the performance.

Note that if you want to build and operate Application Server with a standard pattern, first build and operate a system and then if required, change the settings and perform the tuning. First, at the system design stage, check whether the system to be built matches with the standard pattern described in the *Application Server System Setup and Operation Guide*.

4.2.3 Developing an application

Develop an application depending on the intended business contents.

You can execute J2EE applications developed with a Java language and batch applications on Application Server.

A typical J2EE application consists of a group of programs having components based on the MVC architecture. An individual business program is created as an Enterprise Bean depending on the EJB specifications. You can develop a re-usable J2EE application by developing a complicated application in combination with an Enterprise Bean and by interchanging the Enterprise Bean depending on the changes in the business.

You can develop an application that operates on a system built with Application Server by using the following methods:

- **Developing using the functionality of Developer**
  When using Developer, you can develop an application with WTP, which is an Eclipse plug-in.
  With WTP, you can execute a series of operations from coding to building, and debugging. You can perform operations such as starting and stopping of the J2EE server on the built test environment and starting and stopping of applications.
- **Developing without using Developer and IDE**
  You can develop an application even without using Developer and IDE. In this case, create a source for the program by using a text editor and compile it with the javac command. Create an archive by using the jar command to develop applications.

4.2.4 Building a system

Build a system based on the result of the system design.

Building a system is the stage at which J2EE applications or batch applications can execute the execution environment of Application Server. Building a system that executes J2EE applications includes operations such as installing a product, setting up the J2EE server, linking to the Web server, setting up J2EE resources and J2EE applications, and building an operating environment. Building a system that executes batch applications includes operations such as installing a product, setting up a batch server, and setting up resources.

Application Server provides the following three types of tools for building a system:

- **Setup Wizard**
  This tool builds a system by selecting or setting the items depending on the dialog box displayed on the interactive wizard. You can build the execution environment of a J2EE application.
- **Operation management portal**
  This tool builds a system by selecting the items on the GUI window displayed on the Web wizard, or by entering a value in the text. You can build the execution environment of a J2EE application or batch application.
- **Smart Composer function**
  This tool sets up parameters in a file having an XML format, sets up that file in an argument and executes a command to build the system. You can build the execution environment of a J2EE application or batch application.

You can build a system operating in the production environment with the Setup Wizard and without creating a detailed setting or a definition file. However, there are some systems that cannot be built with the Setup Wizard. Therefore, first check whether the intended system can use the Setup Wizard before building the system. For details on different tools, see the *Application Server System Setup and Operation Guide*.

The systems built using these tools can be managed in a batch by using the functions of a server process called *Management Server*.

4.2.5 System operations and maintenance

After you finish developing an application and building the system, execute the application and start the operation. In the system operations, repeat the routine operations and the system maintenance.

In the routine operations, in addition to starting and stopping the server, perform operations such as monitoring each process, collecting logs, and performing user management for the stable operation of the system. You can also use JP1 to perform integrated operations by collecting the systems having a wider scope.

The system maintenance includes changes in the system scale such as scale in, scale out, scale up, scale down and (*Troubleshooting*) of the errors that occur.

4.3 Characteristics of J2EE application execution environment

This section describes the characteristics of a J2EE application execution environment.

4.3.1 Standard specifications

Application Server is compliant to the standard specifications of **Java EE 6**.

With the execution environment compliant to Java EE, you can execute the commonly used functionality in multiple businesses by using the API of Java EE. The commonly used functionality include functionality such as the functionality for connecting to a database or a mainframe, the session management functionality, and the transaction management functionality.

These functionality are provided in an application, in a format such as the container, service, which is the module included in Application Server. You can reduce the complicated coding of each J2EE application by using the API of Java EE. You can also specify the attributes required for operating an application in an annotation.

In addition to this, a system built by Application Server is compatible with the standard specifications related to items such as the XML, Web service, SSL, and the distribution system. For details on the standard specifications supported by Application Server, see section *4.6.2 Standard specifications supported by Application Server*.

4.3.2 Executing the stable operations of a system

It is very important for a business system to continue the operations with stability. Depending on the business contents, at any given point in time, access might suddenly increase and the requests might be concentrated in a specific process. The following controls are effective for the stable operation of a system by promptly responding to the rapid changes in the system:

- **Flow control**
  The flow control method controls the number of concurrent executions of a process corresponding to the requests such as the access requests from the client and the process requests, and thus ensures the stability of operations by preventing the performance degradation due to increasing requests.
- **Priority control**
  In this method, a particular request is processed on a priority basis when the requests from the client are increasing. A request is processed depending on its urgency and priority.
- **Load balancing**
  In this method, there is more than one server to process the requests so that the requests are not accumulated in one location. The requests are distributed in multiple execution environments to prevent performance degradation and ensure the stability of the operations on a system.

With Application Server, you can execute the flow control and the priority control corresponding to a J2EE application. As a result of this, in addition to realizing stable system operations, you can also use system resources effectively.

This section describes the flow control of a Web application, and flow control and load balancing of an Enterprise Bean with the OLTP technology. For details on the respective functions, see section *2.15 Overview of controlling the number of concurrently executing threads* in the *Application Server Web Container Functionality Guide* and the chapter *3. Scheduling and load balancing of requests by CTM* in the *Application Server Expansion Guide*.

(1) Flow control of a Web application

Application Server executes a Web application that consists of servlets and JSPs with the Web container, which is the Web application execution base of the J2EE server.

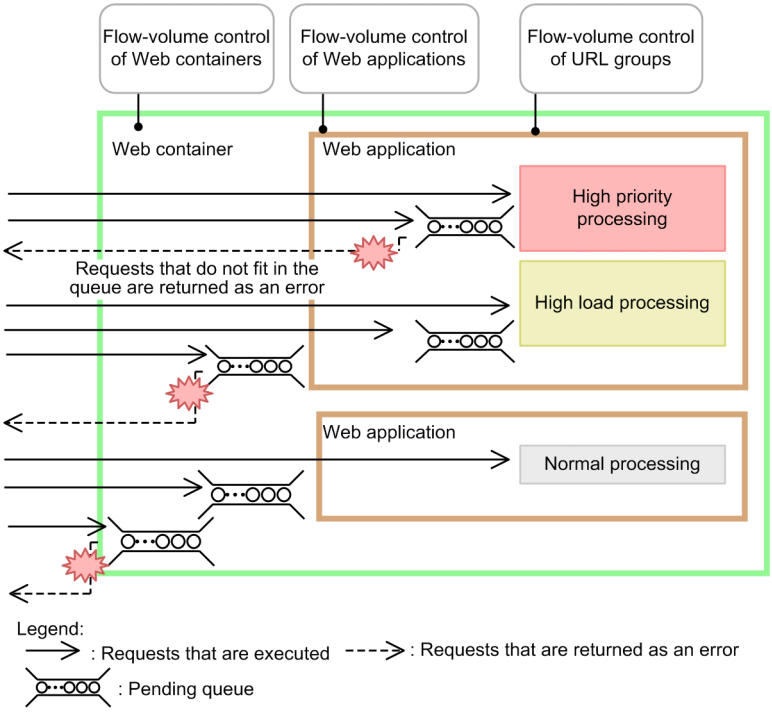
With the Web container of Application Server, you can set the number of requests that can be concurrently processed in a Web container unit, Web application unit, and a **URL group** (business logic) unit in a Web application. You can also manage the requests with the concept of request queue (queuing). As a result of this, you can fully control the number of processes that can be concurrently executed in a Web application according to the process content. With

this, even if the number of sent requests suddenly increases, the system operations can be executed with stability, as it is now possible to control the number of requests to be processed as a fixed quantity.

In addition to this, you can execute the requests depending on their level of importance. For example, you can ensure that very urgent processes are definitely processed, which results in preventing the impact of the high-load business processes on other businesses.

The following figure shows an overview of the flow control of a Web application.

Figure 4–4: Overview of the flow control of a Web application



You can execute system operations with stability by executing the flow control in the Web container unit, Web application unit, and URL group (business logic) unit. In addition to this, you can build a system that can execute the processes on priority even in the case of a high load.

(2) Applying OLTP technology in Enterprise Bean

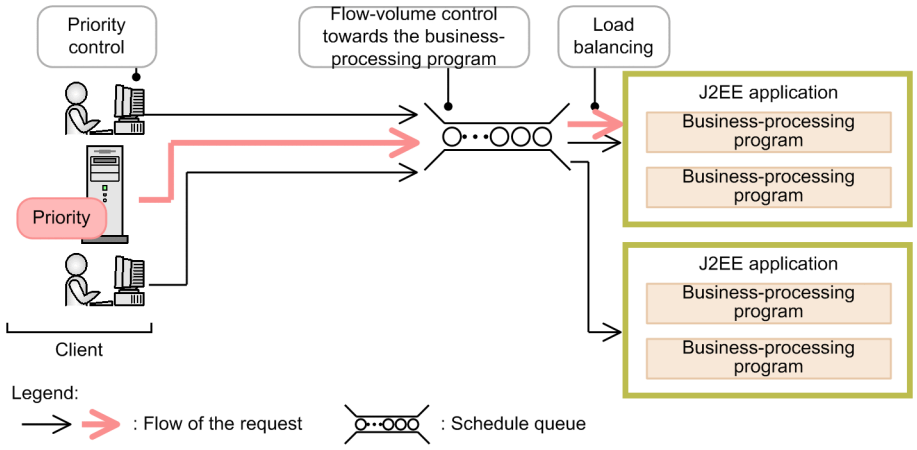
The OLTP technology is mandatory for a large-scale business system. With Application Server, you can apply the advanced OLTP technology for a business process program (Enterprise Bean) in J2EE applications. Thus, you can execute the following processes. Note that the target Enterprise Bean is a Stateless Session Bean.

- **Flow control and load balancing with the request scheduling**

  If many requests are collected in a particular J2EE application, you can process the requests by distributing them on multiple servers, and control the number of requests (flow) to be processed at a time by scheduling the requests depending on the business process program. You can also process the requests sent from the client on a priority basis by setting the order of priority for the clients that send the requests. In addition to this, it is possible to prevent load accumulation by appropriately distributing the processes corresponding to a particular business process program, thereby improving the process performance of the entire system and enabling stable system operations.

  The following figure shows the priority control, flow control, and load balancing of an Enterprise Bean.

Figure 4–5: Overview of priority control, flow control and load balancing of Enterprise Bean



Even if a large volume of requests is sent, you can control the number of requests to be executed concurrently, and distribute the load by executing the requests of a business process program in a J2EE application through a scheduling queue. You can ensure the execution of important requests promptly with the priority control.

- **Service lock**

  When replacing a particular business process program while operating a J2EE application, you can safely lock only the relevant services by controlling the requests corresponding to the target business process program. If an error occurs in a particular business process program, you can localize the relevant location and replace the business process program, without stopping the entire system, by executing degeneration and retrieving.

4.3.3 Improving availability and fault tolerance

In a mission-critical business system, it is mandatory to continue the services provided by the system with stability and without stopping these services.

For example, among the services, there might be services that must operate continuously for 24 hours and services that incur a huge loss if the business system stops due to an error.

As a result of this, for Application Server that is the base of a business system, it is very important to prevent error occurrence, and, even if an error occurs, it is very important to locally control the impact of the error, and to continue the operations without stopping the business system. In addition to this, it is also very important to restore the location of error occurrence.

This section describes the characteristics of a system having high availability and fault tolerance that can be realized with Application Server.

(1) Stopping the occurrence of full garbage collection

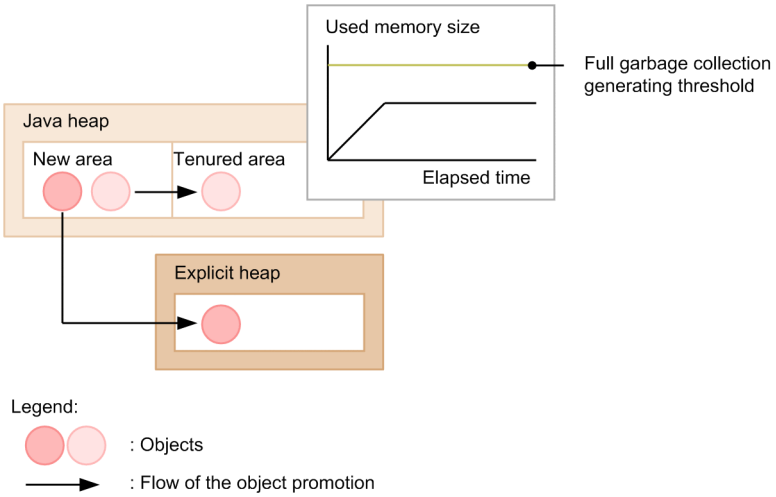
You can control the frequency of stopping the system by stopping the occurrence of full garbage collection.

The full garbage collection occurs because of an increase in objects in the Tenured area of the Java heap. The objects having a long life (objects having long usage time) are set up in the Tenured area. Among the objects having a long life, and which cause the full garbage collection, Application Server places the objects having a definite usage time in a unique memory space. This memory space is called the *Explicit heap*. Since the full garbage collection does not occur in the Explicit heap, it is possible to stop the full garbage collection of these objects.

Application Server places the objects related to an HTTP session and the objects for communication with a redirector in the Explicit heap.

The following figure shows an overview of stopping the occurrence of the full garbage collection with the Explicit heap for the objects related to an HTTP session.

Figure 4–6: Overview of stopping the occurrence of full garbage collection with Explicit heap



A part of the objects upgraded in the Tenured area are moved to the Explicit heap from the new area. As a result of this, the increase in the memory size of the Tenured area can be controlled and the occurrence of the full garbage collection can be stopped.

In addition to this, it is also possible to generate the objects that cause the full garbage collection in the application, in the Explicit heap directly by modifying the user application.

(2) Preventing error occurrence

You can prevent error occurrence by monitoring the operation status of an application and resources and by anticipating the errors. In addition to this, you can also set and monitor a threshold value and define an action for when this threshold value is exceeded.

For example, if full garbage collection occurs in JavaVM because of accumulation of requests, you can set up and monitor a threshold value in the frequency of full garbage collection occurring within a given time. If the threshold value exceeds, it is possible to automatically reduce the number of requests that are concurrently executed and to prevent the slowdown of the system.

(3) Improving availability when an error occurs

You can use the following functions to improve the availability when an error occurs:

- **Settinga correct timeout**

  You can set an appropriate timeout between the client and a Web service, a Web server and a Web container, an EJB client and an EJB container, and an EJB client and a naming service. As a result of this, even if an error occurs on the communication destination machine, you can detect the error in few seconds and the service is not stagnated by not responding.

  You can also set a timeout for the method level processes in an application. As a result of this, if an error of infinite loop occurs in an application, that process gets cancelled and the job can continue.

- **Linking with cluster software**

  You can prevent the stopping of services by linking with the cluster software, setting the cluster configuration for Application Server and Management Server, and by promptly switching the nodes if an error occurs. (1 to 1 cluster configuration, the mutually switching configuration, or the switching configuration considering the host unit management model as the target).

  If an error occurs on Application Server while using the resources, you can conclude the transaction procedure by releasing the resources immediately when an error occurs (N to 1 cluster configuration), by preparing one standby server.

- **Inheriting the session information**

  With a Web system, if an error occurs on a sever on which a session has already been set up from the client; it is possible to inherit the service by inheriting the session information on the other server. The session information

can be managed by the database. If an error occurs on a Web server or a J2EE server, the Web system retrieves the session information from the database and inherits the services.

- **Reconnecting with the backend system**

  If the connection of database with the backend service is abnormally terminated, it can be automatically reconnected.

- **Output of a detailed log**

  Application Server provides a Hitachi-specific JavaVM (hereinafter referred to as *JavaVM*). JavaVM has a function to output a detailed log related to the status of Java objects. You can use this log in tuning for a cause analysis in the case of an error and for improving the performance.

4.3.4 Easy system installation and expansion

A business system consists of multiple elements such as the Web server, J2EE server, load balancer, and the server for operation management. The J2EE server also has various functions. For smooth operations using the characteristics of a business system centered on Application Server, you must appropriately set the parameters of the respective elements by determining the system configuration according to the system application.

This section describes the characteristics of Application Server for the system installation and expansion.

(1) Easy building of the system

You can build a system that executes J2EE applications with the `Setup Wizard`. You can use the Setup Wizard to build a system by entering a setup value in the CUI program dialog box conversational format.

Use the Operation Management Portal or the Smart Composer Function for building a system that uses the configuration and functionalities that are not supported by the Setup Wizard. For details on the characteristics and usage of the respective tools, see the *Application Server System Setup and Operation Guide*.

(2) Setting the attributes of an application in the development environment

You can define the unique definition information of Application Server in the development environment in an XML file (cosminexus.xml). You need not set the attributes in the execution environment, when you import the application that includes the cosmimexus.xml file to the J2EE server of the execution environment.

For example, since the linkage resolution with resources can be defined in the development environment, you can start the application by immediately importing in the execution environment. In addition to this, if you want to switch an application by changing the DD, you need not redefine the Application Server specific definition.

Use CUI (server management command) to set the attributes of an application in the execution environment.

4.3.5 Ensuring the system security by monitoring the system

Since problems such as accounting scandals have been increasing, it is necessary to reinforce an internal control in organizations.

The objective behind internal control is to understand what type of business was executed by whom, and when and how to ensure that the business is implemented by adhering to every law and regulation. The following conditions are required for supporting the internal control:

- It is mandatory to ensure that a person in-charge with proper rights appropriately executes the business with correct operations.
- An auditor or an evaluator must be able to certify that there is no problem in the verified result.

It is mandatory to monitor "What was executed", by "Whom", and "When" and this result must be maintained as a record.

Application Server provides the following functions:

- Output of audit log
- Output of audit trail information linked with database

This section gives an overview of the above functions. For details on the output of an audit log, see section *6.4 Output of audit log* in the *Application Server Operation, Monitoring, and Linkage Guide*. For details on the output of the audit trail information linked with database, see section *7.6.2 Obtaining audit trail information of database* in the *Application Server Operation, Monitoring, and Linkage Guide.*

(1) Output of audit log

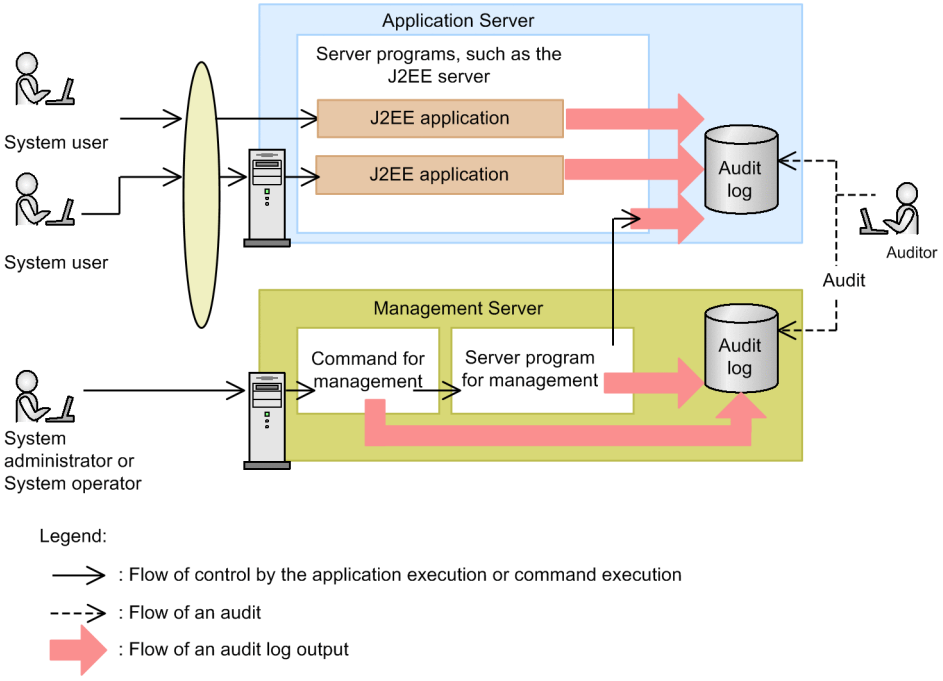
In a business system built with Application Server, the information related to *Who*, *When*, and *What was executed* is output as the **Audit log**.

The following information is output in the audit log:

- The history of the operations executed by the system administrator and the system operator, and the history of the program operations corresponding to these operations
- The history of the operations executed by the system user using a J2EE application and the history of the program operations corresponding to these operations

The following figure shows an overview of the audit log output in Application Server.

Figure 4–7: Overview of the audit log output in Application Server



Note that the audit log output by Application Server can be managed by consolidating with the audit log output by the Hitachi middleware other than Application Server, by linking with JP1.

For details on linking with JP1, see section *6. Linking with Other Products*.

(2) Output of audit trail information linked with the database

In many cases, the database operates in the backend of a business system built by Application Server.

The database includes very important data that must not be leaked and falsified. This information must be managed strictly with an appropriate security management.

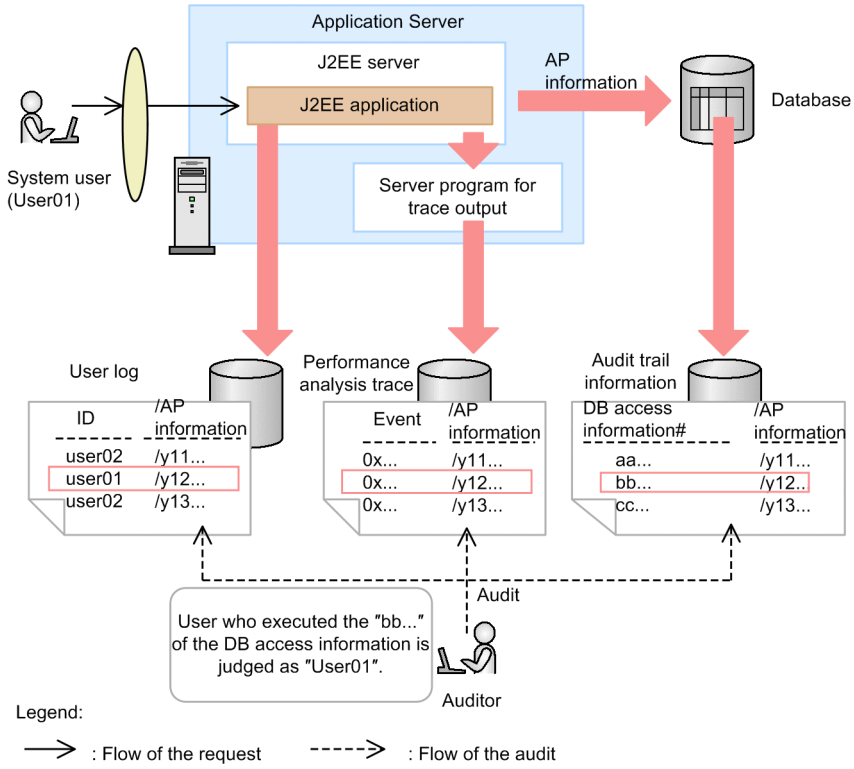
You can output the information related to "What type of database access was executed" by "Whom", and "When" in the database. This information is called the *Audit trail information*.

With Application Server, you can output the information that indicates the request of Application Server in which the database access was executed, to the audit trail information output by the database. If you combine this information with the log information output by a J2EE application, you can trace details such as the user operation of Application Server, which was extended to execute the database access.

Note that the database that Application Server can link with, to output the audit trail information is HiRDB.

The following figure shows an overview of the output of the audit trail information linked with the database.

Figure 4–8: Overview of output of audit trail information linked with the database



Note: AP information implies the root application information (information for specifying a request).

The database operation log that is being managed by the database, is output.

In this example, the following three types of information are output, when the system user (user01) accesses the database through J2EE applications:

- The user log output by a J2EE application
- The performance analysis trace output by the server program (performance tracer) for the trace output
- The audit trail information output by the database

The information to identify all the requests is output in the above types of information (root application information). With this information, the auditor verifies the request extension by which the database was accessed and the user who has executed the request. In addition to this, if you use the performance analysis trace, you can verify the flow of request processing.

4.3.6 Executing the operation management to improve business efficiency

The operation management of a business system becomes complicated with increasing volume. An operation method that is efficient, has a controlled operational cost, and that achieves the maximum system performance is required. Therefore it is mandatory to have an interface with an excellent operability, and a mechanism for automatic operations.

Application Server provides the following operation management functions to fulfil these requirements:

- **Batch operation of a business system**

  An environment built with the Setup Wizard, Operation Management Portal, and Smart Composer Function is managed based on the concept of an operation management domain. The operation management domain is

managed by the processes called *Management Server*. You can manage the start and stop processes of multiple J2EE servers and Web servers in a batch.

In addition to this, you can execute operations such as monitoring, automation of operations, and analysis of operation status of the overall business systems including systems that are not built with Application Server in a batch, by linking with JP1.

- **Output of operation information or resource usage status of a system**

  You can output the operation status and resource usage status of a system. It is possible to prevent the occurrence of trouble and take countermeasures if trouble occurs by monitoring the output contents. The output information is also used for system tuning. You can also output this information to a file.

- **Automation of operations**

  You can automate recovery operations when an error occurs and automate operations to prevent errors by using functions such as the function for restarting the server when an error occurs and functions for issuing events and executing actions linked with the operation management.

  You can automate the operations of all the systems including the systems that are not built with Application Server, by linking with JP1.

- **Log operations**

  You can collect the logs output by the Application Server system in a batch. It is possible to handle the logs of J2EE applications developed by the user in the same manner as that of the logs output by an Application Server system. It is possible to execute a highly reliable log operation by managing the logs of the overall system in a batch.

  In JavaVM used by Application Server, the output contents of logs are extended so that they can be used for cause analysis when an error occurs and for checking the system status. You can improve the system availability by executing appropriate tuning using this log.

- **Performance analysis by trace information**

  When executing a request, the trace information is output at every point starting from the Web server up to the database connection. You can clarify the process bottlenecks and specify the location of error occurrence, by analysing this information.

  Note that you can output the trace information in a system and also in an application developed by the user.

- **Integrated user management**

  You can execute the integrated management of users logged in a system built with Application Server. Since the information of the users managing the respective J2EE applications is linked and managed, you can log in to the various J2EE applications, with a one-time login process. Note that you can use the LDAP directory service and the database as a repository to manage the user information.

- **Efficiency of switching application at the time of maintenance**

  You can manage the applications in two formats; the archive format and the deployable directory format. For an application in the deployable directory format, you need not re-archive when switching the application but the switching is done just by reloading the changed class file. For an application in the archive format, switching (redeploy) is possible without stopping and restarting the application in case of local changes. As a result of this, the switching efficiency increases during maintenance and the system operation cost is also controlled.

4.3.7 Web Services

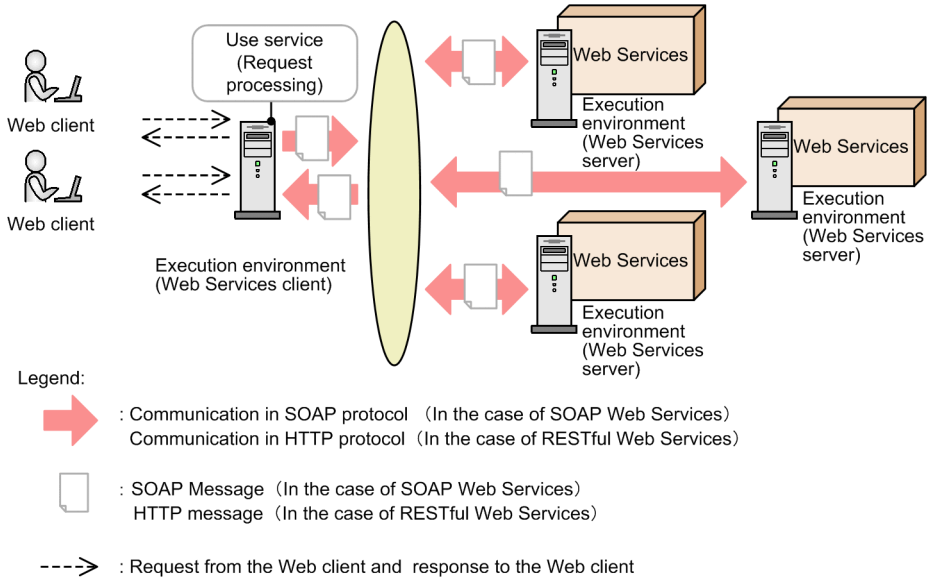
Web Services exchange the information between systems by using Web technologies and is a mechanism for enabling the use of applications of other systems on the Web. An application that can be released and executed through a network, by using the mechanism of a Web Service is also called a Web Service. You can execute SOAP Web Services and RESTful Web Services with Application Server.

SOAP Web Services use a protocol called *SOAP* to exchange messages. The HTTP protocol is used in the lower transport layer of the SOAP protocol. Application Server executes the binding among SOAP messages, SOAP Web Services, and Web Service clients according to the JAX-WS specifications.

RESTful Web Services directly use the GET and POST methods of the HTTP protocol, to exchange messages. Application Server executes the binding among HTTP messages and RESTful Web Services according to the JAX-RS specifications. An execution environment that can serve as the server of RESTful Web Services is required for using Web Services. You can build this execution environment with Application Server. You can develop a client for RESTful Web Services by using either client APIs or the standard Java APIs for RESTful Web services.

The following figure shows an overview of the Web Services usage environment.

Figure 4–9: Overview of Web Services usage environment



For the environments used by Web Services, the execution environment of the Web Services client and the execution environment of the Web Services server are mandatory. Both these execution environments can be built with Application Server.

The execution environment (client of Web Services) that has received the requests from the users of the Web client sends SOAP messages (for SOAP Web Services) or HTTP messages (for RESTful Web Services) to the execution environment that provides Web Services (server of Web Services). The Web Services server executes the process, and sends the result to the Web service client as a response. The Web Services use the SOAP protocol (for SOAP Web Services) or HTTP protocol (for RESTful Web Services) for the communication between the clients and the server of Web Services.

For details on these functions, see the *Application Server Web Service Development Guide*.

Reference note

You can develop and execute Web Services by using the existing SOAP application development support functionality and SOAP Communication Infrastructure. (The application to be developed is called a *SOAP application*.) To develop and execute a SOAP application, see the *Application Server SOAP Application Development Guide*.

4.3.8 Highly reliable asynchronous communication

You can execute asynchronous communication using messages between applications.

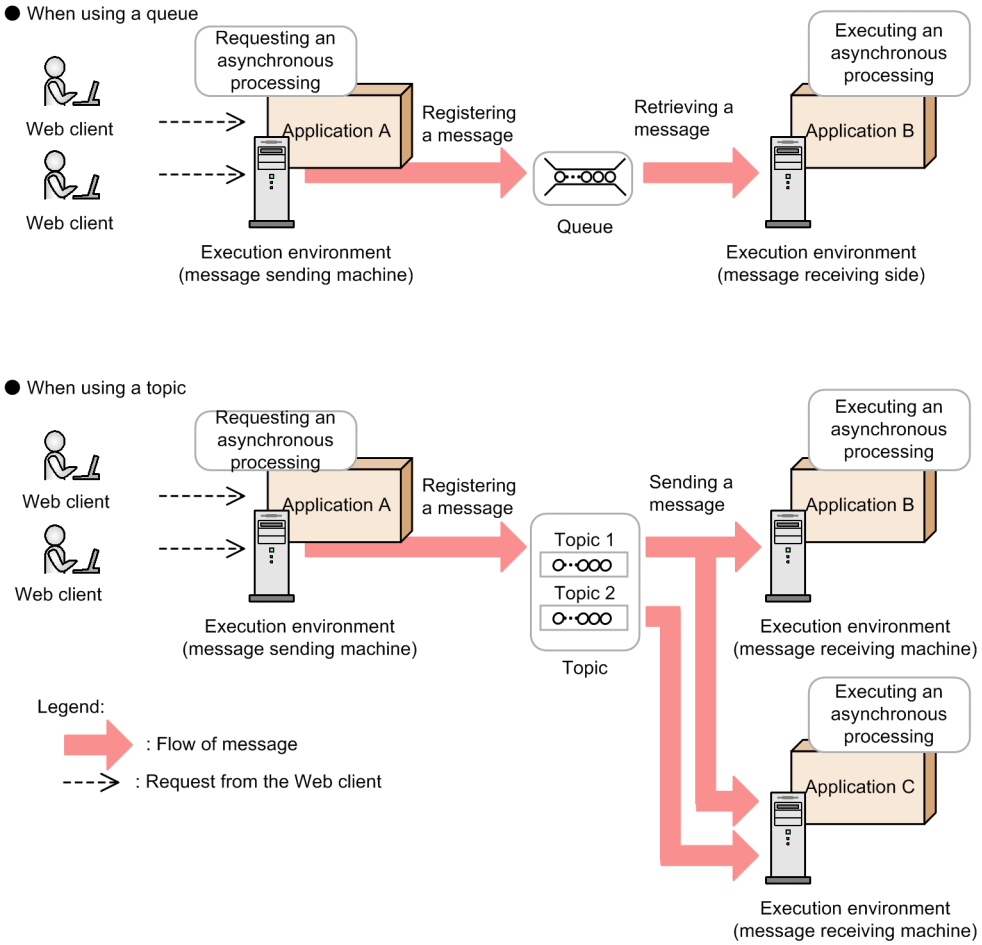
In the asynchronous communication, while sending messages, the application registers the message in a queue or a topic.

When using the queue, the application on the receiving machine receives the message and extracts the message from the queue. As a result of this, the applications on the sending and receiving machines exchange the messages even when the machines are not concurrently operating.

When using a topic, the message is sent to the application on the receiving machine which is subscribed in advance. The subscription can be applied from multiple applications.

The following figure shows an overview of asynchronous communication executed by Application Server.

Figure 4–10: Overview of asynchronous communication executed by Application Server



Note that you can use the topic only when using the functionality of the JMS provider.

When using Reliable Messaging, since the queue that manages the messages is managed by the database, you can achieve a highly reliable message management. You can ensure the reliability of the message communication by QoS (Communication quality) assuring one-time delivery or sequential deliveries. In addition to this, since WS-Reliability which is a highly reliable standard protocol is also supported using the JMS interface, you can flexibly support the linking with other systems and porting of the application from other systems. For details on functions of Reliable Messaging, see the manual *Reliable Messaging*.

Reference note

This chapter describes the asynchronous communication when using the functionality provided by Application Server.

You can also execute the process on Application Server asynchronously by receiving messages from external resources using the optional resource adapter compliant with the Connector 1.5 specifications. For further details, see section *3.16.3 Message inflow* in the *Application Server Common Container Functionality Guide*.

4.3.9 Virtualization

Recently, with the increasing system advancement and complexity, the number of servers used in a system is increasing. However, there is a risk of increase in the maintenance and management cost with an increase in the number of server machines. The server virtualization technology is considered to be a solution to this problem. With the virtualization technology, you can increase the number of servers without physically increasing the number of servers. However, because there is a trend for the number of virtualized servers to increase, the load of building and operating a J2EE application execution environment on virtual servers is not reduced.

If you use the functions of Application Server, you can easily build and operate a system by using the virtual server technology. The following operations can be executed with this virtualization technology:

- **The environment in which the same J2EE application is executed on multiple servers can be built and operated in a batch.**

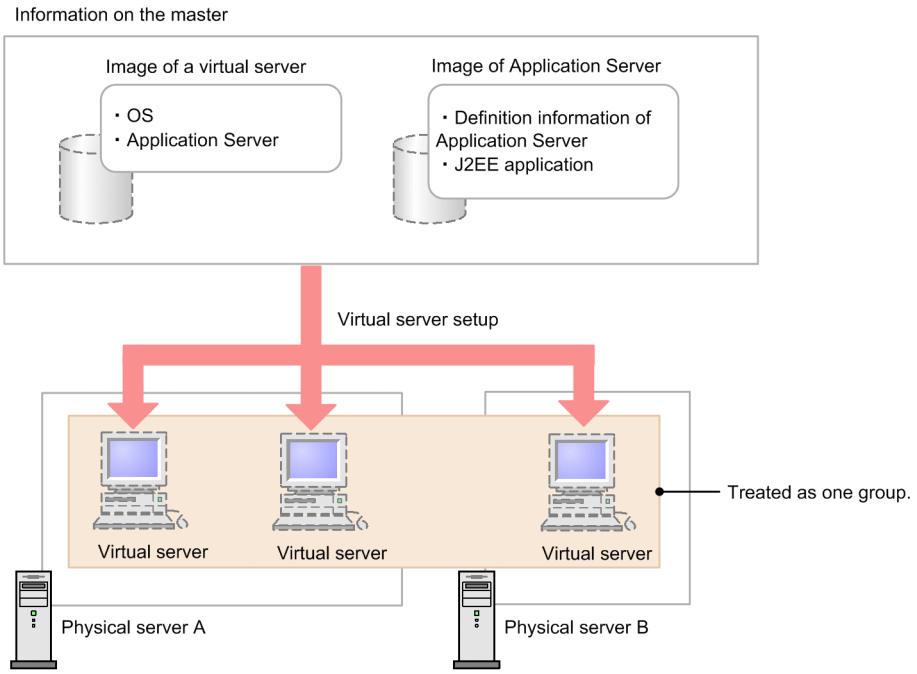
  To build a virtual server with the functionality of Application Server, first register the following information to be set up on the virtual server as the master information:

  - Image of the virtual server (Products to be installed on the OS and the virtual server)
  - Image of Application Server (Definition information of Application Server and the J2EE application)

  You can build multiple virtual servers having the same environment in a batch if the virtual server is built based on the registered master information. Since the virtual servers having the same execution environment in which the same J2EE application is operating are handled as one group, it is possible to start or stop the multiple virtual servers having the same environment, in a batch.

  The following figure shows an example of building and operating the virtual servers from the master information in a batch.

  Figure 4–11: Example of building and operating the virtual servers in a batch

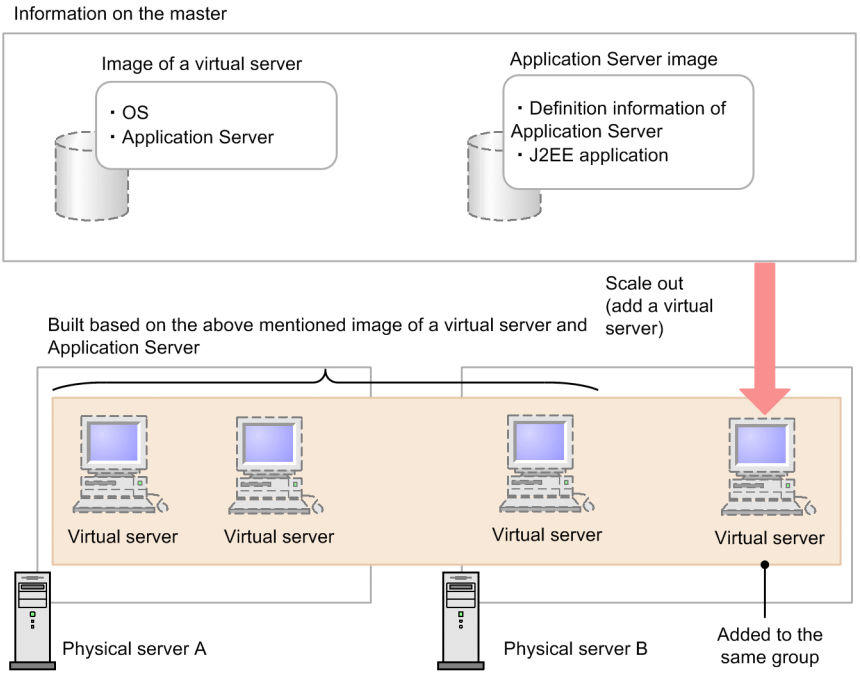
  

  In the above figure, three virtual servers are configured as one group. The virtual servers in this group can be started and stopped in a batch.

- **It is possible to scale in and scale out the virtual server by executing commands**

  If access is accumulated in a particular J2EE application, you can easily increase the number of virtual servers (scale out) by executing commands. The master information is used here. The following figure shows an example of adding one virtual server as a scale out.

Figure 4–12: Example of adding a virtual server



As the virtual server added by scale out is added in the same group as that of the already built virtual server, it is also considered for batch start and batch stop.

Note that when the access count and the load of virtual servers are reduced, you can also reduce the number of virtual servers (scale in).

- **Operation rights of virtual server manager can be managed.**

  You can manage the account of the user who can execute the virtual server manager separately from the OS account. An authority related to the overall functions and the authority related to part of the functions such as building and operating the virtual server can be set as the operations rights of the virtual server manager. You can enhance the security by setting the operations rights of the virtual server manager for every user.

For details on building and operating Application Server that operates on a virtual server, see the *Application Server Virtual System Setup and Operation Guide*.

Reference note

When using a virtual server manager in 08-50 mode, the configuration pattern of the virtual system that can be built by Application Server is predetermined. This section describes the respective system configuration patterns:

- Pattern 1: Providing the test environment on demand

  The test environment provided for the developer of the Web application is virtualized and the usage provided on demand according to the requests of the developer is presumed.

- Pattern 2: Persistent operation of the execution environment

  In this pattern the performance and reliability of an application is secured by building a Web system in the virtual environment and stable operations are maintained in the application by persistently operating the execution environment of each server.

- Pattern 3: Scale out of the execution environment

  In this pattern the flexibility of the system configuration is achieved by building a Web system in the virtual environment. According to the increase in the volume of business, scale out is done (expansion) and the latest operations are always assumed for the environment of each server.

- Pattern 4: Consolidation of small scale section servers

  This pattern assumes a simple configuration and operation in which the scattered section servers are consolidated in a virtual environment and thus the TCO is reduced.

In version 08-53 or later, you must make the settings for using the virtual server manager in 08-50 mode. See *Appendix D Setting* when using the virtual server manager in 08-50 mode in the *Application Server Virtual System Setup and Operation Guide*.

You can execute the following operations with the virtual server manager in the 08-50 mode:

- You can build and operate an environment that executes the same J2EE application on multiple virtual servers, in a batch.
- You can perform scale in or scale out of the virtual servers by executing the command.
- You can easily change the contents of the virtual servers without stopping business.

  When you update the programs of the OS on the virtual server, or change the definition information of Application Server, if you update the image, the updated contents are reflected on multiple virtual servers.

  The feature of this pattern is that you need to execute only one command to apply the image to multiple virtual servers. You can apply the changes even if the virtual server is running or when stopped. By applying the changes on the machines one by one, you can apply the changes without completely stopping the business.

4.3.10 Linking with OpenTP1

You can call business logic operating on Application Server from a system that uses OpenTP1 by linking Application Server and OpenTP1. As a result of using OpenTP1, when adding a new service in the system, efficient development is possible using Java. Note that you can check the flow of processes between OpenTP1 and Application Server by comparing the trace information.

4.4 Characteristics of the execution environment of a batch application

This section describes the characteristics of the execution environment of a batch application.

4.4.1 Executing a batch job in the open environment

Execute the batch job in the execution environment of Java, which is an open environment.

The execution environment of a batch application is an environment that executes a batch application developed as a Java program. You can migrate a batch job to an open environment, by using this environment.

Currently most of the batch jobs that execute the processes of core businesses are executed on mainframes. Application Server builds the execution environment of a batch job as the Java execution environment, which is an open environment. As a result of this, you can build and operate the execution environment for batch jobs endowed with flexibility and easy operability that are the features of Java.

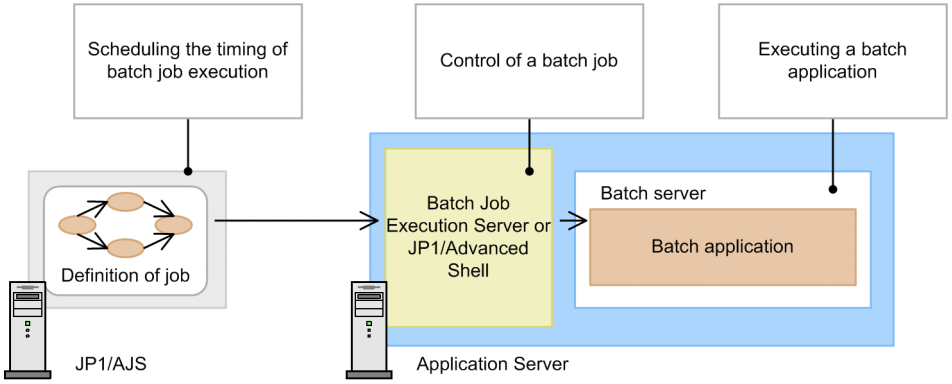
You execute the batch applications with the *Batch server* that is a resident JavaVM process. This enables you to control the cost of starting JavaVM every time when executing a batch job. This is effective when you repeatedly execute batch applications that comparatively have a shorter response time.

You can control or automatically execute processes similar to the contents executed with a mainframe, by linking with the following related programs:

- **JP1/AJS**
  You can schedule the timing for the execution of a batch application. With this, you can automate the business in which the batch application is used.
- **Batch Job Execution Server**
  You can execute the batch application in the image of job management depending on the job control language executed on the mainframe.
- **JP1/Advanced Shell**
  You can execute the batch applications by using a UNIX Korn shell script file that can be used on a cross platform basis.

The following figure shows an overview of the batch job execution in the execution environment of a batch application.

Figure 4–13: Overview of the batch job execution in the execution environment of a batch application



In this configuration, you can control the timing of executing a batch job with JP1/AJS. You can also execute a batch job based on the contents defined by using Batch Job Execution Server or JP1/Advanced Shell.

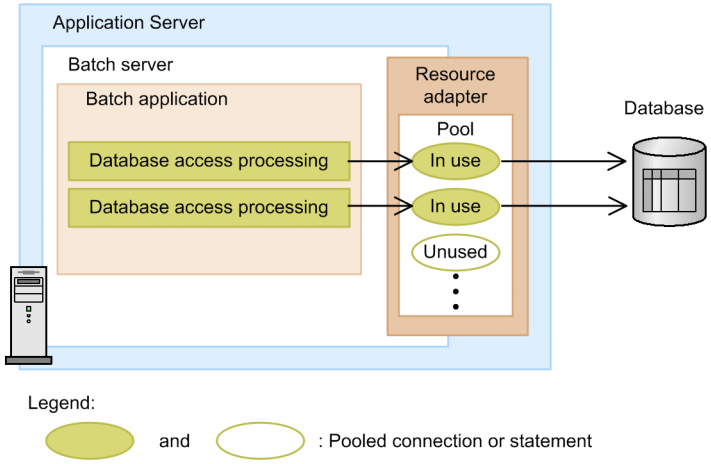
4.4.2 Speeding up the database access that uses the connection pool and statement pool

Generating a connection or a statement to be used for the database connection is a time consuming process. The connection pool and statement pool are the functions to improve the process performance by pooling the generated connections and statements.

In the execution environment of a batch application, you can use the functions such as the connection pool and statement pool. With this, you can quickly access the database from a batch application. Note that you must execute the connection pool and statement pool functions by using the resource adapter.

The following figure shows an overview of accessing the database with the connection pool and statement pool.

Figure 4–14: Overview of accessing the database with the connection pool and statement pool



Use the pooled connection or statement for accessing the database from a batch application.

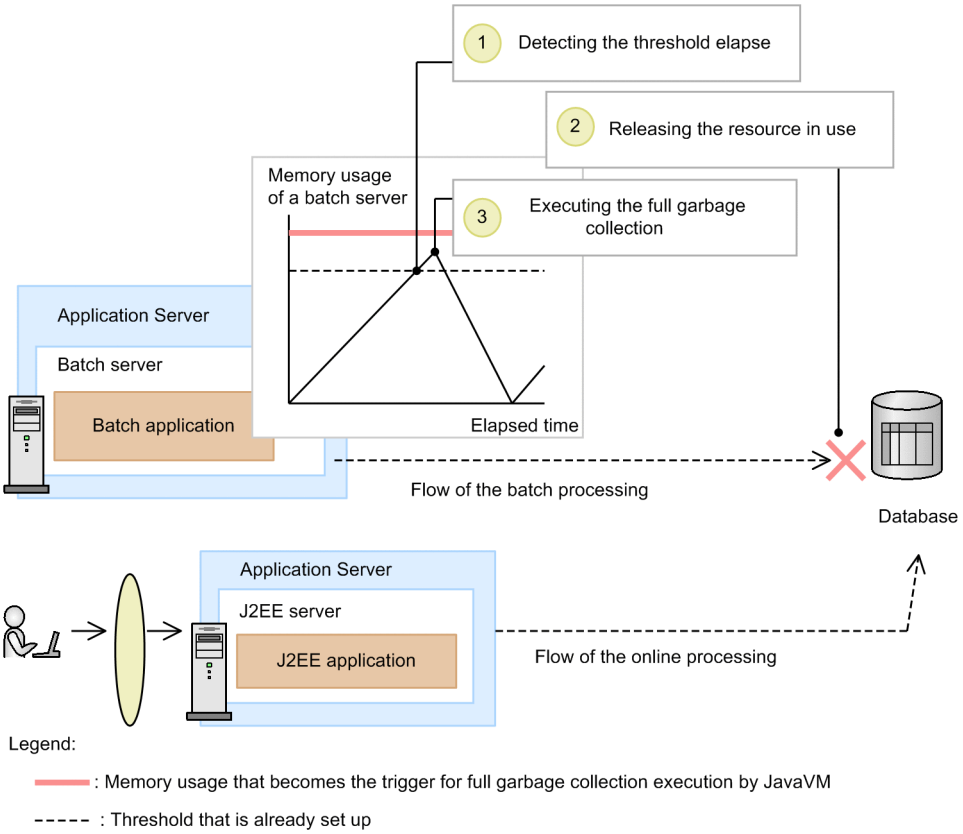
4.4.3 Stopping the execution of full garbage collection in the resource exclusion status

In JavaVM, the full garbage collection is executed when the free memory size becomes less than the fixed size. When the full garbage collection is executed, all the processes on that JavaVM are interrupted. If the full garbage collection is executed while the application on the batch server is using the resources in the exclusion status, the process is interrupted as is in the exclusion status. In this case, execution of the online processes that need to use the same resources is also interrupted.

With the execution environment for batch applications, you can control the timing for executing the full garbage collection on the batch server. With this control, you can execute the full garbage collection exactly when the resources are not excluded, and can release the objects that are not required. This enables you to suppress the execution of the full garbage collection at an unintended timing.

The following figure shows an overview of controlling the full garbage collection on the batch server.

Figure 4–15: Overview of controlling the full garbage collection on the batch server



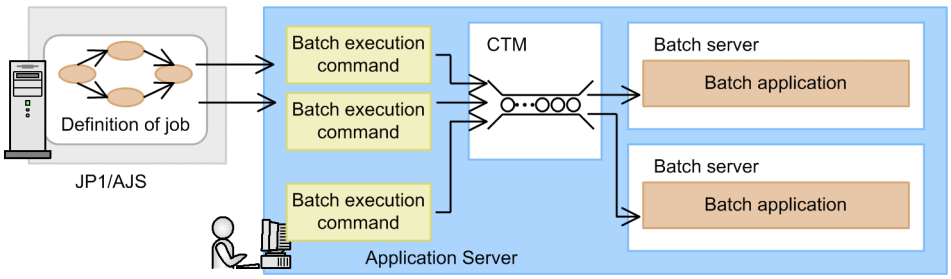
With the control of full garbage collection, the event that the used memory space has exceeded the threshold value is reported as shown in step 1 in the figure. After it is confirmed that the used resources are released on the batch server as shown in step 2, full garbage collection is executed as shown in step 3.

4.4.4 Scheduling jobs in a batch application

You can execute one batch application per the batch server. If you execute batch applications exceeding the number of batch servers, an error is returned. If there is a request for execution of a job for which the number of batch servers has exceeded, you can keep the execution request of that batch job waiting, by using the function of scheduling the jobs of a batch application. This can prevent the event of an error return.

You use the CTM function for scheduling the jobs of a batch application. The following figure shows the scheduling of jobs with CTM.

Figure 4–16: Scheduling of jobs with CTM



A batch application execution request is sent by a batch execution command executed through JP1/AJS or by a batch execution command executed directly. The CTM uses the concept of a queue to manage the execution requests. When an executable batch server exists, the processes are sorted and if no executable batch server exists, the execution requests are retained in the scheduling queue.

4.4.5 Other characteristics

The execution environment of a batch application has the following additional characteristics. The execution environment has the same characteristics as those of the J2EE application execution environment.

- **Improving the availability and fault tolerance**

  See section *4.3.3 Improving availability and fault tolerance*. However, the description of inheriting the session information is not applicable here.

- **Easy system installation and expansion**

  See section *4.3.4 Easy system installation and expansion*.

- **Ensuring the security by monitoring the system**

  See section *4.3.5 Ensuring the system security by monitoring the system*.

- **Execution of operation management to improve business efficiency**

  See section *4.3.6 Executing the operation management to improve business efficiency*. However, the descriptions of items such as the integrated user management and efficiency of switching the application during maintenance are not applicable here.

4.5 Characteristics of application development

You build the Application Server development environment by installing Developer. In Developer, a development environment is built in such a way that the application operating in the execution environment can be efficiently developed with easy-to-understand operations.

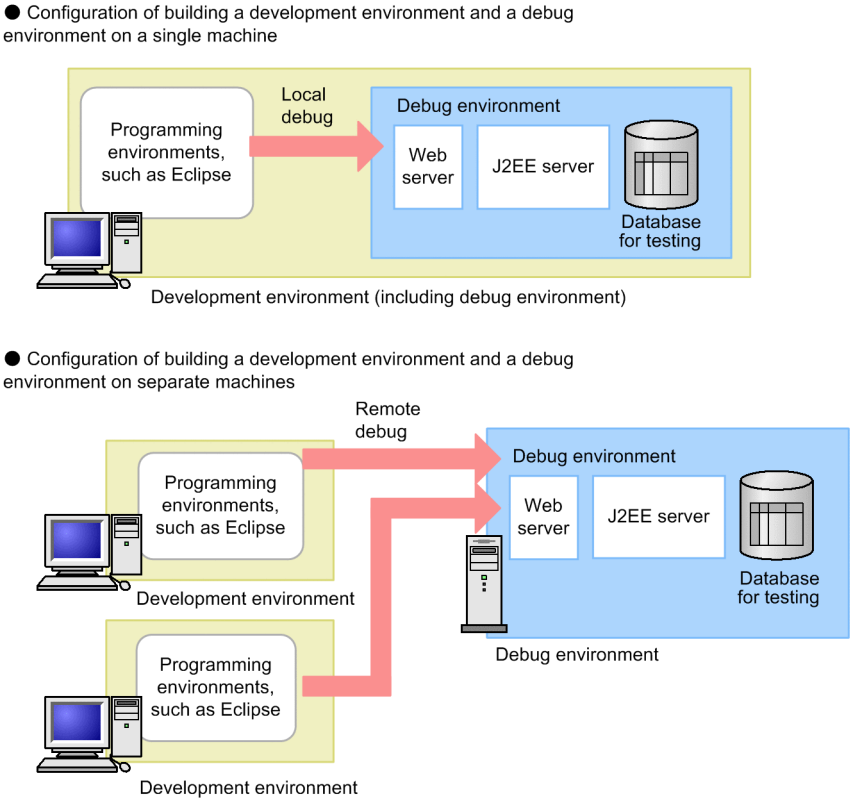
When developing an application using Eclipse as an IDE, you can use install and use the plug-in provided by Developer, which enables you to execute a series of operations during the application development, with Eclipse or the Eclipse Plug-in.

With the development environment built by Developer, you can build an environment having the same functions as that of the execution environment, as the debug environment for the developed application. You can execute the series of operations from the programming of an application up to its testing and debugging in the development environment built by Developer.

4.5.1 Building the development environment of an application

You can build a development environment with the configuration shown in the following figure in accordance with the development method of the application and the debugging method.

Figure 4–17: Configuration of the development environment



- **Configuration for building the development environment and the debug environment on one machine**
  This configuration is applicable when one developer develops an application on one machine. You can execute the local debugging.
- **Configuration for building the development environment and the debug environment on different machines**
  This configuration is applicable when an application is built by a group of developers. You can also execute the remote debugging. Note that you can use this configuration when performing testing on an OS having an execution environment other than Windows.

The following table describes the products installed in the respective cases.

Table 4–1: Products installed for each configuration of development environment

Configuration	Development environment	Debug environment
Configuration for building the development environment and the debug environment on one machine	• Developer • Service Architect	
Configuration for building the development environment and the debug environment on different machines	• Developer • Service Architect	• Application Server • Service Platform

4.5.2 Types of applications to be developed

With Application Server, you can develop J2EE applications, applications corresponding to the Web Services, and applications for asynchronous communication by using messages. This section describes an overview of developing the respective applications.

- **Developing J2EE applications and batch applications**

  You can develop J2EE applications by creating JSPs, servlets, and Enterprise Beans. You can use WTP for developing a J2EE application on Application Server. WTP enables you to execute seamless development from creating a JSP, servlet, and Enterprise Bean up to testing and debugging.

  You can also develop batch applications that run on the batch server.

- **Developing an application corresponding to a Web service**

  You can develop an application (Web service) corresponding to a Web service. With Application Server, you can generate the files and a source code (stub and skeleton) required for developing and executing SOAP Web Services.

  You can also develop an application using Web service security technology.

- **Developing an application for asynchronous communication by using messages**

  You can develop an application for asynchronous communication by using messages. You can develop an application with the JMS interface provided by Application Server.

- **Developing an application with the standard XML API**

  You can develop an application with the standard XML APIs such as JAXP, JAXB, and StAX.

- **Developing CORBA client application operating in the Java EE environment**

  You can develop a CORBA client application.#

For details on the functions that can be used in Application Server, see the *Release Notes*.

4.6 List of Application Server functions and corresponding standard specifications

This section lists the functions of the system that executes J2EE applications and that of Application Server that executes batch applications. The standard specifications corresponding to Application Server are also described.

For details on the functions that can be used with Application Server, see *3.2.5(1) Functions that can be used with Application Server*

4.6.1 List of Application Server functions

The Application Server functions that can be used differ depending on whether the system is executing J2EE applications or executing batch applications.

(1) Functions that can be used in a system executing J2EE applications

The following table describes the main functions that can be used in a system executing J2EE applications.

Table 4–2: Main functions of Application Server (For a system executing J2EE applications)

Function classification	Overview
Java language	This function supports Java SE 6.#
Web server	This functionality provides a Web server (HTTP server) for the mission critical fields and this functionality supports Secured Sockets Layer (SSL) based on Apache HTTP Server.
Web container	You can use a Web container that supports the Servlet 3.0 and JSP 2.1 versions. You can also develop and execute a Web application that uses JSF. It provides unique functions such as the in-process HTTP server function.
EJB container	This function provides an EJB container that can execute the following Enterprise Bean: • Session Bean • Entity Bean • Message-driven Bean You can also use the function of Timer Service.
Resource connection and transaction management	You can perform the resource connection and transaction management supported by the following functions: • Connection pooling • Connection sharing • Look-up by JNDI • Local transaction • 2 phase commitment • Executing distributed transaction You can also execute the connection with OpenTP1 or connection with the destination (queue or topic) compliant with the JMS specifications.
Asynchronous parallel Process threads	You can use the following functions: • Asynchronous timer process that uses TimerManager • Asynchronous timer process that uses WorkManager
XML processor	This functionality is compliant with the JAXP specifications (including each of the DOM/SAX/XSLT/XPath/XMLSchema specifications), StAX specifications, and JAXB specifications.
Web service	You can develop and execute the following Web Services:

Function classification	Overview
Web service	• SOAP Web Services compliant with the JAX-WS specifications • RESTful Web Services compliant with the JAX-RS specifications You can also use the existing SOAP Communication Infrastructure and SOAP application development support functionality.
Applying the OLTP technique	You can use the following functions: • Concurrently executed thread count control of the Web application and URL group (business logic) unit • Concurrently executed thread count control of an Enterprise Bean (use CTM)
Improving availability	You can use the following functions: • Inheriting the session information between J2EE servers • Monitoring the resource depletion • Monitoring the operation information • Output of the performance analysis trace/error analysis trace
Security management	You can use the following functions: • Integrated user management • Output of an audit log • Database audit trail linkage
Security management(SOAP Web Services)	You can use the following functions: • Ensuring the completeness and confidentiality of a SOAP message • Authentication of a SOAP message • Generation and validation of an XML signature data • Encryption and decoding of an XML signature data
Framework Library DI specifications	You can use the following functions: • JSF and JSTL • CDI • Load components provided by Application Server
Application development	You can use the following functions: • Developing an application with WTP • Debugging on a local machine and a remote machine
System setup	You can use the following functions: • Defining a system configuration in a batchandSimple setup(Smart Composer function or Setup Wizard) • Building a system using the GUI window (Operation Management Portal) • Setting all types of servers from the remote environment • Deploying an application in an archive format or deployed directory format • Deploying a resource adapter • Setting up an application • Setting up resources
System operation	You can use the following functions: • Batch start, individual start, automatic start and restart of the server processes • JSP pre-compile • Reloading a J2EE application • Issuing an eventand automatic action control by monitoring the operation information

Function classification	Overview
System operation	• Log/trace collection • Batch management of domain
Linking with the cluster software	You can operate the following cluster systems: • Switching node as 1:1 in cold standby for Application Server or Management Server • Mutual standby configuration • N:1 recovery system configuration in which one recovery-specific server is prepared • Switching node in cold standby for the host unit management model
Linking with JP1	You can use the following functions by linking with all the products of JP1: • Monitoring the error collection • Monitoring the operation performance • Automating the operation by job • Acquiring the operation information with SNMP • Collecting the audit log and uniform management of the audit log

JDK 6 is the supported version of Oracle JDK. For details on the commands that can be used with JDK 6 and how to use APIs, see the corresponding page *(http://docs.oracle.com/javase/6/docs/index.html)*.

(2) Functions that can be used in a system executing batch applications

The following table describes the functions that can be used by a system executing batch applications.

Table 4–3: Main functions of Application Server (for a system executing batch applications)

Function classification	Overview
Java language	Supports Java SE 6.#
Batch server	Provides a batch server that can execute the following functions: • Executing a Java application on which a batch process is executed. • Controlling the execution of the full garbage collection
Resource connection and transaction management	You can execute the resource connection and transaction management supported by the following functions: • Connection pooling • Connection sharing • Look-up by JNDI • Local transaction
XML processor	This functionality is compliant with the JAXP specifications (including each of the DOM/SAX/XSLT/XPath/XMLSchema specifications), StAX specifications, and JAXB specifications.
Improving availability	You can use the following functions: • Monitoring the resource depletion • Monitoring the operation information • Output of the performance analysis trace/error analysis trace
Security management	You can use the following functions: • Output of the audit log • Database audit trail linkage
Application development	You can use the following functions: • Developing an application with WTP

Function classification	Overview
Application development	• Debugging on a local machine and a remote machine
Building a system	You can use the following functions: • Defining and simple building of a system configuration in a batch (Smart Composer Function) • Setting all types of servers from the remote environment • Deploying a resource adapter • Setting the resources
System operations	You can use the following functions: • Batch start, individual start, automatic start and restart of the server processes • Event issue and Automatic action controlby monitoring the operation information • Log/trace collection • Batch domain management
Linking with the cluster software	You can operate the following cluster systems: • Switching node as 1:1 in cold standby for Application Server • Mutual standby configuration • Switching node in cold standby for the host unit management model
Linking with JP1	You can use the following functions, by linking with all the products of JP1: • Monitoring the error collection • Monitoring the operation performance • Automating the operations by job • Acquiring the operation information with SNMP • Collecting the audit log and uniform management of the audit log

JDK 6 is the supported version of Oracle JDK. For details on the commands that can be used in JDK 6 and how to use APIs, see the corresponding page *(http://docs.oracle.com/javase/6/docs/index.html)*.

4.6.2 Standard specifications supported by Application Server

This section describes the standard specifications supported by Application Server. This section also describes the manuals that explain the details and the points to be considered when using Application Server.

(1) Standard specifications of Java EE

Table 4–4: Standard specifications of Java EE

Specifications	Remarks	Important reference location manuals#
Servlet 2.3/Servlet 2.4/Servlet 2.5/ Servlet 3.0	From the functions specified in the Servlet 3.0 specifications, for the functions that can be used with Application Server, see section *6.1 Support range of the functionalities that are added or changed in Servlet specifications and JSP specifications* in the *Web Container Functionality Guide*.	*Web Container Functionality Guide*
JSF 2.1	--	
JSP 1.2/JSP 2.0/JSP 2.1	You cannot use JSP 1.1.	
JSP Debugging 1.0	--	
EJB 2.0	--	*EJB Container Functionality Guide*

Specifications	Remarks	Important reference location manuals#
• Message-driven Bean • Local interface • CMP 1.1 • CMP 2.0	--	*EJB Container Functionality Guide*
EJB 2.1	Among the CMP functions, you cannot use the Web service linkage function that uses expansion part of EJB2.1 and *service-ref* tag.	
EJB 3.0 (Session Bean)	--	
EJB 3.1	--	
Bean Validation 1.0	--	*Common Container Functionality Guide*
CDI 1.0	--	
Common Annotation 1.0	--	
JDBC 2.0 core/JDBC 2.0 option package	--	
JDBC 3.0	The JDBC driver used for connection must support the functions specified in the JDBC 3.0 or JDBC 4.0 specifications.	
JDBC 4.0		
JMS 1.0.2	When using Reliable Messaging or TP1/Message Queue - Access, JMS1.0.2 is presumed. The restrictions are set on the part of the functions including Topic.	
JMS 1.1	The conditions for using JMS1.1 are as follows: • The JMS provider must be compatible with JMS1.1 • The Message-driven Bean must be compatible with EJB2.1 (when using the Message-driven Bean) • The resource adapter to be used must be compatible with Connector 1.5	
Connector 1.0 (JCA 1.0)	--	
Connector 1.5 (JCA 1.5)	--	
JTA 1.0.1 • local • global	You can use a local transaction (local) when LocalTransaction is specified in the transaction-support of the DD (ra.xml) of the resource adapter and JavaVM is not called remotely during the business logic. When the light transaction is activated, the global transaction (global) cannot be used. For details on the light transaction, see section *3.14.5 Light Transaction* in the *Common Container Functionality Guide*.	
JTA 1.1	--	
JPA 1.0	--	
JavaMail 1.2	Use the SMTP and SMTPS protocols for sending (outgoing) emails, and POP3 for receiving (incoming) emails.	
JavaMail 1.3		
JavaMail 1.4		

Legend:

--: Not applicable

#:

The term *Application Server* is omitted from the manual names.

(2) Standard specifications related to XML and Web Services

- JAX-RS 1.1
- JAX-WS 2.2
- JAXB 2.2
- JAXP 1.4 (including StAX)
- JAXR 1.0
- SAAJ 1.2 and SAAJ 1.3
- SOAP 1.1 and SOAP 1.2
- WSDL 1.1
- WS-RM 1.1 and WS-RM 1.2
- UDDI 2.0 and UDDI 3.0
- WS-I Basic Profile 1.1
- WS-Security 1.1
- WS-Reliability 1.1
- XML-Signature Syntax and Processing
- XML Encryption Syntax and Processing

For details on the items to be considered related to the XML standard specifications, see the *XML Processor User Guide*. For details on the items to be considered related to the Web Services of Application Server, see the *Application Server Web Service Development Guide*, the *Application Server SOAP Application Development Guide,* the *Application Server Web Service Security Users Guide* and the *Application Server XML Security - Core User Guide*.

(3) Standard specifications related to SSL

- SSL version 2 and SSL version 3
- TLS version 1, TLS version 1.1, and TLS version 1.2

For details on the items to be considered related to the SSL standard specifications, see the *Application Server HTTP Server User Guide* and the *Application Server Security Management Guide*.

(4) Standard specifications related to OMG distributed object

- CORBA 2.5
- CORBA Object Transaction Service 1.3

For details on the items to be considered related to the standard specifications of the OMG distributed objects, see the *Borland(R) Enterprise Server VisiBroker(R) Developer Guide*, the manual *Borland(R) Enterprise Server VisiBroker(R) Programmers Reference*, and the *Release Notes*.

5 Objective-wise Introduction to Functions of Application Server

This chapter explains the Application Server functions that can be used to achieve each objective of your system.

Based on the contents of this chapter, consider which Application Server functions you want to use so that you can achieve your system objectives. For detailed explanations, see each manual described as a reference location in this chapter.

5.1 Building a system

You can use various methods to build a system on which Application Server runs. To build the system, use the method that most suits your objectives.

The three cases described here are as follows:

- When you want to operate Application Server immediately
- When you want to proceed thoroughly from system design
- When you want the system to also support a virtual environment

5.1.1 When you want to operate Application Server immediately

This case applies to situations where you want to start running the system with the standard configuration. For example, first because you want to execute the required tuning while running applications or you want to immediately try running the Java EE compliant applications that were running on another system.

In this case, first see the *Application Server System Setup and Operation Guide* and check whether the expected system can be configured with the Setup Wizard.

With Application Server, the following patterns are assumed as the standard system configurations that can be configured with the Setup Wizard:

- A system in which one Web server and one J2EE server are set up on the same host
- A system in which hosts with the same configuration are set up on multiple machines (Load distribution function is used for sorting the requests)
- A system in which multiple Web servers and J2EE servers are set up on the same host
- A system in which multiple resources are accessed from the same J2EE server

You can change the settings of these system configurations with the *management portal*, as and when required.

5.1.2 When you want to proceed thoroughly from system design

As a result of referencing the *Application Server System Setup and Operation Guide*, if you observe that the objective system is not a target system to be configured with the Setup Wizard, or you want to work thoroughly from the system design by creating a definition file while validating the parameters, build the system with the functions of the management portal or Smart Composer.

The procedure for building a system with these functions is explained in the *Application Server System Setup and Operation Guide* and the *Application Server Management Portal User Guide*. The concept of parameter setting used in tuning is explained in the *Application Server System Design Guide*. Details of parameters are explained in the *Application Server Definition Reference Guide* and the *Application Server Application and Resource Definition Reference Guide*.

5.1.3 When you want the system to also support a virtual environment

Application Server provides the following functions to support a virtual environment:

- Batch setup of business systems (Application Server) on multiple virtual servers
- Batch start and batch stop of business (applications) in Application Server on multiple virtual servers
- Scaling out and scaling in the virtual server according to the business size
- Managing the account of a user who can execute the virtual server manager

For details on building and operating a system in a virtual environment with the functions mentioned above, see the *Application Server Virtual System Setup and Operation Guide*.

5.2 Improving the system performance

To improve the system performance, first check the procedure for *Performance tuning* described in the *Application Server System Design Guide*. This manual describes the functions corresponding to the concepts for improving the system performance, such as the concepts of optimizing the number of concurrent executions and setting a timeout.

In addition to this, investigate the usage of the following functionality:

- **In-process HTTP server**

  This function is valid for a system that executes a Web application. You can improve the system performance by executing the Web server functions in the same process as the J2EE server. For details on this function, see the *Application Server Web Container Functionality Guide*.

- **Functionality for performance tuning in resource connection and transaction management**

  This function improves the system performance by pooling and re-using the connection and statements, and by optimizing the process of the local transaction. This function is described in the *Application Server Common Container Functionality Guide.*

If you want to further improve the performance using functions other than those mentioned above, check the description of *Relation of system objectives and functions* in the manuals starting with the *Application Server Functionality Guide* and study the usage of the functions relevant to *Performance*.

5.3 Improving the system reliability (availability and fault tolerance)

Use the following functions to improve the system reliability with respect to availability (stable operations) and fault tolerance (disaster tolerance):

- **Session failover functionality**

  This function is valid for a system that executes a Web application.

  You can improve the availability/fault tolerance with redundancy execution by storing the session information in a database or an EADs server, and by inheriting the session information when an error occurs. For details on this function, see the *Application Server Expansion Guide*.

- **Function to deter full garbage collection (Explicit management heap function)**

  You can deter the full garbage collection by storing the object that is the cause of full garbage collection in a unique area other than the Java heap, and thus improve the availability. Details on this function are described in the *Application Server Expansion Guide*.

- **Function for fault tolerance in the resource connection and the transaction management**

  You can improve the fault tolerance by detecting an error in the connection, retrying to obtain the connection, and by automatically closing the connection. Details on this function are described in the *Application Server Common Container Functionality Guide*.

To improve the system reliability by using functions other than those mentioned above, check the description of *Relation of System Objectives and Functions* in the manuals starting with the *Application Server Functionality Guide* and use the functions relevant to *Reliability*.

5.4 Improving the system reliability (Security)

Use the following functions to improve the system reliability (security):

- **Function to prevent access by an invalid user (Authentication function)**
- **Function to ensure the security of a communication path (Encryption function)**
- **Function to prevent the execution of an invalid process**

You can also ensure the system security by investigating the system configuration and the operation method.

For details on these functions, see the *Application Server Security Management Guide*.

5.5 Operating the system efficiently

Use the following functions to operate the system efficiently:

- **Efficiency of regular operations**

  The systems built with the Setup Wizard, management portal, or Smart Composer Function can execute regular operations like starting and stopping a server in a batch, by using the management portal or the Smart Composer Function.

  You can also automate these operations by linking the system with JP1.

- **Tuning and automation of process by monitoring the operation information**

  You can perform tuning of the resources such as memory, for optimum operation of a system, by checking the operation information of a server output by Application Server. In addition to this, you can set a threshold value corresponding to the operation information so as to receive an alert when the usage rate of the resource exceeds a certain value, even if the user is not monitoring. You can also perform the centralized monitoring of the operation information, and automate the process when an alert is output, by linking the system with JP1.

All these functions are described in the *Application Server Operation, Monitoring, and Linkage Guide*. If you want to operate the system more efficiently by using the functions other than those mentioned above, check the description of *Relation of System Objectives and Functions* in the manuals starting with the*Application Server Functionality Guide* and use the functions relevant to *Operations and Maintenance*.

5.6 Troubleshooting

If trouble occurs in a system on which Application Server is operating, you must first determine the cause of the trouble and then perform troubleshooting by referencing the message logs or sending a query by gathering the required information.

Application Server has a function to output a detailed log for determining the cause of the trouble.

If trouble occurs in a system, first check the *Procedure for troubleshooting* described in the *Application Server Maintenance and Migration Guide.* This manual describes the method to remove the trouble based on the log and the typical troubleshooting method. This manual also describes the method to output each type of log used in troubleshooting, the output destination, and the output contents.

5.7 Developing an application

Investigate the usage of the following functions to develop an application:

- **Development Environment Instant Setup function (When developing an application using Developer)**

  You can use this function to develop an application using the Developer functions. You can build an environment to be used for testing and debugging, in accordance with the wizard. For details on this function, see the *Application Server Application Development Guide*.

- **Eclipse setup functionality (When developing an application using Developer)**

  You can build an Eclipse environment wherein you can execute the operations from coding to debugging and testing. For details on this function, see the *Application Server Application Development Guide*.

Also, when developing an application compliant to the standard specifications of Java EE, to check the implementation of Application Server, see the manuals described in section *4.6.2 Standard specifications supported by Application Server*.

If you want to develop an application using a unique API provided by Application Server, see the *Application Server API Reference Guide*.

5.8 Checking details of standard specifications supported by Application Server

Regarding the standard specifications supported by Application Server, to confirm the details and precautions on implementation in Application Server, see section *4.6.2 Standard specifications supported by Application Server*.

6 Linking with Other Products

This chapter explains the linking of Application Server with other products.

You can link, build and operate systems by linking Application Server with databases, JP1, and cluster software.

6.1 Linking with the database

Application Server can connect to the following databases:

- HiRDB Version 9
- HiRDB Version 8
- XDM/RD E2
- Oracle 11g R1
- Oracle 11g R2
- SQL Server 2012
- SQL Server 2008
- SQL Server 2005

The functions that you can use are different for every database. Note that you can use SQL Server only for Windows.

This section describes the databases, to which you can connect from the J2EE server and the JDBC driver you use for connecting. The JDBC driver you use for connecting is different for every database. You can use two connection methods for the database connection, such as the method of connecting to a database by using the JDBC interface and the method of connecting to a database by using the JMS interface in addition to the JDBC interface.

The following table describes the relationship of the databases that can be connected from the J2EE server, the JDBC driver, and the interface used for connection.

Table 6–1: Databases that can be connected from the J2EE server

Database	JDBC driver	JDBC interface		JMS interface and JDBC interface	
		Local transaction	Global transaction	Local transaction	Global transaction
HiRDB Version 9	HiRDB Type 4 JDBC Driver	Y	Y	Y	Y
HiRDB Version 8	HiRDB Type 4 JDBC Driver	Y	Y	Y	Y
XDM/RD E2	HiRDB Type 4 JDBC Driver	Y	--	--	--
Oracle 11g R1 11.1.0	Oracle JDBC Thin Driver	Y	Y	Y	Y
Oracle 11g R2 11.2.0	Oracle JDBC Thin Driver	Y	Y	Y	Y
Oracle 12c 12.1.0	Oracle JDBC Thin Driver	Y	Y	Y	Y
SQL Server 2012	JDBC Driver 4.0 for SQL Server	Y	--	--	--
SQL Server 2008	SQL Server JDBC Driver 3.0	Y	--	--	--
SQL Server 2008 R2	SQL Server JDBC Driver 3.0 or JDBC Driver 4.0 for SQL Server	Y	--	--	--
SQL Server 2005	SQL Server JDBC Driver 3.0	Y	--	--	--

Legend:

Y: Can be used.

--: Cannot be used.

Application Server provides a resource adapter for connecting to these databases. For details on the databases that can be connected and the functionalities that can be used, see section *3.3 Resource connections* in the *Application Server Common Container Functionality Guide*.

6.2 Linking with JP1

JP1 is an integrated operation software from Hitachi that implements the integrated operation management of a complex business system. You can build and operate a system built with Application Server, as a system having high operability, by linking the system with JP1.

With JP1, you can perform integrated management such as the job management, availability management, and the network management of various systems. You can perform operations such as error monitoring and performance monitoring, and automate the starting and stopping of a server or an application by integrating the system built on Application Server with other business systems.

The following table describes the functionalities that you can execute and the products that you can use by linking the system with JP1.

Table 6–2: Functionalities that you can execute and the products that you can use by linking the system with JP1

Functionalities	Overview	Products to be used
Centralized error monitoring	You can now perform centralized error monitoring for the entire system.	JP1/IM
Performance monitoring	You can now perform performance monitoring for the entire system.	JP1/PFM
Automating job based operations	You can now use the JP1/AJS jobs to automate the starting and stopping of the server or the process managed with Application Server.	JP1/AJS
Acquiring the operation information with SNMP	You can now acquire the operation information with SNMP.	JP1/Cm2/ESA
Collection and unified management of the monitoring log	You can now collect and perform unified management of the monitoring log output by the products, such as Application Server in a system.	JP1/Audit Management - Manager

For details on linking with JP1, see *chapter 12. Operation of the Systems Linked with JP1* in the *Application Server Operation, Monitoring, and Linkage Guide*. For details on obtaining the operation information by using SNMP, see chapter *8. Statistical Output using Management Commands* in the *Application Server Operation, Monitoring, and Linkage Guide*.

For details on the products to be used, see the respective product manuals.

6.3 Linking with the cluster software

You can attain more availability of the operations, by linking the system built on Application Server with cluster software.

When you operate a system built on Application Server with cluster software, you can switch to a standby Application Server, if an error occurs in the running Application Server, and you can use the standby recovery server to execute the recovery of the erroneous Application Server. Also, if an error occurs in the operation management server, you can switch to the standby server. With this, you can reduce the non-operational time caused by the error on the server and also prevent the interruption of business processes to the maximum possible extent.

The following table shows the list of cluster software that can be linked to Application Server.

Table 6–3: List of cluster software that can be linked

OS on which Application Server operates	Windows Server Failover Cluster	HA monitor
Windows#	Y	--
AIX	--	Y
HP-UX	--	Y
Linux	--	Y
Solaris	--	--

Legend:

Y: Can be used.

--: Cannot be used.

The following OSs are compatible:

- Windows Server 2012 Standard
- Windows Server 2012 Datacenter
- Windows Server 2008 Enterprise

For details on the functionalities that you can execute by linking with the cluster software, see chapter *16. Linking with the Cluster Software* in the *Application Server Operation, Monitoring, and Linkage Guide*.

7 Overview of SOA

This chapter presents an overview of SOA, the concept on which the service platform is based.

7.1 What is SOA?

Business environments are always dynamically transforming due to factors such as globalization, deregulation, M&A, and TOB. Consequently, enterprises must respond quickly and flexibly to environment transformations, and therefore, the need to change their information systems in a timely manner has increased. At the same time, however, the number of enterprises having IT infrastructures that cannot quickly adapt to business environment transformations is increasing.

In this way, IT has become more closely related to business, and therefore, information systems require an architecture that can quickly and flexibly adapt to transformations in business environments. Here enters *Service Oriented Architecture (SOA)*.

SOA is the technique and concept of creating business-required functionality as reusable services, and then combining the services to build a system. Moreover, to realize business, SOA uses a 'business process' that regulates the type and order of services to be invoked , thus making it easier to add, change, or re-organize services.

The following are the objectives of SOA:

- **Re-using services**
  You can re-use services to avoid overlapping of development, to increase productivity, and to localize each development.
- **Automating business processes**
  Business processes can be automated, thereby decreasing the intervention of users, eliminating the incorporation of mistakes and corruption, and reducing turnaround time.
- **Making business processes visible**
  Business processes can be made visible, so you can analyze conditions in a timely manner and improve continuous processes.

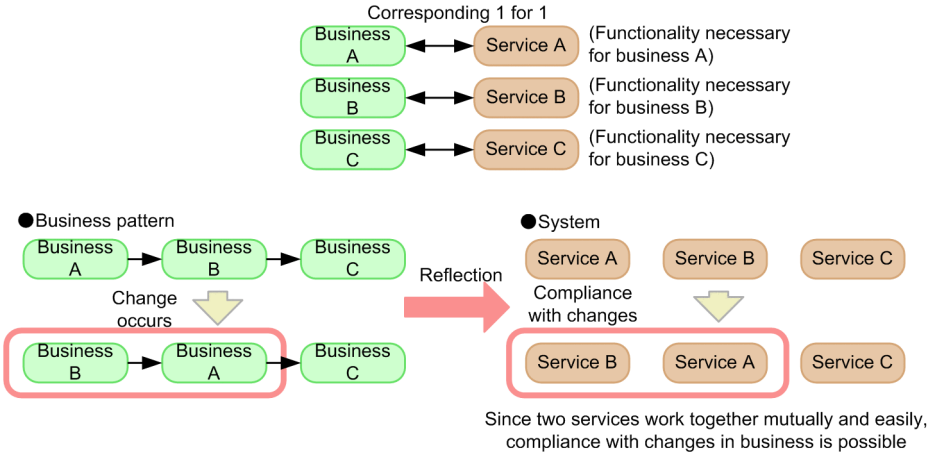
7.2 Objectives and Merits of SOA

SOA has the following three objectives and merits:

7.2.1 Adapting a System to Changes in Business

To implement a system that can support changes in business, you build a business system by combining reusable software components called services that are in a one-to-one relationship with business. Conventionally, you build a system that consists of application units; however, with SOA, you build a system that consists of services for business units. By doing this, you can quickly identify the scope of modifying a system based on the changes in business and then change or expand the system according to business requirements. Figure 1-1 shows the correspondence between businesses and services.

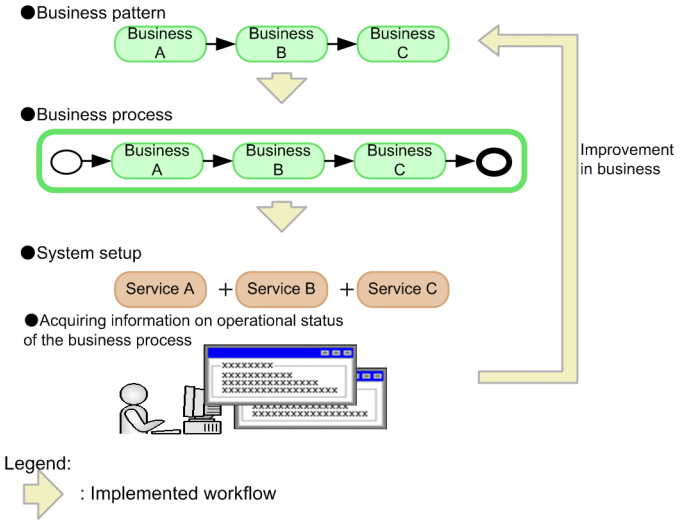
Figure 7–1: Correspondence between businesses and services



7.2.2 Optimizing and Improving the Efficiency of a Business

To implement a system that can optimize and improve the efficiency of a business, apply *business process management* (BPM) a method for building and operating a system. By applying business process management, you can combine various target services based on the flow of the business, and automate them as business processes. As a result, the conventional method of linking systems manually can be automated and expedited. Additionally, the operation status of business processes can be gathered in a unified format and the business processes can be collectively managed at one site. A business can be visualized, so continual support of business improvement is enabled. Figure 1-2 shows the management and improvement of business processes.

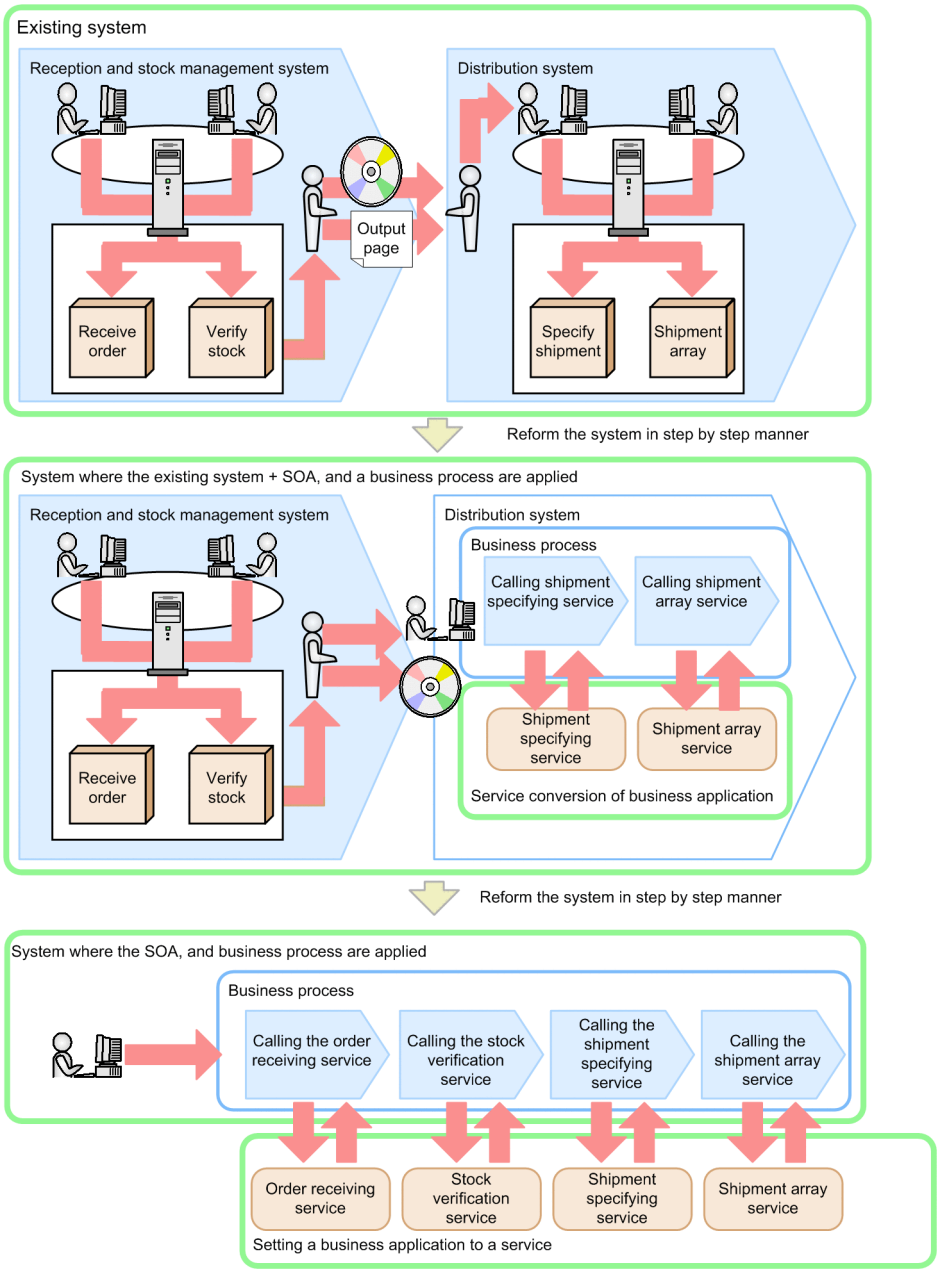
Figure 7–2: Management and improvement of business processes



7.2.3 Step-wise Reformation of a System

You start the optimization of a business system by introducing services and BPM from the parts with relatively higher priority, and reform the system gradually. To optimize the overall system, carry out a cross section examination of the businesses and aim at a planned reformation by setting an overall optimum system configuration, which is the final goal. While using the existing system, by adding the missing functions and systems, and re-creating the old parts, you can gradually migrate to a new system. Figure 1-3 shows the introduction of services and business process management.

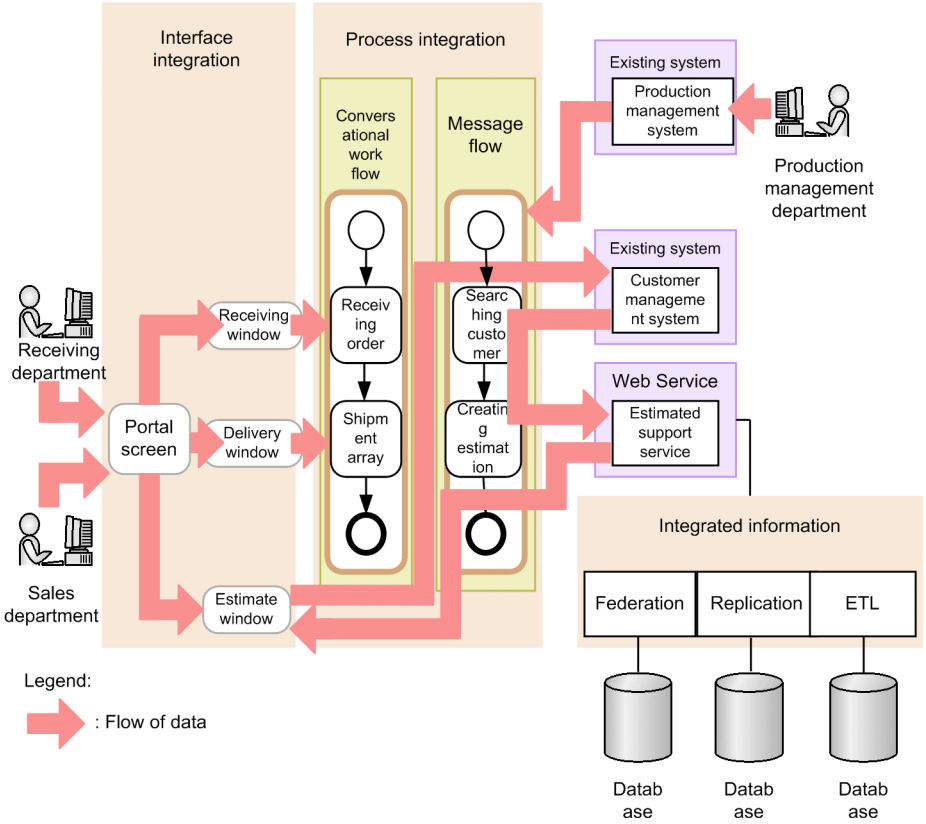
Figure 7–3: Introduction of services and business process management



7.3 Implementing a System Where SOA Is Applied

Figure 1-4 shows an example of implementing a system where SOA is applied.

Figure 7–4: Example of implementing a system where SOA is applied



A system using SOA can be realized by using the Service Platform to integrate interfaces and processes integration.

- Interface integration

  By integrating screen interfaces with focus on multiple businesses assigned to a user, an easy-to-use operation environment can be implemented. Additionally, the productivity of the user can be improved through intuitive operations where services are linked on the screen.

- Process integration

  In SOA, a new system can be built quickly by combining services flexibly; however, the focus is on process integration that integrates services by processes. Apart from the message flow that calls services automatically, an interactive workflow that integrates businesses involving human beings can be used.

- Information integration

  With the help of the following linkage functions, the data distributed within a system can be integrated and unified. Therefore, consistent data can be shared across various services. The data distributed within the system can be used whenever required.

  - Federation

    You can extract required data from the masters that different databases manage, and integrate and reference the data in a virtual table. This can be used in a business where the access frequency is not very high and the volume of data to be acquired is less.

  - Replication

    Create a replica by copying the database. This can be used in a business where the access frequency and volume of the data to be acquired are high.

  - ETL (Extract Transform Loading)

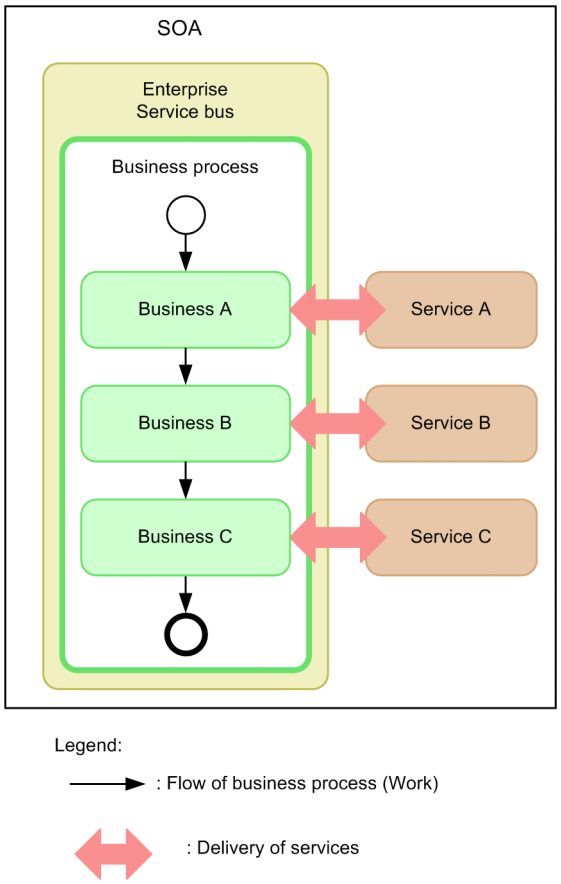
Data is extracted from multiple databases, processed and saved in a new database. This supports the management of master data that is centrally managed by integration of data used in multiple systems.

As a result, you can easily use various services.

7.4 Elements Constituting SOA

SOA applies the concept of services as well as business process management. The relationship between the services and business process in SOA is shown in Figure 1-5.

Figure 7–5: Relationship between the services and business process in SOA



You call and use the required services based on the business process that follows the flow of businesses. The linkage between the service user, business process, and services is implemented through an enterprise service bus (ESB).

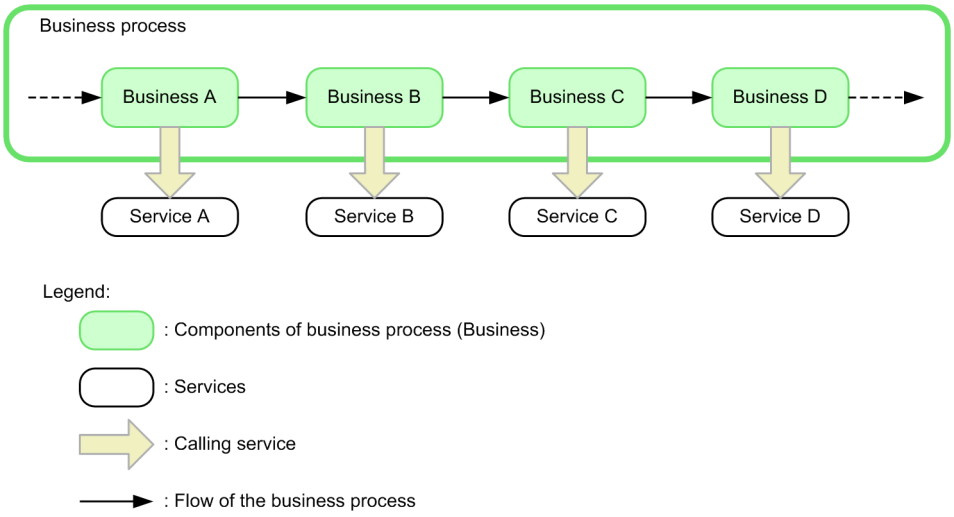
The enterprise service bus that combines and executes a service, constitutes the core part of SOA.

The business process and services of SOA are explained below:

7.4.1 Business Processes

A flow of operations that implements a series of business processing is called a *business process*. The elements constituting a business process are called businesses (or services, since they are in a one-to-one relationship with businesses). Additionally, a business process that consists of multiple services is called a complex service. In a business process, multiple businesses are connected smoothly, and are linked in such a way that they can be inter-changed at any point. The following figure shows an overview of a business process.

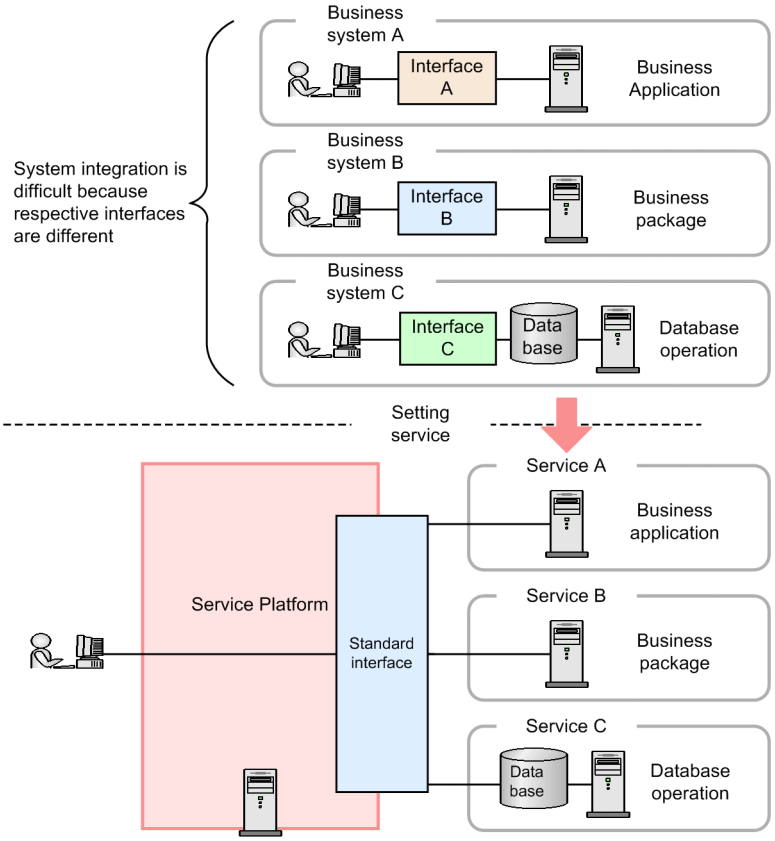
Figure 7–6: Overview of a business process



7.4.2 Services

The *services* used in SOA imply reusable software components with a one-to-one relationship with businesses. By handling a business application as a service (converting them into a service), the systems that have been dispersed can be integrated as a system where SOA is applied. Figure 1-7 shows the conversion of business systems into services by adopting the concept of SOA, as well as the system integration.

Figure 7–7: Conversion of business system into services and system integration



Components such as business applications, business packages, and database operations that are distributed over a network sometimes use different interfaces. Because of this, system integration used to be difficult. With the service platform, by converting the business applications, business packages, and database operations into services, and by using a common standard interface, the distributed business systems can be integrated.

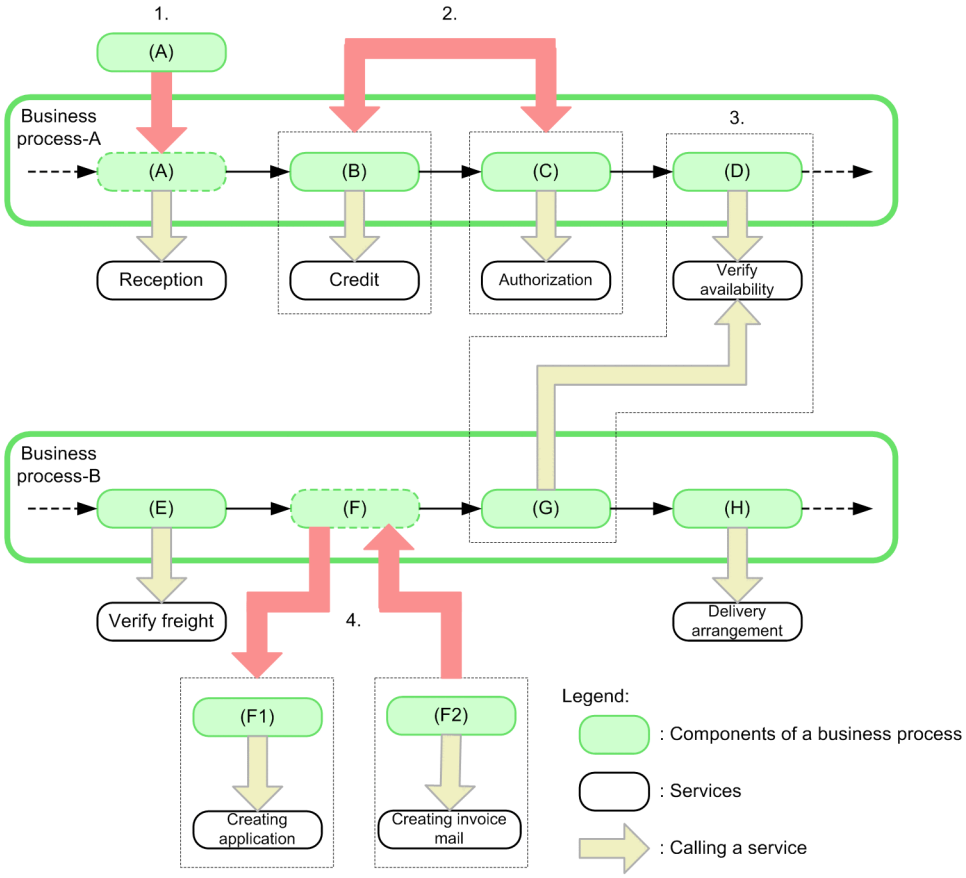
Since services have an isolation level, one of their characteristics is that if a failure occurs in a certain service or if a service is added or modified after a business process is re-evaluated, such an occurrence has little effect on other services. In addition, existing services can be reused.

(1) Relationship between services and a business process

Since the services constituting a business process are inter-linked loosely, even if a failure occurs, or even if the services are replaced when there is a need to change the system, the scope of impact is minimum. As a result, you can change a business process by quickly changing the business procedures.

Also, a service that is being used in a business process can be reused in another business process. The next figure shows an example of adding and modifying services that make up a business process.

Figure 7–8: Example of adding and modifying services that make up a business process



1. Service calling can be added newly.
2. Order of service calling can be changed.
3. Similar service can be called from different BP.
4. Component of business process can be replaced and calling process can be changed.

(2) Service interfaces

A service has an interface for providing functions to a business. The service is linked with the business process through this interface. An *interface* is a connecting point of a service and defines the functions that the service can provide, as well as the required I/O data. The components provide implementation for the functions that a service

provides, and the interface acts as the entrance. A *component* is a part of a service that processes the requested contents. The component consists of online applications or interactive applications that perform synchronous and asynchronous processing, and methods such as new development, reuse of existing systems, and introduction of packages can be used. The following figure shows the structure of a service.

Figure 7–9: Structure of a service

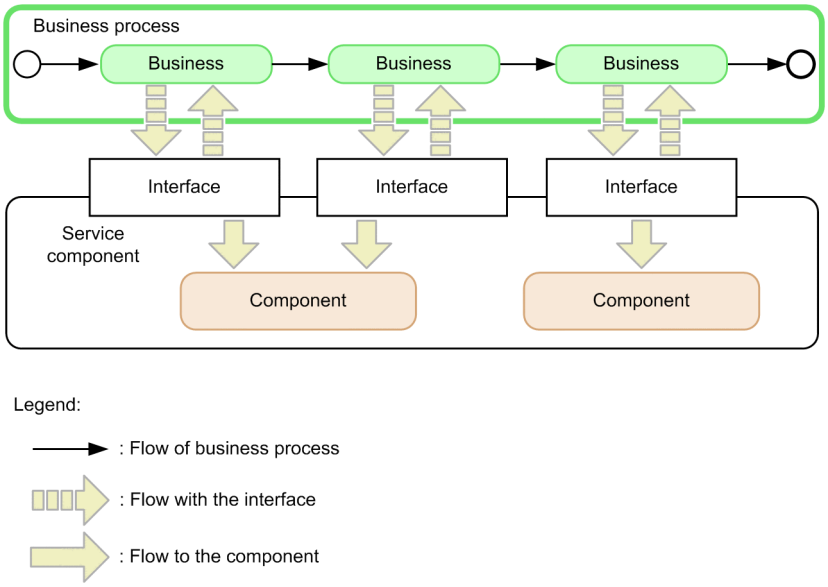


Figure 1-9 shows how each business uses components corresponding to the request, through an interface.

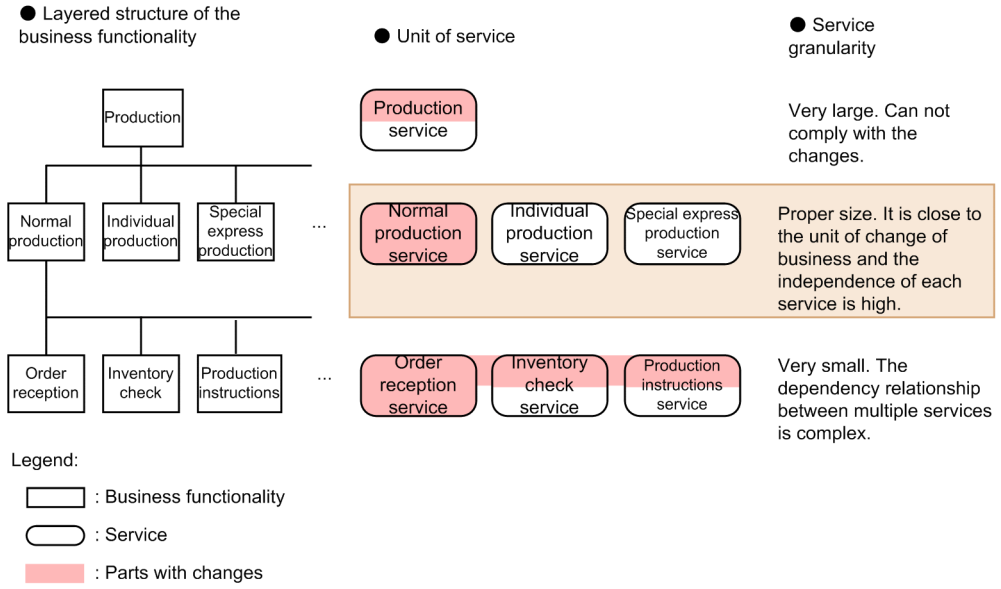
(3) Extraction and granularity of services

In a system where SOA is applied, everything right from development up to operation is considered as a service. As a result, the performance and flexibility of the system varies depending on how the service is divided and the size of the service. This method of dividing a service is called *service extraction*. When you extract a service, the business contents are examined and the functions to be provided by the service are decided. Additionally, the scope (size) of the functions that a service provides is called *granularity*. When developing a system where SOA is applied, whether a system can easily support the changes in business environment is decided depending on the granularity.

The service granularity is decided as a whole from the business system and business flow, and the dependency of the data. Appropriate granularity for a service depends on the business-related changes that occur. When each service is close to the unit in which the business change occurs and the level of independence also is high, the business change can be absorbed through a change brought about by a re-arrangement of services. Such a unit becomes appropriate as the service granularity.

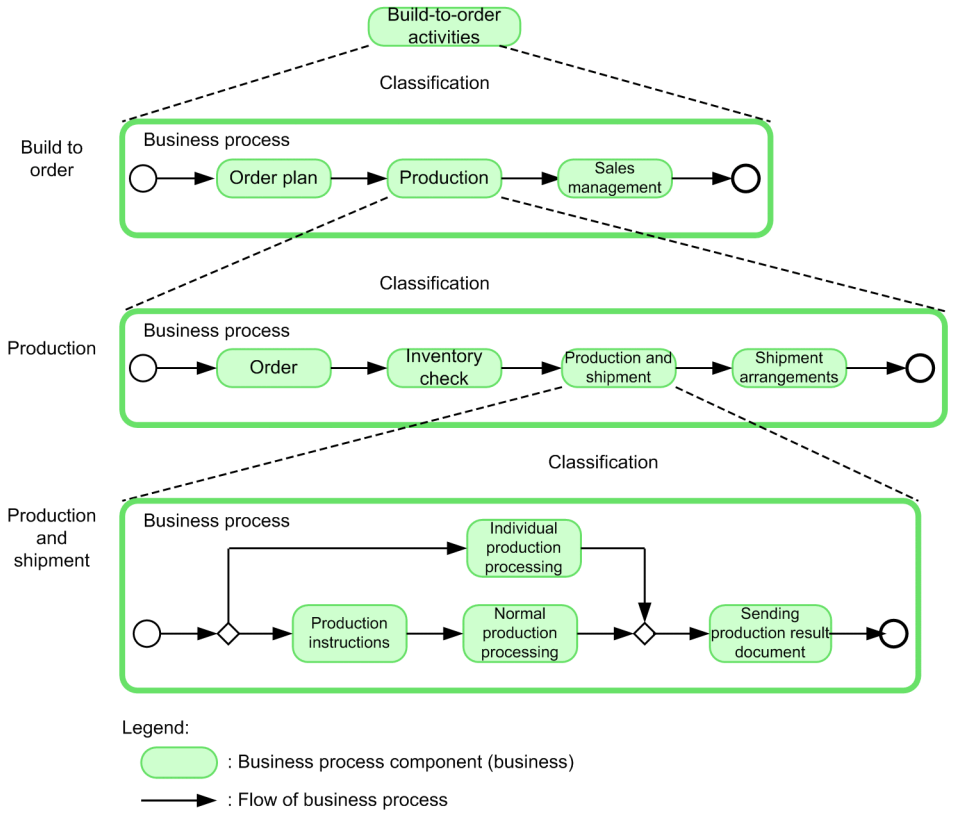
If the service is too big for a business change that occurs, it will not be possible to comply with the change by re-arranging the services. If the service is too small, the dependency between the services will become complex and making changes in a single service may not be sufficient to comply with the change in business. Figure 1-10 shows the relationship between the services and service granularity.

Figure 7–10: Relationship between the services and service granularity



However, depending on the contents of the business change, the upper-layer services may support easily to comply with the change. In such a case, it is possible to comply with the changes by arranging the business process in layers. The following figure shows an example of arrangement of the business process and businesses in layers.

Figure 7–11: Example of arrangement of the business process and businesses in layers



8 Overview of the Service Platform

This chapter presents an overview of the service platform.

8.1 Introduction to the Service Platform

This section explains the purpose and positioning of a service platform.

The service platform is a product that serves as the development and operating infrastructure in systems that implement a Service Oriented Architecture (SOA). With this product, you can perform operations right from building and executing processes up to connecting the services in an integrated development and operating environment. Therefore, by tapping the merits of SOA, you can flexibly integrate services and rapidly build and execute a new system. The pivotal point is process integration that integrates services with processes. The service platform implements process integration. An example of implementing a system using the service platform is shown in Figure 2-1.

Figure 8–1: Example of implementing a system using the service platform

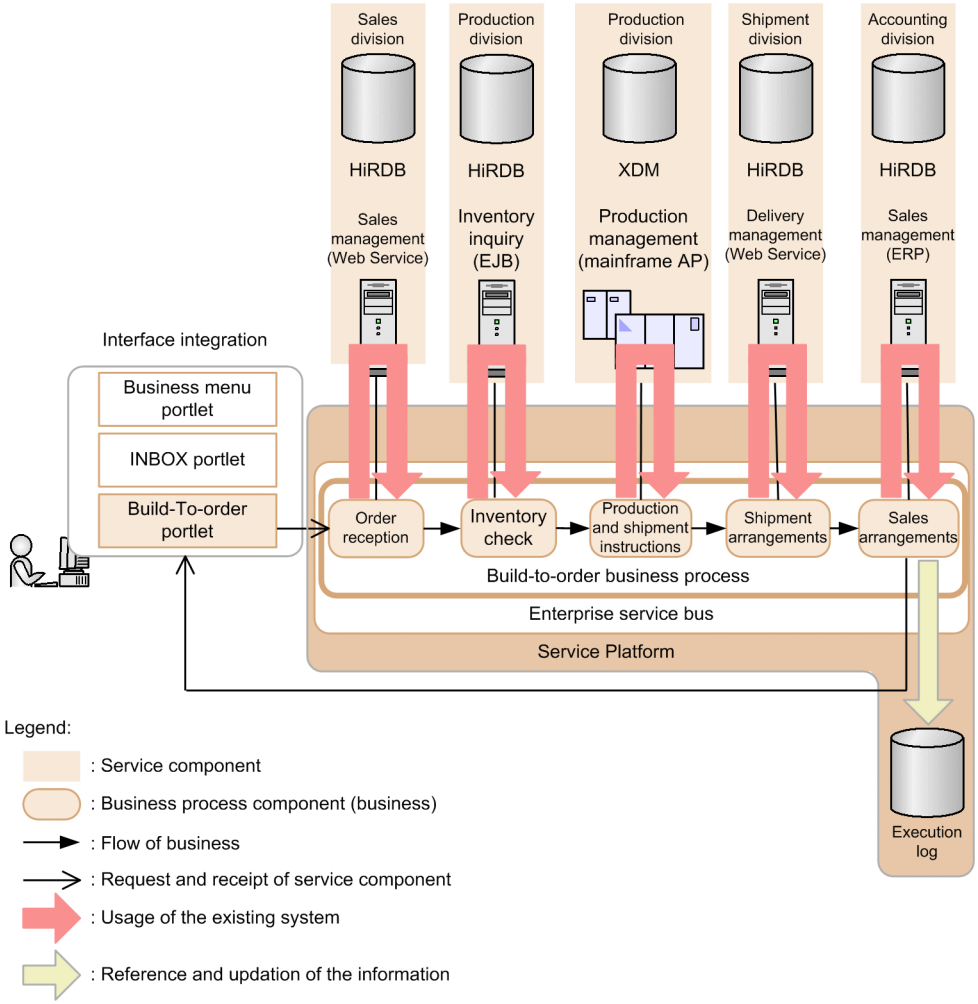


Figure 2-1 shows an example of applying SOA in a product flow system. Services can be invoked automatically along with the flow of business. Figure 2-1 contains the following merits:

- It becomes easy to take action when business opens in phases.
- A business process corresponding to the actual business can be implemented.
- The stock and production progress state can be adequately comprehended and a prompt delivery response can be sent.
- The lead-time can be reduced.

The service platform consists of the Service Platform and Service Architect. In addition to the functionality for the execution environment provided by Application Server, Service Platform includes functionality for integrating

services in the execution and operating environments. The Service Platform contains the enterprise service bus function that is the focal point of SOA, and can build a system in response to the strategic changes to be executed, by freely combining services. With the Service Platform you can also set up a highly reliable system by freely combining the services taken from the existing system and services provided from an external source.

The Service Architect corresponds to the development environment, and has a function for integrating services as well as the functions of the development environment in Developer Professional. The Service Architect can use the definition tools required for process integration, such as business process definitions, data transformation definitions, and service adapter definitions, as plug-in of Eclipse. Process integration can be achieved through a series of operations on Eclipse, ranging from business processes to connection of services.

Among the techniques for developing a system in which SOA is applied, the Service Architect supports the operations right from detailed designing of business processes including interfaces up to the implementation and testing. You can use the functions of service platform to design and implement components.

As a result, you can integrate the services already running in the execution environment of the application and services running in environments other than service platform, and provide new services to users.

8.2 Service Platform Features

This section describes the features of a service platform.

8.2.1 Support for System Development in a Visual Environment

The service platform allows you to use windows to visually develop various definitions in system development. These definitions include definitions for service adapters (used for calling services), for business processes, and for data transformation methods, and other definitions necessary for a system.

As an example, suppose that the structure of the data (request message) used to request execution of a service component differs from the structure of data used by the actual service component. In this case, you need to define how to transform the data of individual elements in the structure. The service platform provides a window in which you can bind data elements (mapping) by using a line when you define a transformation method between data items with different structures.

As another example, suppose you need to define a business process by using BPEL, which is a work flow description language based on XML. The service platform allows you to define a business process by placing, in a window, the elements (activities) that compose a business process, and binding them. You can create BPEL from a business process defined in this way.

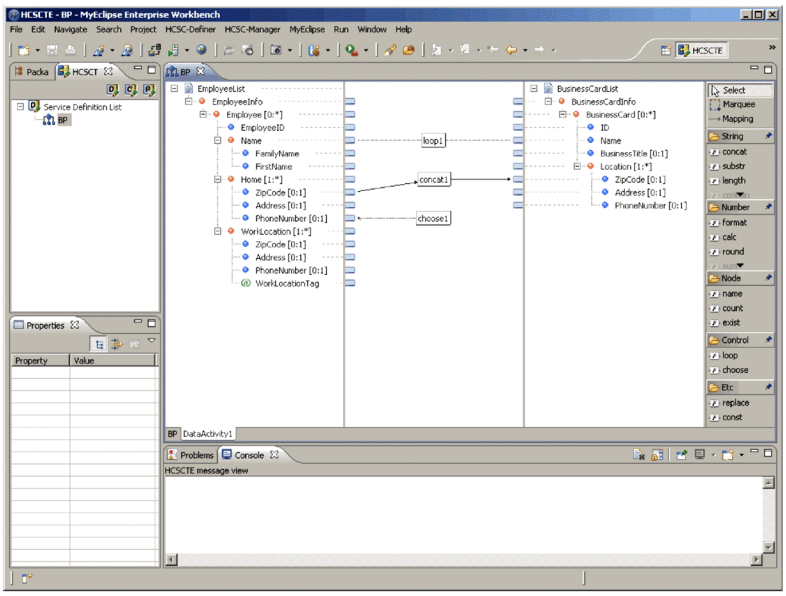
Also, a BPEL definition created in an upper process by using an upstream design tool based on BPMN can be imported into the development environment (Service Architect) of the service platform. The imported BPEL definition is converted to a business process definition. The converted business process definition, for which activities are set, is displayed in the development environment window. For the converted business process definition, check the processing flow and activity settings, and modify the definition as necessary.

In this way, by importing BPEL definitions created in an upper process, you can reuse existing assets. This can save time and avoid problems when redefining in the development environment of the service platform.

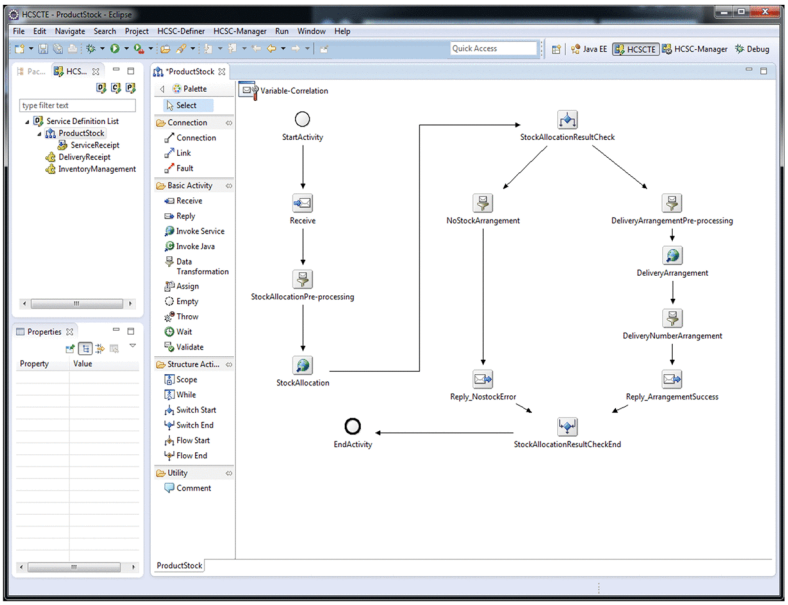
The following figure shows example windows used for system development that uses the service platform.

Figure 8–2: Example of windows used for system development

●Example of windows to define a data transformation



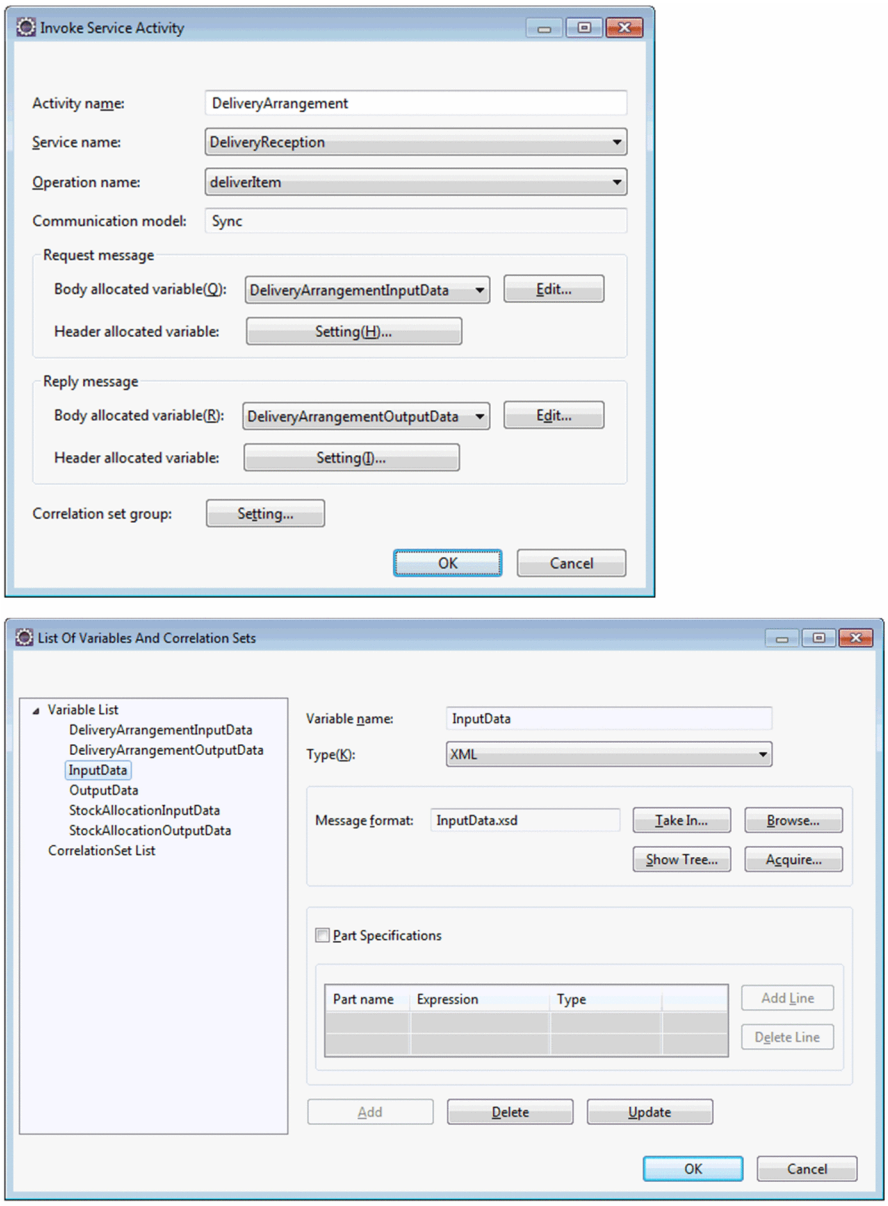
●Example of windows to define a business process



In addition, by opening dialog boxes from a window whenever necessary, you can define detailed parameters for data transformation, business processes, and service adapters (such as the SOAP adapter used to call service components of Web services, and the DB adapter used for operations on databases).

The following figure shows example of dialog boxes used for system development and which are used in the service platform.

Figure 8–3: Example of dialog boxes used to set detailed definitions



Thus, the service platform allows you to develop systems in a visual environment. This reduces the time and effort of system developers in programming jobs. Improving development speed allows you to promptly handle changes in the business environment, and to reduce the costs required for system configuration.

8.2.2 Ensuring Availability and Expandability Through the Use of Industry-Standard Technologies

For the creation of SOA systems, a service platform uses and supports the following technologies:

Fundamental technologies

With regard to data format, a service platform uses the *XML* format, which has a strong affinity with Web Services. Also, since it uses Java as the base in the systems that are set up, portability between platforms is ensured.

Usable services

Systems that are set up using a service platform support *Web Services*, *SessionBean*, and *MDB* as usable services.

Development environment windows

The development environment windows use *Eclipse*. The functions you wish to use in a service platform can be used by installing plug-ins on Eclipse.

Descriptive language for business processes

BPEL is used for business process descriptions. A service platform stores business processes that you define using windows as BPEL descriptions.

By using these standard technologies, a service platform allows you to set up general-purpose systems with high availability.

Since a service platform lets you use existing resources more effectively in developing a system, you can reduce the costs involved in introducing new technologies.

8.2.3 Service Conversion of Database Operations

A service platform allows you to define and use database operations as a single service component. To link with a database, you use a *DB adapter* to convert the database into Web Services, and then you link it up. You can use the windows of a service platform to create a DB adapter.

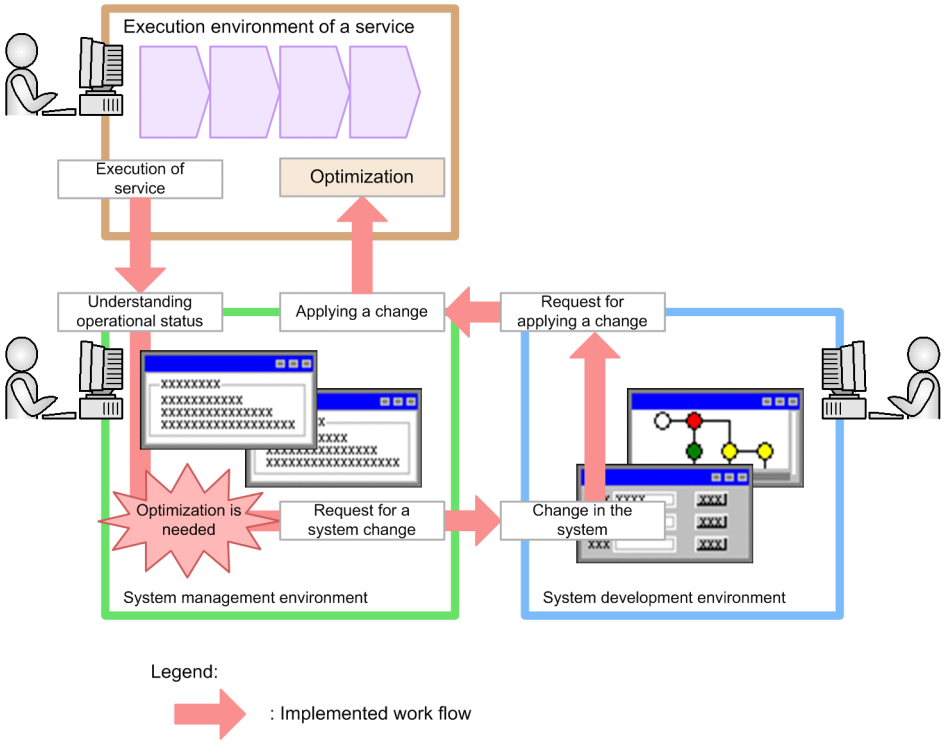
When database operations are converted into a service, service users can operate the database without having to worry about access control to the database tables.

8.2.4 Operation Status Awareness and System Optimization

The service platform allows you to check the operation status of the system by service. By being aware of the system's operation status and analyzing it, you can collect information necessary for enhancing system operations. Using this collected information and taking into consideration the business environment, you can establish a cycle for ongoing optimization of the system.

The next figure shows a system optimization cycle.

Figure 8–4: System optimization cycle



You can also collect various log and trace files. When a failure occurs, you can use the log and trace files to investigate its location and cause.

8.2.5 Intelligent Distribution Control

Messages that are sent as requests for executing service components are sent automatically to the appropriate service components according to the service component type (single service component or business process). Operators can use the service components without knowing the types or locations of the service components they are executing.

8.2.6 Resolving Data Differences by Transforming Data

A service platform has a standard data structure (message format) for accepting requests for executing service components. The structure is called a *message*. However, just as there are various types of service components, there are various types of data structures (message formats) that the service components request. This means that the data structure requested by a service component may differ from that of the message.

In these cases, the data transformation functionality reconciles the differences between the data structure of the message and the data structure requested by the service component.

The data transformation functionality defines how data items with different structures are to be mutually transformed. The system uses such definitions to resolve differences automatically between data structures when service component execution is requested, thus allowing the operator to request for executing service components that have a variety of data structures.

8.2.7 Effectiveutilization of an existing system

Various service adapters as mentioned below are available in the service platform to use service components. By using these service adapters, you can flexibly build a system and effectively utilize data of an existing system as service components.

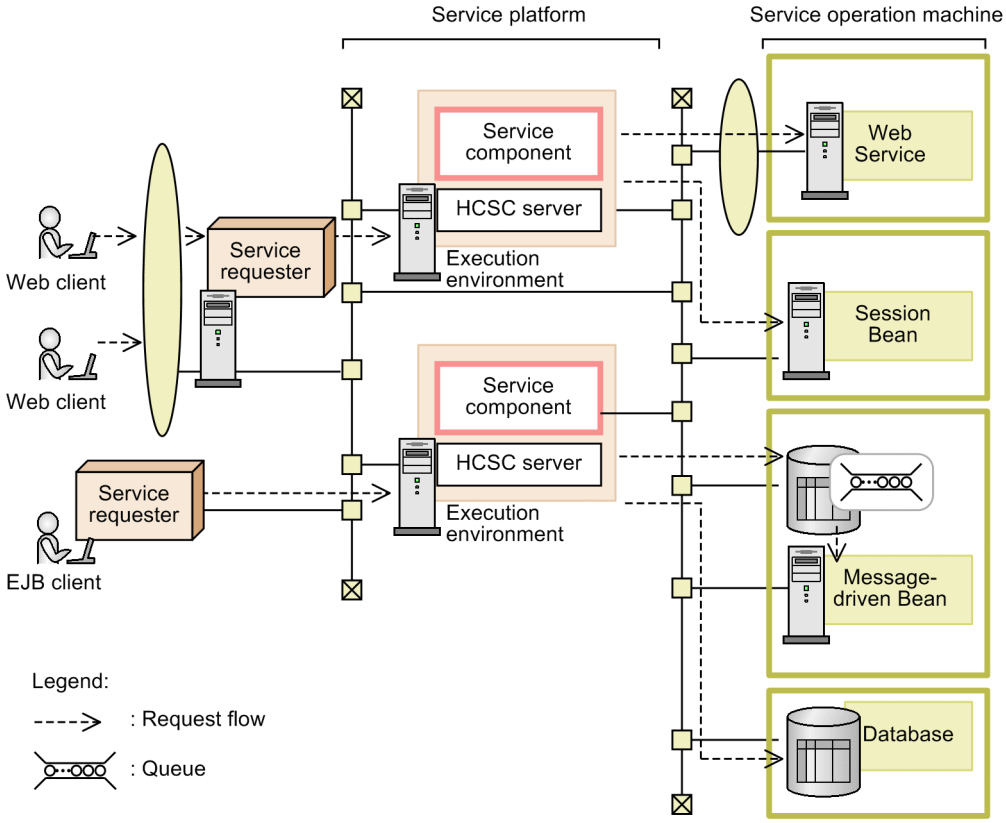
- SOAP adapter for using the service components of Web Services (SOAP communication)
- SessionBean adapter for using the service components created with EJB(Stateless Session Bean or Stateful Session Bean)
- MDB (WS-R) adapter for using the service components of asynchronous MDB (Message Driven Bean) by using WS-R (WS-Reliability)
- MDB (DB queue) adapter for using the asynchronous service components of TP1/EE by using DB queue
- DB adapter for using database as service components
- TP1 adapter for using the system on OpenTP1 or XDM/DCCM3
- File adapter for using the file-mediated business-processing system
- Object Access adapter for using core system such as mainframe
- Message Queue adapter for sending and receiving messages from an existing message queue (IBM WebSphere MQ system)
- FTP adapter for transferring files by connecting to the FTP server
- File operations adapter for performing the format conversion, duplication, and deletion of file
- Mail adapter for sending mails by connecting to the mail server
- HTTP adapter for using RESTful Web Services (Web Services that use JAX-RS engine)

8.3 Flow of Requests Using the Service Platform

The service platform receives a request from a service requester, and sends it to the corresponding service operation machine, based on (for example) the service content or the protocol type. The result of the processing executed on the service operation machine is returned to the service requester via the service platform. If the service requester calls a business process, multiple services running on the service operation machine are called according to the definition of that business process.

The following figure shows the flow of requests in an execution environment built with the service platform.

Figure 8–5: Flow of requests in an execution environment built with the service platform



To prevent requests from being concentrated on one execution environment, you can spread the load of requests by using the execution environment in a redundant configuration. Load balancing is available when a Web service or Session Bean is used. In this case, a load balancer (for a Web service) or CTM (for Session Bean), which are functions in the execution environment of the application server, is used for load balancing, depending on the service type. For details about load balancing of requests in the service platform, see *1.4.1 Redundant configuration of the HCSC server using the load-balancing functionality* in the manual *BPM/ESB Service Platform Overview*.

9 Service Platform Functionality

This chapter presents an overview of the service platform functionality for achieving a system to which SOA is applied.

For details about the functionality described in this chapter, see *Chapter 1. Functional Overview of Service Platform* in the manual *BPM/ESB Service Platform Overview*.

9.1 Overview of the Service Platform Functionality

The main functionality provided by the service platform, which is a basic product for achieving SOA, is as follows:

- Functionality for invoking service components

  Manages business operations and functions as services, calls appropriate service components corresponding to requests, and executes the service components.
- Functionality for executing business processes

  Develops and executes business processes used to control the service call flow.
- Data transformation functionality

  Provides the data transformation function, which is used to absorb differences between message formats.
- Various types of reception

  Provides receptors for receiving execution requests from various types of service requesters.
- Various types of service adapters

  Provides service adapters used to call various types of services.
- Execution log management function

  Permanently manages the execution history of business processes in a database, allowing retries if an error occurs and visualization of business processes.

The following describes details of the above functionality.

9.2 Functionality for invoking service components

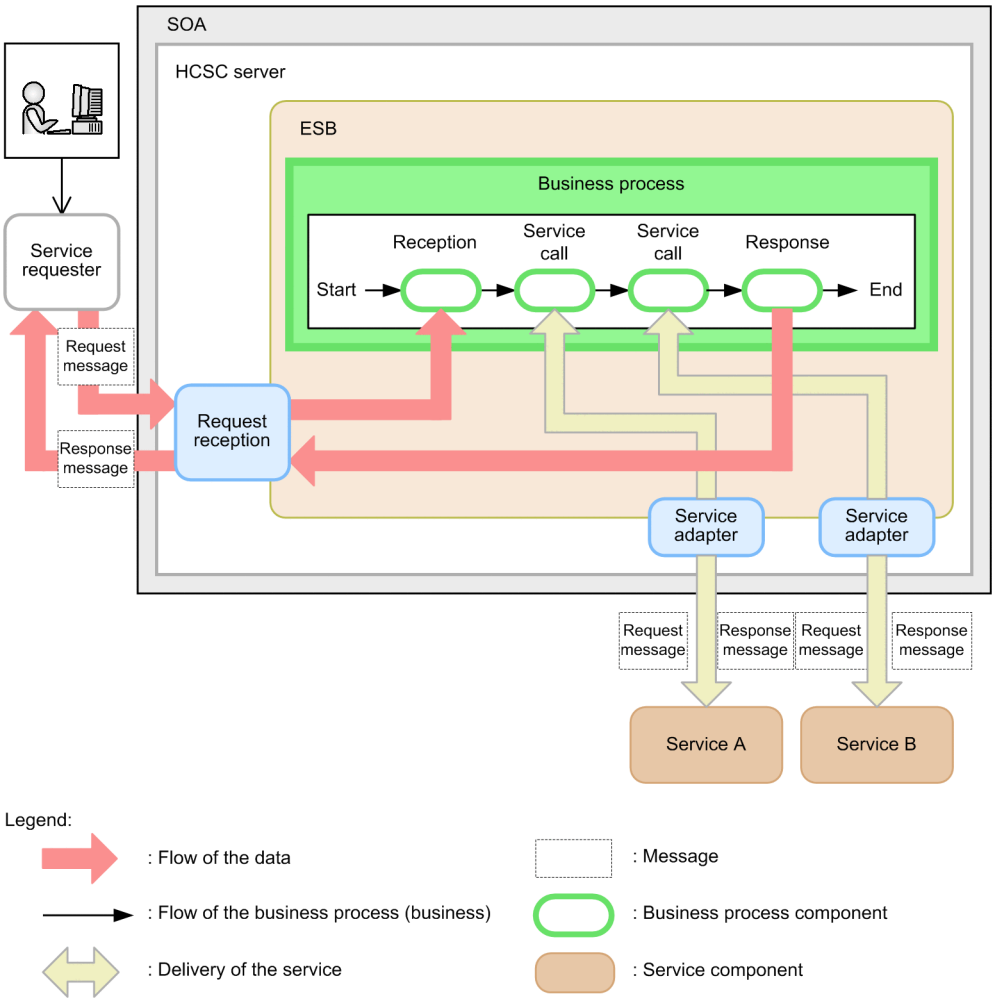
The service platform implements the linkage between the service user, business process, and services through an enterprise service bus (ESB).

In the systems built with SOA, a user-side system for sending requests is called as *service requester* and a provider-side system for providing the functionality is called as *service component*. Also, the SOA environment used for building ESBs, business processes, and adapters is called as *HCSC server*.

If users request for the execution of service components, SOA invokes and executes the appropriate service components from business processes or adapters that are deployed on the HCSC server.

The following figure shows the procedure for invoking service components:

Figure 9–1: Procedure for invoking service components



For details about business processes, see *9.3 Functionality for executing business processes*, and for details about the types of adapters, see *9.6 Types of Service Adapters*.

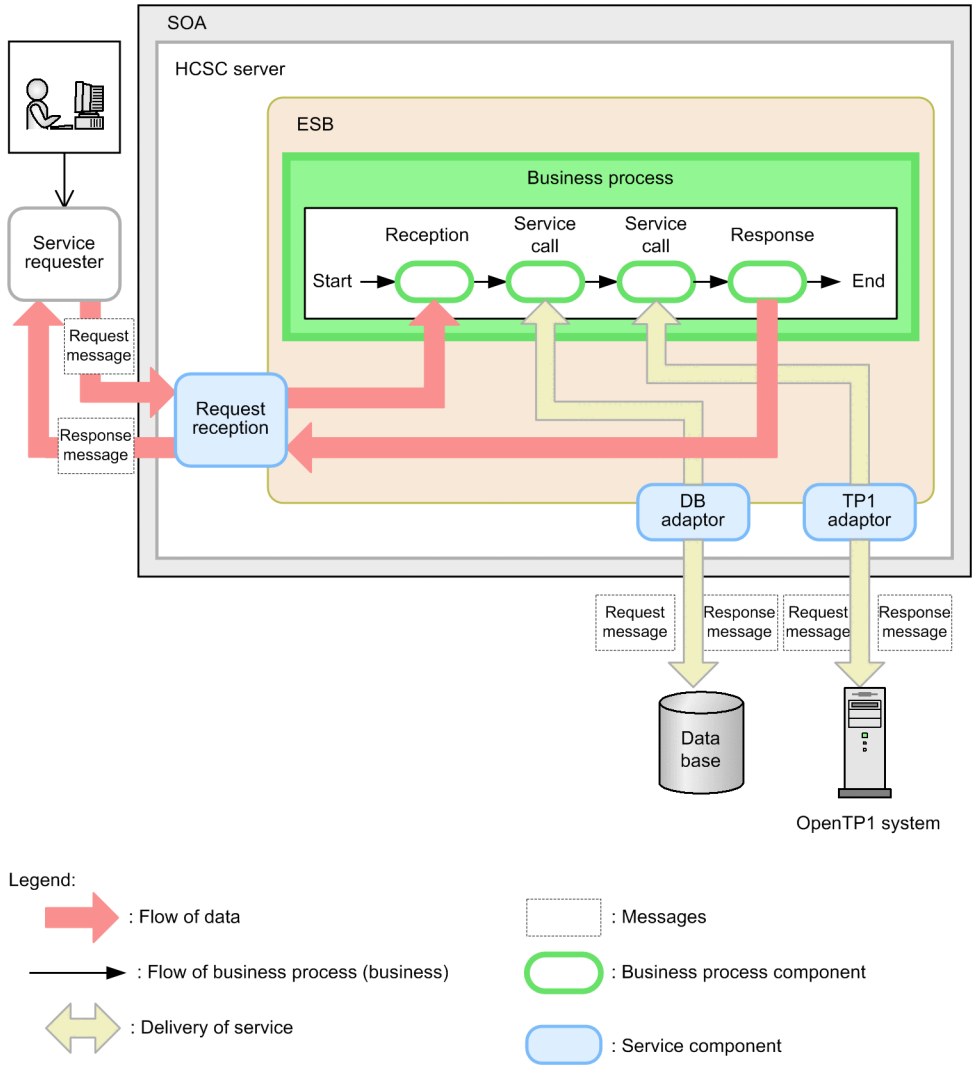
9.3 Functionality for executing business processes

A business process defines the processing order and conditions of multiple service components as a business series.

In the service platform, you can call and execute service components sequentially in the order that they are defined in a business process.

The next figure shows a summary of service component execution using a business process.

Figure 9–2: Executing service components from business processes



If a service requester that is an application for sending requests (request messages) to execute services on the HCSC server, receives a request for executing service components, the service requester will send the request message to a business process.

The business process transmits information sequentially to the individual service adapters, and executes the service components via the service adapters.

Once execution of one service is completed, processing switches to execution of the next service, and service execution continues until the business process ends.

When processing in each service component ends, a response message is sent to the corresponding adapter as the processing result. The response message is reported to the business operator through the service requester.

A business process defines the basic elements of business processes such as [receive], [reply], and [service invoking] or sets the processing of a business process such as [branch process], [parallel process], and [repeat process].

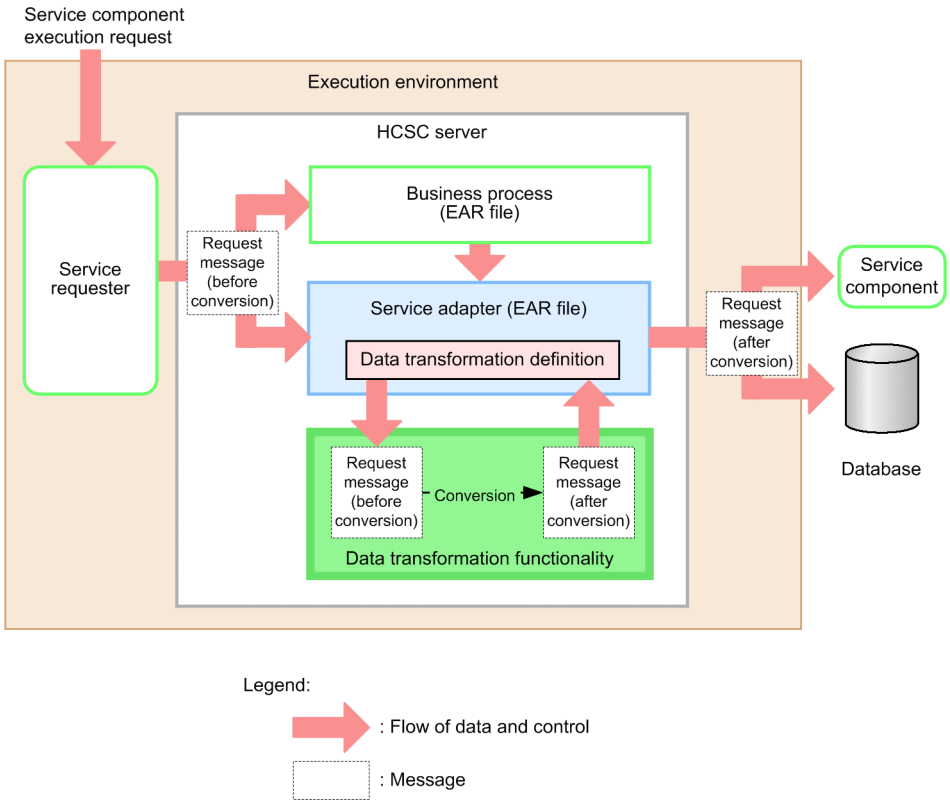
Note that a business process is defined using the GUI tools integrated with Eclipse instead of coding.

9.4 Data Transformation Functionality

Using the Service Platform, you can perform the data transformation of XML and binary data for the smooth processing of businesses in the systems that are built with SOA. You can perform the seamless data integration with existing systems, without thinking about any difference in the data format.

The *Data transformation functionality* transforms a request message to the appropriate format when, in the execution environment, the format of a request message transmitted from a service requester to an adapter differs from the format of a request message transmitted from the adapter to the service component. The following figure shows an overview of the data transformation functionality in the execution environment.

Figure 9–3: Overview of the data transformation functionality



If a service requester that is an application for sending requests (request messages) to execute services on the HCSC server, receives a request for executing service components, the service requester will send the request message to an adapter or a business process.

If the adapter (including an adapter called from a business process) includes a data transformation definition, the data transformation functionality converts the request message.

The functionality executes data transformation according to the contents specified in the data transformation definition. The adapter transmits the transformed request message to the service component to be executed.

If a response such as the execution result is to be returned to the business operator after service component execution, a data transformation is executed that is the opposite of the one performed during the execution request.

9.5 Types of Reception

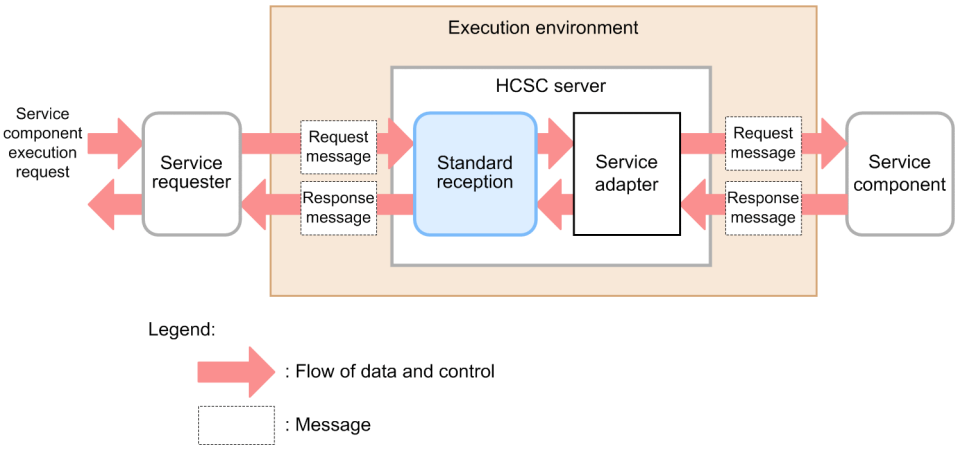
The service platform receives request message from service requester with reception. Reception has standard reception included in HCSC server and user-defined reception that enables user to define any interface.

9.5.1 Standard reception

Standard reception is a function included in HCSC server. Standard reception has synchronous reception (Web service/SessionBean) and, asynchronous reception (MDB (WS-R)/MDB (DB queue)). When standard reception is started, request message from service requester can be received.

The following figure shows an overview of receiving request message from service adapter with standard reception and executing service component:

Figure 9–4: Flow for receiving request message and executing service component (in case of standard reception)



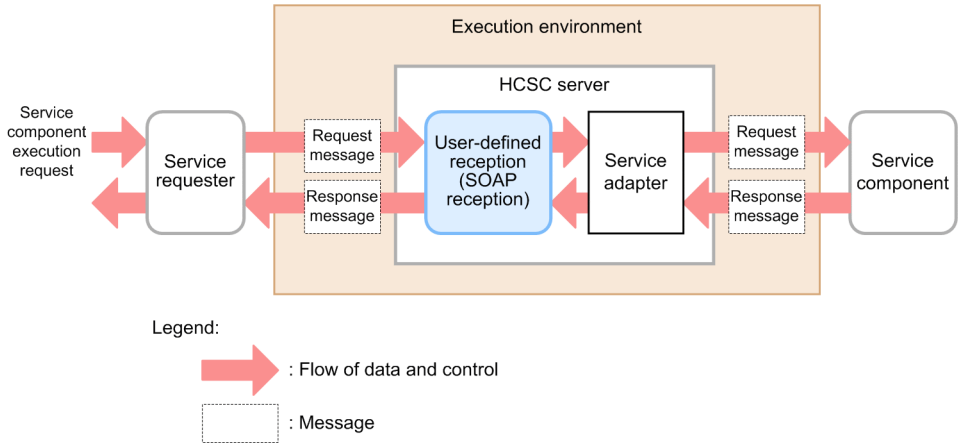
Unlike other reception, standard reception does not require user to define the reception independently. If you create service requesters in accordance with each standard reception format, you can use them as is. However, when you reuse an existing service requester, if service requester does not match the standard reception interface format then, you are required to modify or recreate service requester.

9.5.2 User-defined reception (SOAP reception)

In the service platform, when you request service component execution in a system that uses Web service (SOAP communication), you can use user-defined reception (SOAP reception) that defines any interface.

The following figure shows an overview of receiving request message from service adapter with SOAP reception and executing service component:

Figure 9–5: Executing service component using SOAP reception



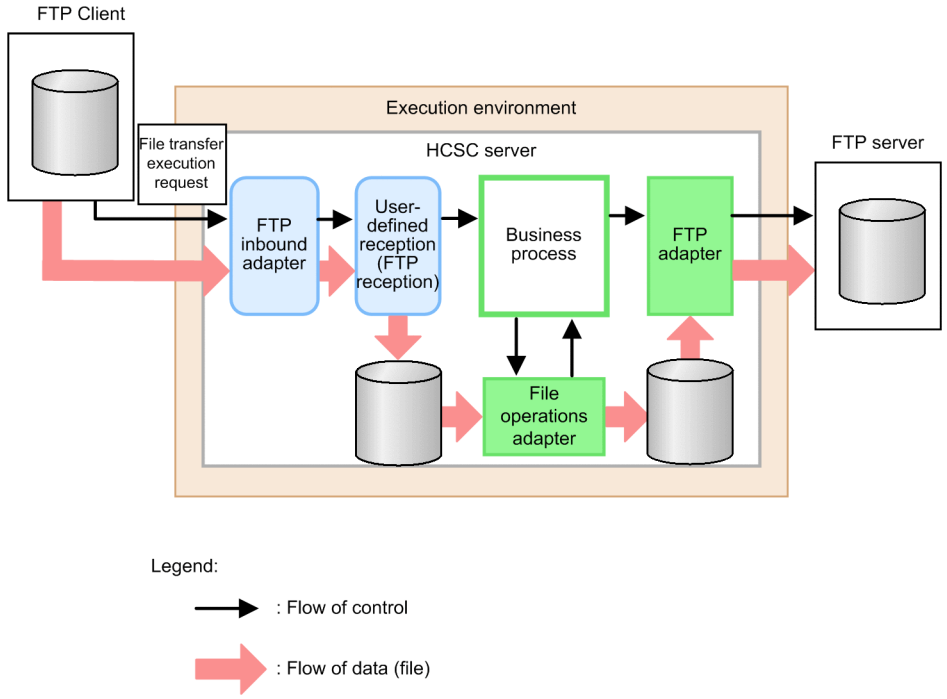
9.5.3 User-defined reception (TP1/RPC reception)

INTENTIONALLY DELETED

9.5.4 User-defined reception (FTP reception)

In the service platform, you can use FTP reception to receive files transferred from FTP client or relay file transfer between FTP client and FTP server. The following figure shows an overview of relaying file transfer:

Figure 9–6: Transferring files between FTP client and FTP server using FTP reception



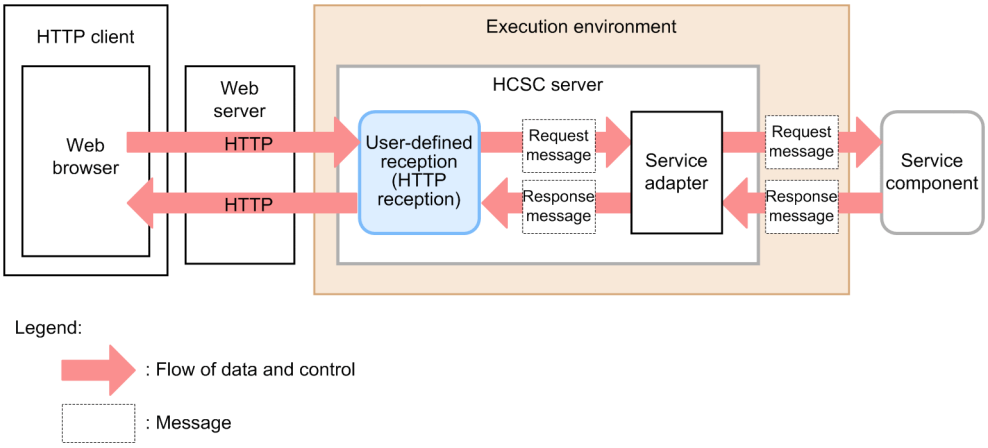
FTP reception receives connection request from FTP client through FTP inbound adapter and, calls the business process. File can be transferred between FTP client and FTP server by linking with FTP adapter or file operations adapter that is called from business process.

9.5.5 User-defined reception (HTTP reception)

In the service platform, if you use HTTP reception, business process can be called directly without going through Web front system or SOAP reception when there is a connection request from mobile terminals or Web browser.

The following figure shows an overview of calling and executing service component with HTTP communication from HTTP client using HTTP reception:

Figure 9–7: Executing service component using HTTP reception



HTTP request originated from Web browser in HTTP client can be received with HTTP reception via Web server. After that, HTTP request is converted into request message (request to call service component) and then service component is called through service adapter.

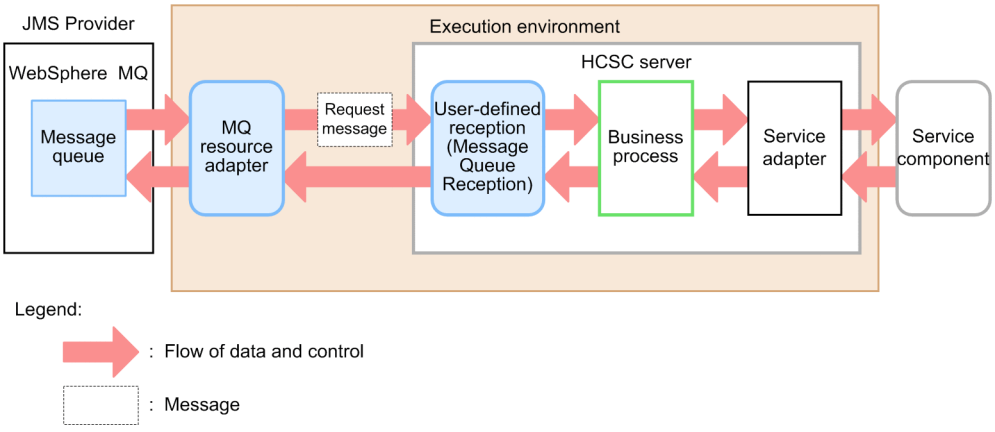
At the time of response, response message passed from service adapter is converted into HTTP response format in HTTP reception and then the response is returned to Web browser.

9.5.6 User-defined reception (Message Queue reception)

In the service platform, if you use Message Queue reception, you can call the business process from message queue (IBM WebSphere MQ system) via the MQ resource adapter and execute each service component.

The following figure shows an overview of calling and executing service component from message queue using Message Queue reception:

Figure 9–8: Executing service component using Message Queue reception



Message Queue reception receives request message via the MQ resource adapter. On receiving service component execution request, Message Queue reception calls the business process and executes various service components through the service adapter.

The receipt of request message uses MDB (Message-Driven Bean).

Message Queue reception does not return a response message.

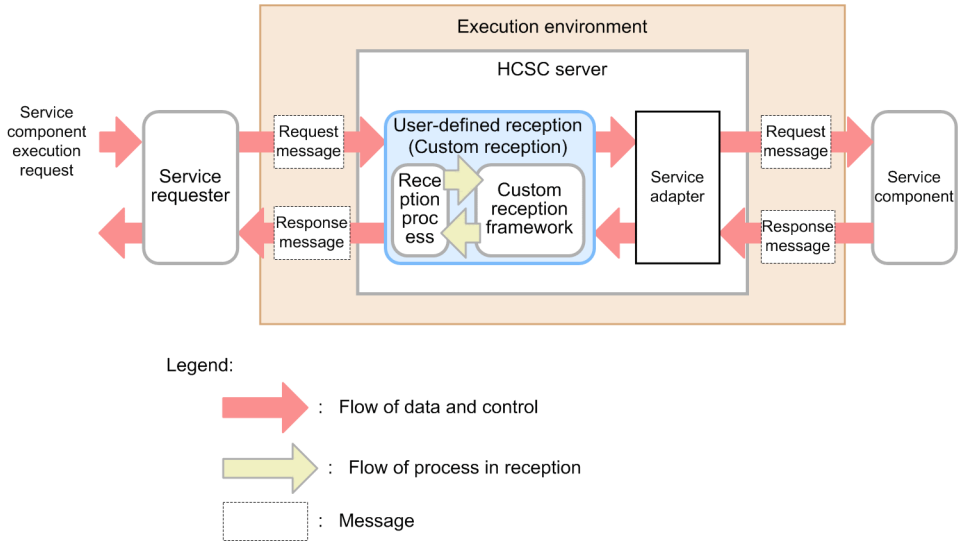
9.5.7 User-defined reception (custom reception)

The service platform provides custom reception framework for executing service components of any system.

You can call and execute any service component from custom reception that combines the provided custom reception framework with reception process created by the developer.

The following figure shows an overview of executing service components from custom reception:

Figure 9–9: Executing service components using custom reception



Reception process receives request message from service requester. The received request message is passed to custom reception framework and after executing process (data transformation); custom reception sends the request message to service component and executes the service component.

9.6 Types of Service Adapters

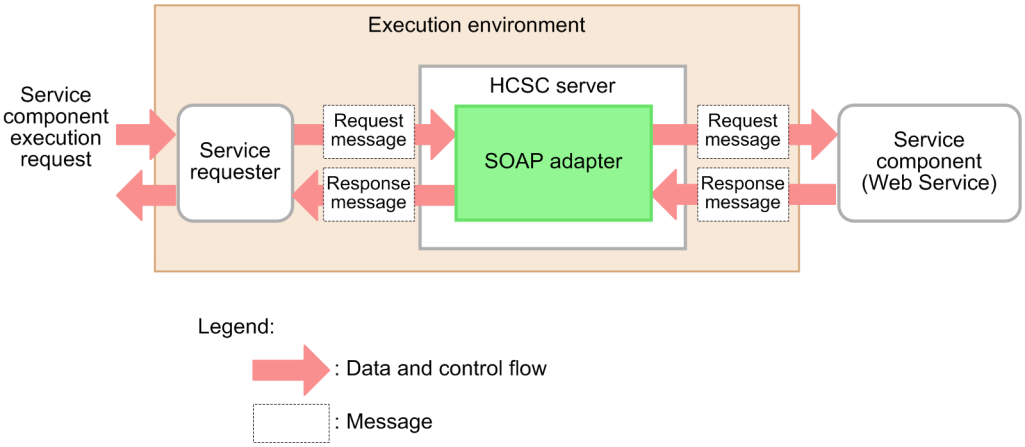
The service platform provides service adapters for available services, based on industry-standard technologies, such as Web services using SOAP communication, Session Beans, and MDB. It also provides various service adapters that can be used to connect existing systems, such as mainframe systems and online systems.

9.6.1 SOAP adapter

The service platform provides a SOAP adapter for calling service components of Web services (SOAP communication).

The following figure shows an overview of calling a service component from the SOAP adapter and executing a service component.

Figure 9–10: Execution of a service component from the SOAP adapter



The service requester (an application that sends execution requests (request messages) to the HCSC server) receives an execution request for a service component (Web service), and then sends a request message to the SOAP adapter.

The SOAP adapter sends a request message to the service component defined in the adapter, and executes the service component.

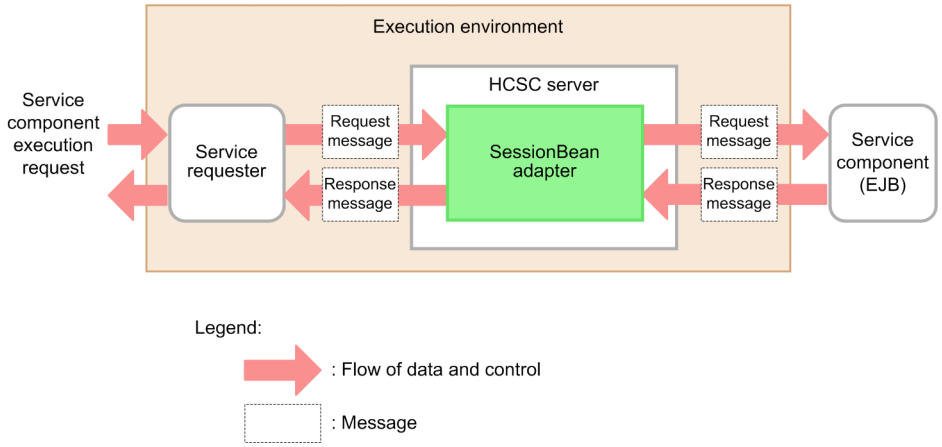
After the processing for the service component finishes, a response message is sent to the SOAP adapter as the processing results. The person responsible for the business operation is notified of the response message via the service requester.

9.6.2 SessionBean adapter

The service platform provides SessionBean adapter for calling service components that are created by EJB (Stateless Session Bean and Stateful Session Bean).

The following figure shows an overview of calling and executing service components from SessionBean adapter:

Figure 9–11: Executing service components from SessionBean adapter



Service requester is an application that sends service execution request (request message) to HCSC server. On receiving service component (EJB) execution request, service requester sends request message to SessionBean adapter.

SessionBean adapter sends request message to service component defined in the adapter and executes the service component.

When the process in service component is complete, response message is sent as a process result to SessionBean adapter. The response message is notified to the business in-charge via the service requester.

9.6.3 MDB (WS-R) adapter

INTENTIONALLY DELETED

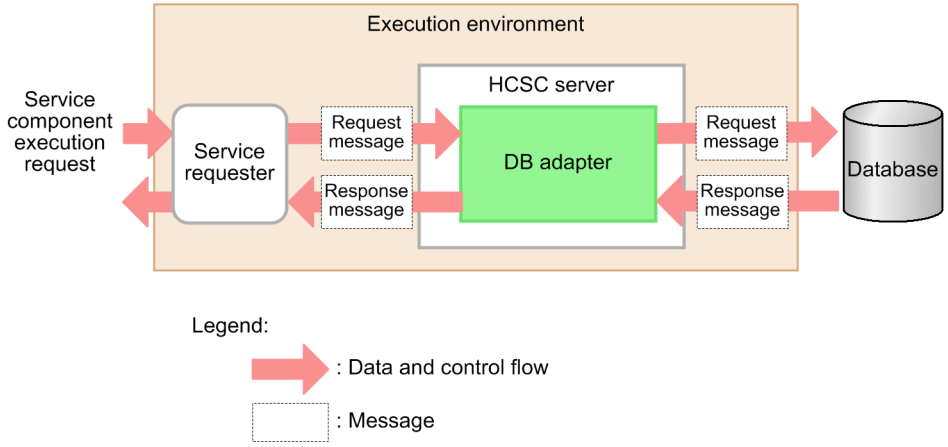
9.6.4 MDB (DB queue) adapter

INTENTIONALLY DELETED

9.6.5 DB adapter

The service platform allows you to execute SQL on a database from a DB (database) adapter. The following figure shows an overview of an SQL execution request for a database from the DB adapter.

Figure 9–12: Execution of SQL from the DB adapter



The service requester (application that sends execution requests (request messages) for services to the HCSC server) receives an execution request for a service component, and then sends a request message to the DB adapter.

The DB adapter sends a message that requests execution of SQL on a database defined in the adapter, and then executes the SQL.

After execution of the SQL finishes, a response message is sent to the DB adapter as the execution result. The person responsible for the business operation is notified of the response message via the service requester.

Important note

Only the following databases can be used as execution targets when you use the DB adapter to execute a service component (database operation):

- HiRDB/Single Server (Version 8)
- HiRDB/Parallel Server (Version 8)
- HiRDB Server (Version 9)
- Oracle 11g
- Oracle 12c

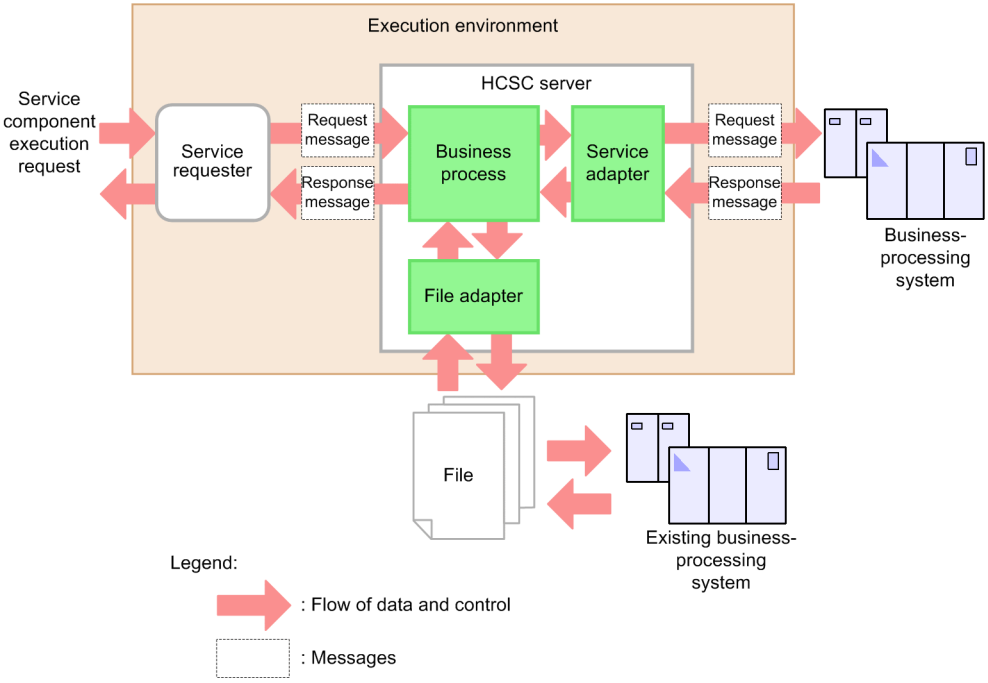
9.6.6 TP1 adapter

INTENTIONALLY DELETED

9.6.7 File adapters

With the Service Platform, you can input and output files of existing business process systems from file adapters. The following figure shows a summary of file I/O with a file adapter.

Figure 9–13: Input and output of files from file adapters



When using files of an existing business process system, you can invoke file adapters from business processes and can read data of the file.

Also, even for passing the service execution results to an existing business process system, invoke a file adapter from the business process and then deliver the file data.

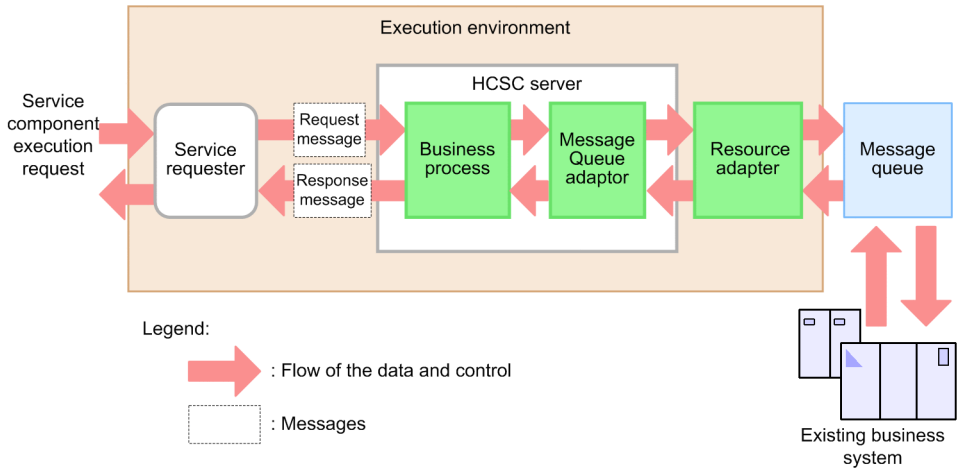
9.6.8 Object Access adapters

INTENTIONALLY DELETED

9.6.9 Message Queue adapters

With the Service Platform, you can send and receive messages for the existing message queue (IBM WebSphere MQ system) from Message Queue adapters. The following figure shows a summary of message transmission and reception using a Message Queue adapter.

Figure 9–14: Message queue control from Message Queue adapters



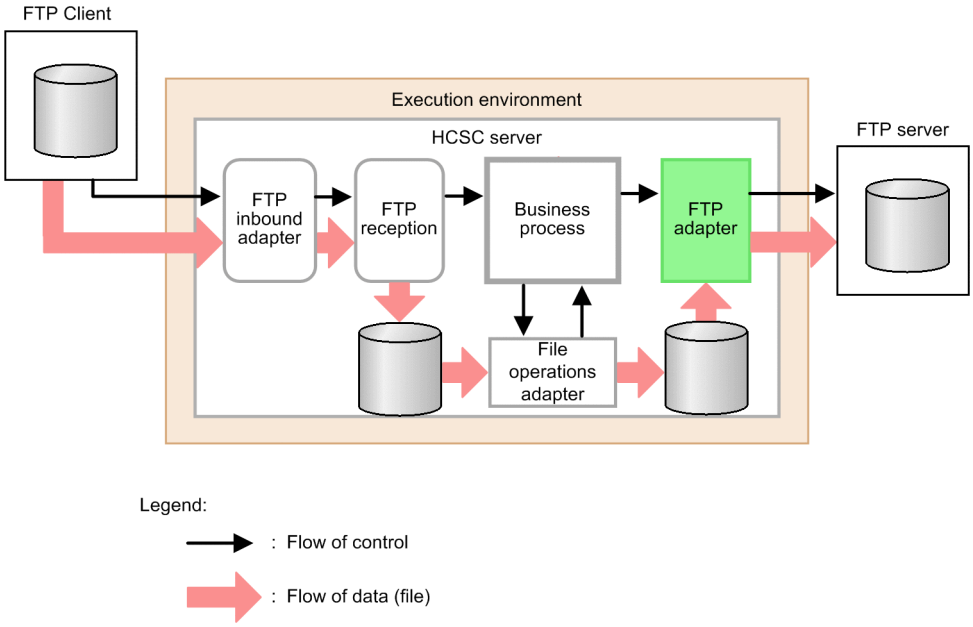
When using an existing message queue, you can invoke a Message Queue adapter from a business process to send and receive messages and to browse.

9.6.10 FTP adapter

The service platform provides FTP adapter for file transfer between FTP client and FTP server. FTP adapter performs the file transfer process with FTP server.

The following figure shows an overview of sending and receiving files between FTP adapter and FTP server:

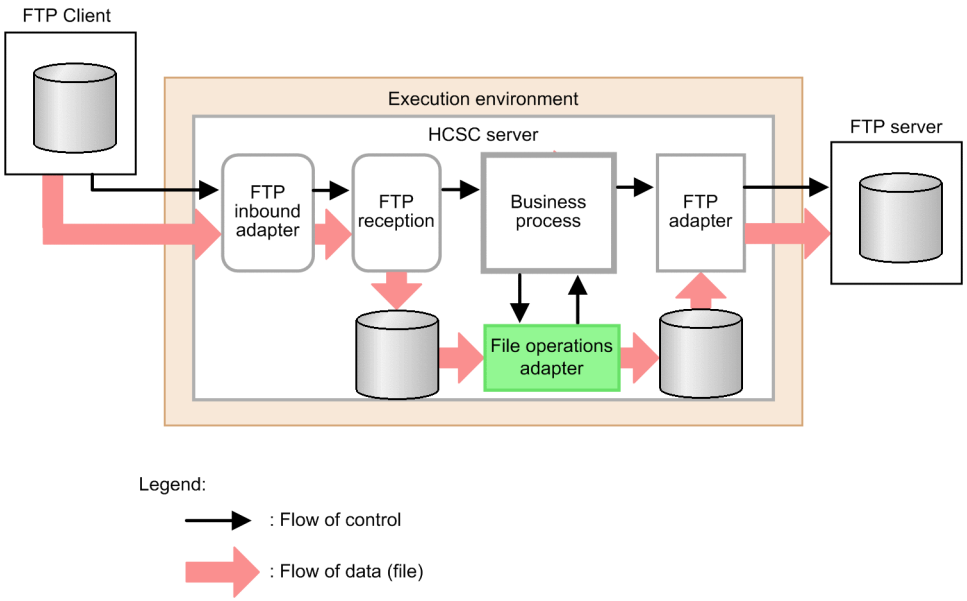
Figure 9–15: Sending and receiving files between FTP adapter and FTP server



9.6.11 File operations adapter

In the service platform, you can convert format, duplicate and delete files to be sent and received from file operations adapter. The following figure shows an overview of operating files from file operations adapter:

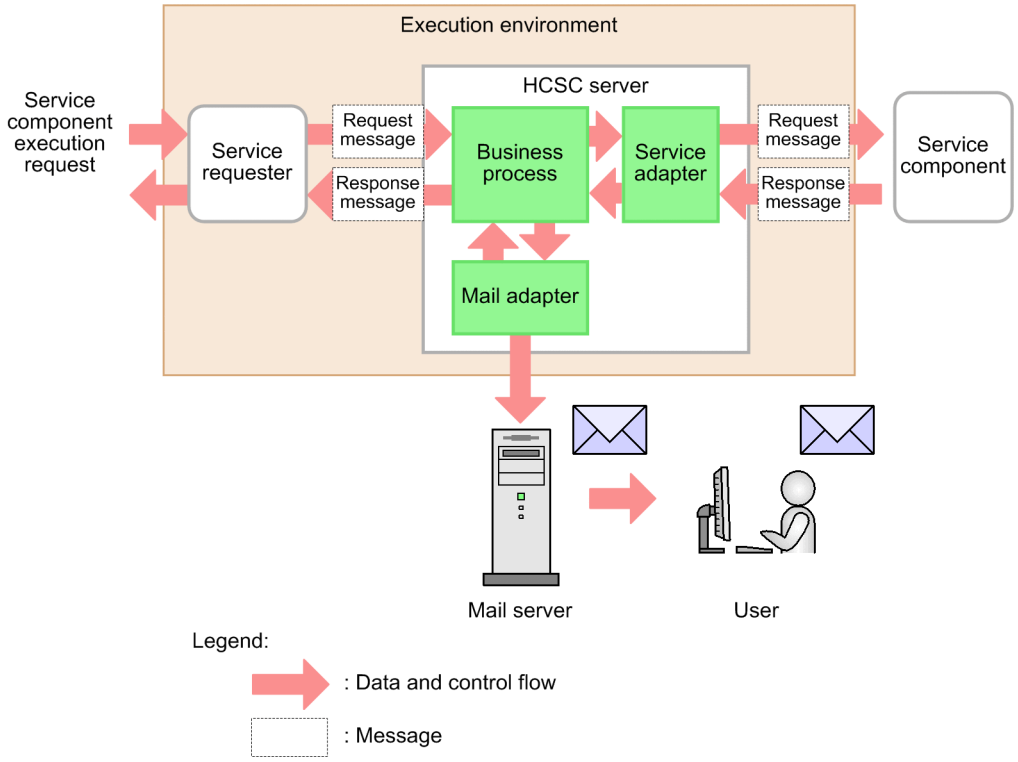
Figure 9–16: Operating files from file operations adapter



9.6.12 Mail adapter

If a mail server supports the SMTP protocol, the service platform allows you to call, as a service, the mail server from the mail adapter, and to send mail. The following figure shows an overview of using the mail adapter to send mail.

Figure 9–17: Sending mail from the mail adapter

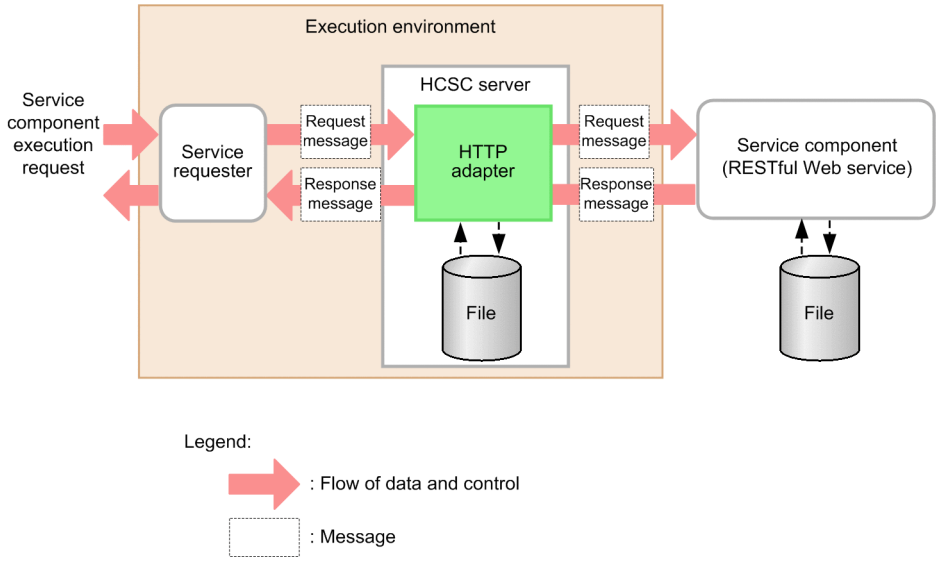


9.6.13 HTTP adapter

The service platform provides HTTP adapter for calling resources that are published on Web server and Web service (RESTful Web service) published in REST style.

The following figure shows an overview of calling and executing service components from HTTP adapter:

Figure 9–18: Executing service components from HTTP adapter



Service requester is an application that sends service execution request (request message) to HCSC server. On receiving service component (RESTful Web service) execution request, service requester sends request message to HTTP adapter.

HTTP adapter sends request message to service component that is defined in the adapter and executes the service component.

When the process in service component is complete, response message is sent as a process result to HTTP adapter. The response message is notified to the business in-charge via the service requester.

Furthermore, you can also send and receive files between HCSC server and service component with HTTP adapter. With this, you can use it to transfer files sent from client to Web server and, send and receive files between Web servers.

9.6.14 General Custom Adapter

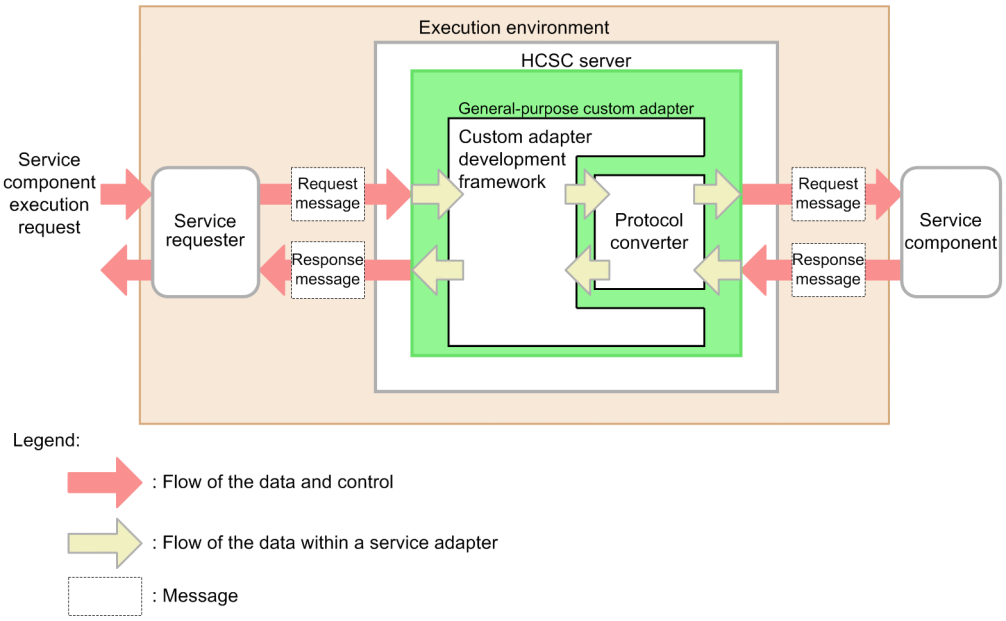
The service platform provides a custom adapter development framework to execute service components in a system that uses a protocol not supported by the following service adapters:

- 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

You can use the general custom adapter, which is a combination of the provided custom adapter development framework and protocol converter, to call and execute a service component.

The following figure shows an overview of executing a service component from the general custom adapter.

Figure 9–19: Execution of a service component from the general custom adapter



The service requester (application that sends execution requests (request messages) for services to the HCSC server) receives an execution request for a service component, and then sends a request message to the general custom adapter.

The general custom adapter that received the request executes the processing defined in the adapter (data transformation and protocol conversion) to allow connection to an arbitrary system. The general customer adapter then sends a request message to the service component, and executes the service component.

9.7 Execution Log Management Function

The service platform provides a function for managing the execution status of business processes and messages that call service components.

9.7.1 Managing the Process Instance Execution Log

You can manage the execution status of process instances of a business process as the execution history.

To manage a business process's execution status, you set up the business process so that execution status information for its process instances is saved (*persistence*) to the database.

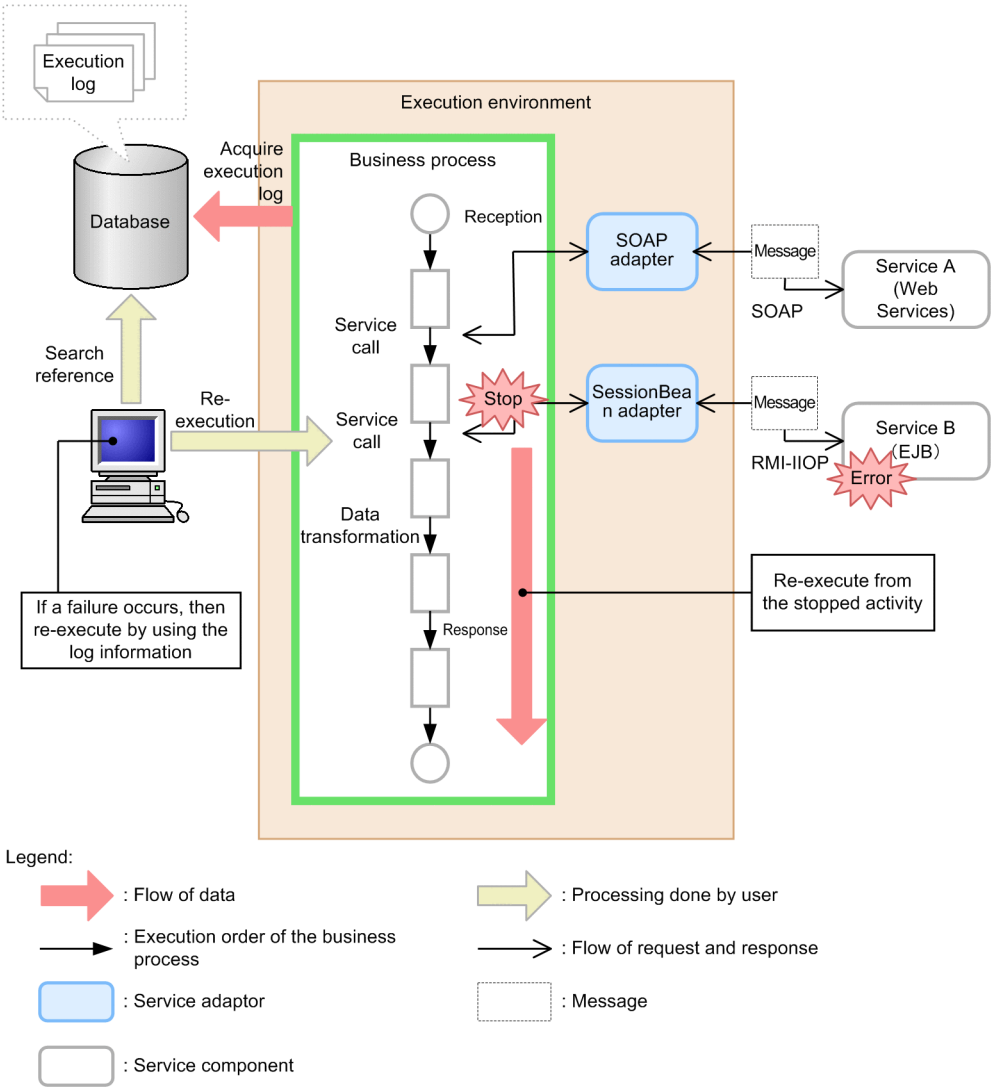
Important note

You can manage the execution status of business processes only when using a database.

Manage the execution history, and if an error occurs while invoking a service to be executed from a business process, you can use the execution history to perform retry from the service in which the error has occurred. You can retry one by one or in a batch.

The next figure shows the management functions for the process instance execution log.

Figure 9–20: Functionality for managing the execution history of process instances



For searching the execution history, you can use the mutual relation set, start date, or status as a search key. You can also reference the format of a message that flows through the business process.

From the contents of the execution history, you can make business processes visible, such as optimum planning of the whole system to learn about the operation processes of each service and to analyze the acquired operation status.

10 System Development and Operations

This chapter describes how to develop and operate a system using a service platform.

For details about the development and operation procedures described in this chapter, see the manuals *Service Platform Development Guide* and *Service Platform System Setup and Operation Guide*.

10.1 Developing a System Where SOA Is Applied

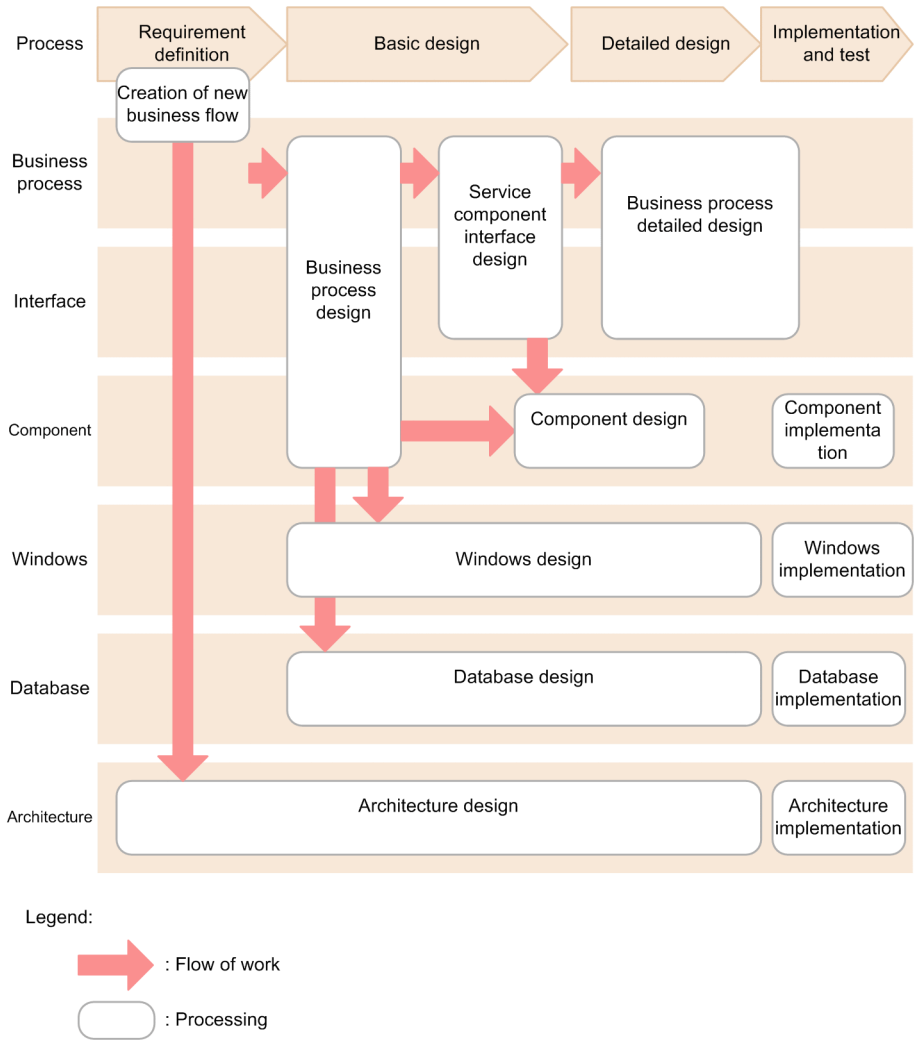
This section explains the merits of developing a system where SOA is applied and the features of such a system.

Even from the point of system development, the SOA application has the following merits:

- **Promotion of service reuse**
  For the new business processes that are to be converted to IT, by finding services of a similar pattern from the business processes that are already converted to IT, reuse of services being used in the existing business processes can be promoted.
- **Improvement in maintainability**
  By matching the unit of business and software, the location in the software where changes are corresponding to the business change are required can be identified easily. Additionally, by designing the service with an appropriate granularity, the service can be made more independent, and a change in the software can be restricted within the service. This can help to improve the maintainability of the information system.
- **Reduction in development cost**
  By using the existing system as a service, the parts to be developed newly can be reduced, thereby reducing the development cost.

To make the best use of such merits, during development of a system where SOA is applied, after designing the business process and service interfaces, design the components of the services. The following figure shows an overall image of how to develop a system where SOA is applied.

Figure 10–1: Overall image of how to develop a system on which SOA is applied



The development of a system where SOA is applied is broadly classified into the following processes:

- **Requirement definition**
  Create a new business flow that forms the base of the business process. Additionally, for architecture designing, examine the requirements other than those for functions, and review the system format.
- **Basic design**
  Design the business process and service interfaces to implement the defined requirements. Also design the functions of the applications (components, screens, database, and architecture).
- **Detailed design**
  Design the detailed specifications for implementation, based on various specifications of the business process and application design.
- **Implementation, and testing**
  Implement the applications and test the operation.

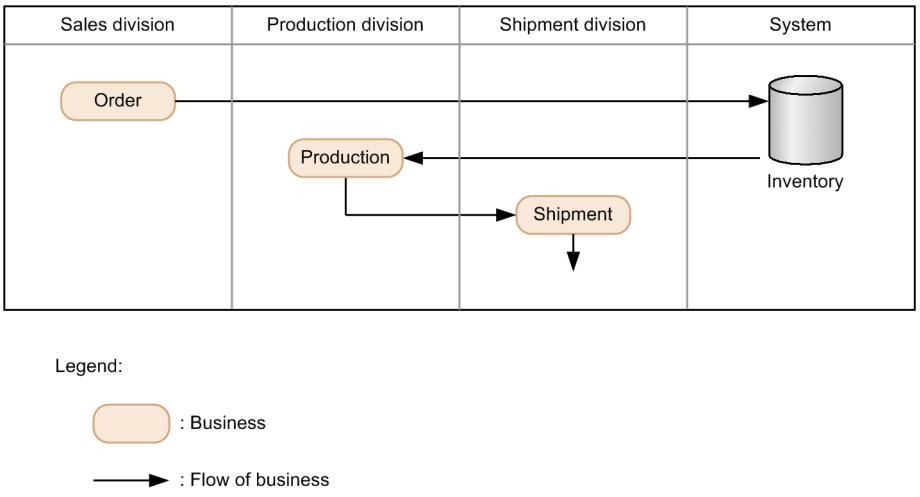
The work to be performed in each process is described below.

(1) Creating a new business flow

Examine the functional requirements based on business reformation, improvement policy, and business problem analysis result and create a new business flow. Create the business flow by using notations, such as the WFA (Work-Flow Architecture). In the business flow, explicitly state the organizations and persons in-charge involved in a business, the flow of a business, and the information that flows.

When you create the business flow, for the service, decide the candidate functions of business to be converted into a system. Additionally, decide the development policy for each service from policies such as new development, reuse of existing system, introduction of package software, and use of service as software. When usage of the policy is decided, decide the target businesses to be converted into services. The following figure shows an example of creating a new business flow.

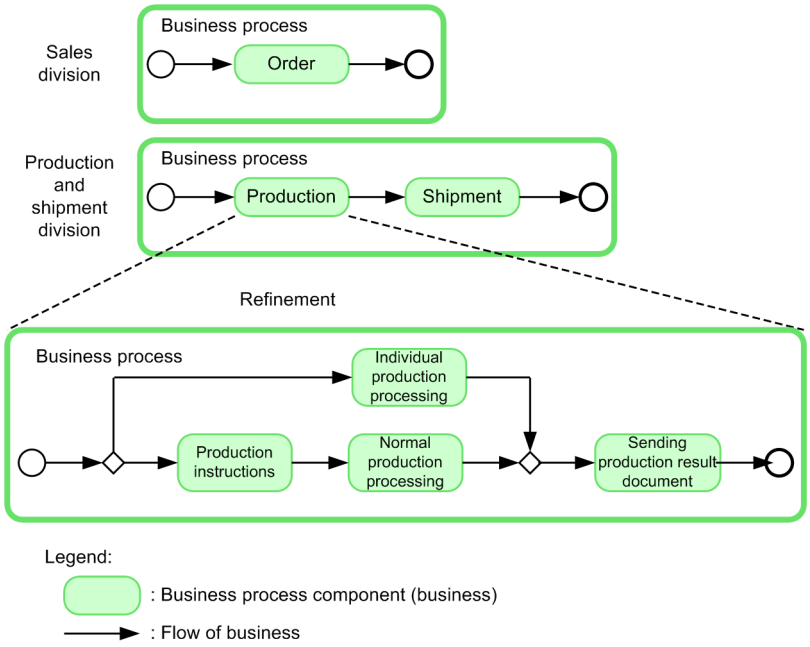
Figure 10–2: Example of creating a new business flow



(2) Designing a business process

Create the business process as a business process figure (basic flow) based on the new business flow that you created, with the help of BPMN (Business Process Modeling Notation). Additionally, from the viewpoint of businesses, arrange the businesses that constitute the business process in various layers as further detailed business processes in a step-by-step manner and revise the service granularity. The following figure shows an example of creating a business process.

Figure 10–3: Example of creating business processes



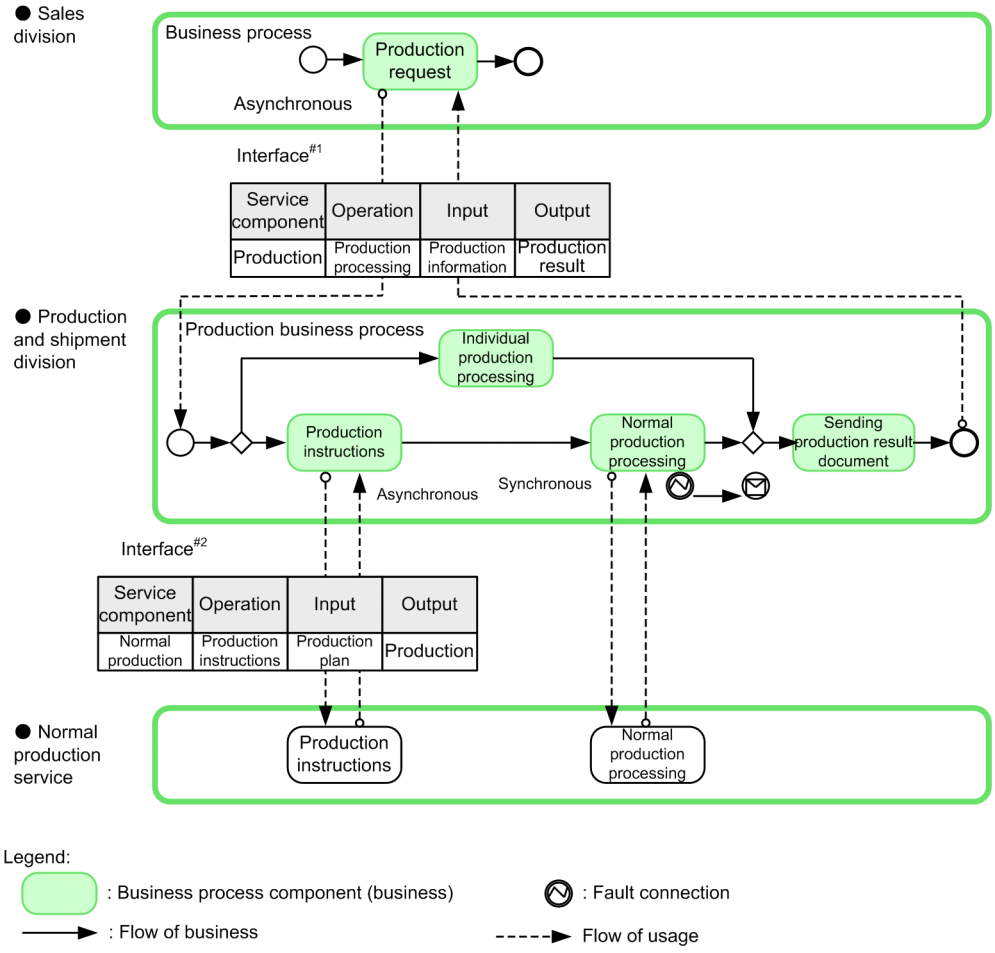
(3) Designing service interfaces

From the viewpoint of the system, revise the business process to make it more detailed. Along with this, decide the interfaces (such as the structure of messages, and addition of exception handling) required to implement the services.

Design the interfaces corresponding to the interactive workflow pattern, interactive application pattern, and online application pattern. Additionally, at this stage, decide about the service granularity for the last time.

The interfaces include an interface of the business process itself, apart from interfaces corresponding to services. Since the business process itself will become a complex service combining the link destination services, it requires an interface. The following figure shows an example of design of service interfaces and business process interface.

Figure 10–4: Example of designing interfaces of services and business processes



#1 Shows the definition contents when calling business process.

#2 Shows the definition contents when calling the functionality with service component.

(4) Detailed designing of business process

Based on various basic design specifications of a business process and interface, design the business process definitions required for implementation, and the detailed definitions of the messages that the services will exchange. For designing and implementation, use BPEL in the case of a business process, and WSDL or XML language in the case of interfaces and messages.

(5) Designing components

Design an application that implements a front system and services. The service components consist of three layers, namely the presentation layer (screen related), function layer (business functions), and data layer, and design each of the three layers. When designing you can assume that various industry standard frameworks such as Struts and JSF are used.

(6) Designing windows

In a front system and services, design the windows and window transitions required for interaction. Design the windows throughout the entire development. Design as follows for each process:

- Requirement definition: Examine the requirements of the end user through a simple layout of windows.
- Basic design: Based on the created scenario, design the windows and window transition of an interactive application.

- Detailed design: Based on the created scenario, design for an implementation that uses HTML and JSP in an interactive application.

(7) Designing a database

Perform operations right from analysis of the data handled in front system and services, up to the table design in the database. Design the database throughout the entire development. Design as follows for each process:

- Requirement definition: Analyze the data, extract key items, and understand the relationship between the data handled in the entire target system.
- Basic design: Based on the scenario created, analyze the required data, extract the components and their relationships, and perform the logical designing of the data.
- Detailed design: Perform the physical design of the data for physical tables.

(8) Designing the architecture

Based on the system conversion plan, examine the requirements other than those for functions, and design the architecture overview of the entire target system. While doing this, consider the development policy of the services nominated in creating the new business flow and examine the linking method with the existing system and packaged software. Continue designing from the basic design up to the detailed design, and design a format of the system corresponding to the requirements other than those for the functions.

10.2 System Configuration and Relationships with Individual Environments

With a service platform you configure three environments: a development environment, an operating environment, and an execution environment. You configure the development environment by installing Service Architect, and the execution environment by installing Service Platform.

This section describes the relationships among the different environments of a service platform.

10.2.1 Relationships Between Software Products and the Individual Environments

(1) Environments composing the service platform

The service platform is composed of the following three environments:

- Development environment

  This is the environment for creating the HCSC components (a general name for the service adapters, business processes, and user-defined receptions created in the development environment) required to integrate the services, and for packaging them in EAR files.
- Execution environment

  This is the environment for invoking service components and business processes corresponding to a request, and then executing the business.
- Operating environment

  This is the environment for deploying the HCSC components created in the development environment in the execution environment, and thereafter executing the operating environment. Furthermore, in this environment you can collect information from the execution environment, and confirm the status of service components.

(2) Relationship between the development environment, execution environment, and operating environment

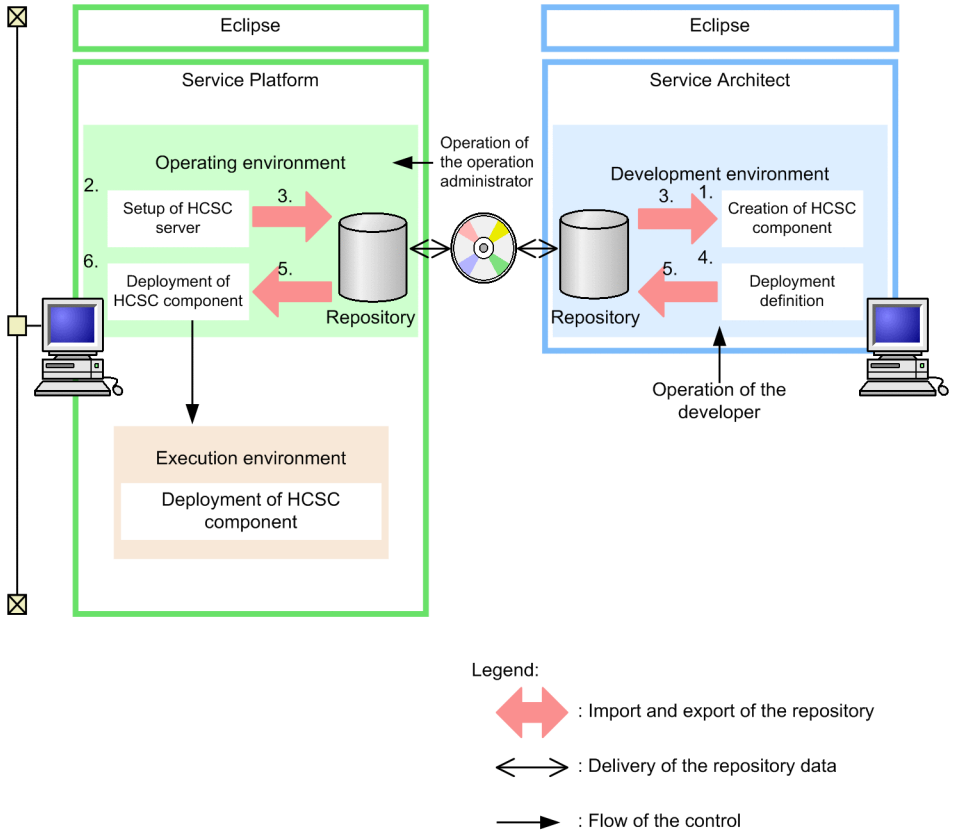
Set up the development environment, execution environment, and operating environment respectively on different computers.

Install Service Architect and Eclipse in the development environment. Install Service Platform in the execution environment, and install Eclipse if the OS is Windows.

To run the execution environment from the operating environment, you must use the data model called a *repository*.

The following figure shows the environments that compose the service platform.

Figure 10–5: Environments composing the service platform



These environments are mutually linked to implement an integrated environment of services. After you set up the environment and follow the steps mentioned below, each environment can actually execute operations:

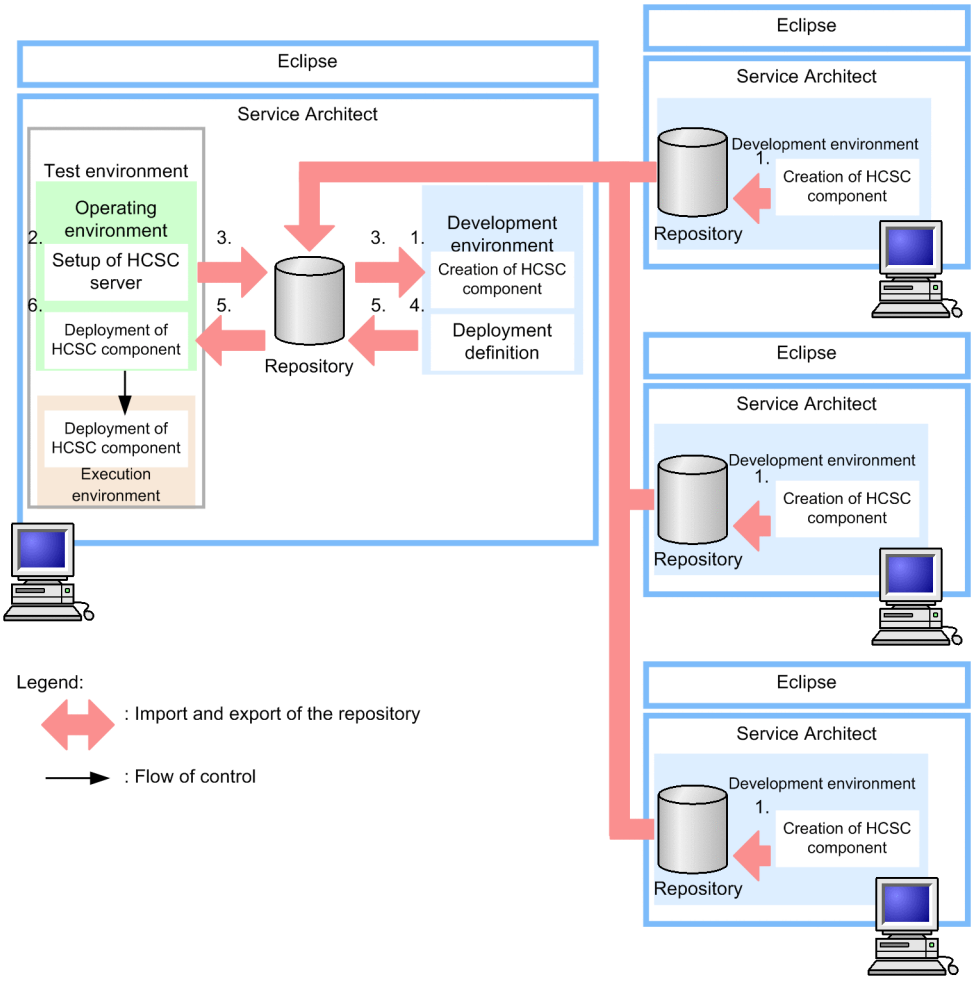
1. Create HCSC components in the development environment.
2. In the operating environment, set up HCSC servers and define the system configuration.
3. Export the repository from the operating environment, and import the system configuration definition, defined in the operating environment, to the development environment.
4. Based on the system configuration definition defined in the operating environment, define the part of the system configuration in which components are deployed, and then update (deployment definition).
5. Export the repository having the deployment definition defined in the development environment, to the operating environment.
6. Based on the deployment definition defined in the development environment, deploy components in the execution environment from the operating environment.

To transfer the information between environments, use the repository that stores the information shared between environments. The repository is saved and read in a ZIP file format via a medium.

(3) Relationship between the development environment and test environment

In the development environment, the information of the HCSC components created on multiple computers can be collected as one through a repository. Moreover, you can set up a test environment for executing testing and debugging of the created HCSC components on the same computer on which you have set up the development environment. The test environment is a simple environment that is required for testing and can be set up at the same time. The following figure shows the relationship between the development environment and test environment.

Figure 10–6: Relationship between the development environment and test environment



For executing operations in the development and test environments, use the following procedure:

1. Create HCSC components in the development environment.
2. Set up the test environment (you can use the HCSC Simple Setup functionality to set up an HCSC server, and define the system configuration). When setting up the test environment also, it is assumed that the execution environment is the actual environment.
3. Export repository from the test environment, and import the system configuration definition defined in the test environment, to the development environment.
4. Based on the system configuration definition defined in the test environment, define the part of the system configuration in which components are deployed, and then update (deployment definition).
5. Export the repository having the deployment definition defined in the development environment, and then import to the test environment.
6. Based on the deployment definition defined in the development environment, deploy components in the test environment.

Also the repository is used for transferring information between such environments. Since the development environment and test environment are set up on the same computer, information is transferred without the use of a medium.

(4) Relationship between the test environment and production environment

With the Service Platform, you first build a test environment and then perform testing and debugging. You can use the HCSC Easy Setup functionality to build the test environment.

In actual system development, you need to configure the production environment in addition to the test environment. Therefore, you are required to move the repository used in the test environment to the production environment.

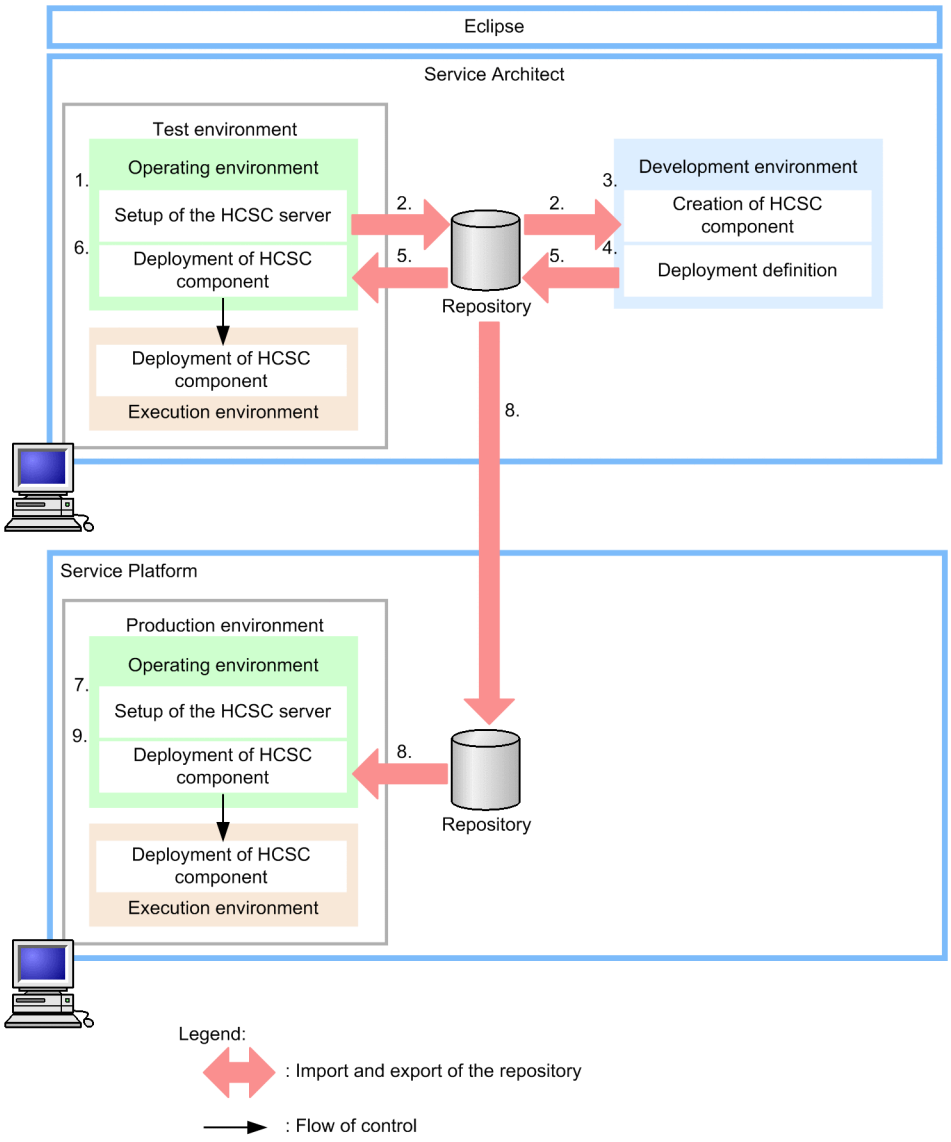
In the following test environments and production environments, if the settings for whether to use the database and Reliable Messaging are the same, you can move the repository used in the test environment to the production environment as it is:

- When both the database and Reliable Messaging are used in the test environment and production environment
- When both the database and Reliable Messaging are not used in the test environment and production environment
- When the database is used but Reliable Messaging is not used in the test environment and production environment

Note that in the test environment and in the production environment, you can move the repository even if the settings for whether to use the database and Reliable Messaging are not the same. For details, see the manual *Service Platform System Setup and Operation Guide*.

The following figure shows how to move a repository when the settings for whether to use the database and Reliable Messaging are the same in the test environment and the production environment:

Figure 10–7: Procedure for moving a repository when the settings for whether to use the database and Reliable Messaging are the same



1. In the test environment, set up HCSC servers and define the system configuration.

You can use HCSC Easy Setup definitions to define the setup of HCSC server and the configuration of systems. However, when configuring a system using a database and not using Reliable Messaging, you cannot use the HCSC Easy Setup functionality.

2. Export the repository from the operating environment, and import the system configuration definition that is defined in the test environment to the development environment.
3. Create HCSC components in the development environment.
4. Based on the system configuration definition defined in the test environment, define the part of the system configuration in which components are deployed, and then update (deployment definition).
5. Export the repository having the deployment definition that is defined in the development definition, and import the repository to the operating environment.
6. On the basis of the deployment definition that is defined in the development environment, deploy HCSC components on the test environment.
7. In the production environment, set up HCSC servers and define the system configuration.

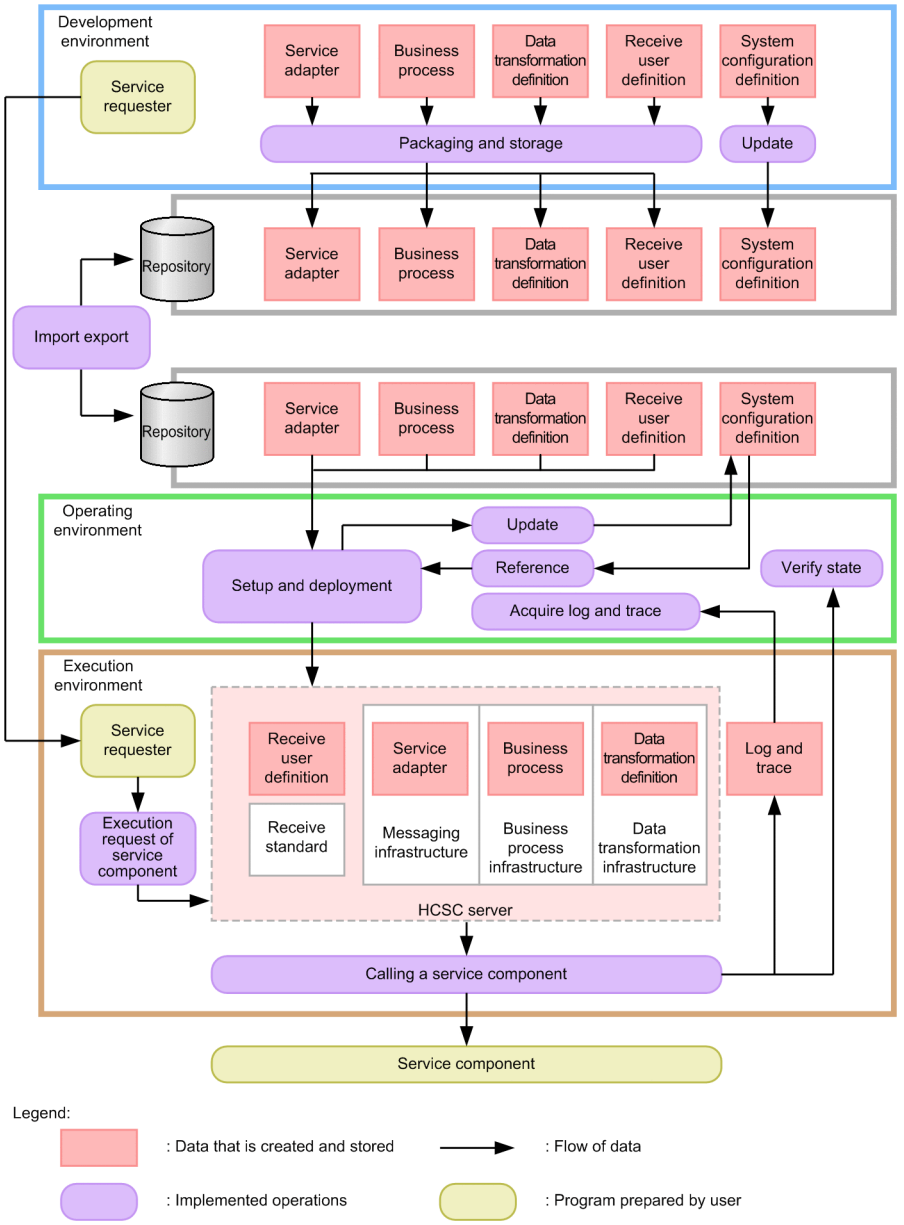
   In the test environment and the production environment, specify the same settings for whether to use the database and Reliable Messaging.
8. Export the repository included in the deployment definition that is defined in the development environment, and then import to the operating environment.

   When you use standard import, you can move the repository of the test environment to the production environment, even if the HCSC server name or IP addresses are different.
9. On the basis of the deployment definition that is defined in the development environment, deploy HCSC components on the production environment.

10.2.2 Relationships Between System Operations and the Individual Environments

A service platform configures the system as a whole by setting up the development, operating, and execution environments so that they are interrelated. The next figure shows how the overall operations of a service platform are related to the development, operating, and execution environments.

Figure 10–8: Relation between the operating and development environments and between the operating and execution environments of a system



Use a repository to import the information that was defined in the development environment into the operating environment. You then set up and deploy the imported information in the execution environment. When a service component execution request is received from a previously created service requester, the HCSC server calls the service component. From the operating environment, you can check and manage the status of service component calling by collecting log and trace information.

The following individually describes the development, operating, and execution environments of the service platform, shown in Figure 5-8.

(1) Development environment

In the development environment, developers create the HCSC components and system configuration definitions required for executing service components and business processes.

HCSC component is a generic term that refers to service adapters, DB adapters, and business processes created in the development environment.

A *system configuration definition* defines how a HCSC component is to be deployed in the execution environment. A system configuration definition includes setup information for the HCSC servers and clusters in the operating environment, and information on how HCSC components are to be deployed in the execution environment. The development environment uses a repository to acquire setup information that was created or updated in the operating environment, and defines how HCSC components are to be deployed.

The HCSC components that have been created are assembled in an EAR file. An *EAR file* assembles files related to HCSC components so that these components can be deployed in the execution environment. The process of creating an EAR file is called *packaging*. The packaged EAR file is stored in a repository.

A *repository* is a directory that stores defined information. You can use repository management functions (repository import and export functions) to pass the stored information between the development environment and the operating environment.

The development environment also creates *service requesters* that accept request messages for executing service components in the execution environment, and that send these messages to DB adapters and business processes.

(2) Operating environment

The operating environment reads the EAR file created in the development environment from the repository, and deploys the file to the execution environment. The operating environment also sets up the HCSC servers that will be used in the execution environment.

After operations are started, the operating environment starts and stops the system, monitors the system status, and collects log and trace information.

(3) Execution environment

The execution environment calls service components and business processes via the HCSC server in response to request messages accepted by the service requester, and executes business operations. The *HCSC server* includes a messaging infrastructure, a business process infrastructure, and a data transformation infrastructure. In addition, HCSC components that were created in the development environment are deployed from the operating environment to the HCSC server.

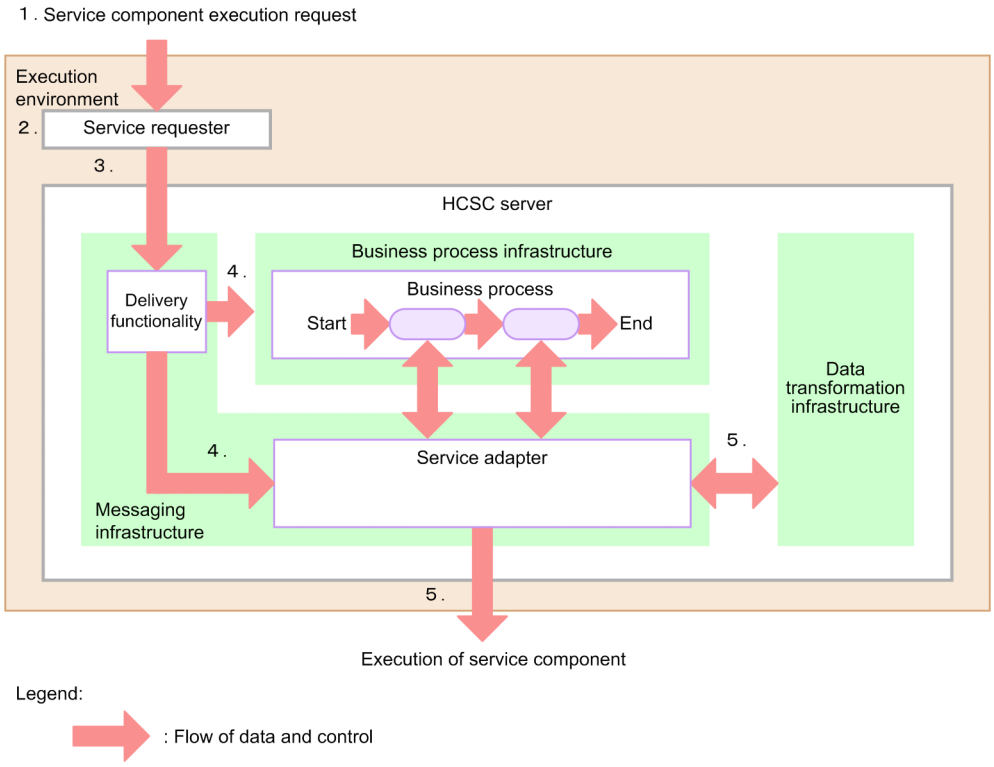
When a service requester accepts a request message, the message is sent to the messaging infrastructure. After that, the distribution function of the messaging infrastructure sends the request to the appropriate adapter or business process, according to the request message.

When a request message is for a request to a business process, the request message is sent to the business process infrastructure. The business process infrastructure then sequentially calls service components via the messaging infrastructure according to the business process definition.

If a service component is set up so that data transformation is performed when the service component is executed, the execution environment uses the data transformation infrastructure to transform the data, and then the service component is executed.

The next figure shows the control flow in the execution environment.

Figure 10–9: Control flow in the execution environment



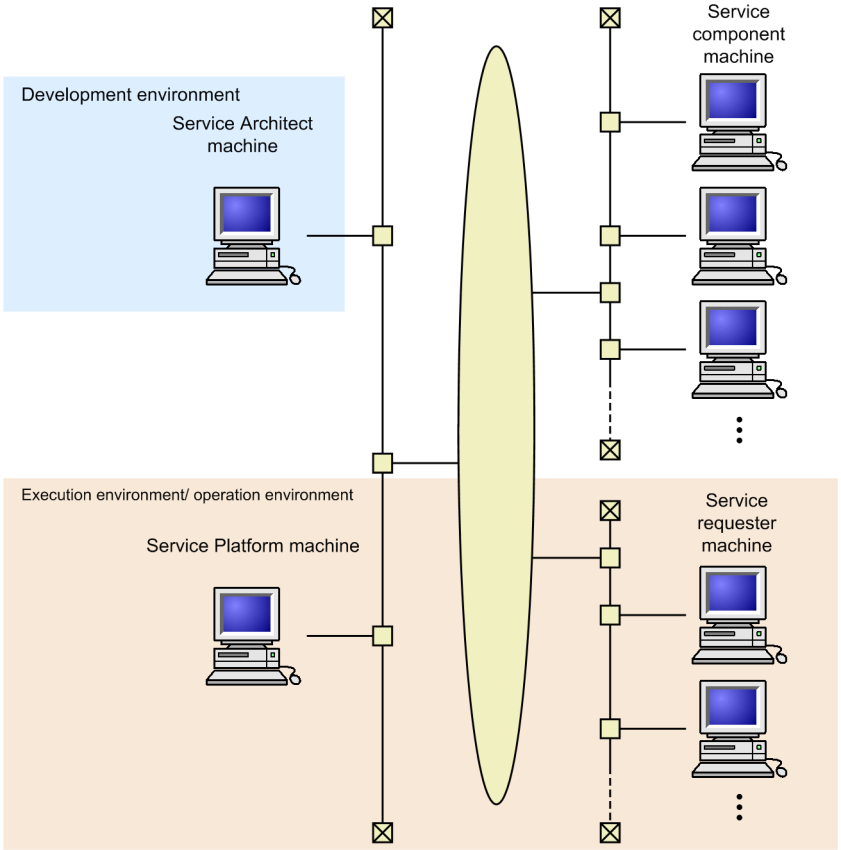
1. A business operations operator requests execution of a service component or business process.
2. The service requester accepts the request message.
3. The service requester sends the request message.
4. The request message is sent to the appropriate service component or business process by the distribution function.
5. If necessary, the data of the request message is transformed according to the data transformation definition, and the service component is called.

10.2.3 Network configuration and relationship between each environment

The service platform consists of a development environment, operating environment, and execution environment. The service platform connects with the network and configures the system. We recommended building the development environment, operating environment, and execution environment on separate machines.

The following figure shows each environment to be built with the service platform, and the hardware configuration:

Figure 10–10: Configuration of each environment and hardware

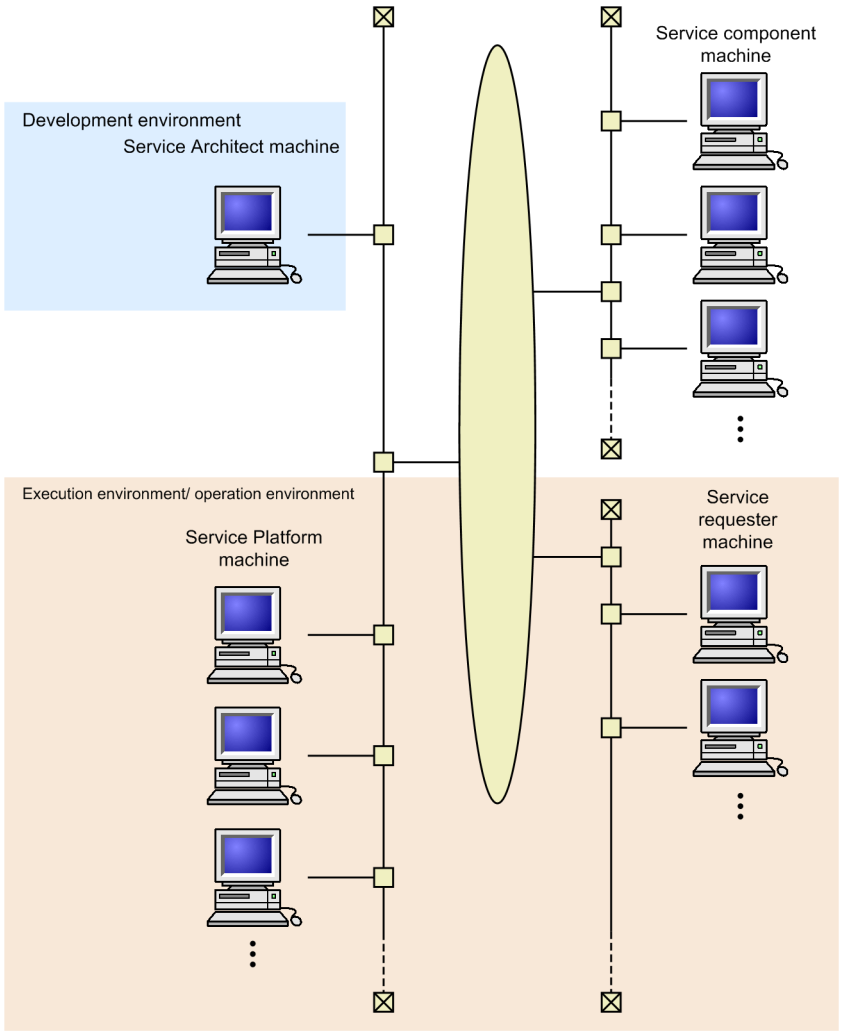


You can build the development environment on a different network as well.

Note that, in execution and operating environments, you can configure a cluster combining two HCSC servers and build a system with a redundant and highly reliable configuration.

The following figure shows each environment when a cluster is configured to combine HCSC servers and hardware configurations:

Figure 10–11: Each environment and hardware configuration (When configuring a cluster combining HCSC servers)



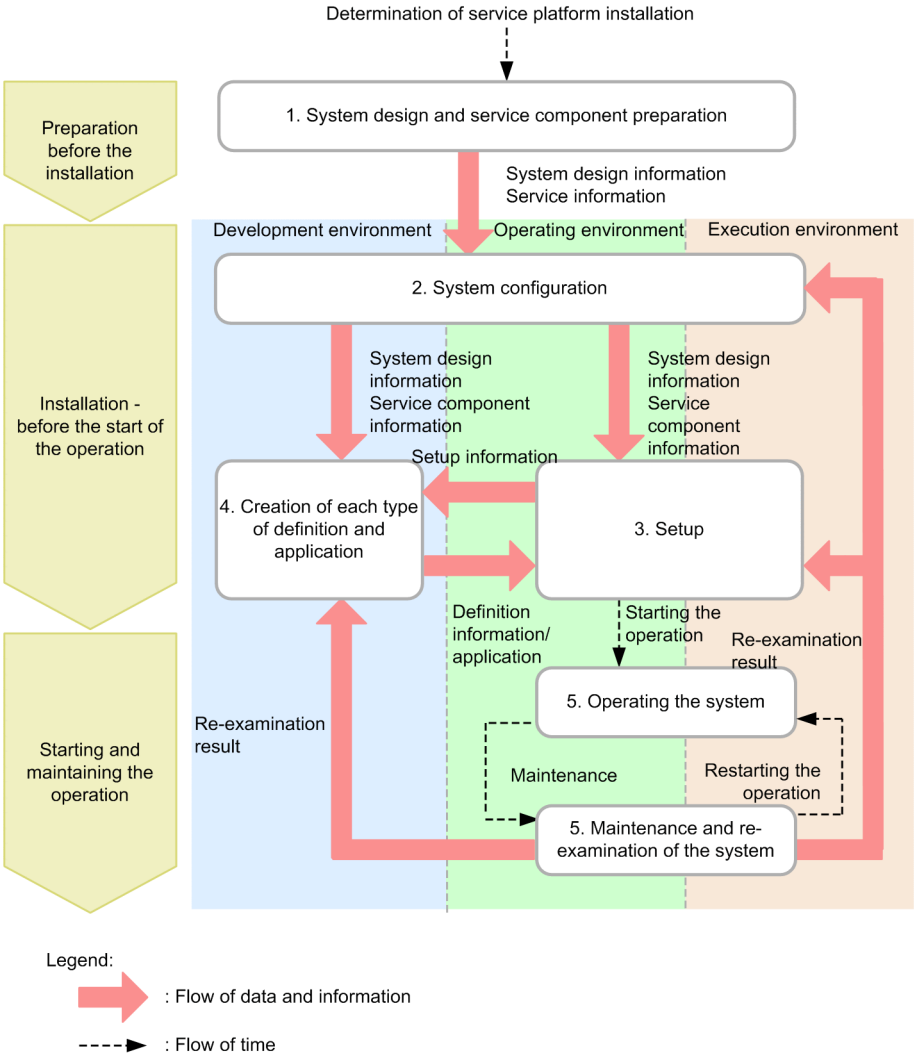
You can also build a development environment in a different network.

For details on systems in which a cluster configuration is used, see *1.4.1 Redundant configuration of HCSC server in which load-balancing functionality is used* in the manual *BPM/ESB Service Platform Overview*.

10.3 Life Cycle of Systems with a Service Platform Installed

The following figure depicts the life cycle of a system in which a service platform is installed.

Figure 10–12: System life cycle of the Service Platform



The life cycle of a system in which a service platform is installed consists of the following phases:

- System design and service component preparation
- System configuration
- Setup
- Creation of various definitions and applications
- System operations
- System maintenance and review

The phases of this life cycle take place as described below.

1. System design and service component preparation
   After you decide to install the service platform, analyze the business operations to be executed in the system and convert the business applications into services. Design the system according to how the services are to be used.

2. System configuration

   Configure the development environment, the operating environment, and the execution environment based on the design information and service component information of the system decided upon in step 1.

3. Setup

   Prepare to start system operations by setting up the operating and execution environments. In addition to the design information and service component information of the system devised in step 1, also use the definition information and applications that were created in the development environment.

4. Creation of various definitions and applications

   Develop the various definitions and applications required to operate the system.

   In developing the definitions and applications, use the setup information of the operating and execution environments, in addition to the application design information and service component information devised in step 1.

5. System Operations, Maintenance, and Review

   Start system operations. During system operations, perform daily operations as well as periodic system maintenance.

   Review the system any time business operations need to be modified, or when the system scale needs to be changed. After reviewing the system, if you decide to reconfigure it and set it up again, or add or modify certain definitions and applications, carry out the procedures described in steps 2 to 4 before restarting system operations.

Continue this process until you decide to renew or discontinue use of the service platform.

The following subsections provide an overview of the various stages (procedures).

10.3.1 System Design and Service component Preparation

Once you have decided to install a service platform, analyze the business operations you intend to execute in the system and convert them to services. With regard to existing business operations applications, you can consider the reusability of each application and use it by converting it to a service.

If you plan to create new services, you can set up a system with higher availability by considering the potential reusability and durability of each service when you create it.

After choosing and creating the services that will be used, design the system according to how the services will be used.

10.3.2 System Configuration

Configure the development, operating, and execution environments based on how you have designed the system.

(1) Configuring the Development Environment

Configure the development environment so that you can use it to develop various definitions and applications.

The development environment includes installation of the prerequisite software and Service Architect, embedding all the plug-in used for various definitions, and the setup of a repository.

(2) Configuring the operating environment

Configure the operating environment so that it can receive information from and exchange it with the development environment. Also, configure the operating environment so that setup of the execution environment can be executed from the operating environment.

During this stage you install the prerequisite software programs and Service Platform, create the definition file required for setting up the execution environment, and set up the repository.

(3) Configuring the execution environment

Configure the execution environment so that it can implement the definitions and applications it will receive from the development and operating environments.

This stage involves installing the prerequisite software programs and Service Platform, setting up the J2EE server, and setting up the database.

10.3.3 Setup

Prepare to start system operations by setting up the execution environment from the operating environment.

The setup of the HCSC server and the setup of executing environment of the defined information that is created in the development environment will be included.

Tip

In the operating environment, you must set up the definition information created in the development environment in the execution environment. Conversely, when you create various definitions in the development environment, use the information for the setup that was carried out in the operating environment.

When setting up the execution environment by using the operating environment and creating various definitions in the development environment, use the repository to transfer information between the operating environment and the development environment.

10.3.4 Creating Various Definitions and Applications

Create the various definitions and applications necessary for setting up the operating and execution environments, and for starting actual operations.

The definitions you create at this stage include the following: definitions for using service components (service adapter and DB adapter definitions), business process definitions, definitions for deploying various definitions to the execution environment (deployment definitions), and applications for accepting business operations execution requests and sending execution requests to service components in the execution environment (service requester applications).

Tip

The development environment contains information to be defined based on information on the execution environment setup that was executed in the operating environment. Conversely, when you set up the execution environment in the operating environment, use the information that was defined in the development environment.

When you set up the execution environment by using the operating environment and creating various definitions in the development environment, use the repository to transfer information between the operating and development environments.

10.3.5 System Operations, Maintenance, and Review

After you create the various definitions and applications in the development environment and complete the setup of the execution environment in the operating environment, you can start system operations.

During system operations, you perform daily operations, system maintenance, and system review processes as needed.

Daily operations involve daily startup and shutdown of the execution environment, as well as process monitoring and log creation to ensure stable system operations.

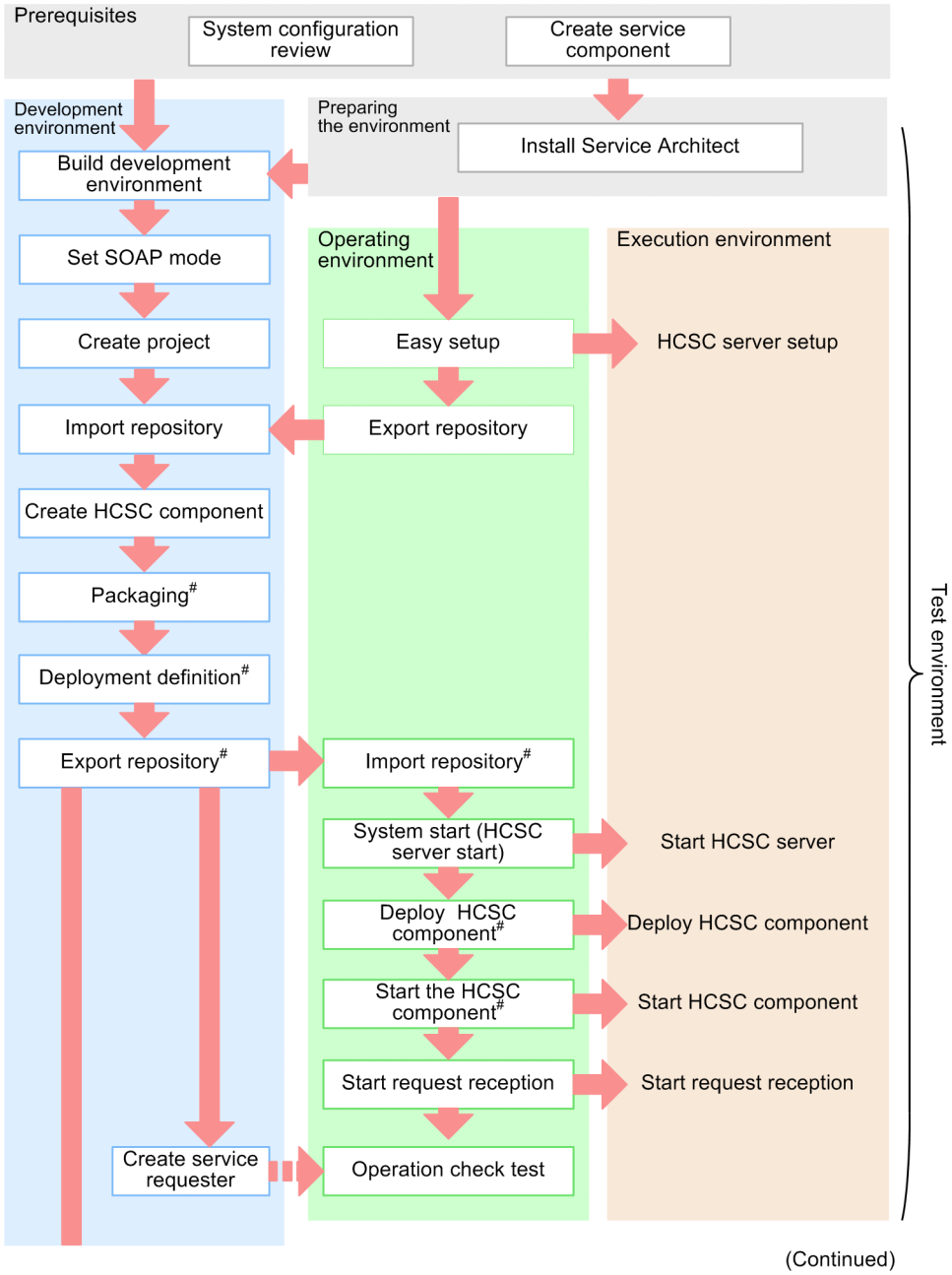
System maintenance and review involve modifying the configuration of each environment, adding and modifying definitions and applications, and correcting any problems that occur.

You can also operate a service platform more efficiently by linking it with JP1.

10.4 Work flow from development to actual operation

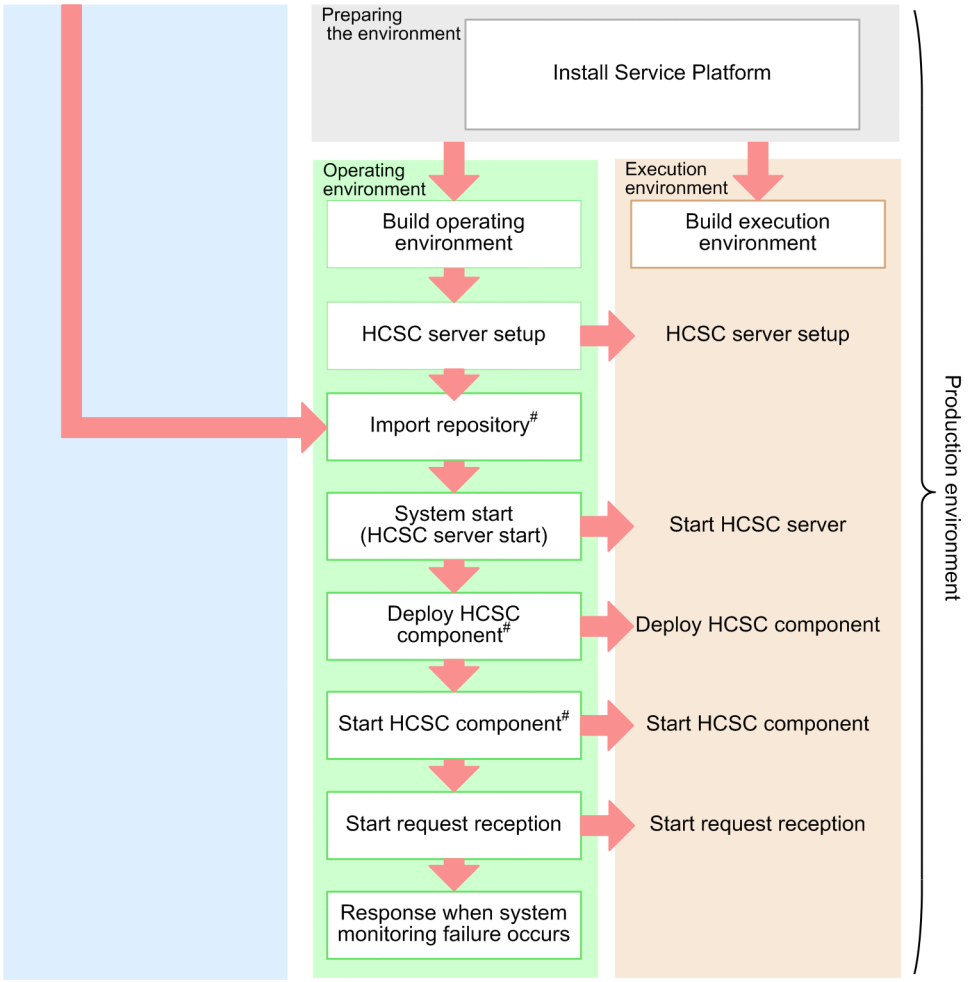
The following figure shows the work flow from development to actual operation of a system that uses the service platform.

Figure 10–13: Work flow from development to actual operation



(Continued)

(Continued)



#

You can also execute these operations in a batch in a development environment. However, you can execute a batch operation only during system development or from the time of a unit test to the time of the join test. For details, see *7.5 Batch execution of processes for deploying HCSC components on the HCSC Server and then starting* in the *BPM/ESB Service Platform Basic Development Guide*.

After the prerequisite preparation, perform the operations necessary for the development environment, operating environment, and execution environment of the service platform.

For details about the operations necessary for a development environment, see the *BPM/ESB Service Platform Basic Development Guide* and *BPM/ESB Service Platform Reception and Adapter Definition Guide*. For details about the operations necessary for an operating environment and execution environment, see the *Setup and Operation Guide.*

Index

Numerics

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W

X