Interstage Collaboration Ring

Process Manager Design Guide
Preface

+ Purpose of this Document

This document has been prepared for Interstage CollaborationRing Process Manager (hereafter, referred to as CollaborationRing PM).

This product is designed to achieve Enterprise Application Integration (EAI), Business to Business (B2B - an XML-based system for trade between companies) integration, and Business Process Management (BPM) integration. It is one of the component parts of the Integration Layer, part of the Interstage Software Platform suite of products.

This document describes the basic information required to design and construct an integrated system by installing CollaborationRing PM.

+ Who Should Read this Document?

This document is intended for users who are planning to install CollaborationRing PM, as well as those who are planning to use it to design a variety of integrated systems.

It is assumed that readers of this document have a basic knowledge of the following.

- Windows(R)
- UNIX
- Interstage Application Server
- C, C++, and Java
- Internet
- Object-oriented technology
- Relational databases
- Web servers

+ Organization of this Document

This document is organized as follows:

- Chapter 1 Chapter 1 Basic System Design
  Describes the basic configuration of CollaborationRing PM and the basic concept of system design for installing CollaborationRing PM.

- Chapter 2 Chapter 2 Process Rule Design
  Describes how to design process rules, which govern how multiple activities are linked.

- Chapter 3 Chapter 3 Application Design
  Describes how to write applications that run under the Process Flow Manager.

- Chapter 4 Chapter 4 Application to Business Design
  Describes how to design business applications that assemble and disassemble forms.

- Chapter 5 Chapter 5 Working Environment Setup and Application Control Operations
  Describes working environment settings and system administration of Process Flow Manager.

- Chapter 6 Chapter 6 The Enterprise Application Pipeline
  Describes the functions provided by the Enterprise Application Pipeline (EAP), and how to use those functions with applications.

- Chapter 7 Chapter 7 Using Adapter Kit to Support Constructing the Adapter
  Describes the functions provided with Adapter Kit, and the programs that can be created with them.

+ Position of this Document

This document is only one component of the information set provided for Interstage CollaborationRing PM. The full information set consists of the following documents:

- Interstage CollaborationRing Process Manager Guide
  (hereafter, referred to as Interstage CollaborationRing PM Guide)

- Interstage CollaborationRing Process Manager Design Guide (this document)
  (hereafter, referred to as Interstage CollaborationRing PM Design Guide)

- Interstage CollaborationRing Process Manager Operation Guide
  (hereafter, referred to as Interstage CollaborationRing PM Operation Guide)

  (hereafter, referred to as Interstage CollaborationRing PM Reference Manual (Process Flow Manager Edition))

  (hereafter, referred to as Interstage CollaborationRing PM Reference Manual (Enterprise Application Pipeline Edition))

- Interstage CollaborationRing Process Manager Web Services Aggregator Guide
  (hereafter, referred to as Interstage CollaborationRing PM Web Services Aggregator Guide)
+ **Product Names**

Actual names of the products and the naming conventions used in this document are as follows:

- Interstage CollaborationRing Process Manager is referred to as CollaborationRing PM.
- MessageQueueDirector is referred to as MQD.

+ **Notations Used in this Document**

In this document, the following expressions are used:

- Points to note
  Things that require your attention are expressed in the following format:

  ![Notice]

  Things that require your attention are described. Make sure to read this.

- Reference information
  Reference information is expressed in the following format:

  ![Point]

  Hints and tips are described.

The table below explains symbols commonly used in this document.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>[ ] (square brackets)</td>
<td>Encloses an option that can be omitted.</td>
</tr>
<tr>
<td>-&gt; (arrow)</td>
<td>Indicates the result of an operation.</td>
</tr>
<tr>
<td>{</td>
<td>} (vertical bar and curly brackets)</td>
</tr>
</tbody>
</table>

+ **Trademarks**

Trademarks of other companies:

<table>
<thead>
<tr>
<th>Product</th>
<th>Trademark/Registered Trademark</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microsoft, Windows, and Windows Server</td>
<td>Trademarks or registered trademarks of Microsoft Corporation in the U.S.A. and other countries.</td>
</tr>
<tr>
<td>Java, other trademarks containing Java</td>
<td>Java and other trademarks containing Java are trademarks of Sun Microsystems, Inc. in the U.S. or other countries.</td>
</tr>
<tr>
<td>Netscape, Netscape Navigator, and Netscape Communicator</td>
<td>Registered trademarks of Netscape Communications Corporation in the U.S.A and other countries.</td>
</tr>
<tr>
<td>ORACLE</td>
<td>Registered trademark of Oracle Corporation, USA.</td>
</tr>
<tr>
<td>UNIX</td>
<td>UNIX is a registered trademark in the U.S.A and other countries, licensed exclusively through X/Open Company Limited.</td>
</tr>
<tr>
<td>CORBA, Object Management Group, OMG, and ORB</td>
<td>Trademarks or registered trademarks of the Object Management Group in the U.S.A and other countries.</td>
</tr>
<tr>
<td>Rational and Rational Rose</td>
<td>Registered Trademarks of Rational Software Corporation.</td>
</tr>
<tr>
<td>EBF workflow linkage</td>
<td>Copyright(C) 1999 Information-technology Promotion Agency, Japan</td>
</tr>
<tr>
<td>IWF process history storage</td>
<td>Copyright(C) 2000 Information-technology Promotion Agency, Japan</td>
</tr>
<tr>
<td>These products are merchandised from the results of work by Information-technology Promotion Agency, Japan (IPA)</td>
<td></td>
</tr>
<tr>
<td>Fujitsu and IPA share the copyright of these products.</td>
<td></td>
</tr>
<tr>
<td>Product</td>
<td>Trademark/Registered Trademark</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Sun, iPlanet</td>
<td>Sun and iPlanet are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. or other countries.</td>
</tr>
</tbody>
</table>

All other trademarks or registered trademarks used in this document are trademarks or registered trademarks of the respective owner.

+ Comment of this Document

In this document, Interstage may refer to the same software product as Interstage Application Server.

In this document, Enterprise Application Pipeline is referred to as EAP and Process Flow Manager is referred to as CRF.

+ Export Control

This document contains technology relating to strategic products controlled by export control laws of the producing and/or exporting countries. This document or a portion thereof should not be exported (or re-exported) without authorization from the appropriate government authorities in accordance with such laws.

+ High Risk Activity

The Customer acknowledges and agrees that the Product is designed, developed and manufactured as contemplated for general use, including without limitation, general office use, personal use, household use, and ordinary industrial use, but is not designed, developed and manufactured as contemplated for use accompanying fatal risks or dangers that, unless extremely high safety is secured, could lead directly to death, personal injury, severe physical damage or other loss (hereinafter "High Safety Required Use"), including without limitation, nuclear reaction control in nuclear facility, aircraft flight control, air traffic control, mass transport control, medical life support system, missile launch control in weapon system. The Customer, shall not use the Product without securing the sufficient safety required for the High Safety Required Use. In addition, Fujitsu (or other affiliate's name) shall not be liable against the Customer and/or any third party for any claims or damages arising in connection with the High Safety Required Use of the Product.

+ Copyright

Copyright (C) 1999-2003 Fujitsu LIMITED

First Edition, October 2003

+ Important Notice

- Do not copy this document without permission.
- This document may be changed without notice.
Chapter 1 Basic System Design

This chapter explains the basic configuration of CollaborationRing and basic system construction concepts associated with its installation.

### 1.1 CollaborationRing Configuration

The composition of CollaborationRing PM components is illustrated in the diagram below.

#### 1.2 Process Flow Operation

This section describes the composition and operating environment of Process Flow Manager (occasionally abbreviated as "CRF" in this manual), the core component of CollaborationRing PM.

#### 1.2.1 Process Flow Manager Component Structure

Process Flow Manager consists of basic functions for process integration and extended functions.

**+ Basic Functions**

The basic functions described below are essential to the successful running of CollaborationRing PM.

**++ Client-type API Libraries**

Client-type API libraries are used by applications to execute processes or UI activities that are active within processes. Applications can take the following forms:

- Applications running on Windows(R) PC client machines
- CGIs, servlets, and other applications running on Web server machines
These libraries can also be used from applications running on servers, including automatic execution applications described later in this manual.

++ Server-type API Libraries

Server-type API libraries make it easy to read and create form data (XML documents) transferred by automatic execution applications via the IDL interface.

++ Control Server

The control server controls process flow, which is central to process automation. In accordance with process definitions, it calls automatic execution applications, accepts generated forms and tasks from the person in charge, and calls subprocesses.

There is also a Collection Server function (one of the components of the global tracking function) to perform batch merging and management of multiple event data obtained by the control server. Normally, the Collection Server is installed on a different machine from that running the control server, however both servers can be installed on the same machine.

++ HTTP Bridge

The HTTP Bridge is a gateway function to connect a Web server and a control server using CORBA. It operates on the Web server.

To simplify the application control of the Process Flow Manager, we recommend that a Web server be set up specifically for the HTTP Bridge.

++ CR Administration Client

The CR Administration Client possesses the following functions to manage the operation of the control server.

- Management Tool
  A tool that uses a GUI to perform application control on the control server.
- Database capacity calculation tool for control server
  A tool that uses a GUI to calculate the capacity of a database on the control server, and outputs a database creation script.
- Database capacity calculation tool for Execution Server (OTS linkage server)
  A tool that uses a GUI to calculate the capacity of a database on the Execution Server (OTS linkage server), and outputs a database creation script. This tool is used when using the execution server (OTS linkage server) described in the extended functions below.
- Process Definition Tool
  A tool that uses a GUI to edit process definitions.
- Process Definition Add-in
  A tool that uses UML to define business processes that comprise Web services.
- EAP definition program
  This program is at the core of EAP, and defines the I/O mediums and format of the leading/following tasks data, association of leading/following tasks, and distribution destination of the data, etc. The definition is performed from the GUI window with connecting to the server on which EAP runs.

+ Extended Functions

The extended functions described below can be used to extend the capabilities of CollaborationRing PM. Use these functions only when required.

++ Execution Server (OTS linkage server)

The Execution Server (OTS linkage server) prevents duplicate execution of automatic execution applications. It possesses an execution control function that helps to prevent duplicate updates to an application database during the re-execution procedure that is performed when a problem occurs during the operation of an automatic execution application. The Execution Server (OTS linkage server) and the automatic execution application should be installed on the same machine. It is possible to distribute load by installing the Execution Server (OTS linkage server) on multiple machines. It is also possible to install the Execution Server (OTS linkage server) on the same machine running the control server.

++ Browser Linkage Function

The Browser Linkage Function enables form generation and the tasks of a person in charge (UI activity) to be accomplished from the browser by using HTML to create form generation documents and work documents (e-form HTML). This is restricted to fixed workflow tasks as prescribed in the Browser Linkage Function. Because there is no need to create a separate application, this is useful for defining processes when designing work systems and
confirming performance in workflow systems.

++ History send agent

One of the components of the global tracking function, this his function sends (or transfers) multiple event data obtained by the control server to the Collection Server at regular intervals. It runs on the control server.

++ Global History Query

One of the components of the global tracking function, this is a query function used to display and search for historical data on the Collection Server using a Web browser. It runs on the Web server.

++ Trading Analysis Function

Importing parts of the data of the management database on the Collection Server, execute the statistical work, and then on the Web browser display and analyze the following:

- Processing time counting on the form exchange between companies and for displaying of the result of statistical work

As the trading database should be located separately from the management database, it is necessary to prepare another database. It is also necessary to make settings in the trading database and Web server in order to process statistics and display the results in a Web browser.

1.2.2 Operating Environment

This section describes the operating environment required to run Process Flow Manager.

For details on installation and required software, refer to the Installation Guide.

+++ Operating Environment when using HTTP Communication

The following tables show connectivity with a Web server via HTTP communication.

[SSL Communications using CA: VeriSign]

<table>
<thead>
<tr>
<th>Web server API</th>
<th>Interstage Application Server InfoProvider Pro</th>
<th>Interstage HTTP Server</th>
<th>iPlanet Web Server, Enterprise Edition</th>
<th>Microsoft(R) Internet Information Services</th>
</tr>
</thead>
<tbody>
<tr>
<td>Without client authentication</td>
<td>Supported</td>
<td>Not supported</td>
<td>Supported</td>
<td>Not supported</td>
</tr>
<tr>
<td>With client authentication</td>
<td>Supported</td>
<td>Not supported</td>
<td>Not supported</td>
<td>Not supported</td>
</tr>
</tbody>
</table>

[PUT/DELETE method support]

<table>
<thead>
<tr>
<th>Web server API</th>
<th>Interstage Application Server InfoProvider Pro</th>
<th>Interstage HTTP Server</th>
<th>iPlanet Web Server, Enterprise Edition</th>
<th>Microsoft(R) Internet Information Services</th>
</tr>
</thead>
<tbody>
<tr>
<td>PUT/DELETE methods</td>
<td>Not supported</td>
<td>Not supported</td>
<td>Supported</td>
<td>Supported</td>
</tr>
</tbody>
</table>

1.3 EDI System Components

With CollaborationRing PM, you can construct EDI systems, particularly Internet-based EDI, that realize various types of electronic data interchange between companies.

Installing a system such as in the diagram below, you can construct speedy and inexpensive EDI systems that use encryption technology, digital authentication technology, international standard format conversion functions, and so on.
To operate FEDIT and EAP, install and run these components on the same machine as that running the Process Flow Manager control server.

### 1.3.1 Enterprise Application Pipelines

When using the Enterprise Application Pipelines (occasionally abbreviated as “EAP”), the task that performs the link is referred to as the leading task, and the task that is linked is referred to as the following task. The table below shows the types of leading and following tasks.

#### [Leading Tasks and Following Tasks]

<table>
<thead>
<tr>
<th>Leading task</th>
<th>Following task</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task created by task flow (UI) Message collection by file transfer Form generation from task created by task flow</td>
<td>Task created by task flow Message distribution by file transfer</td>
<td>The following protocols can be used in file transfer: · Built-in FTP file transfer function FTP The following data linkage is also available: · Asynchronous communications linkage function MQD · Plaintext mail linkage function Plaintext mail · File transfer function For files on same server</td>
</tr>
</tbody>
</table>

It is not possible to link one task to another just by simply passing data. It is necessary to provide mechanisms that convert data formats and transfer data to the applicable following task so that the following task can perform processing. To make this possible, the leading task must perform processing while being aware of the following task. However, in this type of arrangement, any changes to the following task must be reflected in the leading task.

The Enterprise Application Pipelines solve such problems and provide a mechanism to enable the leading task to link without being aware of the following task, while still being able to respond flexibly to additions and changes to tasks. For more details, refer to Chapter 6 The Enterprise Application Pipeline.
1.3.2 FEDIT

FEDIT is a component for converting company data into an appropriate format for exchange with another company. FEDIT uses industry standard formats for exchanging data, and can convert data into the customized formats that companies receiving the data might be using. FEDIT also provides the appropriate software for these conversions, as detailed in the table below.

### [Conversion Formats and Software in FEDIT]

<table>
<thead>
<tr>
<th>Format type for exchange</th>
<th>Format conversion software name</th>
<th>Defined software name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Converting to UN/EDIFACT and ANSI X.12</td>
<td>FEDIT/EDIFACT-SV</td>
<td>FEDIT/EDIFACT-TABLE</td>
</tr>
<tr>
<td>Converting to user-specific format</td>
<td>FEDIT/FL-SV</td>
<td>FEDIT/FL-TABLE</td>
</tr>
<tr>
<td>Converting to XML</td>
<td>FEDIT/FL-XML</td>
<td>FEDIT/FL-TABLE</td>
</tr>
<tr>
<td>When source data are in user-specific format</td>
<td>FEDIT/FL-XML</td>
<td>FEDIT/FL-TABLE</td>
</tr>
<tr>
<td>When source data are in XML format</td>
<td>FEDIT/XMLTRANSLATOR-SV</td>
<td>FEDIT/XMLTRANSLATOR-TABLE</td>
</tr>
</tbody>
</table>

For details on settings, refer to the appropriate manuals, as indicated below.

- Converting to UN/EDIFACT and ANSI X.12
  FEDIT GENERAL DESCRIPTION (EDIFACT)
- Converting to user-specific format
  FEDIT GENERAL DESCRIPTION (User-Specific Format)
- Converting to XML (source data in user-specific format)
  FEDIT GENERAL DESCRIPTION (XML)
- Converting to XML (source data in XML format)
  FEDIT GENERAL DESCRIPTION (XML-XML)

You can execute FEDIT from EAPs and from FEDIT/Controller in the default automatic execution applications. For details on FEDIT/Controller settings, refer to the FEDIT/Controller Operations Guide.

### 1.4 Trading Partner Manager

Trading Partner Manager (occasionally abbreviated as "TPM") provides a framework for Internet-based or XML-based electronic business transactions.

By installing TPM, you can use RosettaNet, ebXML, or CIDX for your electronic business transactions. In addition, by linking with Process Flow Manager, you can use your own company's systems and current EDIs to realize total business integration with all of your existing trading partners.

The diagram below gives an overview of TPM.
[Overview of TPM]

TPM consists of the following components:

- **TPM Server**
  Integrates with the Process Flow Manager control server to execute trading scenarios.

- **Transport**
  Includes communication control and security functions for the exchange of business documents with trading partner systems.

- **Administration Client**
  Manages TPM using the following functions:
  - **Scenario Definition Tool**
    Defines how your own company trades with a partner.
    The method is defined in a GUI window. Individual items that can be defined include company information, business document attributes, and process to run on receipt of requests or orders.
  - **TPM Management Tool**
    Lists and manages scenario definitions, processes, and error events.
    It is possible to confirm details such as current trading status and business document content using a Web browser.

- **TPMAPI Library**
  Used to operate TPM processes from user applications.

- **Profile Manager**
  Manages information set using the Scenario Definition Tool as profiles in the UDDI registry.

- **Scenario Manager**
  Manages scenarios created using the Scenario Definition Tool on the TPM server.

- **Business Templates**
  Templates for the design and construction of business operations.

For more details, refer to the TPM Guide.

### 1.5 Web Services Aggregator

You can easily make work systems to link with Web services using Web Services Aggregator.

Web Services Aggregator includes the following functions:

- **Process Definition Add-in**
  This is a graphic tool for defining processes that include Web services.
  You can define processes that include Web services using OMG’s Unified Modeling Language (UML).

- **Web Service Access Components**
  Normally, people intending to use Web services and UDDI require programming ability and expert knowledge in areas such as networking, XML and protocols. The Web Service Access Components make it possible to use Web services and UDDI by setting uncomplicated parameters.

- **Web Service Publishing Function**
  This is a function for publishing CollaborationRing PM business processes externally as Web services,
allowing those processes to be used from external systems.

- Web Service Performance Analyzer
  This is a function for analyzing the performance of Web services. The analysis results can be used as
  information to help in choosing trading partners.

The diagram below illustrates how a system to integrate with Web services is configured.

[System Configuration for Web Service Linkage]
For details on how to make environment settings for and use Web Services Aggregator, refer to the Interstage
CollaborationRing PM Web Services Aggregator Guide.

1.6 Other Functions
CollaborationRing PM also contains the following functions:

Linkage function with BizTalk Server
This function provides a link between CollaborationRing PM and Microsoft(R) BizTalk server. This gives users the
opportunity to link business processes in ways such as filling BizTalk server schedules from CollaborationRing PM
and performing CollaborationRing PM processes from BizTalk server.

Adapter Kit
This function provides a means of easily constructing adapters to link CollaborationRing PM with existing systems.
Use of CollaborationRing PM API is supported. You can also create adapter to link with systems utilizing the latest
technologies, such as SOAP and EJB.

1.7 High-reliability Systems
This section describes High-reliability system created by utilizing CollaborationRing PM.

1.7.1 Constructing a High-reliability System using Cluster Systems
By using the High Availability function (abbreviated below as "HA function") of CollaborationRing PM in each type of
cluster system, two server machines with CollaborationRing PM installed can be operated as a single virtual server
machine. This allows the construction of high-reliability systems with high availability.

The cluster system supported by CollaborationRing PM is in 1:1 operation-standby mode. In other words, the server
machine providing an application service during normal operation is called the active node, and the server machine
that provides an application service when an error occurs in the active node is called the standby node.
[Hot Standby Configuration using the HA Function]

For information on supported cluster systems, refer to the Installation Guide.

**1.7.2 Server Functions that Support HA Function**

The CollaborationRing PM server functions that support using the HA function are listed below.

- Process Flow Manager Control Server
- Process Flow Manager Execution Server (OTS Linkage Server)
- Process Flow Manager HTTP Bridge
- Process Flow Manager History Send Agent
- Process Flow Manager Global History Query
- Process Flow Manager Trading Analysis
- EAP
- TPM
- FEDIT/Controller
- Adapter Kit (for Component Objects)
- Web Services Aggregator

**1.7.3 High-reliability System Operation**

To use the HA function, it is necessary to register Interstage Application Server, DBMS, each CollaborationRing PM server function, user applications, and other resources as resources in the cluster system. These resources must also be registered together as an application group.

+++ Operation format for SunCLUSTER or PRIMECLUSTER

There are two types of SunCLUSTER or PRIMECLUSTER. One is Standby class provided for high availability and another is Scalable class provided for high performance. Standby class is used in CollaborationRing PM and it operates on 1:1 operation-standby mode. Standby class (1:1 operation-standby mode) provides one standby node for one active node (server machine).

**1.8 Global Tracking**

This section describes Global Tracking system created by utilizing CollaborationRing PM.

**1.8.1 Global Tracking using the Event Collection Method**

Individual business documents and EDI forms are transferred between business processes and are circulated through linked servers. Implementing tracking between organizations and enterprises allows the flow of documents to be followed across multiple servers. This function is referred to as global tracking. If jobs spanning multiple servers are performed via linkages between processes, this function can be used to enable a single operator to manage the progress status and reduce the time and effort required for business operations.

The Process Flow Manager adopts the event collection method for global tracking. In this method, the Process Flow Manager stores information in the control server database every time an activity is invoked. This information is periodically transferred to the server specified in advance as the Collection Server. Thus, users can refer to the
Collection Server to check the status of a batch of business documents that meet the specified conditions. Event data relating to the processing of a form is sent to a pre-determined Collection Server at regular intervals, where it can undergo batch querying. In addition, if processing results are not returned for some business documents due to partial failure of a server, the process up to which processing has advanced can be traced from the original place of issue of the document.

[Overview of Global Tracking using Event Collection]

Using the event collection method has the following benefits:

- Individual forms can be tracked even when processing spans multiple servers. As a result, if a server fails and the processing results are not returned, it is still possible for the issuer of the form to determine how far the processing of the form had progressed, even when the processing route is complex. It is also possible to investigate the status of multiple forms according to conditions specified by a user. For example, if a collection parameter value is specified, forms and business documents whose processing is approaching a deadline can be checked and delays detected early, and tracking can be accomplished for each client without complicated operations.

- Reduced operational workload. If the processing of tasks that span different servers (including EDI) can be linked, progress may be controlled by a single operator, resulting in faster operation and reduced labor. For example, error identification may have been previously handled by more than one person through telephone calls and log-reviewing procedures. By using functions such as global tracking and operation management tools, one person can determine the cause of errors, even if operations are spread over more than one server.

1.8.2 System Configuration

To employ Global tracking, a Collection Server needs to be provided to control the log data for each collected unit. Each control server sends the log (that is, event) data collected whenever an activity is invoked to this Collection Server at regular intervals, where it is managed centrally using a database. Users can query the Collection Server to search or track the status of multiple business documents that satisfy specified conditions. Data is sent to the Collection Server by a history send agent. Requests for information to the Collection Server are made using the history reference function, and the results can be viewed in a Web browser.

An example of a system configuration is shown below.
[Event Collection Tracking Method]

- Collection Server
  More than one Collection Server can be installed, and it is possible to collect log information from each installation. For example, for a business process that traverses multiple control servers, you can set a particular Collection Server to collect log information for a particular tracking range, so that you only need to refer to a particular level of a business process.
  It is possible to install a Collection Server for each control server, or to share a single Collection Server among several control servers. The Collection Server is normally constructed on a separate server from the control server, but it can be constructed on the same server as the control server. The Collection Server is provided as a function (collection control) of the control server.

1.8.3 Environment Setup

To use the global tracking function, it is necessary to create a send agent database on the control server, and a Collection Server database on the Collection Server. Also, in addition to the normal CollaborationRing PM environment setup, it is necessary to set information pertaining to global tracking in the environment of each CollaborationRing PM component.

The relevant components are shown below. Refer to the Interstage CollaborationRing PM Operation Guide for details of the settings.

- Control Server and Collection Server
- History send agent
- Global history query

1.8.4 Operating Notes

+ Managing Information using a Global Key

In the case of single forms, it is possible to search or track each form through the use of a form ID specified by the user. However, when the operation mode requires multiple related forms to be processed together at one time, it becomes necessary to use a unique identifier associated with each form. CollaborationRing can use a global key (process information) to relate forms to one another. Association is achieved by assigning the same global key to each related business process or form.

By using the global key when querying log data, it is possible to search or track the status of multiple forms that have the same key. Log data searches are made using the history reference function.
+ Erasing the Contents of the Collection Database

Global tracking uses a control database on the control server (known as the history send agent database) and a control database on the Collection Server (known as the collection database). Log (event) data collected by the control server is stored temporarily in the history send agent database on the control server. The history send agent monitors this history send agent database regularly to check for the presence of log data. If log data is present, it is sent to the Collection Server for collection. If the transmission is successful, the history send agent then deletes the corresponding data from the history send agent database on the control server.

At the Collection Server, collected log data accumulates in the collection database and is managed centrally from there. Log data that is no longer required must be erased at regular intervals using the application control command of the control server or the Management Tool, or it will become a burden on the database and lead to a shortage of available space.

+ Specifying the Log Data Collection used in Global Tracking

Specifying the log data collection used in global tracking is accomplished in the working environment settings of the control server. If "Collect log data" is specified in a control server working environment, log data for all processes operating in that server will be collected.

Even when "Collect log data" is specified in the control server working environment settings, it is still possible to prevent the collection of log data for specific processes. This is accomplished by using the Process Definition tool to specify "Do not collect log data" in the Process Definitions. Note that the setting to collect or not collect log data must be the same for all related processes. If the log data for some processes in a group of related processes is not collected, the parent-child relationship (i.e., the relationship between a parent process and its subprocesses) cannot be obtained from the log data, which makes it impossible to display process relationships when performing a global history query. The same is true for the working environment settings of a control server: The same settings must be used among all control servers running related processes.

1.9 Trading Analysis

This section describes Trading Analysis system created by utilizing CollaborationRing PM.

1.9.1 General Configuration

The configuration of the trading analysis function is shown in the below figure. The trading log collected on the control server is saved to the trading log temporary storage table in the collection server by the history send agent. During trading analysis processing, the function reads trading log from the table and sends the processed data to the trading analysis processing table. When trading analysis is demanded, the trading analysis processing calculates the requested data from processing tables and shows it.

[General Configuration of Trading Analysis]

1.9.2 Overview

With the trading analysis function, trading between enterprises is defined in advance as a process flow and statistical information on the flow of trade is provided. This information can then be used to work out problems affecting trade
and to improve business.

The following is a description of the functions provided by the trading analysis function to assist intercorporate trading executed in Interstage CollaborationRing:

- Aggregating processing time for determining the progress of forms, transactions and processes in intercorporate trade, and displaying statistical information (trading analysis function).

This function records the processing status of intercorporate trade as a trade log, then calculates the statistical information of the processing time for each process and displays it. This function works to find the location of bottlenecks in businesses' supply chains, and assists the optimization of supply chain and the improvement of business methods in trading processes. The type of trading that can be analyzed in this way is RosettaNet two-action type. This corresponds to the following PIPs in the range of the scenario templates provided with TPM:

- 2A9 Query Technical Product Information
- 3A1 Request Quote
- 3A2 Request Price and Availability
- 3A4 Request Purchase Order
- 3A5 Query Order Status
- 3A8 Request Purchase Order Change
- 3A9 Request Purchase Order Cancellation

Trading analysis can be performed for intercorporate trade using RosettaNet and ebXML.

1.9.3 Intercorporate Trading Analysis

Trading analysis is of practical use for improving business at both macro and micro levels. Analyzing trade between companies means analyzing intercorporate trading carried out by the exchange of forms on the basis of the business process model. The exchanged form information can be obtained wherever trading partners "meet" online, as in Interstage CollaborationRing. Transaction information (such as time and date of transaction, type of form and so on) is logged in the trading log, and the total time of trading or times of specific transaction parts can be obtained.

The Trading Analysis Function targets trading (Business Processes) executed according to a trading agreement between enterprises. Almost all the Business Process models are between two companies with a trade agreement, such as PIPs defined by RosettaNet. In the model, as a rule two enterprises are assumed to be involved; if trading is performed with more enterprises, the analysis is performed using a combination of 2-enterprise models.

[Concept of Intercorporate Trading Analysis]

Intercorporate trading analysis offers the following advantages:

- Overall reduction in transaction time (lead time)

In order to react quickly to fluctuations in demand, transaction time is reduced, stockpiling is minimized, and wasted opportunities are avoided.

To maximize the benefit to business procedures, analysis is required to detect and prioritize the processing sectors that are the most likely to receive the biggest improvements. Generally speaking, processes that take longer to complete and processes that are executed more often receive the greater benefits.

In trading analysis, the processing time for intercorporate trading (for RosettaNet and ebXML) and the number of executions (for Inter-Company Collaboration in ebXML) are displayed. You can use this information as the basis for determining which processing sectors to improve.

12
Note that, when improving processing sectors, you must decide whether or not to carry out improvements on the whole company. Because the trading business process models handled in trading analysis may only represent a part of the overall trade, you are recommended to investigate in advance what effect the improvement of a particular sector may have on the operations of the company as a whole (for example in terms of processing costs).

- Reduction in discrepancy in processing times for transactions
  Discrepancies in delivery times for goods can lead to problems such as waste in processes that are reliant on those times, and inability to fix delivery dates in advance.
  In trading analysis, the processing times are displayed as average, maximum, and minimum times. You can use this information as the basis for determining which processes take varying amounts of time and to target improvements accordingly.

The trading business process model is executed on the basis of the intercorporate trading definition defined by the TPM scenario definition tool. The Trading Analysis Function connects trading logs according to this definition information, then completes the functions of Trading Analysis. (For details on intercorporate trading definitions, refer to the TPM Guide.)

1.10 Linkage with Systemwalker

CollaborationRing PM links with Systemwalker/CentricMGR to provide support for message monitoring and the remote issuing of commands in the event of an error.

1.10.1 Functions Realized by Linkage with Systemwalker

- Configuration Management
  Network settings and the monitoring and control of devices and system can be accomplished via a GUI.
- Error management
  Network, system and application errors can be monitored centrally.

![Linkage with Systemwalker](image)

1.10.2 Settings in Systemwalker

To link CollaborationRing PM and Systemwalker/CentricMGR, you must set up the following environments.

- Monitoring system:
  The server where the Systemwalker/CentricMGR Operation Management Server is installed.
- Monitored system:
  The server where all the CollaborationRing PM servers that are to be monitored (i.e., control server, Execution Server (OTS linkage server), Web server, log server) are installed.
  Systemwalker/CentricMGR agent must be installed in this server as a 'working server'.

Setting the correct information in each system is one of the requirements for running CollaborationRing PM with Systemwalker/CentricMGR.

These settings should be customized to suit the operation. To do so, use the Systemwalker/CentricMGR menus and tools.
Refer to the Systemwalker/CentricMGR manuals for details.

**Note**
You must have administrator rights to make settings in either the monitored or monitoring system.
Chapter 2 Process Rule Design

This chapter explains how to design process rules, which govern how multiple activities are linked together. Designing process rules is referred to as designing a workflow. Process rules define the conditions required to select activities and the order of processes in activities.

Note that process rules are occasionally referred to as processes.

2.1 Elements and Concepts of Process Rules

The main elements of process rules managed with Process Flow Manager are:

- Process Definitions
- Processes
- Activities
- Form Data
- Users
- Access Control
- URI (HTTP connection, Direct connection)
- Subprocesses in a Unified Control Server
- Event Queue

2.1.1 Process Definitions

Process definitions contain activity and form data definitions and define workflow processing.

2.1.1.1 Process Definition Creation

The Process Definition Tool and Process Definition Add-in allow process definitions to be created graphically. The Process Definition Add-in uses UML to create process definitions that use Web services. The Process Definition Tool does not use UML, and is used to create process definitions that do not use Web services. The Process Definition Tool is used from a Windows screen. Refer to the online help of the Process Definition Tool and Process Definition Add-in for more details. To define one business process, the process rules are designed as follows:

- Use arrows or similar to express the flow of control between activities
- Specify the data input and output between the business document bus and the activities.

In practice, the components that define process rules are dragged and dropped onscreen to define the business process.

Normally, the Process Definition Tools described above are used to create process definitions. However, sometimes process definitions may need to fit the task more closely or be simpler, or users may prefer to use CASE or another tool to define processes. The Process Flow Manager is equipped with input and output functions, using an API that can define processes in an open format and XML expression, for use when the Process Definition Tools are not used.

Use of these functions enables the following:

- Users can reference process definition information to check for consistency.
- Users can copy process definition templates to create process definitions dynamically.
- Users can reference, create, and update process definition information to create user-preferred Process Definition Tools.
- Users can use batch translation of titles and so on between the process definitions entered in XML and the process definitions normally used. In addition, process definitions can be converted to XML and used as documents. This function is provided as the process definition XML/Binary conversion tool.

Note that this chapter and subsequent chapters describe operations using the Process Definition Tool.

2.1.1.2 Using Process Definitions

Process definitions are identified and managed by their title in the control server. Therefore, when using a Process Definition Tool to create a process definition, it is necessary to assign the process definition a unique title.

Process definitions contain the information shown in the table below.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process definition ID</td>
<td>Character string that uniquely identifies the process definition within the control server. It is automatically set by the control server when the definition is created. A unique process</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Name Description</td>
<td>definition ID is set for each version of the process definition.</td>
</tr>
<tr>
<td>Title</td>
<td>Process title.</td>
</tr>
<tr>
<td></td>
<td>The process definition title uniquely identifies a group consisting of different versions of the same process definition within the control server. Processing relating to a process definition is conducted according to the title of that process definition, ensuring that only the latest version of the process definition is processed.</td>
</tr>
<tr>
<td>Description</td>
<td>A detailed description of the series of tasks defined in the process definition, used, for example, to select the process definition to use when a process is generated.</td>
</tr>
<tr>
<td>Priority</td>
<td>The priority of the process.</td>
</tr>
<tr>
<td></td>
<td>The value specified here is used if no priority is specified for the generation of processes.</td>
</tr>
<tr>
<td>Owner</td>
<td>User (group) that can access a process definition.</td>
</tr>
<tr>
<td></td>
<td>Only owners can display or update process definitions, or generate processes based on that process definition.</td>
</tr>
<tr>
<td></td>
<td>When the owner is designated as &quot;default&quot;, other users within a defined role may generate processes based on this definition. Only the control server administrator may update the process definition.</td>
</tr>
<tr>
<td>Draft Form</td>
<td>Form used to create processes.</td>
</tr>
<tr>
<td>Default Form</td>
<td>Form used to browse processes.</td>
</tr>
<tr>
<td>Input data definition</td>
<td>Definition of process creation input data.</td>
</tr>
<tr>
<td></td>
<td>In accordance with the input data definition, the Process Flow Manager applies the input data to the process data during process creation.</td>
</tr>
<tr>
<td>Output data definition</td>
<td>Definition of output data as processing results when a process is referred to or when a process is completed.</td>
</tr>
<tr>
<td></td>
<td>In accordance with the output data definition, the system extracts output data from process data.</td>
</tr>
</tbody>
</table>

In addition to the above, the process definition contains activity definitions, role definitions, form definitions and data definitions.

### 2.1.2 Processes

A process is created when initial data is passed to a process definition, using the CreateProcess command of the client-type API library. The object created is called a process or process instance.

Users generate processes by using client-type APIs to create forms (CreateProcess). When a user generates a form more than once, a new process is created each time. These processes are distinguished by the process ID, an identifier automatically allocated by the control server when the process is created. This ID is reported to the user who created the process, and must be submitted to the control server when a user wants to retrieve information relating to processing status, or summarize results.
[Relationship between Process Definitions and Processes]
Processes contain the following information.

[Other Information Contained in a Process]

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Process ID</td>
<td>Character string that uniquely identifies the process within the Process Flow Manager. The process ID can be specified only when the process is created. If a process ID is not specified, the control server will assign one automatically. However, if a user attempts to create a process using an ID with the same value as an existing ID, an error occurs.</td>
</tr>
<tr>
<td>Global key</td>
<td>Unique character string associated with multiple processes. The global key can only be specified when a process is created. If a global key is not specified, the process ID will be assigned as the global key. When creating a subprocess, the global key of the parent process is also set for the new subprocess. By setting the same global key for related processes, when performing global tracking data queries it is possible to track the status of multiple processes using this key.</td>
</tr>
<tr>
<td>Title</td>
<td>As described in the table above, this is specified during process definition, and is used to indicate the content of the process in process lists.</td>
</tr>
<tr>
<td>Description</td>
<td>Described in the table, Information Contained in a Process Definition.</td>
</tr>
<tr>
<td>Priority</td>
<td>Described in the table, Information Contained in a Process Definition.</td>
</tr>
<tr>
<td>Status</td>
<td>Status of executing processes. You can change this status from client-type API. Refer to the “Process State Transitions” figure below for descriptions regarding status of execution and its transition status.</td>
</tr>
<tr>
<td>Draft Form</td>
<td>Described in the table, Information Contained in a Process Definition.</td>
</tr>
<tr>
<td>Default Form</td>
<td>Described in the table, Information Contained in a Process Definition.</td>
</tr>
<tr>
<td>Input data definition</td>
<td>As described in the table, Information Contained in a Process Definition, in accordance with the input data definition, the control server applies the input data to the process data during process creation.</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Output data definition</td>
<td>As described above, in accordance with the output data definition, the control server extracts output data from process data and returns it as processing results.</td>
</tr>
<tr>
<td>Custom attributes/fields</td>
<td>Data that can be used freely by the user application. These appear on the summary line of a workflow, and allow users to add additional information.</td>
</tr>
<tr>
<td>Creator</td>
<td>User who created a process. This attribute is set automatically in the control server when the process is created.</td>
</tr>
<tr>
<td>Event Log</td>
<td>Events that occurred during processing are added successively to the log.</td>
</tr>
<tr>
<td>Creation date</td>
<td>Date and time the process was created.</td>
</tr>
<tr>
<td>Form data</td>
<td>Data sent from and received by forms during a process.</td>
</tr>
</tbody>
</table>

**[Process State Transitions]**

- **Initialized state (open.notRunning.notStarted)**
  A process has been created but processing has not commenced. In this state, no activities exist in active state. It is possible to refer to or modify processes.

- **Suspended state (open.notRunning.suspended)**
  Processing has been halted temporarily after commencement. At least one activity is running. When a process is in suspended state, activities that are running are also suspended at the same time. When a process returns from a suspended state to an active state, activities also return to their previous state. It is only possible to refer to processes.

- **Active state (open.running)**
  Processing is being performed. A minimum of one running activity exists. It is possible to both refer to and modify a process. When the state changes from initialized to active, activities change to become active according to process definitions.

- **Completed state (closed.completed)**
  All running activities have finished, and processing has completed normally. It is only possible to refer to processes.

- **Terminated state (closed.terminated)**
  Processing was terminated by an application. When a process is terminated, all active, non-active and suspended (open.*) activities are terminated simultaneously. Automatic execution activities wait until the automatic execution application has finished processing. In this state it is possible to refer to processes only.

- **Aborted state (closed.aborted)**
  Processing was forcibly terminated by an application or tool such as the Management Tool. When a process is aborted, the activity is interrupted. Subprocess activities and automatic execution activities are interrupted without waiting for them to complete processing. Aborting processing may result in inconsistencies between the subprocess and automatic execution application states, and damage to process information, including process data.

### 2.1.3 Activities

Activities represent each task within the workflow. Activities are distinguished within a process by means of a unique activity title.
There are three distinct types of activity: For details, refer to 2.2 Functions Used in Process Rules.
- UI (User Initiative) activities
- Automatic execution activities
- Subprocess activities

Activities contain the information described in the following table:

[Information Contained in Activity Definitions]

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Activity ID</td>
<td>Character string that uniquely identifies the activity within the process with which it is associated. The control server automatically assigns this ID during definition.</td>
</tr>
<tr>
<td>Title</td>
<td>Short description of the activity, suitable for indicating the content of the activity in activity lists.</td>
</tr>
<tr>
<td>Description</td>
<td>Detailed description of the activity.</td>
</tr>
<tr>
<td>Priority</td>
<td>Number indicating the priority of the activity. The control server does not use this integer for control purposes. Instead, it is used as a sorting parameter, for example, when listing activities using a client application. The Process Flow Manager uses a value between 1 and 5, with 1 indicating the highest priority.</td>
</tr>
<tr>
<td>Status</td>
<td>Execution status of the activity. Refer to the &quot;Activity State Transition&quot; figure below for descriptions regarding status of execution and its transition status.</td>
</tr>
<tr>
<td>Input data definition</td>
<td>Definition of input data used to process an activity. In accordance with the input data definition, the control server extracts the input data from process form data.</td>
</tr>
<tr>
<td>Output data definition</td>
<td>Definition of output data as the result of processing an activity. In accordance with the output data definition, the control server returns output data as processing results to process form data.</td>
</tr>
<tr>
<td>Role</td>
<td>Applicable to UI activities only. Specifies the owners of an activity. Access permission to the activity is restricted to this user or group.</td>
</tr>
<tr>
<td>Form</td>
<td>Applicable to UI activities only.</td>
</tr>
<tr>
<td>Process definition</td>
<td>Applicable to subprocess activities only. Process definition title or URI of a subprocess.</td>
</tr>
<tr>
<td>Synchronization setting</td>
<td>Applicable to UI activities only. This setting determines whether to wait (synchronous) or not wait (asynchronous) for the completion of a subprocess.</td>
</tr>
<tr>
<td>Application Name</td>
<td>Applicable to automatic execution activities only. Registers the name in the CORBA application naming service.</td>
</tr>
<tr>
<td>Invocation method</td>
<td>Applicable to automatic execution activities only. Setting of application invocation method.</td>
</tr>
<tr>
<td>Execution Queue Name</td>
<td>Applicable to automatic execution activities only. Setting of queue name to manage execution of an application.</td>
</tr>
<tr>
<td>Mail</td>
<td>A UI activity will notify the user by mail when it has become active, and other activities send mail when an error has occurred in a subprocess or during automatic execution. The mail contains error messages or information relating to these transmissions.</td>
</tr>
<tr>
<td>Option arrows</td>
<td>Option arrows join an activity to other activities or nodes. The activity that becomes the optional transition destination is determined according to the activity processing results.</td>
</tr>
<tr>
<td>Default arrow</td>
<td>Single option arrow selected if an option is not specified when an activity is completed.</td>
</tr>
<tr>
<td>Expiration date</td>
<td>Expiration date of an activity.</td>
</tr>
</tbody>
</table>

Activities undergo state transitions as described below.

From a client application, it is possible to change the state of an activity to the completed state only. Transition to other states is carried out by the control server according to changes in the state of a process or process results.
### Activity State Transition

- **Initialized state (open.notStarted)**
  Workflow control has not reached the activity, and activity processing has not yet begun. It is only possible to refer to the activity.

- **Suspended state (open.suspended)**
  A process has been halted temporarily. It is possible to refer only to the activity.

- **Active state (open.running)**
  Processing is possible, or processing is being performed. It is possible to refer to or complete an activity. An automatic execution activity will call an automatic execution application, and a subprocess activity will create a subprocess form. A UI activity will wait for form data updating by an application using the client-type API. When processing has been completed, the state of the activity changes to the completed state.

- **Completed state (closed.completed)**
  Processing has finished. It is possible to refer only to an activity.

- **Terminated state (closed.terminated)**
  Processing has been terminated. It is possible to refer only to an activity.

- **Aborted state (closed.aborted)**
  Processing has been terminated forcibly.

### 2.1.4 Process Data

Process data (form data) are user data managed by the control server during workflow processing. The user can refer to and update process data by means of a client application or an automatic execution application.

Form data are stored and managed for each process. Classified and managed within a process as a variable, the initial value of the process data variable can be set in the process definition properties, and can also be set at form creation by the client application.

The type of process data variable varies according to the data to be stored, as indicated in the following table:

#### Variable Types

<table>
<thead>
<tr>
<th>Variable type</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text</td>
<td>Simple string data.</td>
</tr>
<tr>
<td>Binary</td>
<td>BASE64 encoded binary data.</td>
</tr>
<tr>
<td>XML document</td>
<td>XML fragments.</td>
</tr>
<tr>
<td>Attached file</td>
<td>Information in attachments.</td>
</tr>
<tr>
<td>String</td>
<td>A loosely defined category that can store data as for &quot;Text&quot;, &quot;Binary&quot;, and &quot;XML document&quot; above.</td>
</tr>
</tbody>
</table>

Which process data variable is passed to or received from the application is determined when the process definition is defined. Because, depending on the application, the persons in charge of developing and maintaining applications and process definitions may be different, it is sometimes preferable to handle applications and process definitions separately. In such a case, the variables for passing and receiving are omitted from the definition of the activity in the process definition. By omitting the variables, all variables in a process are passed to the application and all variables specified in the application are received by the process. This results in variables that do not exist in the process being newly added to the process. The type of variables added in this way is "String."
2.1.5 Users

The Process Flow Manager manages users in order to control access to process definitions, processes and activities. Users managed by the Process Flow Manager and concepts relating to users are described in the table below.

<table>
<thead>
<tr>
<th>User Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Owner</td>
<td>The user or group permitted to generate processes based on a process definition. If the owner is set to &quot;default&quot; when creating a process definition with a Process Definition Tool, all users will be able to refer to that process definition and generate forms from it. However, only the control server administrator will be able to update the process definition.</td>
</tr>
<tr>
<td>Form issuer</td>
<td>Creator of the process.</td>
</tr>
<tr>
<td>Assignee</td>
<td>A user permitted to view or update form data when control passes to the UI activity. When workflow processing moves to the corresponding UI activity, the assignee is obligated to perform the tasks of the UI activity.</td>
</tr>
<tr>
<td>Group</td>
<td>A unit used to manipulate multiple users. A group has at least one user associated with it, and a user can belong to more than one group. Groups can become assignees, but cannot become form issuers, linkage users, or linkage approvers (see below). When a group becomes an assignee, all users belonging to that group become assignees for the UI activity.</td>
</tr>
<tr>
<td>Role</td>
<td>A collection of all users and groups that can become assignees for a given UI activity. A role does not have to be associated with a specific UI activity, and does not have to be specified as assignee of any UI activity within a process. To transform a form issuer into a participant (see below), register the form issuer in a role that is not designated as the assignee of any UI activity in a process.</td>
</tr>
<tr>
<td>Participant</td>
<td>Collective term for all users of all roles contained in a given process.</td>
</tr>
<tr>
<td>Linkage User</td>
<td>Used when linking a main process and a subprocess according to the action of a subprocess activity. This nominal user initiates subprocesses and sends the results of those subprocesses to control servers other than the CollaborationRing control server. Refer to 2.2.3 Subprocess Linkage Function for details.</td>
</tr>
<tr>
<td>Linkage Approver</td>
<td>This nominal user is recognized as the initiator of a subprocess and reporter of the subprocess's results from the receiving control server's viewpoint. Permits linkage by a subprocess activity. When a control server that owns a subprocess receives a form generation request for that subprocess, the issuer (i.e., the linker on the main process side) must be registered as a linkage approver. Also, because the control server receives the process results for a subprocess when that subprocess is synchronous, the linker on the subprocess side must be registered as a linkage approver in the control server that owns the main process.</td>
</tr>
</tbody>
</table>
2.1.6 Access Control

Process Flow Manager limits the number of users who are allowed to perform a task associated with a process definition, process or activity, according to the content of that task.

The correspondence between each task and the users who are allowed to perform it is shown in the tables below.

[Access control for Process Definition Tool]

<table>
<thead>
<tr>
<th>Task</th>
<th>User type permitted to perform task according to control server environment settings (access check)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>All OFF</td>
</tr>
<tr>
<td>Edit process definition</td>
<td>Owner and control server administrator</td>
</tr>
</tbody>
</table>

[Access control for Subprocess]

<table>
<thead>
<tr>
<th>Task</th>
<th>User type permitted to perform task according to control server environment settings (access check)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>All OFF</td>
</tr>
<tr>
<td>Create subprocess</td>
<td>Linkage approvers only</td>
</tr>
<tr>
<td>Receive subprocess results</td>
<td>Linkage approvers only</td>
</tr>
</tbody>
</table>
### Access control for Client-type API Library

<table>
<thead>
<tr>
<th>Type of Function</th>
<th>Task</th>
<th>User type permitted to perform task according to control server environment settings (access check)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Process definition operations</strong></td>
<td>Retrieve process definition list</td>
<td>All OFF</td>
</tr>
<tr>
<td></td>
<td>Retrieve process definition</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Start process</td>
<td></td>
</tr>
<tr>
<td><strong>Process operations</strong></td>
<td>Retrieve process list</td>
<td>Participant</td>
</tr>
<tr>
<td>Retrieve all process lists</td>
<td>Retrieve process</td>
<td>Participant</td>
</tr>
<tr>
<td>Retrieve process definition binary data</td>
<td>Retrieve process log</td>
<td></td>
</tr>
<tr>
<td>Start process</td>
<td>Update process</td>
<td>Participant</td>
</tr>
<tr>
<td>Update route dynamically</td>
<td>Move process to optional activity</td>
<td></td>
</tr>
<tr>
<td>Suspend process</td>
<td>Restart process</td>
<td></td>
</tr>
<tr>
<td>Interrupt process</td>
<td>Lock process</td>
<td></td>
</tr>
<tr>
<td>Unlock process</td>
<td>Delete process</td>
<td>User who locked the process</td>
</tr>
<tr>
<td><strong>Activity operations</strong></td>
<td>Retrieve activity list</td>
<td>Assignee</td>
</tr>
<tr>
<td>Retrieve activity</td>
<td>End activity</td>
<td></td>
</tr>
<tr>
<td>Update route dynamically</td>
<td>Update activity</td>
<td></td>
</tr>
<tr>
<td>Send back activity</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Agent functions</strong></td>
<td>Process agent</td>
<td>User who received agent rights</td>
</tr>
<tr>
<td>Grant agent rights</td>
<td>Remove agent rights</td>
<td></td>
</tr>
<tr>
<td>Retrieve delegate list</td>
<td>Retrieve delegator list</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Granted agent rights</td>
<td></td>
</tr>
<tr>
<td>Type of Function</td>
<td>Task</td>
<td>User type permitted to perform task according to control server environment settings (access check)</td>
</tr>
<tr>
<td>------------------</td>
<td>-----------------------</td>
<td>------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td>All OFF</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Allow viewing of all process lists</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Allow viewing of all processes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Allow update of all processes</td>
</tr>
</tbody>
</table>

- **Global history query**
  - Retrieve process list

| Global history query users |

2.1.7 URI (HTTP Connection, Direct Connection)

**+ URI of Object in Control Server (HTTP Connection)**

In basic operations, the Process Flow Manager uses HTTP to enable communication between client applications and control servers, and between control servers using subprocess linkage. Because HTTP is used, the objects contained in the control server are identified from outside the control server by means of a URI.

Based on this, there are a number of variations in the scheme portion of the head of the URI (that is, the section beginning "http:.") It is possible to use SSL in addition to HTTP to strengthen security, and in this case, "http" in the URI scheme is replaced by "https". It is also possible to specify direct connection between the control server and applications using client-type API libraries.

It is possible to specify faster second and subsequent requests from a client application to an object in a control server running on the same machine. (The nonstandard scheme "xiiop" is specified. Refer to Direct Connection for details.)
The URI format of an object in the control server is shown in the following figure. The format changes according to the type of Web server receiving the request.

Case (A) applies to Interstage HTTP Server (IHS), iPlanet Web Server Enterprise Edition (iPlanet), or Microsoft(R) Internet Information Services (IIS).

Case (B) applies to Interstage Application Server InfoProvider Pro (IPP).

**General URI Format of Objects within the Control Server**

<table>
<thead>
<tr>
<th>(A) IHS, iPlanet, IIS:</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://https://host">http://https://host</a> name [:port no.]/control server object reference name.crf?details</td>
</tr>
<tr>
<td>(a)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(B) IPP:</th>
</tr>
</thead>
<tbody>
<tr>
<td><a href="http://https://host">http://https://host</a> name [:port no.]/cgi-bin/libcrbrgipp.so?control server object reference name.crf?details</td>
</tr>
<tr>
<td>(a)</td>
</tr>
<tr>
<td>(g)</td>
</tr>
</tbody>
</table>

**Description**

a) **Scheme**: Determined by the protocol by which the control server receives requests. This is normally "http". When the direct connection function is used between the control server and the client-type API library, "xiiop" must be specified explicitly.

b) **Host name**: Host name of the server in which the Web server and the HTTP bridge exist
(Example: host1.xxx.yy.zz.fujitsu.co.jp)

c) **Port number**: Specifies the number of the port where the Web server operates. (If omitted, the port number defaults to 80.)
d) CGI identifier: Specifies the CGI identifier specified in the environment definition file of the Web server (InfoProvider Pro).

e) HTTP bridge library name: Specifies the library name of the HTTP bridge when using InfoProvider Pro as the Web server. This is fixed at "libcrbrgipp.so".

f) Control server object reference name: Specifies the object reference name of the control server to register in the Interstage Application Server (ObjectDirector). This name can be set with the commands for creating the operating environment or with the Management Tool (for example: SWAP).

g) Fixed character string "crf".

h) Details: Details that are used to make fine distinctions between objects in the control server.

The specifics of the URI value are necessary in the following three situations:

- When specifying the Control Server URI in the operating environment definition of the Control Server
- In the initialization stage when an application begins to use the client-type API libraries
- When the Process Definition Tool is used to define a subprocess invocation

Each of the above three descriptions explains a type of URI to be specified.

+++ When specifying the Control Server URI in the operating environment definition of the Control Server

The value entered in the commands for creating the operating environment is found in the sections of the URI assigned to the object in the control server, excluding the "?" character and "details". This corresponds to (a) through (g) in the above diagram.

+++ In the initialization stage when an application begins to use the client-type API libraries

It is possible to specify the control server URI, which becomes the default destination address, when initializing the client-type API library. The control server object specified in the client-type API library initialization is known as an Extension object, and the format of its URI is as shown below with the value of the "details" section fixed as "Extension".

**URI Format of the Extension Object**

<table>
<thead>
<tr>
<th>(A) IHS, iPlanet, IIS:</th>
</tr>
</thead>
<tbody>
<tr>
<td>**http://<strong>host name [[:port no.]/control server object reference name.crf?Extension</strong></td>
</tr>
<tr>
<td>(a)          (b)        (c)                    (f)                   (g)    (h)</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(B) IPP:</th>
</tr>
</thead>
<tbody>
<tr>
<td>**http://<strong>host name [[:port no.]/cgi-bin/libcrbrgipp.so?control server object reference name.crf?Extension</strong></td>
</tr>
<tr>
<td>(a)          (b)        (c)       (d)        (e)                    (f)</td>
</tr>
<tr>
<td>(g)    (h)</td>
</tr>
</tbody>
</table>

Once this initialization is performed, the client application no longer requires the actual value of the URI to use it. There is no longer any need to specify the URI of the destination control server each time operations such as list searches are conducted.

Even when generating a process form, it is sufficient to specify using a short name that is unique in the control server (the process definition title). For other client-type API library operations, it is necessary to use parameters to specify the actual value of the URI assigned to the object in the control server. Since it is usual to use the values obtained as list search operation results, it is not necessary for the application to specify the actual value directly.

Note that the URI returned from the control server is assembled based upon the URI specified when the request was made. As a result, if the URI contained in an operation result is used without modification, the scheme specified in the client-type API library initialization will be used consistently from then on.

+++ When the Process Definition Tool is used to define a subprocess invocation

When calling a subprocess, it is necessary to directly specify the value of the URI assigned to the definition of that subprocess. The format of the process definition URI is shown in the following diagram. The details section (h) consists of the character string "ProcessDefinition" followed by ",," and the title of the process definition (a short name for the process definition that is unique in the control server).

**URI Format of Process Definition**

<table>
<thead>
<tr>
<th>(A) IHS, iPlanet, IIS:</th>
</tr>
</thead>
<tbody>
<tr>
<td>**http://**host name [[:port no.]/control server object reference **</td>
</tr>
</tbody>
</table>
+ Direct Connection

Client applications can use the direct connection function. When communicating between an application and a control server, this function uses a CORBA connection to connect to the control server directly, instead of using HTTP to communicate via the Web server.

Note that even when the direct connection function is used, HTTP communication is used initially for authentication purposes.

[Direct Connection Function]

++ Using the Direct Connection Function

To use the direct connection function, specify "xiiop" and not "http" in the scheme of the URI of the control server that is being communicated with. This causes the direct connection function to be used within the client-type API libraries.

To use the direct connection function, the ObjectDirector function of the Interstage Application Server must exist in the environment where client-type API libraries run. Therefore, before using the direct connection function, it is necessary to install the CORBA client function of the Interstage Application Server.

2.1.8 Subprocesses in a Unified Control Server

When using process definitions existing in a unified control server by subprocess linkage, it is possible use x-plan schemes. Specify an x-plan scheme as follows:

```
x-plan:process definition name
```

When you specify an x-plan scheme, it is possible to search process definitions registered in the control server and generate process instances without passing through the Web server.

An x-plan scheme has an advantage over an http scheme in that there is no need to modify it if the process definition name is registered in multiple control servers because only the process definition is specified. With http, because the URI contains information such as the Web server name, if a process definition is registered in multiple control servers, the URI for subprocess linkage must be modified to suit each environment.
2.1.9 Event Queue

The control server uses event queues to execute activities. After the control of workflow reaches activities, events are executed according to the information set in the activity properties. They can be executed simultaneously by multithreading. The number of threads for executing events can be specified by using the Management Tool. If the number of events exceeds the number of threads, the surplus events are queued in the event queue. The following figure shows the flow of event executions into the event queue:

[Event Queue]

There are three types of event: events for executing automatic execution activities, events for executing subprocess activities, and events for executing mail transmission. For details on each type of event, refer to "2.1.3 Activities".

2.2 Functions Used in Process Rules

Process rules govern how multiple activities work together, and define the conditions required to select activities and the order of processes in activities.

2.2.1 Standby Function for Client-type Applications (UI Activity)

The standby function for client-type applications stops processing in the middle of the workflow, and waits for a person (or an application program) to access the workflow under the direction of the user and finish processing the pending tasks.

The standby function for client-type applications is expressed as a UI activity. When control of the workflow reaches this UI activity, forms accumulate and must be processed by a user or a client application before workflow proceeds. If no assignee is set in the UI activity, processing proceeds immediately to the next activity.

The UI activity can be set up to send an email notifying all relevant assignees that a job has been generated as the activity receives the control of the workflow. This is known as the mail transmission function.

For details on defining UI activities, refer to the online help of the Process Definition Tool.
[UI Activity and Mail Transmission Function]

2.2.2 Call Function for Server-type Applications (Automatic Execution Activity)

In contrast to the standby function for client-type applications, which requires human intervention, the call function for server-type applications automatically calls a business application at the direction of the workflow.

Expressed as an automatic execution activity, the business application called from the automatic execution activity is known as an automatic execution application.

When processing reaches an automatic execution activity, the activity extracts the data (input variables), notifies the automatic execution application, and waits. Control will not proceed beyond the activity until processing results are returned from the automatic execution application. While an application is running, the control server exercises exclusive control over each process unit to avoid conflicts and inconsistencies occurring in the information contained within the same process.

Which automatic execution application being invoked is specified in the "application name" or "variable name" of the automatic execution activity properties (defined using the Process Definition Tool). When a variable name is specified, the application name must be stored in that variable. If no information is set in the variable, processing proceeds immediately to the next activity.

The application name specified in the diagram below is the object name that is registered in the Interstage Application Server Naming Service when the automatic execution application is created.

For details on defining automatic execution activities, refer to the online help of the Process Definition Tool.
[Business Application Invocation]

+ Execution Queue

The execution queue is used to execute automatic execution activities. Apart from the execution of normal activities, specific automatic execution activities can be executed with the execution queue while the event queue is used for executing normal activities. For the setup of execution queues, refer to How to use execution queues.

The following figure shows the flow of event executions using execution queues:

![Execution Queue Diagram]

(1) ~ (5): Order when Activity is Active

Execution Queue

Event Queue (Default Queue)

Event Execution Thread

Activity that uses Default Queue

Automatic Execution Activity that uses Execution Queue

By using the execution queue, automatic execution activities are executed without affecting the default queue status.

[Execution Queue]

The following are examples of automatic execution applications (business applications) that use execution queues:

- Execution queues for a business application that requires a long processing time.
  If a business application requiring a long processing time is executed in the event (default) queue, other activities in the queue have to wait. Such processes should be defined to use an execution queue for application. This lessens the load on other activity events.

- Execution queues for a business application that requires emergency executions.
  If a business application that requires an emergency execution, such as error notifications, is executed in the event queue, other activities in the queue have to wait. In this case, the process should be defined to use an emergency execution queue for the application. If emergency executions are needed, they can be executed without affecting other activity events.

- Execution queues for a business application that executes standby processes.
  If a business application that executes a standby process is executed in the event queue, other activities in the queue have to wait. In this case, the process should be defined to use an execution queue for the business application. This lessens the load on other activity events.

<table>
<thead>
<tr>
<th>Note</th>
</tr>
</thead>
<tbody>
<tr>
<td>Special care is needed for the following business applications that execute a standby process. If a business application issues client-type API functions (such as SearchActivities, SearchProcesses, or GetProcess) and executes a standby process for other activities, the same queue cannot be used for processes for auto-execution activity events operated by the application that executes the standby process and other standby activity events. If the same queues are used and the number of event processing threads is insufficient, it can cause a deadlock. For this reason, define an execution queue for the applications that execute the standby process and an event queue for other standby activities.</td>
</tr>
</tbody>
</table>
+ How to use Execution Queues

The administrative tools create an execution queue on the control server by setting the queue name and the number of threads. Multiple execution queues can be created. To use an execution queue, the queue name has to be set in the properties for the auto-execution activities of process definitions. When the auto-execution activities become active, the business application executes the execution queues set in the properties. If there is no execution queue set in the properties, the event queue will be used. For more details of the execution queue setup, refer to the online helps of the Process Definition Tool and the Management Tool.

2.2.3 Subprocess Linkage Function

When a process definition is modularized hierarchically in the manner of a function or subroutine, the creation of a child process from a parent process is called subprocess creation, and the child process is called a subprocess.

A subprocess is created using a subprocess activity: when processing reaches the subprocess activity, the data (input variables) specified in the subprocess activity definition is extracted from among process data, and used as the initial data to create forms for the subprocess.

While a subprocess is being created, exclusive control is exercised over parent processes to avoid conflicts and inconsistencies occurring in the information contained within the same process.

Subprocess activities can be either synchronous or asynchronous. Synchronous subprocess activities wait for process results from the subprocess, and processing does not progress to the next activity until a response returns from the subprocess. In asynchronous subprocess activities, processing progresses to the next activity immediately without waiting for a response from the subprocess.

The subprocess to be invoked is specified in the "URI" or "variable name" properties of the subprocess activity. When a variable name is specified, the subprocess URI must be stored in that variable. If no information is set in the variable, processing proceeds immediately to the next activity.

As stated above, function that links with subprocess is called Subprocess linkage function. For information about setting environment for subprocess linkage function, refer to 5.1.3 Subprocess Linkage Environment Settings and "5.1.2 User Information Settings" User Information Relationships During Subprocess Linkage.

For details on defining subprocess activities, refer to the online help of the Process Definition Tool.

2.2.4 Conditional Branching Function

The Process Flow Manager can use the result of each activity's processing to dynamically select subsequent activities. This is called conditional branching. There are two methods of conditional branching: conditional branching by option value, and conditional branching by condition node.
+++ Conditional Branching by Option Value

The conditional branching by option value method determines which activity is next on the basis of an option value set in the activity processing results. The option value is included in the business document set by a client application, automatic execution application, or by a subprocess. If the option value is not set, the next activity is selected on the basis of the default value.

The Process Definition Tools use option arrows to define condition branching on the basis of option values. If arrows are stretched from one activity to multiple activities, the arrows become option arrows. The title of an option arrow is used as the option value enabling selection of that activity. Select one of the option arrow titles and set it as the option value of the automatic execution applications or client applications. In addition, one of the multiple arrows must be specified as the default arrow. If an option value is not set for the activity processing results, or if there is no appropriate option value, the default arrow is selected.

For details on defining option arrows, refer to the online help of the Process Definition Tool.

[Conditional Branching by Option Value]

+++ Conditional Branching by Condition Node

Conditional branching by condition node uses values other than option values to determine condition branching. The types of data that can be referenced for determining condition branching are:

- process status
- custom attributes (1 to 5)
- process title
- global key
- text or string type variables (without internal XML tags) corresponding to the form data.

The Process Definition Tool can define condition branching on the basis of condition nodes and condition arrows. Arrows stretched from the condition node to the next activity are condition arrows. Conditional expressions can be entered in condition arrows, and, control passes to the activity indicated by that arrow when a conditional expression is satisfied.

If more than one conditional expression are defined from a single condition node, the order in which the conditional expressions are evaluated becomes crucial. In such a case, you can specify the order by assigning different levels of priority to the conditional expressions.

For details on defining condition arrows and conditional expressions, refer to the online help of the Process Definition Tool.
Tag Content (Data) in XML Document Variables

Tag content (data) located in XML document type variables cannot be used by the Process Flow Manager to evaluate conditions. Use an automatic execution application or similar to extract the required data part and convert it to string type or move it to a custom attribute before using the data for condition evaluation.

2.2.5 Function for Parallel Execution and Merging of Work Steps

It is possible to run the activities in a process in parallel, alternately it is also possible to wait until all the activities operating in parallel finish processing before proceeding.

When defining a process, parallel operation of activities is represented by program nodes and arrows. Waiting until all activities finish processing is represented by And nodes.

For details on defining program nodes, arrows, and And nodes, refer to the online help of the Process Definition Tool.

2.2.6 Work Timer Monitoring Function

The timer function limits the time that control remains with an activity. If the activity has not finished after the specified period has elapsed or the specified time has been reached, control branches to a point specified in the timer function. If the workflow forms a loop, an application can be executed periodically. Because it is possible to specify a particular
time and date by means of variables, the next action can also be set to start after the time calculated by the automatic execution application. If no information is set in the variable, processing proceeds immediately to the next activity.

In the Process Definition Tool, work timer monitoring is represented by a timer arrow.

For details on defining timers, refer to the online help of the Process Definition Tool.

### 2.2.7 Automatic Execution Application Error Monitoring Function

This function is used when an application error occurs in an automatic execution activity. When the error occurs, control moves to the next activity according to the branch specified in error monitoring (i.e., the direction of the error arrow in the Process Definition Tool).

For details on defining error arrows, refer to the online help of the Process Definition Tool.

### 2.2.8 Dynamic Modification Function for Process Rules Using Variables

Variables refer to the input and output variables that contain the I/O data of an application specified by a process definition activity's properties. These variables can be used in other properties of the process definition. For example, these variables can be defined as an automatic execution application name or subprocess URI. By specifying output variables for applications, the automatic execution application name or subprocess URI can be changed automatically.

The following table lists the definitions that can be used as variables in a process definition.

#### [User-defined Variables]

<table>
<thead>
<tr>
<th>Item</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conditional expression</td>
<td>A variable can be used as the conditional expression in conditional branching. Refer to 2.2.4 Conditional Branching Function for details.</td>
</tr>
<tr>
<td>Assignee</td>
<td>A variable can be used as the assignee in a role definition. Only one assignee can be defined with each variable. It is also possible to define an assignee using multiple variables in a single role. Refer to 2.1.5 Users for details.</td>
</tr>
<tr>
<td>Automatic execution</td>
<td>A variable can be used as the object name of an application specified in an automatic execution application activity. When control reaches the automatic execution activity, the automatic execution activity associated with the object name specified in the variable is called. Refer to 2.2.2 Call Function for Server-type Applications (Automatic Execution Activity) for details.</td>
</tr>
<tr>
<td>Subprocess</td>
<td>A variable can be used as the URI specified by a subprocess activity. When processing reaches the subprocess activity, the subprocess activity associated with the URI specified in the variable is called. Refer to 2.2.3 Subprocess Linkage Function for details.</td>
</tr>
<tr>
<td>Timer</td>
<td>A variable can be used as the timer setting definition. Refer to 2.2.6 Work Timer Monitoring Function for timer monitoring.</td>
</tr>
<tr>
<td>Mail address of an</td>
<td>A variable can be used as the mail address of an activity.</td>
</tr>
<tr>
<td>activity</td>
<td></td>
</tr>
<tr>
<td>Mail template of an</td>
<td>A variable can be used as the mail template of an activity.</td>
</tr>
<tr>
<td>activity</td>
<td></td>
</tr>
<tr>
<td>Expiration date of an</td>
<td>A variable can be used as the expiration date of an activity.</td>
</tr>
<tr>
<td>activity</td>
<td></td>
</tr>
<tr>
<td>Error arrow error</td>
<td>A variable can be used as the Error Arrow error information storage location. Error information is stored when an automatic execution application fails to start. For error monitoring, refer to 2.2.7 Automatic Execution Application Error Monitoring Function.</td>
</tr>
<tr>
<td>information storage</td>
<td></td>
</tr>
<tr>
<td>location</td>
<td></td>
</tr>
</tbody>
</table>

The Process Flow Manager can utilize the following system variables in addition to user-defined variables. These system variables can be used in process definitions in the same way as user-defined variables.

#### [System Variables]

<table>
<thead>
<tr>
<th>System variable name</th>
<th>Setting</th>
</tr>
</thead>
<tbody>
<tr>
<td>_piid</td>
<td>Process ID</td>
</tr>
<tr>
<td>System variable name</td>
<td>Setting</td>
</tr>
<tr>
<td>----------------------</td>
<td>----------------------------------------------</td>
</tr>
<tr>
<td>_globalKey</td>
<td>Global key</td>
</tr>
<tr>
<td>_title</td>
<td>Process title</td>
</tr>
<tr>
<td>_description</td>
<td>Process description</td>
</tr>
<tr>
<td>_priority</td>
<td>Process priority</td>
</tr>
<tr>
<td>_createdAtTime</td>
<td>Process creation date and time</td>
</tr>
<tr>
<td>_owner</td>
<td>Process creator</td>
</tr>
<tr>
<td>_field1</td>
<td>Process custom attribute 1</td>
</tr>
<tr>
<td>_field2</td>
<td>Process custom attribute 2</td>
</tr>
<tr>
<td>_field3</td>
<td>Process custom attribute 3</td>
</tr>
<tr>
<td>_field4</td>
<td>Process custom attribute 4</td>
</tr>
<tr>
<td>_field5</td>
<td>Process custom attribute 5</td>
</tr>
</tbody>
</table>

### 2.2.9 Dynamic Modification Function for Process Routes

This function allows you to add activities to a process after drafting it, and to modify the distribution route of the process dynamically.

This is useful for example in a situation where, during drafting of a process, the distribution route is undecided, but this route becomes known when adding activities.

The diagram below illustrates this example.

The modified distribution routes are only valid in the process in which they were modified, even if an identical process definition is specified at the time of planning.

![Dynamic Modification Function for Process Routes](image)

**[Dynamic Modification Function for Process Routes]**

Process routes are modified dynamically using the client-type API libraries and process definition read/write APIs. For more details, refer to the Interstage CollaborationRing PM Reference Manual (Process Flow Manager Edition).

### 2.2.10 Activity Process Transfer Function

This function allows you to transfer a process from the activity currently being processed to a different activity even after the process has been designed.

This is useful for example in special cases where you want to transfer a process from one activity to another without being bound by the process route defined in the process definition.

The diagram below illustrates this example.
This function uses the client-type API libraries and process definition read/write APIs. For more details, refer to the Interstage CollaborationRing PM Reference Manual (Process Flow Manager Edition).

### 2.2.11 Agent Function

In any work situation, cases arise where, for one reason or another, a person in charge is absent, and the authority of that person in charge must be temporarily transferred to another (the ‘agent’).

The agent function performs this transfer (and the subsequent rescinding) of authority so that the agent can perform tasks such as drafting processes and approving requests. The actual transfer and rescinding of agent authority is carried out with the Management Tool and management commands.


### 2.2.12 Mail Linkage Function

The mail linkage function is used to inform a person in charge by email that work has begun and to report the occurrence of errors in automatic execution activities. The corresponding email addresses are set in the UI activity properties and the automatic execution activity properties of the Process Definition Tool respectively. In each case, more than one email address can be set.

By using a variable in the email address in the UI activity properties, it is possible to specify that mail is or is not sent to a particular person in charge depending on the details of the work.

For more details, refer to the online help of the Process Definition Tool.

### 2.3 Relationship between Applications and Process Rules

The process rules are for linking multiple activities.

#### + Automatic Execution Applications

Automatic execution applications are CORBA server applications that have an IDL specifying the server-type API and a fixed method name. Their object names are defined in a process definition as automatic execution activities so that when control reaches that activity, the application is automatically called and returns process results.

#### + Client Applications

Client applications specify a process or the object name of a UI (User Initiated) activity and act upon a process. These applications use the client-type API library to perform the following main operations:

- Form generation (CreateProcess)
- Reading lists of tasks (worklist) (SearchActivities)
- Reading task input data (GetActivity)
- Updating task result data (CompleteActivity)

Automatic execution applications can also be called from a process and, using client-type API, communicate with the process.

The relationship between applications and processes is illustrated in the following diagram.
2.3.1 Relationship between Process Rules and Automatic Execution Applications

Automatic execution applications can select the required business data from the data managed for each process (process data), and pass this data as application input and output.

Process data is specified in each variable in the process definition. The specified data is referred to as process variables. Those variables that are handled by applications are specified as the input variables and output variables of an automatic execution activity. In turn the automatic execution activity invokes an automatic execution application using this data as the process data handled by the application.

The name of the called application is the CORBA application object name specified in the automatic execution activity definition.

The relationship between business data and automatic execution applications is shown in the illustration below.
2.3.2 Relationship between Process Rules and Client Applications

Client applications create processes by specifying the name of a process definition (the process definition title) and generating a form. By transmitting the initial value of the business data at the same time a form is generated, it is possible to write the initial information to the data managed by the process.

By defining a UI (User Initiated) activity in the process definition, a client application can specify a UI activity that is active within a process and extract data required by the application from the process data. It can also write to the process data.

Process data is specified in each variable set in the process definition. The initial value of the business data is transmitted by the application at the same time as form generation is selected from among the process variables. It is specified as an input variable of the process definition. In addition, data passed by an application via a UI activity which was active within a process is specified in the input variables and output variables of that UI activity.

When all activities of a process have completed, the process automatically enters completion status. When an application takes the data that it needs from the data of a completed process, it selects from the process variables and specifies an output variable of the process definition. The process output variable can be retrieved using the process acquisition function of the client-type API (GetProcess).

The relationship between business data and client applications is illustrated in the following diagram:
2.3.3 Overview of Form (Process Creation) Applications

An application identifies which process definition it needs to create by referring to the CRF host name, the control server object name and a key that contains the process definition title. Process definitions also have a defined owner name, which is registered by the application in the control server.

When a process has been created by an application, a process key (URI) identifying the process will be posted to the control server. The application can refer to the status of the created process by specifying this key.

When viewing the status of a process, the list acquisition function of the client-type API (SearchProcesses) is used to obtain a list of keys (URI) associated with processes created by that application. The status of a particular process can be selected from the list and viewed. The process list can be narrowed by using the search criteria setting function (SetFilter).

When an application generates a form with the initial data values assigned, the data is managed within the created process. Refer to Chapter 3 Application Design for details.

The relationship between process definitions and client applications is illustrated in the following diagram.

---

**Point**

A client application specifies objects in the control server (process definitions, processes, activities, etc.) by means of a key. The key is expressed as a URI. Refer to "2.1.7 URI (HTTP Connection, Direct Connection)" URI of Object in Control Server (HTTP Connection) for details.
2.3.4 Overview of UI Activity Reference/Update Applications

The UI activity definition component of a process definition contains an assignee name indicating the operator. The application registers user information corresponding to the name of the assignee.

When a process is created by the application generating a form, a key (URI) is assigned to the activities within that process. The application can specify this key (URI) to read process data and update task results.

The read worklist function of the client-type API (SearchActivities) can be used to obtain a list of keys (URI) corresponding to tasks managed by the application. It is possible to read business data and update task results by selecting the key (URI) that identifies the necessary activity. The worklist can be narrowed by using the search criteria setting function (SetFilter) when reading the worklist. See Chapter 3 Application Design for details. The relationship between an activity's assignee and the client application is illustrated in the following diagram.
[Relationship between an Assignee and the Client Application]

+++ (a,c) Initialization (Init)

Specifies the URI of the control server being communicated with. For details on URI, refer to 2.1.7 URI (HTTP Connection, Direct Connection).

+++ (b) Form creation (CreateProcess)

If the application specifies the process definition title and issues a form (CreateProcess) as in (b) after it has issued the initialization function (Init) as in (a), the request is sent to the control server after BASIC authentication by the Web server. At the control server, a process is created from the process definition and a unique key added. This is called the process key.

This key is returned to the client-type application as the form creation result. From then on, the client application can use this key to access the process. When a process is created, its first task (activity) is becomes active. Each time a UI activity becomes active, a key is assigned to the activity.

Usually, the key specified after form generation uses the process definition list acquisition function (GetProcessDefinitionList) of the client-type API to obtain a list of keys identifying the process definition. By using the search criteria setting function (SetFilter) beforehand, it is possible to narrow the list obtained.

+++ (d) Read worklist (SearchActivities)

Of the tasks (activities) that are active within a process, collections of tasks (UI activities) which need to be performed by each task assignee are referred to as worklists. In a normal workflow, the assignee determines which task to start based on this worklist.

To read the worklist, an application issues the initialization function (Init) as in (c) to initialize the client-type API, followed by the read worklist function (SearchActivities) as in (d). Because the user ID previously registered in the Web server are set in user information (g) in the control server, it is possible to obtain a list of keys associated with active tasks (activities) that are the responsibility of the assignee.

By specifying each key from the list of keys obtained, it is possible to read task input data from process data (GetActivity) as in (e) and update task result data to process data (CompleteActivity) as in (f). The worklist can be narrowed by using the search criteria setting function (SetFilter), the result count limiting function (SetMaxList), and the sort item setting function (SetSorter).

+++ (e) Read task input data (GetActivity)

In order to execute the managed task, the application reads its input data from the process data. Reading is
accomplished by issuing the read task input data function (GetActivity) of the client-type API as in (e). When doing so, the task is identified by using GetActivity to specify the activity key obtained by the read worklist function (SearchActivities) as in (d). As a result of GetActivity, input business data is obtained as an XML document. The manager performs the task using this business data as the original, and the task result data is created as output business data in the form of an XML document. This is then stored in the process data by issuing the update task result data (CompleteActivity) function as in (f).

+++ (f) Update task result data (CompleteActivity)

Issue the update result data function (CompleteActivity) of the client-type API to apply task results (output business data) to the process business data as in (e). When doing so, specify the key obtained by the read task input data function (GetActivity) to specify the task (UI activity) to be updated.

2.3.5 Overview of Automatic Execution Applications

This section explains the processing of an automatic execution application using the server-type API.

Automatic execution applications are CORBA applications that possess an IDL interface and fixed method name that specify server-type API. Automatic execution application input is expressed in the IDL as octet-type sequences containing business data in XML document format. By specifying the object name of the automatic execution application in the automatic execution activity definition, the application is called automatically when processing reaches the activity. The automatic execution application can modify process business data by receiving the XML document (input data) and returning the process results as an XML document (output data).

[Relationship between Activities and Automatic Execution Applications]

+++ (a) Read task input data (in DATA activityDataIn)

The application receives the process data it requires using the input data (in DATA activityDataIn) of the CORBA IDL interface.

+++ (b) Update task result data (out DATA activityDataOut)

By setting the task result data (business data) in the CORBA IDL interface output data (out DATA activityDataOut) and returning it, the application stores the data in the process data.
Chapter 3 Application Design

This section describes options for developing applications using the Application Program Interfaces (API) provided by CollaborationRing Process Flow Manager.

3.1 Application Development Styles

This section explains the application development styles of CollaborationRing PM (occasionally referred to as “CRF” hereafter).

3.1.1 API Overview

The following two types of CollaborationRing PM API are available:

- **Client-type API**
  - An API that operates on a process or a UI activity active within a process by specifying a UI activity in a process definition.

- **Server-type API**
  - An API that is called automatically and returns a result by specifying an automatic execution application activity in a process definition.

+++ Positioning of Applications within CollaborationRing

The positioning of applications is shown in the diagram below.

![Position of Applications within CollaborationRing](image)

3.1.2 Server-type API

Used when an automatic execution activity is specified, the server-type API is invoked automatically by the automatic execution application invoked by the Process Flow Manager. Using a fixed method name the API is called from the Process Flow Manager using a CORBA IDL (Interface Definition Language) interface, and creates automatic execution applications as Interstage Application Server distributed objects.

Business data forming the input is received, processed and returned as IDL parameters, which are passed as XML documents. These may be viewed or set using the Interstage Application Server XML processor (also called the XML parser), an object designed to access XML data such as DOM. A server-type API library is a function that makes easier the handling of XML documents in the server-type API and makes it possible to view frequently used elements and to set document information without using the XML parser.
[Input/output Data Passed in IDL]

[Input and Output Business Data IDL Parameters]

<table>
<thead>
<tr>
<th>Data</th>
<th>IDL parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input data</td>
<td>ActivityDataIn</td>
<td>Control data with a CRF-fixed tag and form data specified in the activity input variables in a Process Definition Tool are reported as an XML document.</td>
</tr>
<tr>
<td>Output data</td>
<td>ActivityDataOut</td>
<td>By returning control data with a CRF-fixed tag and form data specified in the activity output variables in a Process Definition Tool, the form data managing the process are updated.</td>
</tr>
</tbody>
</table>

3.1.2.1 Server-type API Libraries

An overview of the processing performed by an application using server-type API libraries is shown in the following diagram.
+ **Explanation of Diagram**

1. **Register IDL input data (activityDataIn)**
   - Register the input information (activityDataIn) received by the CORBA IDL interface in the server-type API library object.

2. **Acquire (read) control element (CRF_GetControlElement)**
   - Read the control element from the input information (activityDataIn) registered in the server-type API library object. The control element is information determined by the control server, such as the activity key or activity title.

3. **Acquire (read) text-type user element (CRF_GetUserTextElement)**
   - Read the text-type user element (the information (business data determined by the user) set in the input variables specified in the activity properties of the Process Definition Tool) from the input information (activityDataIn) registered in the server-type API library object.

4. **Acquire (read) user element of XML document type (CRF_GetUserXMLFragment)**
   - Read the XML document type user element from the input information (activityDataIn) registered in the server-type API library object.

5. **Acquire (read) user element of binary data type (CRF_GetUserBinaryElement)**
   - Read the binary data type user element from the input information (activityDataIn) registered in the server-type API library object.
Read the binary document type user element from the input information (activityDataIn) registered in the server-type API library object.

(6) Set (write) control element (CRF_SetControlElement)

Write output information (activityDataOut) returned to the Process Flow Manager by the CORBA IDL interface to the server-type API library object.

The control element is information determined by the Process Flow Manager, such as the activity key or activity title.

(7)-(9) Set (write) control element (text, XML document, binary data)

Write output information (activityDataOut) returned to the Process Flow Manager by the CORBA IDL interface to the server-type API library object.

The user element written here is the information (business data determined by the user) set in the output variables specified in the activity properties of the Process Definition Tool.

The user element can be set with text-type, XML document-type and binary-type data.

(10) Create IDL output data (activityDataOut) (Fetch XML document-type data) (CRF_SaveOctets)

Create string<octet> XML document-type data from the output information (activityDataOut) that is written to the server-type API library object and that is to be returned to the Process Flow Manager (control server). The application sets this string<octet> data to the CORBA IDL interface output information (activityDataOut).

(11) Return IDL output data (activityDataOut)

Returns the output data (activityDataOut) set in the CORBA IDL interface to the control server.

Refer to the Interstage CollaborationRing PM Reference Manual (Process Flow Manager Edition) for details of the server type API library methods associated with (1) to (11) above.

As shown in (3) and (7) in the above diagram, if the XML structure for frequently used business data items is simplified (or the elements of the input and output variables for the automatic execution activities defined using the Process Definition Tool are converted to text format), it is possible, without the use of an XML parser, to view XML documents (activityDataIn) received by the Control server and IDL, and to create XML documents (activityDataOut) returned using the IDL. It is also possible to handle XML documents (as shown in (4) and (8) above), and binary data (as shown in (5) and (9) above).

Refer to 3.1.2.2 Relationship between I/O Variables of the Server-type API and I/O Variables of Process Definitions for an explanation of input variables and output variables of automatic execution activities defined by the Process Definition Tool.

3.1.2.2 Relationship between I/O Variables of the Server-type API and I/O Variables of Process Definitions

The relationship between server-type API I/O variables and Process Definition Tool I/O variables is illustrated in the below figure. Refer also to the overview of processing performed by applications using server-type API libraries in 3.1.2.1 Server-type API Libraries.
3.1.3 Client-type API

The client-type API is one of the APIs provided by the Process Flow Manager and is used for business document operations under the leadership of applications. (Application Leadership refers to the situation where invocation from an application is the start point of an operation.)

The organization of the application and client-type API are shown below.
3.1.3.1 Client-type API Libraries

An overview of the processing performed by an application using client-type API libraries is shown in the diagram below.
+ **Explanation of Diagram**

1. **Create request data for reading task data (GetActivity)**
   - Create a prototype of request data for GetActivity in the client-type API library object.

2. **Send task data read (GetActivity) request**
   - Send a GetActivity request to the Control server, and read task data from the Control server as GetActivity response data. The task data read from the Control server is registered in the client-type API library object.

3. **Acquire (read) control element**
   - Read a control element from the GetActivity response data registered in the client-type API library object. The control element is information determined by the CRF, such as the activity key or activity title.

4. **Acquire (read) text-type user element**
   - Read a text-type user element from the GetActivity response data registered in the client-type API library object. The user element read here is the information (business data determined by the user) set in the input variables specified in the activity properties of the Process Definition Tool.

5. **Acquire (read) XML document-type user element**
   - The user element can also be set with XML-document-type data. Used when an application uses an XML parser (DOM, etc.) to handle XML document-type data.
(6) Acquire (read) binary-type user element
The user element can also be set with binary-type data.

(7) Create request data for task result data reflection (CompleteActivity)
Create a prototype of request data for CompleteActivity in the client-type API library object.

(8) Set a control element (write)
Write applications' task result data in a control element of request data for CompleteActivity. The control element is created in the client-type API library object.
The control element is information defined by CRF such as activity key and activity title.

(9)(10)(11) Set a user element (write)
Write applications' task result data in a user element of request data for CompleteActivity. The user element is created in the client-type API library object.
User elements to be written here is information (business data determined by users) to set to an output variable, which is specified in the properties of process definition activity.
You can set text-type, XML document-type and binary-type data to the user element.

(12) Send task result data reflection (CompleteActivity) request
Send CompleteActivity request to the control server.
Refer to the Interstage CollaborationRing PM Reference Manual (Process Flow Manager Edition) for details about methods of client-type API library object described (1) through (12) above.

By simplifying XML structure of items often used in business data, you can view task data, XML document (GetActivity) and create task result data, XML document (CompleteActivity) without using XML parser. Simplification means changing the elements of input and output variables of UI activity defined by Process Definition Tool to text format. Example of this simplification is shown in the (4) and (9) in the above figure. Also, XML documents shown in (5) and (10) and binary data shown in (6) and (11) can also be simplified.

Refer to 3.1.3.2 Relationship between Client-type API I/O Variables and Process Definition I/O Variables about input and output variables of automatic execution activity to define by Process Definition Tool.

3.1.3.2 Relationship between Client-type API I/O Variables and Process Definition I/O Variables

The relationship between the I/O variable of the client-type API and Process Definition Tool I/O variables is illustrated in the following diagram. Refer also to the overview of processing performed by client applications using client-type API libraries in 3.1.3.1 Client-type API Libraries.
3.1.4 Using Variables in Process Definitions

Variables contain the input and output information of an application (input and output variables) specified in the activity properties of a Process Definition Tool. They can be used in other properties of a Process Definition Tool, for example, they can be defined as a Process Definition Tool's automatic execution application name, or the URI of a subprocess.

If the output variable is set as the automatic execution application name or subprocess URI, etc., the application can change these variable settings automatically.

The Process Definition Tool items that can use variables are shown in the following table.

<table>
<thead>
<tr>
<th>Elements that can use Variables</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Conditional expression</strong></td>
</tr>
<tr>
<td><strong>Assignee</strong></td>
</tr>
<tr>
<td><strong>Automatic execution application</strong></td>
</tr>
<tr>
<td><strong>Subprocess</strong></td>
</tr>
<tr>
<td><strong>Timer</strong></td>
</tr>
</tbody>
</table>

3.1.5 Structure of I/O Data Passed by the API

This section describes the structure of the I/O data that is passed between CollaborationRing APIs.
3.1.5.1 I/O Data Structure

The data takes the form of an XML document comprising a fixed tag component determined by the Process Flow Manager, and a business data tag component specified in input and output variables of the Process Definition Tool. In this manual, the fixed tag element determined by the Process Flow Manager is referred to as the control element, and the business data tag element specified by the Process Definition Tool I/O variables is referred to as the user element.

The control element is set with the attributes or content of a process or activity, such as a process activity key or title name. The user element is the business data handled by an application.

3.1.5.2 Control Elements

The application can refer to process or activity information by specifying the control element tag name using an XML parser or client-type API or server-type API library parameters. It can also update the process or activity information by specifying this tag name. Below is an explanation of the basic control element tags used by applications.

+ Control Element Tags Used with the Server-type API

The following control element tags are used when an automatic execution application refers to or sets IDL input information (activityDataIn) and output information (activityDataOut).

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Automatic execution activity</td>
<td>Automatic execution activity title</td>
</tr>
<tr>
<td>Custom attribute</td>
<td>Global key</td>
</tr>
<tr>
<td>Process creator</td>
<td>Process description</td>
</tr>
<tr>
<td>Process ID</td>
<td>Process title</td>
</tr>
</tbody>
</table>

+ Control Element Tags Used with the Client-type API

The main control element tags used when an application performs the following processes are:

<table>
<thead>
<tr>
<th>Tag Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Custom attribute</td>
<td>Global key</td>
</tr>
<tr>
<td>Process creator</td>
<td>Process description</td>
</tr>
<tr>
<td>Process ID</td>
<td>Process title</td>
</tr>
<tr>
<td>UI activity assignee</td>
<td>UI activity ID</td>
</tr>
<tr>
<td>UI activity status</td>
<td>UI activity title</td>
</tr>
</tbody>
</table>

3.1.5.3 User Element (Business Data)

Business data is managed as XML documents in which process variable names (input variables and output variables) specified in the Process Definition Tool have been converted to tags. User elements are components (elements) of these tags (variables). That is, the business data handled by an application is stored as elements of these tags (variables) in a database managed by the Process Flow Manager.

An application can receive business data by specifying these tag names in an XML parser or in parameters of a client-type API library or a server-type API library. It is also possible to specify these tag names and store them as task result business data in a database managed by the Process Flow Manager. The user element tags used by applications are described below.

- Relationship between Process Variables and Business Data
  The business data managed by the Process Flow Manager is managed by assigning a variable name (tag name) to each item of business data handled by an application. These variables are called process variables, and are defined by the Process Definition Tools.
  An application can refer to and store the business data handled by an application from the business data of a process by specifying these process variables (tag names).

- Relationship between Activity I/O Variables and Business Data
  From among the process variables described above, the business data variables (tag names) received by an application are defined in the activity input variables of a Process Definition Tool. Further, the variable names (tag names) of the task result business data returned by an application are defined in the output variables of the activity.

3.1.6 Multi-tasking Applications Running on Multiple Threads

When creating multi-tasking applications that run on multiple threads, note that client-type API libraries and server-type API libraries are not designed for multi-thread operation. Therefore, when calling an API library method, always do so on the same thread the API library object was created on.
3.2 Basic API Functions

This section describes the basic functions required to operate processes and activities.

3.2.1 Basic Functions for Processes and Activities

This section describes the basic client-type API functions required to operate processes and activities.

- **Initialization**: Initializes the client-type API by setting user information and client authentication certificates.
- **Operation of Process Definition Tools**: Performs operations on business processes such as viewing processes and creating processes. When an application generates a form, it can obtain a list of the processes and view these processes.
- **Activity Operation**: Performs operations on the task (activity) that is active within a process, such as reading task input data (business data) and updating task result data (business data). Before initiating a task, the application can obtain a list of the tasks to be managed.
- **Process Operation**: Performs operations on created processes, such as reading a list of created processes, viewing process status and terminating a process. The application can confirm how far a task within a process has progressed, or suspend a task within a process.
- **List searches**: It is possible to read a list of Process Definition Tools to be created, read a list of created processes and read a list of tasks (worklist). The list can be narrowed by setting search conditions.

These fundamental client-type API functions can be executed with client-type API libraries. For details of how to use the client-type API libraries, refer to the Interstage CollaborationRing PM Reference Manual (Process Flow Manager Edition).

3.2.2 Process/Activity Search Filter Function

The elements that can be focused in a client-type API search filter function search vary according to the information being searched for (Process Definition Tools, processes, and activities). The following table shows which conditional expressions can be specified.

<table>
<thead>
<tr>
<th>Conditional Conditional Expressions</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Value matching</td>
<td>• Equal to the specified value</td>
</tr>
<tr>
<td>Size comparison</td>
<td>• Greater than the specified value</td>
</tr>
<tr>
<td></td>
<td>• Less than the specified value</td>
</tr>
<tr>
<td></td>
<td>• Greater than or equal to the specified value</td>
</tr>
<tr>
<td></td>
<td>• Less than or equal to the specified value</td>
</tr>
<tr>
<td>AND of conditional expressions</td>
<td>• Logical AND of listed conditional expressions</td>
</tr>
</tbody>
</table>

3.2.3 Attachment Function

It is often the case in e-business transactions that product catalogs are sent as attachments of business documents. The information contained in these attachments needs to be managed so that it is always up to date, and the Attachment function is an API that does this by retrieving and updating information in attachments set in processes stored in business documents. This information includes the file name, path, file type, file comment and so on. The name of the file server in which the file is stored is set with the Management Tool.

When using a Web server as the file server, it is also possible to store, extract and delete files using HTTP libraries.

3.2.3.1 Using the Attachment Function

Follow the procedure below.

1. Control server environment settings
   - Using the Management Tool, set the file server details.
2. Process definition design
   1. Variable settings for attachment information
      - Using the Process Definition Tool, set the name of the variable in which attachment information is stored. Select Attachment for the variable type.
   2. Variable settings for input/output of activity (UI, automatic execution) that operates attachment information.
      - Using the Process Definition Tool, set the variable in which attachment information is stored in the input variable or output variable.
3. Client-type application design
   - Create applications for retrieving and updating file information and forwarding files.
     1. Set attachment information in the variable (of type Attachment).
To do this, use the methods CRF_SetUserAttachmentElement and CRF_SetControlElement. You can set the following file information:

- Comment
- File name
- Relative path
- Type
- File size

2. Update process data (file information)
   Update (i.e., set, modify or delete) the file information in the process data using the client-type API operations below.
   - CRF_CreateProcess
   - CRF_StartProcess
   - CRF_UpdateProcess
   - CRF_UpdateActivity
   - CRF_CompleteActivity
   - CRF_SendActivityBack

3. Retrieve process data (file information)
   Retrieve the file information from the process data using the client-type API operations below.
   - CRF_GetProcess
   - CRF_GetActivity

4. Retrieve the attachment name and file server name from the variable
   Retrieve the file information from the variable set with the Process Definition Tool using the methods CRF_GetUserAttachmentElement and CRF_GetControlElement.

5. Forward the attachment
   When using a Web server as the file server, it is also possible to upload and download attachments using HTTP libraries.

4. Automatic execution application design
   Create applications for retrieving and updating file information and forwarding files.
   1. Retrieve the attachment information and file server name from the variable
      The attachment information and file server name are reported in the server-type API input data (activityDataIn).
      Retrieve the attachment information from the variable set with the Process Definition Tool using the methods CRF_GetUserAttachmentElement and CRF_GetControlElement.
   2. Forward the attachment
      When using a Web server as the file server, it is also possible to upload and download attachments using HTTP libraries.
   3. Set attachment information in the variable
      Set attachment information in the server-type API output data (activityDataOut) and update process data.
      You can set the attachment information in the variable set with the Process Definition Tool using the methods CRF_SetUserAttachmentElement and CRF_SetControlElement.

The diagram below illustrates the above process.
[Using the Attachment Function]

3.2.3.2 Using Subprocesses

When linking subprocesses over different systems, it is sometimes necessary to send and receive attachments. This means that, when the subprocess is started, the attachment must be sent from the file server of the main process to that of the subprocess. Similarly, when the subprocess ends, the attachment must be sent from the file server of the subprocess to that of the main process.

This section explains how to do this.

- **To send the attachment from the file server of the main process to that of the subprocess**
  In the process definition, in the Activity Properties of the subprocess, set a user exit. This means that the application to forward files (the user exit routine) can be called when the subprocess starts.

- **To send the attachment from the file server of the main process to that of the subprocess**
  In the process definition of the subprocess, in the Activity Properties, set a user exit. This means that the application to forward files (the user exit routine) can be called when the subprocess ends.

When an internal subprocess starts, or when the different systems share the same file server, there is no need to forward files, and you can make settings as for a normal subprocess.

+ User Exit Routine Design

User exit routines are designed in the same way as automatic execution applications, as described under 3.2.3.1 Using the Attachment Function above.

The names of the file servers that send and receive files are notified to the user exit routine.

3.3 Retry Process Design

When restarting an application after an application error or a forced termination operation, it is necessary to perform certain operations, such as retry processes. This section describes the procedures that need to be carried out.

Process Flow Manager commits information to a database at the end of each activity, and maintains the transactional capability of business documents. This concept is explained further in the Interstage CollaborationRing PM Guide.

Nevertheless, in order to maintain close consistency between the control server and applications, it is still necessary to implement careful system design that includes preventing duplicate processing caused by recovery/re-execution procedures.

3.3.1 Re-executing Activities

To ensure consistency between the control server and applications and activities, re-execute the potentially inconsistent activities using one of the following procedures. The restarting procedure varies according to the type of activity.
+ a) UI Activities (Client-Type API)

Re-execute using one of the following methods.

- Continuous retries from a client that continues to operate.
  When an error occurs on the server, the client request will fail. Recover by retrying with exactly the same data.
  This may result in the restarted server rejecting the duplicated request using one of the following codes. This means that the previous request succeeded.
  - Operation mode: Complete task ("Complete") and update ("Update")
  - Error type: InvalidKey
  - Error code: 0x5002 or x080045202
- Reissue a request from an application on the failed server.
  Reissue the request when a business application on a restarted server has issued the Process Flow Manager API.
  - Handle by means of an application processing an uncompleted UI activity
    Restart from retrieving the list (SearchActivities or SearchProcesses).
  - Handle by repeating form submission (process creation)
    Attach the same number to the requested parameter and resubmit the request (CreateProcess). When the server receives the duplicate request, the server will return the same result as before.

+ b) Automatic Execution Activities

- Re-execute after checking the error event.
  An automatic execution application that has terminated abnormally has its failure to execute recorded as an error event in the control server. The administrator can use the Management Tool to verify the content of the error event, and then re-execute using either the Management Tool or the event retry command (crsrevent).
  An error event is recorded for automatic execution applications that have failed due to other problems as well, so the decision to restart or not must be left up to the administrator.
- Automatic rescheduling of automatic execution applications that were running
  The control server will automatically reschedule execution of automatic execution applications that were being called when the server failed, without the need for any additional operation by the administrator. Measures to prevent duplicate processing by automatic execution applications are described in the following section.

+ c) Subprocess Activities

If the server fails when the control server was in the process of starting a subprocess, the control server will reschedule the subprocess automatically when it restarts, without the need for any additional operation by the administrator.

3.3.2 Duplicate Execution Prevention Function

The duplicate execution prevention function is intended to prevent automatic execution applications (applications that use the server-type API) and the control server from reprocessing the same content when the process is retried following system errors (including communications errors).

Depending on the type of Process Flow Manager API, there are two types of duplicate execution prevention function: server-type and client-type. The following sections provide a description of each type.

3.3.2.1 Server-type Duplicate Execution Prevention Function

The server-type duplicate execution prevention function prevents an automatic execution application from reprocessing the same content when the process is retried following an error.

This section describes the duplicate execution prevention function as used in different cases:

- Using the auxiliary duplicate execution prevention function (See 3.3.2.1.1)
- Using the Execution Server (OTS Linkage Server) (See 3.3.2.1.2)

3.3.2.1.1 Preventing Duplicate Processing Using the Server-type API Auxiliary Duplicate Execution Prevention Function

Using the server-type API duplicate execution prevention function, you can prevent automatic execution applications from reprocessing the same content when a process is retried following an error. Using this supplementary function, you can detect and prevent duplicate processing in automatic execution applications.

With the auxiliary duplicate execution prevention function, the automatic execution application uses the following interfaces:

- Message ID
- Notification of End of Automatic Execution Application
++ Message ID

The message ID is a unique 28-digit number attached by the control server to transmitted data, so that an automatic execution application can recognize if the data have already been processed.

When an automatic execution application has finished processing a reply to the control server, a communication error or other type of error can occasionally prevent data from reaching the control server. In response to the error, the control server may resend the data to the automatic execution application for processing. In such cases, the application pairs the output data with a message ID and remembers the pair. The application then uses this information to determine if data sent by the control server are being re-sent as a result of error recovery. If so, the application sends the data back to the control server without processing the data again.

++ Notification of End of Automatic Execution Application

Notification of End of Automatic Execution Application is a server-type API that notifies an application that an automatic execution activity has finished (even in the control server) when the control server receives correct processing results from the automatic execution application. (Note that notification is only sent if requested by the application.)

The automatic execution application temporarily retains log information (such as the message ID and input data) in order to prevent processing being duplicated. The automatic execution application temporarily deletes this information after receiving notification of the end of automatic execution activity.

Note that when the auxiliary duplicate execution prevention function is used, there is no need to use the Execution Server (OTS Linkage Server).

The flow of application processing not using the server-type duplicate execution function is shown below:
++ Sample Program

A sample program for not using the server-type duplicate execution prevention function is stored in the following location:

Distribution medium/CRF/sample/SendCompleteMessage

[Flow of Application Processing when not using the Server-type Duplicate Execution Prevention Function]
following settings:
- Select Process Definition Properties, then DB Write Timing, then Always.
- Select automatic execution activity Activity Properties, then Call Method, then Call Not Via Execution Server (OTS Linkage Server).

3.3.2.1.2 Preventing Duplicate Processing Using the Execution Server (OTS Linkage Server)

You can also use the duplicate execution prevention function with the Execution Server (OTS Linkage Server) to prevent automatic execution applications from duplicating processing without having to worry about negative effects on performance.

However, calls to automatic execution applications may be slower when compared to the method described in 3.3.2.1.1, because this method requires the Execution Server (OTS Linkage Server). In addition, with this method the targets for prevention are duplicate database updates including transactions that use OTS.

For this method, the control server calls the automatic execution application via the Execution Server (OTS Linkage Server).

Accordingly, duplicate execution is prevented in the following cases:

++ If a problem occurs during processing of an automatic execution application

If a problem occurs during the processing of an automatic execution application and the process results are not reported to the Execution Server (OTS Linkage Server), the Execution Server (OTS Linkage Server) will determine that an error has occurred in the automatic execution application and notify the control server. The control server will then treat this as an error event.

In response to the error event, the system administrator of the Process Flow Manager should instruct that the event be checked and attempted again using the Management Commands or the Management Tool. When the system administrator orders the retry, the control server calls the automatic execution application again, and the same processing is retried.

However, problems can occur after the automatic execution application has completed processing - such as when a problem occurs during communications between the automatic execution application and the Execution Server (OTS Linkage Server). In such a case, by retrying the event as described above it is not possible to determine if the error occurred in the automatic execution application, which can lead to the automatic execution application duplicating its processing.

In response to this problem, the duplicate execution prevention function with the Execution Server (OTS Linkage Server) uses the Interstage Application Server Database Linkage Service to manage the automatic execution application database, and to roll back the automatic execution application database when a problem is detected in the automatic execution application.

In the case of an automatic execution application that uses a database to perform application processing, even if a problem such as that referred to above occurs, retrying the error event will result in the automatic execution application performing new processing because the database is rolled back. See the diagram below.
[Duplicate Execution Prevention Function Using the Execution Server (OTS Linkage Server) (1)]

++ Communication Problems between Control Server and Execution Server (OTS Linkage Server)

If the automatic execution application completes normally but communication problems occur when reporting the process results to the control server, the control server will detect the problem and treat it as an error event.

If, in response to the error event, the administrator of the Process Flow Manager directs that the event should be retried using the Management Commands or the Management Tool, the control server will call the automatic execution application again regardless of whether or not the automatic execution application completed its processing normally.

At this time, if the Execution Server (OTS Linkage Server) is being used, the Execution Server (OTS Linkage Server) prevents duplicate processing. Because the Execution Server (OTS Linkage Server) stores the process results of the automatic execution application, it can detect a duplicate request to start the automatic execution application is issued from the control server, in which case it does not call the automatic execution application, but instead reports the stored automatic execution application process results to the control server. This prevents duplicate starting of an automatic execution application. See the diagram below.
Preparing to Use the Duplicate Execution Prevention Function with the Execution Server (OTS Linkage Server)

The following operations must be performed to use the duplicate execution prevention function with the Execution Server (OTS Linkage Server):

- Construct Execution Server (OTS Linkage Server).
  The environments for the Execution Server (OTS Linkage Server) and the Database Linkage Service must be constructed.
- Create an automatic execution application that uses the Database Linkage Service.
  To use the server-type duplicate execution prevention function, it is necessary for the automatic execution application to use the Database Linkage Service.


3.3.2.2 Client-type Duplicate Execution Prevention Function

The client-type duplicate execution prevention function prevents the duplicate creation of a process at the control server, even when an application using the client-type API creates duplicate forms.

The client-type duplicate execution prevention function is effective in situations when a problem such as a break in communications occurs in the middle of form creation, preventing the form creation result being received.

If an application using the client-type API creates a form again using the same content and the process has not yet been created, the control server creates a new process. If creation of the process is already complete, the control server returns the same result as when the process was created. See the diagram below.
[Process Flow of the Client-type Duplicate execution prevention Function]

The client-type duplicate execution prevention function is organized as follows.

The control server examines the request ID set in the form data that is reported when an application using the client-type API creates a form. If a process with this ID has already been created, no new process is created, but the application using the client-type API is notified that the process was created successfully.

If no process with the request ID specified in the form data has been created, creation of a new process takes place.

Therefore, when a form is created, the application using the client-type API must set a request ID in the form data that is unique within the Process Flow Manager.

The client-type API supports the CRF_GetUniqueID method, which is a function to create the request ID automatically. When creating a form with an application using the client-type API, it is recommended that a request ID be set using this client-type API function.

3.3.3 Automatic Execution Application Automatic Retry

When processing of an automatic execution application cannot proceed because of a temporary resource shortage (due to excessive load or other cause) or a communication error, and an error code is set in the control server and recovery takes place, the control server can automatically invoke the same automatic execution application (automatic retry) after a fixed time interval.

The Management Tool can be used to set the error code recovery values that determine whether or not the control server executes the automatic retry function when the automatic execution application sets an error code in the control server and is recovered.

The following table shows the information to be set in the automatic execution application page of the Management Tool.

[Information set by the Management Tool]

<table>
<thead>
<tr>
<th>Setup information</th>
<th>Value range</th>
<th>Default value</th>
</tr>
</thead>
<tbody>
<tr>
<td>User defined error code for retry</td>
<td>1 - 99</td>
<td>Not set (no retry)</td>
</tr>
<tr>
<td>Setup information</td>
<td>Value range</td>
<td>Default value</td>
</tr>
<tr>
<td>-----------------------------------------</td>
<td>---------------</td>
<td>---------------</td>
</tr>
<tr>
<td>Retry count of user defined error</td>
<td>More than 0</td>
<td>0 (no retry)</td>
</tr>
<tr>
<td>Retry interval of user defined error (sec.)</td>
<td>1 - 120</td>
<td>1</td>
</tr>
</tbody>
</table>
Chapter 4 Application to Business Design

This chapter describes form types and how to design business processes for processing forms.

4.1 Form Type

This section describes single/batch forms and assembly and disassembly forms.

4.1.1 Single Forms and Batch Forms

This section describes single and batch forms.

Because forms can exist both as single forms and as batch forms, it is necessary to provide functions to assemble forms into a batch form and to disassemble a batch form into single forms.

CollaborationRing PM can perform form assembly and disassembly operations, and create business processes that can link to a variety of functions, such as message collection and distribution in batch, format conversion, and message sorting.

4.1.2 Assembly and Disassembly of Forms

Form assembly involves collecting multiple forms according to a number of conditions, and creating a batch form. Conversely, disassembly involves separating a batch form into single forms.

A batch form is a collection of forms in the same format. The assembly and disassembly operation does not alter this format, and forms that have undergone assembly or disassembly processing maintain their chronological sequence. This prevents forms that have undergone assembly or disassembly from being processed in reverse order.

There are two types of batch form: The form itself, and a form that contains just the application data extracted from forms. In the former case, forms have control information added to them, and they can be assembled and disassembled in that state. However, when performing EDI with a transaction business that does not have CollaborationRing PM installed, it becomes necessary to communicate the application data only, so the latter type is used. CollaborationRing PM allows assembly and disassembly processing to be designed according to both these operation modes.
4.2 Designing Applications to Assemble and Disassemble Forms

This section explains how to create single form processes from batch processes, and how to create batch processes from single form processes.

4.2.1 Single Form Processes from Batch Processes

The following diagram illustrates the design of an application to disassemble the received batch form into single forms, and create a process according to each form.
[Single Form Processes from Batch Processes]

- Processing of client-type application A  
  After the application receives the batch form, it uses the client-type API to set the batch form in process input variable <l> and create the process that will operate on the batch form.

- Processing of automatic execution application B  
  The application performs a batch conversion after it receives the batch form indicated by input variable <l> of activity (B). The form that underwent batch conversion is specified in output variable <m> of activity (B) and sent to the Control server, where it is stored in process control form (E).

- Processing of automatic execution application C  
  The application disassembles the batch form into single forms after it receives the batch form indicated by input variable <m> of activity (C). Also, for each separated form, form is set in process input variable <X>, and a process to operate on single form is created.

- Processing of automatic execution application D  
  The application creates an output form after it receives the single form (initial input form) indicated by input variable <X> of activity (D). The single output form (intermediate form) is set in output variable <Y> of activity (D) and sent to the Control server, where it is stored in process control form (F).

- Processing of automatic execution application E  
  After the application receives the single form (intermediate form) indicated by input variable <Y> of activity (E), final output form for performing batch processing of single forms is set in output variable <Z> of activity (E) and sent to the Control server, where it is stored in process control form (F).

Depending on the content of the form I/O variables, it is also possible to design complex Process Definition Tools with conditional branching, and to use other functions of the client-type API to design applications that search process control form that matches specified conditions.
4.2.2 Batch Processing from Single Form Processing

The explanation below is based on the example in the following diagram, which shows the design of an application that collects single form, creates batch form, and sends it.

- **Processing of client-type application F**
  The application uses the client-type API function, and, depending on the content of the form I/O variables, receives single form $<Z>$ that matches specified conditions from activity (F). After the received single form items are assembled into batch form, the client-type API is used to set batch form in process input variable $<o>$, and create the process that will handle the batch form.

- **Processing of application G**
  The application performs a batch conversion of the form after it receives the batch form indicated by input variable $<o>$ of activity (G). The form that underwent batch conversion is set in output variable $<p>$ of activity (G) and sent to the Control server, where the batch-converted form is stored in process control form (E).

- **Processing of automatic execution application H**
  After the application receives the batch form (form that has undergone batch conversion) indicated by input variable $<p>$ of activity (H), it sends it to an address that complies with the content of the form.

Depending on the content of the form I/O variables, it is also possible to design complex Process Definition Tools with conditional branching, and to use other functions of the client-type API to design applications that search process control form that matches specified conditions.
4.3 Designing Applications to Perform Batch Processing

This section describes how to design applications that process batch forms.

4.3.1 Processing Modes

The processing of batch forms can achieve a wide range of functionality, depending on the design of the application. The following are common usages.

- FTP linkage
  Communicates batch application data via FTP to accomplish simple linkage with another system.
- Process linkage
  Communicates batch form to use CollaborationRing PM and establish links with processes distributed across multiple systems.

4.3.2 FTP Linkage

This section provides an overview of the FTP linkage function and application flow.

4.3.2.1 Function Overview

FTP linkage communicates batch application data with another system.

The main functions of FTP linkage are:
Data reception and delivery
Performs FTP transmission of batch application data between business systems.

4.3.2.2 Application Flow

[Message Distribution (Assembly and delivery)]

+++ <Form generation>
(a) The assembly/disassembly function generates batch form and creates a predefined process.

+++ <Delivery>
(b) A message distribution activity is performed according to a Process Definition Tool.
(c) An FTP linkage process is called from the message distribution activity.
(d) Based on the information passed from the message distribution activity, the FTP linkage process extracts application data from batch form and performs FTP transmission (PUT).

[Reception (Reception and Disassembly)]

+++ <Form generation>
(a) As a result of message collection monitoring, a predefined process is created after batch application data is received.

+++ <Disassembly>
(b) A disassembly activity is performed according to a Process Definition Tool.
(c) The assembly/disassembly function is called from the disassembly activity. Based on the information passed from the disassembly activity, the assembly/disassembly function calls a disassembly process, which disassembles the batch form into single forms, and creates a process for each single form.
(d) The process results are returned to the disassembly activity.

4.3.3 Process Linkage

This section provides an overview of the process linkage function and application flow.
4.3.3.1 Function Overview

Process linkage generates forms of processes that span business systems (servers).

The main functions of process linkage are:

- Process transmission
  Creates forms of processes that span business systems (servers).
4.3.3.2 Application Flow

++<Assembly>
(a) An assembly activity is performed according to a Process Definition Tool.
(b) The assembly/disassembly function is called from the assembly activity. Based on the information passed from the assembly activity, the assembly/disassembly function calls an assembly process, which assembles single forms into a batch form, and creates a process. It is also possible to create a process on a different server.
(c) The process results are returned to the assembly activity.

++<Disassembly>
(d) A disassembly activity is performed according to a Process Definition Tool.
(e) The assembly/disassembly function is called from the disassembly activity. Based on the information passed from the disassembly activity, the assembly/disassembly function calls a disassembly process, which disassembles the batch form into single forms, and creates a process for each single form.
(f) The process results are returned to the disassembly activity.

4.4 Format Conversion
This section describes about format conversion and its procedures.

4.4.1 Defining Conversion Information
Format conversion refers to converting items to and from different record formats.
The FEDIT series of software is used to perform format conversion. The FEDIT series consists of a GUI software component and a conversion software component.
The GUI software is used on a PC to enter in correct sequence the items, attributes and number of digits of the input and output data formats (i.e., the collection of record items). The user then specifies which of the output items corresponds in meaning to which of the input items. The action of defining a correspondence between items is
referred to as mapping. The results of the mapping are output to a conversion table and sent to the server. The conversion software on the server then refers to the conversion table and converts the data format of the actual message.

[Format Conversion]
The FEDIT series of software can convert between the following formats:

- UN/EDIFACT format (and ANSI X.12)
- User-specific formats (fixed format, compound fixed format, structured format, line-delimited variable format, variable format - specified number of repeats)
- XML documents

+ Format Types
For details on the types of format handled by the FEDIT series, refer to the FEDIT Series manual.

+ Software
The software used differs according to the combination of input and output formats, and is shown in the following table.

[Format Conversion Software]

<table>
<thead>
<tr>
<th>Input format</th>
<th>Output format</th>
<th>GUI software</th>
<th>Conversion software</th>
</tr>
</thead>
<tbody>
<tr>
<td>User-specific format</td>
<td>User-specific format</td>
<td>FEDIT/FL-TABLE</td>
<td>FEDIT/FL-SV</td>
</tr>
<tr>
<td>Input format</td>
<td>Output format</td>
<td>GUI software</td>
<td>Conversion software</td>
</tr>
<tr>
<td>--------------</td>
<td>---------------</td>
<td>--------------</td>
<td>---------------------</td>
</tr>
</tbody>
</table>
| · Fixed format  
· Compound fixed format  
· Structured format  
· Line-delimited variable format  
· Variable format  
· Specified number of repeats | · Fixed format  
· Compound fixed format  
· Structured format  
· Line-delimited variable format  
· Variable format  
· Specified number of repeats | | |
| UN/EDIFACT standard format | User-specific format (*1) | FEDIT/EDIFACT-TABLE | FEDIT/EDIFACT-SV |
| User-specific format (*1) | UN/EDIFACT standard format | | |
| XML document (*2) | User-specific format (*3)  
· Fixed format  
· Structured format  
· Line-delimited variable format | FEDIT/FL-TABLE | FEDIT/FL-XML |
| User-specific format (*3)  
· Fixed format  
· Structured format  
· Line-delimited variable format | XML document (*2) | | |
| XML document (*2) | XML document (*2) | FEDIT/XML TRANSLATOR-TABLE | FEDIT/XML TRANSLATOR-SV |

*1 Structured format and Variable format - Specified number of repeats do not support input and output of UN/EDIFACT.

*2 Only XML documents possessing a hierarchical structure supported by FEDIT.

*3 Compound fixed format and Variable format - Specified number of repeats do not support XML data input.

### 4.4.2 Format Conversion Procedure

Format conversion is performed according to the procedure shown in the below figure.
[Format Conversion Procedure]

1. Register input data layout and output data layout
   Use the GUI software to enter in correct sequence the items, attributes and number of digits of the input and output formats (i.e., the collection of record items) of the data.
   - FEDIT/TABLE: Table data creation; master data creation
   - FEDIT/EDIFACT-TABLE: Master registration; registration of information about own company
   - FEDIT/FL-TABLE: Local layout registration
   - FEDIT/XML TRANSLATOR-TABLE: XML format registration

2. Register mapping
   Use the GUI software to specify which of the output items corresponds in meaning to which of the input items. The action of defining a correspondence between items is referred to as mapping, and the GUI uses arrows to show the mapping visually.
   - FEDIT/TABLE: Table data creation (Local format registration)
   - FEDIT/EDIFACT-TABLE: Mapping registration
   - FEDIT/FL-TABLE: Conversion information setting
   - FEDIT/XML TRANSLATOR-TABLE: Conversion information setting

3. Compile and send
   When mapping is complete, compile and create a conversion table in a format that the server can read. Also, if the directory where the conversion software is installed is specified beforehand, the conversion table is sent by FTP at the same time.
   - FEDIT/TABLE: SM table creation
   - FEDIT/EDIFACT-TABLE: Compiling and compilation data creation
   - FEDIT/FL-TABLE: Conversion information check and activation
   - FEDIT/XML TRANSLATOR-TABLE: Conversion information check and activation

4. Register operation, create execution parameter file, and send
   Use the GUI software to enter such information as the names of the storage files for the input and output data, and which mapping data will be used, then create the execution parameter file. Also, if the destination directory name is specified beforehand, the execution parameter file is sent by FTP at the same time.
   - FEDIT/TABLE: Operation registration; operation
   - FEDIT/EDIFACT-TABLE: Operation registration; linkage file registration
   - FEDIT/FL-TABLE: Operation registration; operation
5. **Revise EAP definitions**
   In the EAP definition used on the server, specify the execution parameter file name and the directory where the execution parameter file is to be sent.

6. **Execute**
   Perform the format conversion using CollaborationRing PM and automatic execution applications.

---

**Point**
- When the number of formats input and output by FEDIT/FL-TABLE or the number of format conversion definitions is large, use FEDIT/FL-TABLE-Client to share the definition registration work among multiple managers.
- You cannot use conversions of user-specific formats and XML or conversions of XML documents with XML documents in EAPs. Use FEDIT/Controller. See 4.4.3 Executing Conversion Between User-specific Format and XML for details.

---

### 4.4.3 Executing Conversion Between User-specific Format and XML or Between XML Document and XML Document

Use FEDIT/Controller to convert between user-specific formats and XML and between XML document and XML document.

FEDIT/Controller is an automatic execution application provided with the FEDIT series. Using FEDIT/Controller makes it easy to include the format conversion function provided with the FEDIT series in business processes.

---

#### [Conversion Between User-Specific Format and XML]

### 4.5 Linking to BizTalk Server

In order to link business processes with Microsoft(R) BizTalk Server for EAI systems, it is necessary to use a function that can link to the BizTalk Server. CollaborationRing PM provides the Linkage function with BizTalk Server for this purpose.

With this function, it is possible to fill in forms in BizTalk Server from CollaborationRing PM, and perform business processes in CollaborationRing PM from BizTalk Server such as sending and receiving form data.

#### 4.5.1 Positioning of Linkage function with BizTalk Server

The Linkage function with BizTalk Server is positioned as illustrated below.
The Linkage function with BizTalk Server is installed in the computer that runs BizTalk Server and exchanges messages with CollaborationRing PM by HTTP or HTTPS protocol.

Messages are received from CollaborationRing PM via Internet Information Services (IIS). Messages are sent to CollaborationRing PM via the Web server on the CollaborationRing PM side.

For details on the software required to run the Linkage function with BizTalk Server and the BizTalk Servers that can be linked, refer to the Installation Guide.

4.5.2 Applying Linkage function with BizTalk Server

The Linkage function with BizTalk Server acts as the CollaborationRing PM control server in BizTalk Server, using a subprocess linkage mechanism to implement business process linkage with CollaborationRing PM. For this reason, when a CollaborationRing PM subprocess form is filled, the form in BizTalk Server is filled from CollaborationRing PM. Conversely, business process forms are filled in CollaborationRing PM from BizTalk Server by means of calling COM+ applications provided with this function from XLANG schedule actions.

The diagrams below give an overview of each operation.
Chapter 5 Working Environment Setup and Application Control Operations

This chapter explains the working environment setup and application control operations performed by the Process Flow Manager.

5.1 Working Environment Setup

This section explains the operations and thinking behind the working environment setup.

- Naming Service Installation when using Execution Server (OTS linkage server)
- User Information Settings
- Subprocess linkage environment settings
- Environment settings of the mail linkage function

The following sections explain these elements.

5.1.1 Naming Service Installation when using Execution Server (OTS linkage server)

The following two system configurations exist to run automatic execution applications on server machines:

- System configuration in which all automatic execution applications run on a single server machine (Centralized system configuration)
- System configuration in which automatic execution applications are distributed across multiple server machines to distribute load (Decentralized system configuration)

Note that if you do not use the Execution Server (OTS linkage server), there are no special considerations regarding naming service installation and registering objects in the naming service.

5.1.1.1 Centralized System Configuration

This section describes the system configuration and Naming Service installation where all automatic execution applications run on a single server machine.

+ Naming Service Installation

It is recommended that a centralized configuration in which the control server, Execution Server (OTS linkage server) and automatic execution application objects are managed by a single Naming Service be used.

Refer to the Interstage Application Server Operator’s Guide for a detailed explanation of the Naming Service.

+ System Configuration Example 1

The diagram below shows a system configuration in which a control server, an Execution Server (OTS linkage server) and automatic execution applications are run on a single server machine.
[System Configuration Example 1]

+ System Configuration Example 2

To distribute load, it is also possible to operate a control server and automatic execution applications on two machines. In this configuration, the Execution Server (OTS linkage server) and the automatic execution applications should be installed on the same machine. The automatic execution applications will run more efficiently if run on the same machine as the Execution Server (OTS linkage server) due to the transaction control resulting from the Execution Server (OTS linkage server)'s support for duplicate processing prevention.

As in example 1, a centralized Naming Service installation is recommended.

The diagram below shows a system configuration distributing operations over two server machines.
5.1.1.2 Decentralized System Configuration

This section describes the system configuration and Naming Service setup used when automatic execution applications are distributed over multiple server machines. This configuration allows the load on the machines to be distributed.

- Location of automatic execution applications and Execution Server (OTS linkage server)s
  Because automatic execution applications are subject to the transaction control of the Execution Server (OTS linkage server)'s duplicate processing prevention function, automatic execution applications and Execution Server (OTS linkage server)s should be installed on the same server machine to improve performance.

- Location of Naming Service
  It is recommended that a Naming Service should be installed on each server machine so that objects in the entire system are controlled by multiple Naming Services.

- Example of Naming Service installation method
  Installation of the Naming Service when automatic execution applications are distributed over HOST_A and HOST_B is described below.

1. Registering application object names in the Naming Service of the hosts where the applications will run (HOST_A and HOST_B), register the object names of the automatic execution applications running on each host in the following format.
   (a) Register APPL1, APPL2, APPL3 and APPL4 in the root directory of the namespace.
   (b) Create a context, and register APPL1, APPL2, APPL3 and APPL4 under the context.

2. Registering Execution Server (OTS linkage server) object names which will run (HOST_A and HOST_B), register the object names of the Execution Server (OTS linkage server)s running on each
host in the following format.
(c) Register CRF_ApplServer in the root directory of the namespace.
(d) Create a context, and register CRF_ApplServer under the context.
(e) In the Naming Service of the control server (HOST_X), create the context name created in (d), and register the Execution Server (OTS linkage server) object name (CRF_ApplServer) running on each host.

- Specifying the Automatic Execution Activity Object Names
  The automatic execution application object names specified by the automatic execution activity in the process definition are defined starting from the namespace root (with a context name added).
  The diagram below shows the specification method.

![Diagram showing Naming Service Specification]

**[Naming Service Specification]**

Naming Service development is explained in the Interstage Application Server Operator's Guide.

### 5.1.2 User Information Settings

**+ User Information Relationship of End Users**

User access control is performed in the Process Flow Manager (refer to Chapter 2 2.1.6 Access Control for details). As a result, the information of users who permit access must be registered in the control server. Further, it is also necessary to set up basic authentication or client authentication with SSL2 and SSL3 in the Web server. The user information registered in the control server at this time consists of the user name and login name. When the Web server (HTTP bridge) employs Basic authentication, the login name reported from the client is posted to the control server without alteration. When client authentication by SSL2 and SSL3 is used, the distinguished name of the subject
of the user certificate is reported to the control server as the user name. When the control server is notified of a login name, it uses the login name to search for a user name, and when it receives a distinguished name, it uses the distinguished name as a user name. The control server controls access to process definitions and activities on the basis of the user name.

In contrast, applications using the client-type API and users using the Browser Linkage Function must provide information to the control server proving that they are registered users. (For a Web server employing Basic authentication, this means a login name and a password; and in the case of SSL2 and SSL3 client authentication, a user certificate.) The flow of user information between the client and the control server when performing Basic authentication on a Web server is shown in the diagram below.

[User Information Relationships]

The flow of user information between the client and the control server when performing SSL2 and SSL3 client authentication on a Web server is shown in the following diagram.
[User information relationships in SSL2 and SSL3 client authentication]

The user information registered in the Process Flow Manager is as shown in the table below.

[User Information to be Registered in the Control Server]

<table>
<thead>
<tr>
<th>Type of user information</th>
<th>Registered item</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>User name</td>
<td>User name</td>
<td>This name can be specified as the process definition owner, process creator and activity manager. Any name can be specified when performing Basic authentication at a Web server. When performing SSL2 and SSL3 client authentication on a Web server, specify all elements included in the DN of Subject in the user certificate according to the rules.</td>
<td>Register this item with User on the Management Tool.</td>
</tr>
<tr>
<td>User name</td>
<td>Login name</td>
<td>When performing basic authentication with a Web server, specify the login name registered in the Web server. When performing SSL2 and SSL3 client authentication on a Web server, specify any name. Specify this login name when using the Process Definition Tool as well.</td>
<td></td>
</tr>
</tbody>
</table>

The user information to be registered in the Web server differs according to the user authentication method set in the Web server. See the following table "User Information to be Registered in the Web Server" for the user information that needs to be registered in the Web server. Refer to 5.4 SSL Communication Environment Setup for how to set up SSL environment.
[User Information to be Registered in the Web Server]

<table>
<thead>
<tr>
<th>Type of user authentication</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic authentication</td>
<td>Register the login name and password of the user or the linkage approver.</td>
<td>Refer to the relevant Web server manual for the method used to register in the Web server.</td>
</tr>
<tr>
<td>SSL2 and SSL3 client authentication</td>
<td>Register the certificate of the CA that issued the certificate of the user or the linkage approver.</td>
<td></td>
</tr>
</tbody>
</table>

+ User Information Relationships During Subprocess Linkage

When performing subprocess linkage, the control server works as a user to create a Subprocess and report the results of the Subprocess. This user is distinct from an ordinary user and referred to as a "linkage user". When performing subprocess linkage, it is necessary to register the linkage user information in the Process Flow Managers on both sides.

The linkage user of the opposing Process Flow Manager that is registered in order to link subprocesses is referred to as a "linkage approver".

Information relating to the linkage user and the linkage approver is registered in the Process Flow Manager, but the registered content differs according to the communication mode.

When the opposing Process Flow Manager performs basic authentication, the login name and password of the linkage user are registered in the control server. This login name and password are registered even in the Web server of the opposing Process Flow Manager.

Also, when the local Process Flow Manager performs Basic authentication, the login name of the linkage approver of the opposing Process Flow Manager is set as the login name, and any name may be set as the user name.

If the opposing Process Flow Manager performs SSL2 and SSL3 client authentication, specify the nickname and PIN of the control server certificate at the control server. For the Web server of the opposing Process Flow Manager, register the certificate of the CA that issued the control server certificate.

If the local Process Flow Manager performs SSL2 and SSL3 client authentication, specify any name as the login name of the linkage approver and the DN of Subject of the control server certificate as a user name.

The flow of user information in the case of Basic authentication is shown in the diagram below.
[User Information Relationships in Basic Authentication]

The flow of user information in SSL2 and SS3 client authentication during subprocess linkage is shown in the following diagram.
[User information relationships in SSL2 and SSL3 client authentication during Subprocess linkage]

[User Information to be Registered in the Control Server]

<table>
<thead>
<tr>
<th>Type of user information</th>
<th>Registered item</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linkage approver name</td>
<td>User name</td>
<td>Refers to the name of the user permitting the creation of a subprocess when subprocess creation is requested from another control server. Any name can be specified when performing Basic authentication with a Web server. When performing SSL2 and SSL3 client authentication at a Web server, specify all elements included in the DN of Subject in the certificate of the control server that requested subprocess creation, according to the rules.</td>
<td>Register with Linkage Approver on the Management Tool.</td>
</tr>
<tr>
<td></td>
<td>Login name</td>
<td>When performing Basic authentication with a Web server, specify the login name registered in that Web server. When performing SSL2 and SSL3 client authentication at a Web server, specify any name.</td>
<td></td>
</tr>
<tr>
<td>Type of user information</td>
<td>Registered item</td>
<td>Description</td>
<td>Remarks</td>
</tr>
<tr>
<td>--------------------------</td>
<td>-----------------</td>
<td>-------------</td>
<td>---------</td>
</tr>
<tr>
<td>Linkage user</td>
<td>Linkage user name</td>
<td>User information reported to a control server when another control server asks it to create a subprocess. Specify the login name registered in the opposing Web server. The opposing Web server must be able to perform Basic authentication.</td>
<td>Set this item with Server Control in the Management Tool.</td>
</tr>
<tr>
<td>Password for linkage</td>
<td></td>
<td>Password corresponding to a linkage login name</td>
<td></td>
</tr>
<tr>
<td>Certificate nickname (*1)</td>
<td></td>
<td>The nickname of the certificate used when performing subprocess linkage by SSL communication.</td>
<td></td>
</tr>
<tr>
<td>User PIN of token (*1)</td>
<td></td>
<td>The user PIN used when performing subprocess linkage by SSL communication.</td>
<td></td>
</tr>
</tbody>
</table>

Refer to 5.4 SSL Communication Environment Setup for how to set up SSL environment.

The user information to be registered in the Web server differs according to the user authentication method set in the Web server. See the following table.

[User Information to be Registered in the Web Server]

<table>
<thead>
<tr>
<th>Type of user authentication type</th>
<th>Description</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Basic authentication</td>
<td>Register the login name and password of the user or the linkage approver.</td>
<td>Refer to the relevant Web server manual for the method used to register in the Web server.</td>
</tr>
<tr>
<td>SSL2 and SSL3 client authentication</td>
<td>Register the certificate of the CA that issued the user or linkage approver certificate.</td>
<td></td>
</tr>
</tbody>
</table>

5.1.3 Subprocess Linkage Environment Settings

When using a subprocess activity to link to another Process Flow Manager, it is necessary to understand the following points to set up the working environment:

- Ownership of the process definition of the subprocess
  - When creating a subprocess, the owner of the process definition must be "default". The owner of the process definition can be set or changed using the process definition tool.
- Subprocess result options
  - When a subprocess activity is synchronous, the results of a subprocess are reported to the subprocess activity upon completion of processing. At this time, the subprocess activity option is set to the value of the subprocess end node event. As a result, among the option arrows deriving from the subprocess activity of the main process, there must be an option arrow with the same title as the name of this end node event. If one does not exist, the subprocess activity in the main process will not terminate even when the subprocess terminates, and it will not be possible to proceed to the next task. However, when the end node event name is not set, the default arrow is selected as the transition destination.

The diagram below shows the process flow during subprocess linkage.
5.1.2 User Information Settings

User Information Relationships During Subprocess Linkage

It is necessary to register linkage users and linkage approvers in the control server.

In the Web server, it is also necessary to register the login name and password of a linkage approver.

5.1.4 Environment Settings of the Mail Linkage Function

The following environment settings are required to activate the mail linkage function:

+++ Email environment settings

It is necessary to register in the control server the location of the email template (specify the full pathname of the file on the machine where the control server is installed), the appropriate SMTP server, and the email address of the person sending the email.

These are all set by selecting Server Control and Change Environment from the Management Tool.

The email template is an email document sent to each manager by the control server. Such information as the process title and description, and variable values can be embedded in the template. See the Interstage CollaborationRing PM Reference Manual (Process Flow Manager Edition) for details.

5.2 Management Tool

To enable the Process Flow Manager to perform application control at the control server, it is provided with a Management Tool.

The Management Tool performs the following tasks to operate and manage the Process Flow Manager.

Refer to the online help of the Management Tool and the Interstage CollaborationRing PM Operation Guide for further details on how to use the Management Tool.

[Functions Executed by the Management Tool]

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Management Tool operation</th>
</tr>
</thead>
</table>
| Change the working environment of the control server | It is possible to change the working environment of the control server. Changes are made valid only after the control server is restarted. The following items can be changed using the Management Tool:  
  - Linkage user name and password (Control | From the menu bar of the Management Tool, select Server control followed by Environment change to display the Environment change window. Select the tab, and change the content as required. |
<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Management Tool operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>User management</td>
<td>It is possible to view, update, add and delete user information registered in the control server to view, grant and grant authorities, and to save lists to file. When displaying the user list, you can specify conditions to limit the number of users in the list. The information that can be displayed is the user name, login name, email address and associated group.</td>
<td>From the menu bar of the Management Tool, select Display. A list of objects that can be managed will be displayed. Select the object to be managed from the list. For example, to manage users, click on Users. Following this, click on Processing on the menu bar of the Management Tool, and select the process to be executed.</td>
</tr>
<tr>
<td>Group management</td>
<td>It is possible to view, update, add and delete group information registered in the control server, and to save lists to file. When displaying the group list, you can specify conditions to limit the number of groups in the list. The information that can be displayed is the name of the user belonging to the group.</td>
<td></td>
</tr>
<tr>
<td>Process management</td>
<td>It is possible to view, abort, delete an update processes, view status and variables, replace, add and delete the operator name of processes registered in the control server, and to save lists to file. When displaying the process list, you can specify conditions to limit the number of processes in the list. Information that can be displayed as process information includes the following: The total number of processes; the process ID; the global key; the process title; the process description; the priority; the creator; the status; the process creation date and time; the process completion date and time; the title of process definitions used in process creation; the version number of the process definition used in process creation; the process size; process attach document information and the process history.</td>
<td></td>
</tr>
<tr>
<td>Process definition management</td>
<td>It is possible to download, delete, change the title, and replace, add and delete the operator name of process definitions registered in the control server, view variables, save lists to file, as well as register process definitions created on a PC local machine to the control server. The process definition information that can be displayed using the Management Tool is the title name, the description, the revised version, the owner name and the URI.</td>
<td></td>
</tr>
<tr>
<td>Locked process management</td>
<td>It is possible to view, unlock and save to file locked processes registered in the control server. The information that can be displayed is the process ID, name of user who locked, lock key and date of locking.</td>
<td></td>
</tr>
</tbody>
</table>
### Task Description Management Tool operation

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>Management Tool operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Event management</td>
<td>It is possible to display a list of error events detected by the control server and to retry or delete error events, and save lists to file. You can also display a list of execution events (events that are being or have been executed). Refer to 3.3.2 Duplicate Execution Prevention Function in Chapter 3 for information on retrying error events. When displaying the error event and execution event list, you can specify conditions to limit the number of events in the list. The information displayed in an error event list includes the event message ID, the event generation date and time, the event start and end date and time, the event name, the process ID, the activity name that generated the error, the error message, the error code, and the global key.</td>
<td></td>
</tr>
<tr>
<td>Linkage approver management</td>
<td>It is possible to view, update, add and delete linkage approver information registered in the control server and save lists to file. When displaying the linkage approver list, you can specify conditions to limit the number of linkage approvers in the list. The information that can be displayed is the user name, the login name and the email address.</td>
<td></td>
</tr>
<tr>
<td>Send Agent database management</td>
<td>It is possible to delete process information on the Send Agent database and save lists to file. When displaying the process information list, you can specify conditions to limit the number of process information items in the list. The information that can be displayed is the process ID and the forwarding URL.</td>
<td></td>
</tr>
<tr>
<td>Collection database management</td>
<td>It is possible to delete process information on the collection database and save lists to file. The information that can be displayed is the process ID, the process end date and time, and the global key.</td>
<td></td>
</tr>
</tbody>
</table>

### 5.3 Application Control Commands

The Process Flow Manager provides the following commands to perform application control on the control server.

**[Application Control Commands]**

<table>
<thead>
<tr>
<th>Name of application control command</th>
<th>Processing performed by application control command</th>
<th>Actual command name</th>
</tr>
</thead>
<tbody>
<tr>
<td>History send agent application control command</td>
<td>This command executes the following processes:  · Suspends the data detection function and the transmission function of the history send agent. It is possible to suspend the data detection and transmission functions without stopping the history send agent. Two suspension modes are available; one mode saves the transmission queue information, and the other does not.  · Restarts the data detection function and the transmission function of the history send agent. It is possible to cancel the suspended status of the data detection and transmission functions of the history send agent.</td>
<td>crateadmin</td>
</tr>
<tr>
<td>Name of application control command</td>
<td>Processing performed by application control command</td>
<td>Actual command name</td>
</tr>
<tr>
<td>------------------------------------</td>
<td>-----------------------------------------------------</td>
<td>---------------------</td>
</tr>
</tbody>
</table>
| Command to change the transmission schedule of the history send agent temporarily | This command executes the following processes:  
  - Displays the content of the history send agent transmission schedule.  
    The content of the current transmission schedule of the history send agent can be displayed.  
  - Sets the history send agent event data search time interval  
    It is possible to temporarily change (or set) the time interval at which the history send agent searches for event data in the control server.  
  - Sets transmission start conditions of the history send agent  
    It is possible to temporarily change (or set) the number of event data items queued that is the transmission start conditions of the history send agent.  
  - Displays queue information in the history send agent  
    It is possible to display the current queue information in the history send agent (number of event data items, etc.).  
  - Sets information of batch transmissions to the Collection Server  
    It is possible to temporarily change (or set) the maximum number of event data items that the history send agent can send to the Collection Server at one time.  
  The content changed (or set) with this command is not inherited when the history send agent is restarted. When the history send agent is restarted, operation is determined by the transmission schedule specified in the environment definition file of the history send agent. | cragtenvadmin |
| History send agent transmission queue control command | This command executes the following processes:  
  - Displays the status of the transmission queue of the history send agent.  
    The transmission status of the history send agent can be displayed. The transmission queue can be in the following states:  
      - Ready to send  
      - Transmission stopped  
      - Forced transmission  
  - Suspends queued transmission of history send agent.  
    It is possible to suspend the queued transmission of a history send agent with a specified destination server ID. When “all” is specified, the queued transmissions of all destination servers are suspended.  
  - Restarts the queued transmission of the history send agent.  
    It is possible to restart (or start) the queued transmission of a history send agent with a specified destination server ID. When “all” is specified, the queued transmissions of all destination servers are restarted.  
  - Forced queued transmission of history send agent  
    It is possible to force the queued transmission of a history send agent with a specified destination server ID. When “all” is specified, the queued transmission of all destination servers is enforced. | cragttsend |
| History send agent status display command | Displays the operating status of the history send agent. The history send agent can be in the following operating states:  
  - Operating (Ready to transmit)  
  - Operating (Transmission stopped)  
  - Stopped | cragtstatus |
<p>| History send agent start command | Starts the history send agent. | cragtstart |</p>
<table>
<thead>
<tr>
<th>Name of application control command</th>
<th>Processing performed by application control command</th>
<th>Actual command name</th>
</tr>
</thead>
<tbody>
<tr>
<td>History send agent stop command</td>
<td>Stops the history send agent.</td>
<td>cragtstop</td>
</tr>
</tbody>
</table>
| Execution Server (OTS linkage server) application control command | This command executes the following processes:  
  · Terminates Execution Server (OTS linkage server). This command can be used to forcibly terminate the Execution Server (OTS linkage server). If forced termination is carried out, the Execution Server (OTS linkage server) will terminate even if automatic execution applications are running.  
  · Displays Execution Server (OTS linkage server) status. This command can be used to display the operating status of the Execution Server (OTS linkage server). The Execution Server (OTS linkage server) can be in the following operating states:  
    - Executing  
    - Starting  
    - Stopped  
  · Displays the status of the Execution Server (OTS linkage server) database. This command can be used to display the information registered in the Execution Server (OTS linkage server) database. This is the information stored by the Execution Server (OTS linkage server) for the duplicate processing prevention function.  
  · Deletes the information stored in the Execution Server (OTS linkage server) database. This command can be used to delete the above information. | crasvradmin |
<p>| Process creation command           | Creates a process. Creates a process according to process definitions retrieved using criteria such as the process definition title and process definition description. When multiple process definitions are retrieved, process generation is performed for all the process definitions retrieved. | crfcrtproc |
| Command to replace the process definition manager name and change the title | Replaces the name of the manager of a process definition and changes the title. It reads an update file and changes the process definition information according to the content of this file. The information to be changed is replaced by the content of the [ITEM] section in the update file. | crfschgprcdf |
| Command to change the process manager name and information | Changes the manager name and information of processes. It reads an update file and changes the process information according to the content of this file. The processes whose information is to be changed can be narrowed by the content of the [FILTER] section within the update file. The information to be changed is replaced by the content of the [ITEM] section in the update file. | crfschgproc |
| Event deletion command             | Deletes an event. The event is deleted according to a specified event message ID and error code. Event message ID can be directly specified or the file that includes the description of event message ID can be specified when inputting crfsdelevent command. | crfsdelevent |</p>
<table>
<thead>
<tr>
<th>Name of application control command</th>
<th>Processing performed by application control command</th>
<th>Actual command name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command to delete log information from the Collection Server</td>
<td>Deletes information from the Collection Database on the log server. The information to be deleted is specified using the process ID. The process ID can be specified directly using parameters, or passed as list data in a file. This command can use the output data of the crfslshista command, which outputs a log information process list on the log server, as the process ID list data.</td>
<td>crfsdelhista</td>
</tr>
<tr>
<td>Command to delete log transmission queue database information from the control server</td>
<td>Deletes information from the log transmission queue database on the control server. The information to be deleted is specified using the process ID. The process ID can be specified directly using parameters, or passed as list data in a file. This command can use the output data of the crfslshistq command, which outputs a log transmission queue database process list on the control server, as the process ID list data.</td>
<td>crfsdelhistq</td>
</tr>
<tr>
<td>Process delete command</td>
<td>Deletes processes on the control server. It is possible to delete all processes on the control server, and to delete only those processes that match conditions specified in the command, such as initial status, suspension status, the process ID, the global key, the name of the owner of the process definition used when the process was created, the process definition title, and the process creation date. By specifying the file containing the list of process IDs, it is possible to delete processes. If the process has an attachment set, a warning message can be set to appear when the process is deleted.</td>
<td>crfsdelproc</td>
</tr>
<tr>
<td>Command to delete multiple server processes with a global key</td>
<td>Deletes closed processes with an identical global key from multiple servers. This command can also display a list of processes with an identical global key existing on each server without performing delete processing.</td>
<td>crfsdelprocgk</td>
</tr>
<tr>
<td>Command to output an error event list</td>
<td>Outputs an error event list. Using the event message ID obtained with this command, you can retry error events (crfsrevent) and delete error events (crfsdelevent).</td>
<td>crfslsevent</td>
</tr>
<tr>
<td>Command to output a log information process list on a Collection Server</td>
<td>Outputs a list of process IDs whose information exists in the Collection Database on the log server. The process ID list output can be made conditional by specifying all processes, processes in closed status, process creation dates, process termination dates, or global keys. The process ID list output with this command can be used as data for the crfsdelhista command, which deletes log information from the Collection Server.</td>
<td>crfslshista</td>
</tr>
<tr>
<td>Command to output a process list to a log transmission queue database on the control server</td>
<td>Outputs a process ID list for processes whose information exists in the log transmission queue database on the control server. The process ID list is output by specifying the destination (URI) with parameters. The process ID list output with this command can be used as data for the crfsdelhistq command, which deletes log transmission queue database information from the control server.</td>
<td>crfslshistq</td>
</tr>
<tr>
<td>Command to output a process list to a log transmission queue database on the control server</td>
<td>Outputs an address (URI) that exists in the log transmission</td>
<td>crfslshists</td>
</tr>
<tr>
<td>Name of application control command</td>
<td>Processing performed by application control command</td>
<td>Actual command name</td>
</tr>
<tr>
<td>-----------------------------------</td>
<td>-----------------------------------------------------</td>
<td>---------------------</td>
</tr>
<tr>
<td>log transmission destination on the control server</td>
<td>queue database on the control server.</td>
<td></td>
</tr>
<tr>
<td>Process list output command</td>
<td>Outputs a list of processes. It is possible to display all processes, and to display only those processes that match conditions specified in the command, such as lock status, suspension status, initial status, active status, the process ID, the global key, the name of the owner of the process definition used when the process was created, the process definition title, user-defined custom field value, and the process creation date. The user can specify whether output information is to be either process IDs or process IDs and detailed information. Information can also be output in CSV format.</td>
<td>crfslsproc</td>
</tr>
<tr>
<td>Output command for process information</td>
<td>Narrows down optional information on a specified process ID and retrieves the process information from the control server. XML or CSV can be selected as the output format.</td>
<td>crfprocinfo</td>
</tr>
<tr>
<td>Batch process command for multiple servers</td>
<td>Performs batch command processing for multiple servers. The two commands supported in this version are crfsmntuser and crfsaddplan. The maximum number of servers supported is 255. This command is executed by specifying a server file that designates the servers, and a command file that designates the commands.</td>
<td>crfsmltsvr</td>
</tr>
<tr>
<td>User registration information maintenance command</td>
<td>Deletes groups and users registered in the control server and updates information and then registers user information in the control server. You can register information of multiple users at the same time. You can also import and export user registration information, as with the Management Tool.</td>
<td>crfsmntuser</td>
</tr>
<tr>
<td>User/group information output command</td>
<td>Outputs in default format the list of user/group information registered in the control server. You can also output information in CSV format.</td>
<td>crfslsuser</td>
</tr>
<tr>
<td>Get user registration information from directory server command</td>
<td>Gets user registration information from an InfoDirectory or iPlanet directory server and creates a user registration file.</td>
<td>crfsgetldap</td>
</tr>
<tr>
<td>Batch register user registration information command</td>
<td>Gets user registration information from an InfoDirectory or iPlanet directory server and imports it to the control server. In effect, combines the operations of crfsgetldap (for extracting user registration information from directory servers) and crfsmntuser (for saving user registration information).</td>
<td>crfsimportldap</td>
</tr>
<tr>
<td>Process unlock command</td>
<td>Unlocks a locked process. The process to be unlocked is specified using the process ID. By specifying the file containing the list of process IDs, it is possible to unlock processes.</td>
<td>crfsrellock</td>
</tr>
<tr>
<td>Group reset command</td>
<td>Deletes all groups and users registered in the control server.</td>
<td>crfsresetgrp</td>
</tr>
<tr>
<td>Event re-execution command</td>
<td>Re-executes an error event detected at the control server. Re-executes the event corresponding to the event message ID or error code that is specified in the command. The event message ID can also be input from file.</td>
<td>crfsetevent</td>
</tr>
<tr>
<td>Name of application control command</td>
<td>Processing performed by application control command</td>
<td>Actual command name</td>
</tr>
<tr>
<td>------------------------------------</td>
<td>-----------------------------------------------------</td>
<td>---------------------</td>
</tr>
</tbody>
</table>
| Control server status display command | Displays the operating status of the control server. The control server can be in the following operating states:  
  - Undergoing initialization  
  - Operating  
  - Stopping  
  - Stopped  
  - Error status  
  - Unable to connect (Stopping complete) | crfsstatus |
| Control server stop command | Stops the control server. | crfsstop |
| Process start command | Starts or restarts a process.  
  Searches for a process according to criteria such as the process ID or global key, process definition title, process creator, process creation date and time, and custom attributes, and starts the process retrieved.  
  If a process has been suspended using the process suspension command or the client-type API command SuspendProcess, this command can be used to restart the process. | crfstartproc |
| Process suspension command |Suspends a process.  
  Searches for a process according to criteria such as the process ID or global key, process definition title, process creator, process creation date and time, and custom attributes, and suspends the process retrieved.  
  If a process is suspended using this command, it can be restarted with the process start command or the client-type API command StartProcess. | crfsuspproc |
| Process termination command |Terminates a process.  
  Searches for a process according to criteria such as the process ID or global key, process definition title, process creator, process creation date and time, and custom attributes, and terminates the process retrieved.  
  If a process has been terminated using the process termination command, it cannot be restarted with the process start command or the client-type API command StartProcess. | crftermproc |
| Global history query application control command | This command executes the following processes:  
  - Deletes all global history query connection management information.  
  - Deletes all expired global history query connection management information stored on a Web server.  
  - Deletes the global history query connection management information prior to a specified date.  
  - Deletes the global history query connection management information stored on a Web server that has expired prior to a specified date.  
  - Deletes the connection management information of log queries within a specified period.  
  - Deletes the global history query connection management information stored on a Web server that has expired within a specified period.  
  - Displays the connection management information of log queries.  
  This command can display a list of the global history query connection management information stored on a Web server.  
  - Deletes the global history query connection management information items with a specified ID. | crhcladmin |
<table>
<thead>
<tr>
<th>Name of application control command</th>
<th>Processing performed by application control command</th>
<th>Actual command name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Command to process transaction analysis</td>
<td>This command stores in the transaction analysis database the transaction logs used for transaction analysis and processes the logs.</td>
<td>crpaexec</td>
</tr>
<tr>
<td>Command to manage transaction analysis users</td>
<td>This command manages definition information for the users who log in to the Web screen for transaction analysis.</td>
<td>crpauser</td>
</tr>
<tr>
<td>Command to encrypt trading analysis passwords</td>
<td>This command encrypts the password for access to the database of the trading analysis that is used.</td>
<td>crpacrypt</td>
</tr>
<tr>
<td>Set delegate command</td>
<td>Makes it possible to set a delegate (a user to perform a process as an agent) for a delegator (the user originally intended to perform the process).</td>
<td>crfsaddelegate</td>
</tr>
<tr>
<td>Delete delegate command</td>
<td>Deletes a delegate setting.</td>
<td>crfsdeledelegate</td>
</tr>
<tr>
<td>Output delegate list command</td>
<td>Outputs a list of delegates.</td>
<td>crfslsdeledelegate</td>
</tr>
<tr>
<td>Output delegated user list command</td>
<td>Outputs a list of delegators.</td>
<td>crfsisdelegetator</td>
</tr>
<tr>
<td>Output attachment list command</td>
<td>Outputs a list of attachments set for a process.</td>
<td>crfslsattach</td>
</tr>
</tbody>
</table>

### 5.4 SSL Communication Environment Setup

The SSL protocol can be used to provide a high level of security for communications between an application using the client-type API and a control server or HTTP communications between control servers. SSL protocol versions 2.0 and 3.0 are supported. This section describes how to set up a communication environment when using SSL protocol.

Refer to 5.1.2 User Information Settings for details on user authentication methods used in SSL communications and user information settings in each server.

SSL protocol environment setup involves setting up the SSL environment for Web servers and the SSL environment for client-type API libraries.

- **Web Server Environment Setup**
  - Refer to the documentation of the product used as the Web server for details on Web server functions and operation procedures, and SSL terminology and commands.
- **Client-type API Library Environment Setup**
  - Set up the environment for client machines using client-type API libraries or server machines.
Chapter 6 The Enterprise Application Pipeline

This chapter discusses the Enterprise Application Pipeline (EAP) and the functions it provides, and explains how to use it with applications.

6.1 Overview of Typical Functions

The Enterprise Application Pipeline contains the following functions to enable flexible connections between applications, and to allow the operation status to be verified with ease.

- **Data processing**
  - Performs translation (format conversion) of data suited to the following task, data sorting, etc.
- **User exits**
  - You can combine user applications before and after data processing. This makes possible procedures such as the following on data passed to following tasks:
    - Conversion to customized formats
    - Data backup
    - Deleting unneeded data
- **Process commencement timing according to application**
  - The time when data processing starts can be selected according to how the data is used. The timing of data processing can be set to suit batch-type applications that process data all at once, or real-time applications that process data as soon as they are generated. The instruction can be made by using the application created with the linkage instruction API or by using a command.
- **Tracking of processed data**
  - Data processed by the Enterprise Application Pipeline can be tracked using the Global Tracking function of the process flow manager.
- **EAP Management Tool**
  - EAP services can be viewed and deleted using a Web browser.

The application linkage definitions required to realize these functions can be manipulated from a GUI window to facilitate installation and operation by the user. For basic information on application linkage definitions, refer to 6.2 Overview of EAP Definitions.

The maximum amount of data that you can process in EAP is two gigabytes.

6.1.1 Data Processing

The following two types of data processing are performed:

- Translation (format conversion)
- Splitting/merging/sorting

Because the order in which these two processes are performed is determined by such factors as the form format types of the leading and following tasks, and the presence of sort processing, they are handled as patterns in the Enterprise Application Pipeline. The user can perform data processing from the leading task to the following task easily by using the application linkage working environment definitions and the application linkage definitions to select the pattern to process.

6.1.1.1 Format Conversion

When the form formats of the leading task and the following task differ, the format must be converted according to the form of the following task. Refer to the FEDIT manuals for a detailed description of form formats. The following table shows the form formats that the Enterprise Application Pipeline can handle.

### [EAP-manageable Form Formats]

<table>
<thead>
<tr>
<th>Form format</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>UN/EDIFACT standard format</td>
<td>UN/EDIFACT (United Nations/Electronic Data Interchange For Administration Commerce and Transport): Created as ISO09735 and ISO07372 to be an international standard for EDI.</td>
</tr>
<tr>
<td>User-specific format</td>
<td>The following three types can be handled in EAP:</td>
</tr>
<tr>
<td></td>
<td>- Structured format</td>
</tr>
</tbody>
</table>
6.1.1.2 Data Splitting/Merging/Sorting

Merging refers to collecting multiple forms into one data. Splitting is the opposite operation and refers to subdividing one piece of data containing multiple forms into individual forms.

Sorting is the process of separating data according to destination address when passing data created by a leading task to applicable following tasks. The relevant data is separated based on destination information contained in the data. The content of these splitting/merging/sorting processes is also specified in application linkage definitions.

The following formats can be sorted:
- User-specific format

The diagram below provides an overview of sorting and batch processing.

[Sorting and Batch Processing]

If there are no leading task data, the Enterprise Application Pipeline will handle the processing normally. However, whether or not there is linkage to the following task is specified in the output logical node definition of the EAP definition.

6.1.2 User Exit

Two types of function are provided for when EAP calls a user application, as described below.

+ User Exit Functions

The linkage from the leading task to the following task calls the specified user application before or after data processing. The user application executes the following operations on the forms that are transferred to the following task:
- Reference
- Delete
- Update
- Add

The those operation in the user application can add the following process to the forms that are transferred to the following task:
- Using the reference operation
The form backup, monitoring the specified form etc.
· Using the delete operation
  Deleting the specified form etc.
· The update operation
  The user-specific format conversion, sorting the forms etc.
· The add operation
  Adding the specified form etc.

+ Post-EAP User Exit Functions

In addition, the Post-EAP User Exit functions can be used to add user processes to forms asynchronously once EAP is complete. It is also possible to combine customized processes (such as notifying results to leading and following tasks or system administrators and recovering after errors) in user processes.

6.1.3 Process Initiation

The Enterprise Application Pipeline can initiate processing in the following three situations:
· When a linkage instruction is issued from an application
· When the real-time linkage function is used
· When the message collection monitoring function is used (message collection terminates normally)

Use the processing commencement timing that is appropriate to the operation. For example, to perform batch processing of multiple data, use a linkage instruction from an application. To process each item of data as soon as it is generated, use the real-time linkage function or the message collection monitoring function.

6.1.3.1 Linkage Instruction Issued by an Application

By using the linkage instruction API provided by the Enterprise Application Pipeline, it is possible to output a linkage instruction request from an application. When the Enterprise Application Pipeline receives the linkage instruction request, it starts a sequence of data processing according to the content of the request.

This API is suited to operations requiring batched processing of multiple data at specified times, such as on a daily or monthly basis. To process multiple leading tasks or following tasks, specify that processing in the EAP definition.

[Linkage Instruction Request from an Application]

6.1.3.2 Using the Real-time Linkage Function

If an automatic execution application for real-time linkage provided by the Enterprise Application Pipeline is set in an automatic execution activity, the Enterprise Application Pipeline will start data processing immediately.

This function is suited to data that requires immediate processing. It can only handle a single leading task, but it can link the leading task to multiple following tasks. To process multiple following tasks, specify the logical node group definition in the application linkage definitions.

The diagram below provides an overview of real-time linkage processing.
6.1.3.3 Normal Termination of Message Collection when Message Collection Monitoring Function is used

The Enterprise Application Pipeline monitors file transfers (collection) started by a remote station, and starts data processing when a transfer has terminated normally. This function is also suited to data that requires immediate processing. It can only handle a single leading task, but it can link the leading task to multiple following tasks. To process multiple following tasks, specify the logical node group definition in the application linkage definitions. In addition, the files whose collection is to be monitored are also specified in the logical node definition in the application linkage definition.

The diagram below provides an overview of message collection monitoring.

6.1.3.4 Relationship between Leading/Following tasks and the Time Processing is Initialized

Depending on the type of leading task, the time at which processing can commence will vary. The user must consider the characteristics of each item of data and its use when selecting the time processing is to commence. The following table shows the relationships between the leading and following tasks and the time processing starts.

<table>
<thead>
<tr>
<th>Leading task</th>
<th>Time Processing Commences</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>An operation in a CollaborationRing PM process that waits for user processing.</td>
<td>Upon linkage instruction from application</td>
<td>Starts data processing of the relevant process in response to a linkage instruction</td>
</tr>
<tr>
<td>An operation that uses an automatic execution application for real-time linkage with a CollaborationRing PM process.</td>
<td>Upon execution of an automatic execution application for real time linkage</td>
<td>Data processing starts as soon as the automatic execution application for real time linkage has finished processing</td>
</tr>
<tr>
<td>An operation that collects messages by file transfer</td>
<td>When the message collection monitoring function is used and message collection terminates normally.</td>
<td>Data processing starts when message collection by file transfer terminates normally.</td>
</tr>
</tbody>
</table>

[Real-time Linkage Processing]

[Message Collection Monitoring]

[Relationship between a Leading task and the Time Processing Commences]
Leading task | Time Processing Commences | Remarks
--- | --- | ---
 | Upon linkage instruction from application | Start processing from the local station message collection by linkage direction. When using the built-in FTP file transfer function, the application continues on to perform local station initiated message distribution after data processing terminates. When using the built-in FTP file transfer function, processing starts from local station initiated message collection.

[Relationship Between a Following task and the Time Processing Commences]

<table>
<thead>
<tr>
<th>Following task</th>
<th>Time Processing Commences</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>An operation consisting of CollaborationRing PM processes</td>
<td>Same time as leading task</td>
<td>When data processing of the leading task terminates, a form is generated for the process of the following task.</td>
</tr>
<tr>
<td>An operation that distributes messages by file transfer</td>
<td>Same time as leading task</td>
<td>Operation will be continued to local station startup message distribution after the completion of data processing.</td>
</tr>
</tbody>
</table>

6.1.4 Tracking

The following diagram provides an overview of tracking. Tracking records which following task inherits data input from a leading task. Tracking with the Enterprise Application Pipeline is accomplished using the global history query function of the Process Flow Manager.

[Tracking using the Global History Query Function]

Tracking can take the form of a service log, received data querying and transmitted data querying. The service log is used to view the history of processes executed by the Enterprise Application Pipeline. Received data querying is used to view the state of a linked following task from a leading task. Transmitted data querying is the reverse operation, and is used to view the status of a linking leading task from a following task. Use these functions as appropriate.

Note that in order to perform global history querying, it is necessary to construct environments for the CollaborationRing PM history send agent and Collection Server. Refer to the Interstage CollaborationRing PM Operation Guide for information on constructing these environments.

6.1.5 High-reliability Systems

The High Availability function (abbreviated "HA" in this manual) enables two server machines with Enterprise Application Pipelines to be operated as a single virtual server machine. This makes it possible to construct uninterruptible systems possessing high scalability.

In cluster systems, the server machine providing an application service during normal operation is called the active node, and the server machine that provides an application service when an error occurs in the active node is called the standby node.

The following diagram shows the hot standby configuration supported by the HA function.
[HA Standby Configuration]

+ Supported systems
For details on supported cluster systems, refer to the Installation Guide.

+ Note on operation
To use the HA function, it is necessary to register Interstage Application Server, Database, Enterprise Application Pipelines, applications, and other components as resources in the cluster system, and also to collect these resources together into an application group.

6.1.6 EAP Management Tool
This tool is provided for the management of EAP functions. Using it allows you to view and delete EAP services from a Web browser. Refer to the online help of the tool or the Interstage CollaborationRing PM Operation Guide for more details.

Note that, if you use this tool, you must also construct an appropriate environment for Interstage Application Server Servlet service and so on. Refer to the Interstage Application Server J2EE User's Guide for details.

6.2 Overview of EAP Definitions
EAP definitions are required in order to use EAP. They define the linkage method between the tasks and the form formats and EAPs handled in those tasks for leading and following tasks linked by EAP. They also define the required translation information and message distribution information for leading and following task combinations.

6.2.1 Physical Nodes and Logical Nodes
There are various task modes and types. The tasks of receiving orders from customers, inventory management and so on, as well as the work of each department (production, management, and planning) within a company are all individual tasks. Each of these tasks may be completed in a server, or some tasks may extend across several servers. In the EAP definitions, these tasks are assigned the following definitions

- Physical node: Refers to each server connected by the network.
- Logical node: Refers to a task in a physical node or a task that extends across physical nodes.

The EAP provides inter-task linkage for leading tasks or following tasks in logical node units.

In EAP definitions, the logical node corresponding to a leading task is called an "input logical node", and the logical node corresponding to a following task is called an "output logical node".

By combining the output forms and types for the input logical nodes and output logical nodes, the EAP performs the translation, merging, and splitting of the input logical node forms and passes the forms to the output logical nodes according to "rules" (discussed later in this chapter).
6.2.2 Logical Node Group

Depending on the task mode, there may be multiple leading tasks for a following task, or there may be multiple following tasks for a leading task. Using the EAP, it is possible to bulk the forms for multiple leading tasks and link them to a following task, or sort the leading task output forms and link them to multiple following tasks. In EAP definitions, multiple predecessor process units are called an "input logical node group", and multiple following process units are called an "output logical node group".

When input logical nodes are grouped (input logical node group), the EAP processes all the input logical node forms in the group together, and passes the forms to the output logical nodes. When output logical nodes are grouped (output logical node group), the EAP distributes the forms to the output logical nodes in the group, according to "rules" to be discussed later.
Leading task
(Input logical node group)

EAP
Format conversion
Merging/splitting

Following task
(Output logical node)

Form
Form
Form
Form
Form

[Input Logical Node Group]
6.2.3 Actions and Rules

The combinations of input logical nodes (or input logical node groups) and output logical nodes (or output logical node groups) that are linked by the EAP are called "actions".

The translation, merging, splitting, and sorting processing content and the processing order required for the combinations of input logical nodes (or input logical node groups) and output logical nodes (or output logical node groups) are called "rules".

The EAP passes the forms received from the input logical nodes (or input logical node groups) to the output logical nodes (or output logical node groups), according to rules. The detail information (parameters) required for the translation and data sorting performed according to these rules is called "action detail information".

The following figure shows the relationship between physical nodes, input-output logical nodes, input/output logical node groups, actions, and rules.

[Relationship between Logical Nodes, Actions, and Rules]

6.2.4 Rules

The rule contents are determined by the presence or absence of the logical node grouping selected by the action definition, the format of the forms to be handled by the input-output logical nodes, and the output form of the output logical nodes.

Once the combinations of input logical nodes (or input logical node groups) and output logical nodes (or output logical node groups) in the action definition have been specified by the user, the EAP Definition Tool selects the applicable
rules and displays them to the user. The user then selects the most appropriate rule from the list. The following figure shows the rule contents selected by the EAP definition.

As shown in the figure below, errors may occur as a result of the combinations. These errors are notified when the erroneous combinations of input logical nodes (or input logical node groups) and output logical nodes (or output logical node groups) are selected in the action definition.

---

**[Rules]**

**6.2.5 Rule Contents**

To link leading tasks and following tasks, users select the rule that best describes the link they wish to make, distinguished by the rule identifier. See below.

<table>
<thead>
<tr>
<th>Rule</th>
<th>Link Pattern (Rule Identifier)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rule 1</td>
<td>EAP_RULE1a</td>
</tr>
<tr>
<td></td>
<td>EAP_RULE1b</td>
</tr>
<tr>
<td></td>
<td>EAP_RULE1c</td>
</tr>
<tr>
<td>Rule 2</td>
<td>EAP_RULE2a</td>
</tr>
<tr>
<td></td>
<td>EAP_RULE2b</td>
</tr>
<tr>
<td></td>
<td>EAP_RULE2c</td>
</tr>
<tr>
<td>Rule 3</td>
<td>EAP_RULE3a</td>
</tr>
<tr>
<td>Rule 4</td>
<td>EAP_RULE4a</td>
</tr>
<tr>
<td></td>
<td>EAP_RULE4b</td>
</tr>
</tbody>
</table>
1. Rule 1

a) Rule Identifier: EAP_RULE1a

(1) Merging of forms in input logical node units (bulk form creation)
(2) Translation to output logical node output form
(3) Splitting of forms (single form creation)

Example: Input logical nodes 1 and 2: User-specific format (fixed format)/single form
Output logical node 1: User-specific format (variable-length format)/single form

b) Rule Identifier: EAP_RULE1b

(1) Splitting of forms in input logical node units (single form creation)
(2) Translation to output logical node output form
(3) Splitting of forms (single form creation)

Example: Input logical nodes 1 and 2: User-specific format (variable-length format)/bulk form, single form
Output logical node 1: User-specific format (fixed format)/single form

[EAP_RULE1b]

++ c) Rule Identifier: EAP_RULE1c

(1) Splitting of forms in input logical node units (single form creation)
(2) Translation to output logical node output form

Example: Input logical nodes 1 and 2: User-specific format (fixed format)/bulk form, single form
Output logical node 1: User-specific format (variable-length format)/single form
2. Rule 2

++ a) Rule Identifier: EAP_RULE2a

(1) Merging of forms in input logical node units (bulk form creation)
(2) Translation to output logical node output form
(3) Merging of forms (bulk form creation)

Example: Input logical nodes 1 and 2: User-specific format (fixed format)/single form
Output logical node 2: User-specific format (variable-length format)/bulk form
[EAP_RULE2a]
++ b) Rule Identifier: EAP_RULE2b

(1) Splitting of forms in input logical node units (single form creation)
(2) Translation to output logical node output form
(3) Merging of forms (bulk form creation)

Example: Input logical nodes 1 and 2: User-specific format (fixed format)/bulk form, single form
Output logical node 1: User-specific format (variable-length format)/bulk form
++ b) Rule Identifier: EAP_RULE2c

(1) Translation to output logical node output form

Example:
Linkage for format conversion only
Input logical node: User-specific format (fixed format)/bulk form, single form
Output logical node: User-specific format (form format)/single form
+ 3. Rule 3

++ a) Rule Identifier: EAP_RULE3a

(1) Merging of forms in input logical node units (bulk form creation)
(2) Translation to output logical node output form

Example: Input logical node 1: User-specific format (fixed format)/bulk form, single form
Input logical node 2: User-specific format (variable-length format)/bulk form, single form
Output logical node 1: EDIFACT format/single form
+ 4. Rule 4

++ a) Rule Identifier: EAP_RULE4a

(1) Sorting of forms in input logical node units
(2) Translation to output logical node output form
(3) Splitting of forms in output logical node units (single form creation)

Example: Input logical nodes 1 and 2: variable format/bulk form, single form
Output logical node 1: User-specific format (fixed format)/single form
Output logical node 2: User-specific format (variable-length format)/single form
++ b) Rule Identifier: EAP_RULE4b

(1) Sorting of forms in input logical node units
(2) Translation to output logical node output form

Example: Input logical nodes 1 and 2: User-specific format (fixed format)/single form, bulk form
Output logical nodes 1 and 2: User-specific format (variable-length format)/single form
5. Rule 5

a) Rule Identifier: EAP_RULE5a

1. Sorting of forms in input logical node units
2. Translation to output logical node output form
3. Merging of forms in output logical node units (bulk form creation)

Example: Input logical nodes 1 and 2: User-specific format variable-length format/bulk form, single form
Output logical node 1: User-specific format (fixed format)/bulk form
Output logical node 2: User-specific format (variable-length format)/bulk form
[EAP_RULE5a]

++ b) Rule Identifier: EAP_RULE5b

(1) Sorting of forms in input logical node units
(2) Merging of forms in output logical node units (bulk form creation)
(3) Translation to output logical node output form

Example: Input logical nodes 1 and 2: User-specific format (fixed format)/bulk form, single form
Output logical nodes 1 and 2: User-specific format (variable-length format)/bulk form
[EAP_RULE5b]

+ 6. Rule 6

++ a) Rule Identifier: EAP_RULE6a

(1) Translation to output logical node output form
(2) Splitting of forms in output logical node units (single form creation)
Example: Input logical nodes 1 and 2: EDIFACT format single forms
Output logical node 1: User-specific format (fixed format)/single form
Output logical node 2: User-specific format (variable-length format)/single form
7. Rule 7

++ a) Rule Identifier: EAP_RULE7a

(1) Translation to output logical node output form
(2) Merging of forms in output logical node units (bulk form creation)

Example: Input logical nodes 1 and 2: EDIFACT format single forms
Output logical node 1: User-specific format (fixed format)/bulk form
Output logical node 2: User-specific format (variable-length format)/bulk form
[EAP_RULE7a]

6.2.6 Conversion Content

The conversion content is determined by the format combinations to be handled by the input logical nodes and output logical nodes in each rule.

The figure below shows the format combinations and the translation content.
6.3 Definitions

It is necessary to define the working environment and other items before using the Enterprise Application Pipeline. Some definitions are mandatory and some are optional depending on the functions used. Related definitions are listed in the following table.

<table>
<thead>
<tr>
<th>Definition name</th>
<th>Overview</th>
<th>Mandatory/Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td>EAP Working environment definition</td>
<td>Defines the environment in which the Enterprise Application Pipeline runs. There are two files, EAPUSER1.CONF and EAPUSER2.CONF, and it is necessary to customize these files directly to suit the particular operation before the Enterprise Application Pipeline are started. Note that the control server URL and the login user ID and password are registered as EAP access definition information.</td>
<td>Mandatory</td>
</tr>
</tbody>
</table>
| EAP definitions                        | Defines the data format of the leading and following tasks, and the destination of a data dispatch from a leading task to a following task. These definitions can be manipulated from a GUI window on a terminal connected to the server on which the Enterprise Application Pipeline runs. The following three EAP definitions are available:  
  - Logical node definition  
  - Node group definition | Mandatory          |
<table>
<thead>
<tr>
<th>Definition name</th>
<th>Overview</th>
<th>Mandatory/Optional</th>
</tr>
</thead>
<tbody>
<tr>
<td>· Action definition</td>
<td></td>
<td></td>
</tr>
<tr>
<td>FEDIT definitions</td>
<td>Defines the I/O formats (items, attributes, number of characters) of data to be converted when translating data received from a leading task. This definition is manipulated from a GUI window on a personal computer. A conversion table created at the GUI is transferred to the server where the Enterprise Application Pipeline runs. Refer to the Interstage CollaborationRing PM FEDIT Reference Manual for details.</td>
<td>Optional</td>
</tr>
<tr>
<td>Working environment definition of built-in FTP file transfer function</td>
<td>Defines the environment in which the FTP file transfer function built in EAP runs. The parameters relevant to the FTP server/client and so on are set.</td>
<td>Optional</td>
</tr>
<tr>
<td>Working environment definition of plaintext mail linkage function</td>
<td>Defines the environment in which the plaintext mail linkage function built in EAP runs. The parameters relevant to sending and receiving mail and so on are set.</td>
<td>Optional</td>
</tr>
</tbody>
</table>
Chapter 7 Using Adapter Kit to Support Constructing the Adapter

This chapter explains the procedure to use Adapter Kit to create programs. The types of Adapter Kit are shown below:

- Adapter Kit (for general applications)
- Adapter Kit (for component objects)

7.1 Adapter Kit (for general applications)

Adapter Kit (for general applications) is a kit for constructing adapters to integrate existing systems easily.

First, use the Wizard to select the program information you want to create. You can generate a program skeleton according to your needs. Import business logic to the skeleton to complete the adapter. Adapter Kit (for general applications) supports various functions, in particular the automatically executed applications from the creation of a program to the start of the program.

7.1.1 Generating skeletons using the Wizard

A skeleton is a program framework. The parts that can be used in any common program are already coded. In the skeleton offered by Adapter Kit, the CollaborationRing PM API is already entered. This means that the developer can simply code the integration part with the existing system to construct the adapter.

The type of skeleton you should use depends on your purpose. By responding to a number of questions asked by the Wizard, you can generate a skeleton using the Adapter Kit Wizard to suit your needs. You can integrate the Wizard with the process definition tool to get required information from the process definition information. This enables you to reduce the amount of development coding.

7.1.1.1 Generated skeleton types

This section explains the skeleton types generated by Adapter Kit (for general applications).

- Target languages:
  - C
  - Java

Note that the language for adapters for EAP is C only.

- Supported application types:
  - Applications that execute process forms.
  - Applications that are executed automatically.
  - Applications that integrate with UI activities (waiting for processing).

You can also create EAP adapters for applications to add processes to forms and for automatic execution applications.

There are three types of EAP adapter:

- Message collection monitoring type adapter (application for adding processes to forms)
- Message collection type adapter (automatic execution application)
- Message distribution type adapter (automatic execution application)

Note: Not all CollaborationRing PM APIs are included in the skeleton. At the very least, APIs that are required for the application type are included. You can add any APIs you want to use to the skeleton.

+++ Automatically executed application skeleton

If you select "Automatic Execution Application" in the Wizard, the following program skeletons are generated:

- Business logic programs
  Programs that describe business logic.
- Recovery programs
  Programs that are called automatically to carry out recovery processes when a business logic program ends abnormally. For details on recovery processes, refer to "7.1.2 Automatically executed application structures that support Adapter Kit (for general applications)" Recovery processing below.
<table>
<thead>
<tr>
<th>Language</th>
<th>Type</th>
<th>Source name</th>
<th>Initialization processing</th>
<th>Stop processing</th>
<th>Main processing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Java</td>
<td>Business logic program</td>
<td>application name.java</td>
<td>Enter logic in the application name class, Init() method.</td>
<td>Enter logic in the application name class, Term() method.</td>
<td>Enter logic in the application name class, Execute() method.</td>
</tr>
<tr>
<td></td>
<td>Recovery program</td>
<td>application name_recovery.java</td>
<td>Enter logic in the application name_recovery class, Init() method.</td>
<td>Enter logic in the application name_recovery class, Term() method.</td>
<td>Enter logic in the application name_recovery class, Execute() method.</td>
</tr>
<tr>
<td>C</td>
<td>Business logic program</td>
<td>application name.c</td>
<td>Enter logic in the CRSDK_Init() function.</td>
<td>Enter logic in the CRSDK_Term() function.</td>
<td>Enter logic in the CRSDK_Execute() function.</td>
</tr>
<tr>
<td></td>
<td>Recovery program</td>
<td>application name_recovery.c</td>
<td>Enter logic in the CRSDK_Init() function.</td>
<td>Enter logic in the CRSDK_Term() function.</td>
<td>Enter logic in the CRSDK_Execute() function.</td>
</tr>
</tbody>
</table>

For details about what data you should enter for business logic program main processing, refer to the sample explanations (tutorials) that are installed with the Wizard.

### 7.1.1.2 Integration with the Process Definition Tool

You can get information input in the properties of an activity or process using the programs that you create when you link them with created activities using the Process Definition Tool. Using the Adapter Kit Wizard, select the process definition file name, process, or activity to get information such as the activity type, application name, and input output variable name.

Use the variable name you got to avoid the needless trouble of inputting input and output information and input errors when you create the program.

Note that there is no need to integrate with a process definition when using an EAP adapter.

[Integration with the Process Definition Tool]

### 7.1.1.3 Integration with EAP Definitions

When you create an EAP adapter, you integrate with an EAP definition. You can select the adapter name input in the Wizard when you select the EAP definition input/output type. Also, if, in the Wizard, you set as properties the variables that you want to add as data to pass to a program, those property names are displayed in the Wizard and you can enter values for them. This means that values are automatically set for properties when a program is executed.

By using property names set in the Wizard, you can reduce the task of entering input/output information and avoid the problem of input errors when a program is executed.
7.1.2 Automatically executed application structures that support Adapter Kit (for general applications)

Automatically executed application structures that support Adapter Kit (for general applications) contain the features shown below:

+++ CORBA programming concealment

CORBA has many advantages, such as allowing you to execute other server programs. To realize this function, you must learn the unique CORBA programming technique. A person who wants to simply create an adapter might find it difficult to learn a programming technique from scratch.

The CORBA program part is not described in the skeleton generated by Adapter Kit. The developer does not need to add it because the CORBA program part is contained in the Adapter Kit connector. For this reason, the developer can concentrate on business logic and programs.

+++ Recovery processing

As well as creating programs that describe business logic, you can also create recovery programs. If the business logic program exits because of an abnormality, a recovery program is called and recovery processing starts. Recognizing that a business logic program has abnormally exited and the calling of the recovery program are executed by the Adapter Kit connector.

A skeleton is also generated is for the recovery program. The developer adds the required recovery processing to the skeleton.

+++ Log output

Adapter Kit connector outputs a trace log as a debugging log. You can also output the log from a business logic program or a recovery program.

The contents that are output are shown below:
When the Adapter Kit connector is started.
When the Adapter Kit connector is stopped.
When the initialization processing is called.
When main processing is called.
When main processing is returned.
When stop processing is called.
When an abnormality occurs.
When a recovery program is called.
When a recovery program is returned.
When user settings are made.

The user settings are output when the user uses an API.
The trace log is output in a file in automatically executed application units.


+++ Scripts for starting and stopping

Script is offered for starting and stopping the automatically executed application you have created. The script is generated using the Wizard at the same time as the skeleton. Script is also generated in automatically executed application units.

7.1.3 Operating environment of the created application

Use the Adapter Kit to create the adapter in the operating environments as shown in the following table:

<table>
<thead>
<tr>
<th>Application type</th>
<th>Operating environment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Applications that execute process forms.</td>
<td>Environment in which the following software is installed:</td>
</tr>
<tr>
<td></td>
<td>· CRF API Library</td>
</tr>
<tr>
<td></td>
<td>· Java 2 Runtime Environment (*1)</td>
</tr>
<tr>
<td></td>
<td>· EAP (*2)</td>
</tr>
<tr>
<td></td>
<td>Other software may also be necessary - install as required.</td>
</tr>
<tr>
<td>Applications that integrate with UI activity (waiting for processing).</td>
<td>Environment in which the following software is installed:</td>
</tr>
<tr>
<td></td>
<td>· CRF API Library</td>
</tr>
<tr>
<td></td>
<td>· Java 2 Runtime Environment (*1)</td>
</tr>
<tr>
<td></td>
<td>Other software may also be necessary - install as required.</td>
</tr>
<tr>
<td>Applications that are executed automatically.</td>
<td>Environment in which the following software is installed:</td>
</tr>
<tr>
<td></td>
<td>· CRF Server</td>
</tr>
<tr>
<td></td>
<td>· CRF API Library</td>
</tr>
<tr>
<td></td>
<td>· CRF Adapter Kit (for General Applications)</td>
</tr>
<tr>
<td></td>
<td>· Server Functions</td>
</tr>
<tr>
<td></td>
<td>· Java 2 Runtime Environment (*1)</td>
</tr>
<tr>
<td></td>
<td>· EAP (*2)</td>
</tr>
<tr>
<td></td>
<td>Other software may also be necessary - install as required.</td>
</tr>
</tbody>
</table>

(*1) Required for Java programs. (Refer to the Installation Guide for details on appropriate versions.)

(*2) Required for the EAP adapter.

7.1.4 Sample Programs

Adapter Kit (for general applications) offers sample programs. The sample programs are based on a skeleton that is generated using a Wizard and encoded.

The sample explanation (tutorial) that is installed together with the Wizard contains the items shown below:

· Wizard input contents:
  How to input and create a sample program using the Wizard.
· Integration with the process definition tool:
  How to input the process definition tool activities and properties.
7.2 Adapter Kit (for Component Objects)

The CollaborationRing PM Adapter Kit for Component Objects is provided to assist solution designers and developers to integrate applications with CollaborationRing PM. This Kit provides a framework that makes it easier to:

- Work with CollaborationRing PM Activity Documents
- Create and deploy code that interfaces with a range of application environments, including packages and custom-built applications of SOAP, EJB (Enterprise Java Beans), and JCA (J2EE Connector Architecture).

In this document, the term “adapter” is used to mean the runtime modules or adapters that are created and set in the Adapter Kit for Component Objects Wizard.

In this document, the term “Activity Document” is used to mean the XML documents that are exchanged between CR control servers and the adapters in each direction. In practice, they are the contents of the parameters passed within the Server-type API named as “activityDataIn” and “activityDataOut”.

7.2.1 Working with Activity Documents

The requirements for a process flow definition may require the manipulation of data in an Activity Document, in order to test a variable value, or send data to an application. The developer can use the CR adapter to execute conversions according to XSLT stylesheets to perform this data manipulation.

For example, the XML document sent to CollaborationRing PM control server via EAP will be stored in an Activity Document as a variable called eap_UserData. To extract one or more fields from this XML document, the developer can create an XSLT stylesheet to process the XML document or to filter it into a smaller document containing just the fields required, and to format the resulting document to meet the requirements of and pass the results to CollaborationRing PM control server.

Normally, in a server-type application, the adapter that represents the corresponding automatic execution application is called from an activity in a process definition. The developer creates a behavior in the Adapter Kit for Component Objects Wizard that will run the XSLT stylesheet when this adapter is executed. This behavior consists of a set of parameters that define the behavior of the adapter and contains information such as XSLT stylesheets, plug-in specifications, and input/output specifications. Note that, when you define a behavior, the information is saved in the adapter kit repository and the XSLT stylesheet is copied there.

7.2.2 Adapter Processing (Linking with Applications)

The developer can create an adapter that interfaces between a CollaborationRing PM process definition and an application running anywhere in the network. The CollaborationRing PM processing model involves passing an Activity Document (formatted based on an activity's definitions) to an adapter, which processes it as per the requirements for accessing another application. The Adapter Kit for Component Objects enhances this model, by providing a framework for:

- Pre-processing and post-processing of the Activity Document
Calling a plug-in module that understands how to interface with its target application, given an XML document (formatted as required) as input, and providing an XML document (in its own format) as output.

The Adapter Kit for Component Objects allows the developer to avoid the overhead of developing this adapter as a CORBA server. Instead, the developer need only put the required logic in a Java module, called a pluggable module, and create a CR adapter behavior definition that will run that module.

Also, recognizing that modern applications are being developed for SOAP and EJB environments, the Adapter Kit for Component Objects provides coded and tested pluggable module combinations that can communicate with a SOAP service and with an EJB stateless session bean. The developer need only write code for these environments, using and producing XML documents as described above. In addition, sample code is provided to show how to use JCA (J2EE Connector Architecture).

[Adapter Processing]

7.2.3 Adapter Kit (for component objects) Functions

The CollaborationRing PM Adapter Kit for Component Objects consists of a runtime, a repository and several wizards and tools as shown below. For an overview of CollaborationRing PM adapters, refer to the Interstage CollaborationRing PM Guide.

[Adapter Kit Functions]

The central component of the Adapter Kit for Component Objects is the Adapter Repository, which acts as a store for CR adapter behavior and association definitions. The Development Wizards use the repository to store these definitions. The Deployment Tools extract these definitions from the repository and the CR adapter on stand-alone computer or with the repository server deploys execution files in the server.

The Component Wizard is used to define the items that will be later collected into a behavior. Initially, base items
including XSLT stylesheet files and pluggable module files are registered into the repository using this wizard. Then a behavior can be created by combining specific XSLT stylesheets and/or pluggable modules, together with some additional attributes needed by the CR adapter executable.

The Association Wizard is used to associate a behavior with an activity in a process definition. To make this association, the activity must already have been defined in CollaborationRing PM control server; the Title of each activity must be unique within a process definition. The wizard ties the behavior's inputs and outputs to the input and output Activity Document variables defined for the activity.

Behaviors and associations are assigned to one or more groups, based on how the administrator wants to deploy the CR adapters that handle these behaviors and associations. These groups are called "behavior groups". The Assemble Tool is used to select which behaviors and associations will be in a group.

A behavior must be deployed into an execution environment before it can be used by an activity. The execution environment is on a server running Interstage Application Server. Deployment decisions would be based on criteria such as frequency of usage, or location of application. For more details about the processing flow of a CR adapter, refer to 7.2.4 CR Adapter Structure below.

The Assemble Tool generates a JAR file for a set of behavior groups, containing all the server code and definition information relating to the behaviors and associations in those groups. This JAR file can then be copied, using standard copy mechanisms like FTP or file copy, to the target runtime server.

To create the execution environment, the administrator uses the Deployment Tool on the JAR file created by the Assemble Tool. Each JAR file is a set of self-contained associations and behavior groups, as it includes the CR adapter executables. The Deployment Tool is used to:

Update the repository with behavior and association information from the JAR;

Generate the execution environment for the CR adapter (The execution environment can be a standalone environment that does not require a repository, or a non-standalone environment that uses a repository.).

For the execution environment, the Deployment Tool creates a directory structure for the CR adapter, populates it with the files and JARs it needs, and generates several script (or .bat) files that will register the server in the Interstage Application Server naming service, and start the CR adapter. Note that the deployed CR adapter requires Fujitsu's XSLT processor or another vendor's XSLT processor.

In order to use a behavior, the automatic execution activity of a process definition must be configured to call the deployed CR adapter. Either the behavior is associated with a specific automatic execution activity in the process definition (case 1), or the automatic execution activity is configured to call that specific behavior in the group (case 2).

1. Specify the name (set with the Assemble Tool) of the group that contains the behavior in the automatic execution activity's Application Name attribute (for example "MyDeployedGroup"), and use the Association Wizard to associate this activity with a behavior. The wizard ties the behavior's inputs and outputs to the input and output variables of the Activity Document defined for this automatic execution activity.

2. Specify the name (set with the Assemble Tool) of the group that contains the behavior, followed by the behavior name, in the automatic execution activity's Application Name attribute (for example "MyDeployedGroup::MyBehavior"). If an input or output variable is needed for the behavior, then use crakc_Input_Variable as the input variable and crakc_Output_Variable as the output variable. If the variable names are different from these, use the binding function of the Process Definition Tool to map the variable names appropriately.

To use an association and its behavior, an activity in a process definition merely nominates in its Application Name attribute the group that does, or will, contain the behavior. This group name is also the CORBA name defined for the CR adapter. Refer to Chapter 2 Process Rule Design for more details about how to define a business process using the Process Definition Tool.

The diagram below shows how the Adapter Kit for Component Objects functions is used to create and deploy a CR adapter in a system using two computers.
7.2.4 CR Adapter Structure

The CR adapter is a CollaborationRing PM automatic execution application. The server instance of each adapter handles a specific behavior group. Each behavior results in the processing of CollaborationRing PM Activity Documents (XML) using a configurable combination of one to two XSLT stylesheets and a pluggable module. How the Activity Document is processed depends on the input and output types defined for a particular behavior.

The various ways that the components defined in a behavior can be structured allow for different types of processing, as summarized below:

Pack part or all of an Activity Document into a single Activity Document variable. This is useful for when an Activity Document must be sent to an Enterprise Application Pipeline (EAP) function.

Unpack an Activity Document variable, and expand it into an XML document that updates one or more other variables in the Document. This is useful when a process definition receives an XML document from EAP, and needs to extract one or more tags into Activity Document variables in order to test their values.

Use pluggable modules to implement small units of business processing based on the contents of an Activity Document. This is useful if the process definition has many steps that combine calls to back-end applications and fragments of processing that prefix or postfix such calls. If each fragment is implemented as a pluggable module, then separate behaviors can be defined, and grouped together into one CR adapter instance. This eliminates the need to deploy many CORBA servers.

If there is a need to compute values, filter, or transform an Activity Document, XSLT stylesheets can be used. This is useful in preparing input data to a pluggable module, processing output data from a pluggable module, or initializing Activity Document variables.

A pluggable module can be used to access a back-end application that is running on a technology that is not native to CollaborationRing PM. The combination of XSLT stylesheets, and a pluggable module is a framework for an application adapter.

The flow of processing in the CR adapter is shown in the diagram below.
7.2.4.1 Input and output types

The input and output types defined for a CR adapter behavior control how data are transferred between a process definition activity and the CollaborationRing PM control server. The valid types are shown in the table below.

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Activity Document</td>
<td>A CollaborationRing PM Activity input/output document that contains such items as server-type API-specific headers, control information, and business data variables. This type is used to map whole Activity Documents, not process definition variables.</td>
</tr>
<tr>
<td>XML Fragment</td>
<td>An Activity Document variable that extracts (as a fragment) part of an XML document, maps variables defined in the activity, and sends it on. When this fragment is presented to the CR adapter, the fragment will have an XML header prefixed to it. This type can only map activity variables - it cannot map Activity Documents.</td>
</tr>
<tr>
<td>Binary</td>
<td>An Activity Document variable that contains data that have been encoded in Base-64 format. For example, eap_UserData. This type can only map process definition activity variables - it cannot map Activity Documents.</td>
</tr>
<tr>
<td>None</td>
<td>For Output type only, signifies that no data is expected to be returned to CollaborationRing PM as a result of the CR adapter's processing. For example, the pluggable module sends data to a back-end application, and expects no return data. The fact that processing finishes and does not return an error status to CollaborationRing PM is enough for the process definition's needs. This type cannot map activity variables or Activity Documents.</td>
</tr>
</tbody>
</table>

7.2.4.2 Activity document format

When "Activity Document" is specified as an Input type for a CR adapter behavior, the XML document passed to an XSLT stylesheet should be the style indicated on "Input information" of "C.1 Automatic Execution Activity" in the Interstage CollaborationRing PM Reference Manual (Process Flow Manager Edition).

The basic points to take note of about the input Activity Document XML are:

- The root element has the name "activityDataIn".
- The input variables are contained in corresponding elements within the "contextData" element. This will usually take the form of an XML fragment built up from the input variables defined for the properties in a process definition activity.
- There are various control elements, containing information about the activity and the process.

When "Activity Document" is specified as an Output type for a CR adapter behavior, the XML document must be in the style indicated in "output" of "C.1 Automatic Execution Activity" in the Interstage CollaborationRing PM Reference Manual (Process Flow Manager Edition).

The basic points to take note of about the output XML document are:

- The root element has the name "activityDataOut".
- The modified Activity Document variables are contained in corresponding elements within the "resultData" element. To be useful, this must take the form of an XML fragment that has, as top level elements, tag names that are the same as the variable names defined for the output variables in the process definition activity.
- There are various control information elements that may also be updated. Refer to Chapter 2 Process Rule Design in Chapter 2 or the Interstage CollaborationRing PM Reference Manual (Process Flow Manager Edition) for more
information about which control elements can be updated.

7.2.4.3 Processing combinations

The valid and invalid combinations of input type and output type are listed below. An optional component can be an XSLT or pluggable module component. The simplest processing combination for CR adapter is when there are no XSLT stylesheets and no pluggable module.

<table>
<thead>
<tr>
<th>Input Type</th>
<th>Optional Component</th>
<th>Output Type</th>
<th>Validity</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>(Any)</td>
<td>(Any)</td>
<td>Invalid</td>
<td>It is not valid to have “none” as the input type</td>
</tr>
<tr>
<td>(Any)</td>
<td></td>
<td>None</td>
<td>Valid</td>
<td>It is valid to have “none” as the output type</td>
</tr>
<tr>
<td>(Any)</td>
<td>At least one optional component included</td>
<td>(Any)</td>
<td>Valid</td>
<td>So long as there is at least one component included, and that component produces the expected output type, then all combinations of input and output type are valid</td>
</tr>
</tbody>
</table>

**Activity document**

<table>
<thead>
<tr>
<th>Input Type</th>
<th>Optional Component</th>
<th>Output Type</th>
<th>Validity</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td>Activity Document</td>
<td>Invalid</td>
<td>This is an invalid combination since the Activity Input Document and Activity Output Document formats are different.</td>
</tr>
<tr>
<td>XML fragment</td>
<td></td>
<td>Invalid</td>
<td>This is not an allowed combination, since there is little benefit in copying an entire Activity document into a single variable.</td>
<td></td>
</tr>
<tr>
<td>Binary</td>
<td></td>
<td>Valid</td>
<td>The entire input Activity Document is packed up as a Base-64 encoded string, and copied to a single output variable. This is useful if the document needs to be sent to EAP for routing to a remote destination.</td>
<td></td>
</tr>
</tbody>
</table>

**XML fragment**

<table>
<thead>
<tr>
<th>Input Type</th>
<th>Optional Component</th>
<th>Output Type</th>
<th>Validity</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td>Activity document</td>
<td>Invalid</td>
<td>This is not an allowed combination, although it could be valid in rare circumstances.</td>
</tr>
<tr>
<td>XML fragment</td>
<td></td>
<td>Valid</td>
<td>The variable is copied.</td>
<td></td>
</tr>
<tr>
<td>Binary</td>
<td></td>
<td>Valid</td>
<td>The variable is copied and encoded in Base-64 format.</td>
<td></td>
</tr>
</tbody>
</table>

**Binary**

<table>
<thead>
<tr>
<th>Input Type</th>
<th>Optional Component</th>
<th>Output Type</th>
<th>Validity</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td></td>
<td>Activity document</td>
<td>Invalid</td>
<td>This is not an allowed combination, although it could be valid in rare circumstances.</td>
</tr>
<tr>
<td>XML fragment</td>
<td></td>
<td>Valid</td>
<td>Extracts a Base-64 encoded variable and copies it to the output variable.</td>
<td></td>
</tr>
<tr>
<td>Binary</td>
<td></td>
<td>Valid</td>
<td>Copies the variable from one binary variable to another.</td>
<td></td>
</tr>
</tbody>
</table>

7.2.4.4 Using XSLT

Richer behavior can be achieved in the CR adapter by using XSLT stylesheets. Generally, the use of XSLT is to extract and/or transform an XML document into another XML document. There are many ways an XSLT stylesheet can be used in the CR adapter:

1) To convert an input Activity Document into an XML activity document that can be recognized by the control server. This is especially useful for extracting specific tags from XML fragments embedded in the Activity Document, and updating their values to output variables that have the same name as the tag.

For example, the Activity Document contains an XML fragment that has a `<MyField>` tag, and the process definition needed to test the value of this tag. The solution is to create an XSLT stylesheet that extracts the tag, and formats it in the Output type Activity Document shown in “Activity Document format” above.

To carry out this processing, the behavior parameter set can specify an XSLT stylesheet on the Input or Output side (Refer to the diagram in 7.2.4 CR Adapter Structure above).

2) To convert an input Activity Document into an XML document that can be recognized by a pluggable module. Since XSLT can convert data into many formats, this is especially useful for preparing data in formats that can be easily processed by the pluggable module. This saves coding for different data structures in a generalized Module.

To carry out this processing, the behavior parameter set specifies an XSLT stylesheet on the Input side (Refer to the diagram in 7.2.4 CR Adapter Structure above).
3) To convert an XML Document into an Activity Document that can be recognized by the control server. The pluggable module has the option of either returning no data to the process definition, or returning an XML document in the format of the Output type Activity Document shown in "Activity Document format" above. This is especially useful when returning data from a pluggable module to the Activity Document in the calling process definition.

For example, the pluggable module is used to extract data from a back-end application, and needs to update the variable with this data. The solution is to create an XSLT stylesheet that reformats the data output from the pluggable module into the Output type Activity Document format shown in "Activity Document format" above.

To carry out this processing, the behavior parameter set specifies an XSLT stylesheet on the Output side (Refer to the diagram in 7.2.4 CR Adapter Structure above).

4) A combination of 2) and 3) above represents the most sophisticated behavior parameter set supported by the CR adapter. To do this processing, the behavior parameter set specifies an XSLT stylesheet on both the Input and Output side.

7.2.5 Pluggable Modules

The pluggable module is where purpose-specific processing is implemented. The XSLT support discussed above is directed at data handling requirements for a CR adapter. The pluggable module is directed at processing requirements.

The CollaborationRing PM Adapter Kit for Component Objects supports the following component types:

- Custom-built components
- SOAP "straight-thru" components (server)
- EJB "straight-thru" components

7.2.5.1 Custom-built components

The custom-built pluggable module is directed at processing requirements, which can include:

- Application logic that processes Activity Document data.
- Database access.
- A simple call to a back-end application.
- A complex interaction with a back-end application, involving multiple calls.
- Specific transformations of Activity Document data that cannot be easily done using XSLT stylesheets.

Characteristics for this type of module are as follows:

- The Module must implement the interface described in the pluggable module Interface API section in the Reference Guide. The resultant class(es) must be put in a JAR, and defined to CR adapter using its Component Wizard.
- If the document received from the CR adapter framework is an Activity Document, the Module can use the CRF Server API class to process it, if desired. This is useful if the developer does not want to use XML parsers explicitly, or if the document contains Base-64 data.
- If the document to be returned to CR adapter framework is an Activity Document, the Module can use the CRF Server API class to create it, if desired. This is useful if the developer does not want to use XML parsers explicitly, or if the document must contain Base-64 data.
- The list of properties relating to the custom-built component should be specified in the Custom Module Properties dialog of the Adapter Kit for Component Objects Component Wizard.

7.2.5.2 SOAP "straight-thru" server components

![SOAP "straight-thru" Server Components]

The SOAP "straight-thru" component, which is part of the automatic execution application framework, allows an activity to make a call to a SOAP service that implements the CRF_ServerAppl method. Refer to the section SOAP "Straight-thru" Service Method Signature in the Reference Guide for more information on this method's signature.
The SOAP "straight-thru" component constructs a SOAP message containing the Activity Document data that represents the automatic execution application input and sends the message as an HTTP request to the SOAP Service.

The SOAP service is responsible for extracting the Activity Document, and can use either the CRF Server API class library or any XML parser to analyze the Activity Document. After processing the Activity Document, a response SOAP message containing the processed Activity Document should be returned to the SOAP "straight-thru" component. This component will pass it back unchanged to the CR adapter, which can pass it back to the CollaborationRing PM process flow.

A number of properties are required by the SOAP "straight-thru" component in order to access a specific SOAP service. These property settings must be specified in the SOAP "straight-thru" properties dialog in the Component Wizard.

<table>
<thead>
<tr>
<th>Properties</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>SoapAddress</td>
<td>A mandatory property. The URI that identifies the SOAP service. For example, http://Interstage_Server:81/soap_admin/servlet/SOAProuter identifies an Interstage Application Server SOAP service. This value is used in the transport layer to route the SOAP request to the target SOAP server.</td>
</tr>
<tr>
<td>SoapTargetNameSpace</td>
<td>A mandatory property. The SOAP target object URI that is used to define the XML inside the SOAP Envelope Body of both the SOAP Request and Response. It is used to qualify the input parameters and output value for the &quot;CRF_ServerAppl&quot; method. An appropriate value for this property can be obtained from the administrator of the SOAP service or retrieved from the corresponding UDDI repository. Example: SOAPS\erverAdapterTest</td>
</tr>
<tr>
<td>SoapAction</td>
<td>An optional property. Use of the SOAP Action feature is OPTIONAL SOAP services may use it as a hint to optimize processing, but should not require its presence. If the property is set the value is used to set the SOAPAction value in the HTTP header of the SOAP Request sent to the SOAP service. An appropriate value for this property can be obtained from the administrator of the SOAP service or retrieved from the corresponding UDDI repository. Example: SOAPS\erverSomeActionURI</td>
</tr>
<tr>
<td>Timeout</td>
<td>A mandatory property. Used to specify the inactivity timeout value for the HTTP connection between the client (i.e pluggable module) and the SOAP service. The timeout value is specified in milliseconds in the range (0 - 2147483647). The default value is 0. If the Timeout property value is 0 the timeout period is assumed to be infinite. The timeout value should be less than the CORBA server timeout, otherwise the CORBA server will time out before the HTTP Connection.</td>
</tr>
</tbody>
</table>

Examples of these property settings are given below.

- To access the Interstage Application Server SOAP service:

  ```
  soapAddress=http://Interstage_Server:81/soap_admin/servlet/SOAProuter
  soapTargetNameSpace=SOAPS\erverAdapterTest
  soapAction=
  timeout=31000
  ```

- To access the .NET SOAP service:

  ```
  soapAddress=http://NET_Server/AdapterTest/TestSOAPService.asmx
  soapTargetNameSpace=http://tempuri.org/
  soapAction=http://tempuri.org/CRF_ServerAppl
  timeout=31000
  ```
7.2.5.3 EJB "straight-thru" server components

The EJB "straight-thru" component, part of the automatic execution application framework, allows an activity to make a call to an EJB stateless Session bean that has an interface equivalent to the CRF_ServerAppl interface supported by the CollaborationRing PM CORBA Interface. Refer to section EJB "Straight-thru" Bean Method Signature in the Reference Guide for more information on this method’s signature.

The stateless Session Bean is responsible for extracting the Activity Document from the automatic execution application and can use either the CRF Server API class library or any XML parser to analyze the Activity Document. After processing the Activity Document the stateless Session Bean constructs an output object of type class CRF_OutputParams, which contains the CollaborationRing PM return code value and the processed Activity Document and sends the output object back to the EJB "straight-thru" server adapter.

By setting appropriate values in the jndi.properties file accessible by the CR adapter, it is possible to make a call to an EJB stateless Session Bean running in a container other than that provided by Interstage Application Server. The values will be used by the CR adapter to access the appropriate JNDI service provided by that third party container.

The list of properties relating to the EJB "straight-thru" server adapter is given below. The property settings must be specified in the EJB properties dialog of the Adapter Kit for Component Objects Component Wizard.

<table>
<thead>
<tr>
<th>Properties</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>BeanName</td>
<td>Bean JNDI registration name</td>
</tr>
<tr>
<td></td>
<td>Example) SampleSessionBean</td>
</tr>
<tr>
<td>HomeInterface</td>
<td>Home Interface class name of the Bean including the package</td>
</tr>
<tr>
<td></td>
<td>Example) com.fujitsu.interstage.CollaborationRing.SampleSessionBeanHome</td>
</tr>
</tbody>
</table>

Examples of these property settings are given below.

- To access the Interstage Application Server EJB Bean:

  ```
  beanName=SampleBMPSessionBean
  homeInterface=com.fujitsu.interstage.SessionBean.SampleBMPSessionHome
  ```

7.2.6 SOAP to CR Adapter

The SOAP to CR adapter allows an external SOAP program to make a single call to CollaborationRing PM and start a Process Flow instance.

The function supported is similar to that provided by the crfcrtproc command-line function.

[SOAP to CR Adapter]

The adapter implements the following functions:

A SOAP server that wraps the Client API is provided. This server is compatible with SOAP functionality in Interstage Application Server and Microsoft .NET.
The functions to start a Process Flow instance are provided.

The client must know the name of the Process Definition.

The client must supply an XML fragment that is used to initialize the business document variables of the started process flow instance.

For UI Activities the client must also set the roleTitle and list of Assignees for that process flow instance.

Glossary

Activity
An activity represents a task in an Interstage CollaborationRing workflow or process. There are three distinct types of activity: UI activities, automatic execution activities and subprocess activities.

Activity Document
In this document, the term "Activity Document" is used to mean the XML documents that are exchanged between CR control servers and the adapters in each direction. In practice, they are the contents of the parameters passed within the Server-type API named as "activityDataIn" and "activityDataOut".

Adapter function
One of the basic component functions of CollaborationRing PM. Adapter functions provide interfaces for a variety of connections, such as to business applications, packages, and systems. Adapters are provided for all of the communications protocols and intermediary middleware products used in CollaborationRing PM.

Adapter kit
This function easily creates adapters that link with an existing system. It supports the creation of adapters CollaborationRing PM APIs, and can also link with SOAP and EJB systems.

The following two types of adapter kits are supported:
  - Adapter kit (for general applications)
  - Adapter kit (for component objects)

API
Application Programming Interface: The API is a collection of functions that can be used from other application programs, and is provided in operating systems, and so on. The functions defined in the API can be coded within programs so that applications can use the various functions.

The process flow controller has the following two types of API:
  - Server-type API
    The server-type API is called automatically from automatic execution activities specified in process definitions, and returns the processing results.
  - Client-type API
    The client-type API is activated by a process, or a UI (User Initiative) activity activated within a process, by means of a UI activity specification in the process definitions.

Assemble Tool
A GUI tool used to gather and/or group a number of associations for packaging into a single JAR file. See Deployment Tool.
**Association Wizard**

A GUI tool which enables an association to be made between a CR Automatic Execution activity and a CR Adapter behavior.

---

**Automatic execution activity**

Automatic execution activities are one type of activity, classified according to start method. This type of activity is used for business applications that are invoked from process definitions via CORBA. The operating timing for these activities is determined by the process progress status. The applications use CORBA to receive XML documents and to return results. When results are returned, processing advances to the next activity.

---

**Automatic execution application**

For automatic execution applications, the object name of the application is defined in an automatic execution activity in the process definitions so that, when control passes to that activity, the application is automatically invoked and returns the processing results using the server-type API. Automatic execution applications are CORBA server applications that have server-type API specification IDL and fixed method names.

---

**Base64 format**

The Base64 format is regulated as ISO-8859-1, and consists of 64 characters, being the uppercase and lowercase alphabets, numerics, and the plus (+) and slash (/) symbols, with the equals sign (=) used as an end pad.

---

**Basic authentication**

This is an authentication method that determines whether or not users are allowed access to a client based on a check of the user's user ID and password.

---

**Behavior**

Also Adapter Behavior. Specific combination of optional components (XSLT stylesheets, pluggable modules) and selection of input/output types to produce a specific result. The behavior definition configures the processing flow of the CR Adapter for a specific request from an activity in a process definition.

---

**BizTalk**

This function enables process linkage between CollaborationRing PM and Microsoft (R) BizTalk Server.

---

**Business document**

The collection of data items (i.e. business data defined by variables of a Process Definition, and associated control information) that pass from activity to activity in a process flow. A CR Adapter accesses and updates the state of the business document through input and output activity documents, which are in the form of XML documents.
**Business document bus**

In the Interstage CollaborationRing EAI model, the business document bus is defined as a virtual transmission path that circulates multiple business documents. In practice, the business document bus manages the business documents separately for each process. When each business process is generated from the process definitions, the business documents are managed separately for each process.

**Business document data**

Business document data is the user data managed by the control server and used in work flow processing.

The business document data is split into units consisting of the variables in the process definitions for management purposes. The initial values of the variables in the business document data can be specified in the process definition properties. Alternatively, they can be set at start sign time by applications that use the client-type API.

Users can reference and update business document data by means of applications that use the client-type API or by means of automatic execution applications.

**Business linkage functions (EAP)**

A variety of tasks, such as order processing between transaction partners, inventory management, in-house production, management, and planning department processes, and so on, are constructed on computers. Depending on operations, a variety of such tasks will exist on a user's system. The business linkage functions facilitate implementation of these tasks and linkages between these tasks.

The tasks that can be linked by the business linkage functions are the leading tasks and the continuation tasks. A leading task is a task that generates the data which is input to the business linkage functions. A continuation task is the output destination task for the data processed by the business linkage functions.

**Business process**

In the Interstage CollaborationRing EAI model, a business process is defined as an independent business system made from multiple applications. An accounting system or a production management system, for example, corresponds to one business process.

**Business process analyzer**

The business process analyzer is one of the basic functions comprising the CollaborationRing PM process flow manager.

The business process analyzer consists of the Management Tool of the process flow manager, and the history display and history search functions of the history reference functions.

The business process analyzer can be used to search for and display business process information such as business document status and numbers of items. The business process analyzer, in conjunction with the business process designer and the process flow controller, implements process integration.

**Business process designer**

The business process designer is one of the basic functions comprising the CollaborationRing PM process flow manager. It consists of a function that defines business processes in accordance with the process rules, and a function that defines format conversion for business documents. These functions define the business documents handled by the business processes, the process rules, and the business document conversion formats. The business process designer, in conjunction with the business process analyzer and the process flow controller, implements process integration.
CIDX

*Chemical Industry Data eXchange.* A group based in the United States developing standards in XML and EDI for the chemical industry, mainly concerned with developing standards for digital transactions in the industry.

---

**Client-type API**

The client-type API is one of the APIs provided by the process flow controller. The process flow manager API library is used for a client-type interface that operates with processes and servers. This API contrasts with the server-type API. With the client-type API operations, business documents are led by applications. In this context, lead by applications refers to operations that are started when they are invoked from an application. The main functions are process generation (business document storage), business document searches, and fetch, update, and similar processes.

---

**Client authentication**

This is an authentication method that determines whether or not users are allowed access to a client based on a check of the client certificate.

---

**Cluster system**

A cluster system consists of multiple servers that are grouped and used as one system. From the point of view of the client computers, there is only one server. If a fault occurs in one server, the applications being executed on that server are switched to another server, such that server resources are provided continuously.

---

**Component Wizard**

A GUI tool that creates/updates CR Adapter behavior definitions as well as defining the basic components that make up an Adapter behavior. The allowable components are XSLT stylesheets, EJB, SOAP, and/or pluggable modules.

---

**Control element**

Control elements are fixed tag elements handled by the process flow manager. These tag names are specified in the XML parser or in the client-type API library parameters or server-type API library parameters, thereby enabling applications to reference process and activity information. In addition, these tag names can be specified to update process and activity information.

In contrast to control elements, user elements are the business data tag elements specified in the input and output variables of process definitions.

---

**Control server**

CollaborationRing PM is a server product, and the control server is one element in the server configuration. Many business documents and process rules are placed in the management database and managed by the control server. This server is the practical embodiment of the business document bus.

The control server also incorporates history collection server functions that collect and manage in batch form the event data collected by the control server. Usually, the history collection server is mapped to a different machine than the control server, but the history collection server may be mapped to the same machine as the control server.
**CORBA**

*Common Object Request Broker Architecture*: CORBA consists of Object Request Broker (ORB) specifications proposed by the Object Management Group (OMG), which is an American object-oriented technology standardization organization. These are standard specifications for managing distributed objects and enable applications on different hardware and operating systems to be linked in networks. The applications are all handled as objects, and ORB products conforming to CORBA specifications provide the means for linking the distributed objects.

**CR**

Occasionally used to refer to CollaborationRing PM.

**Deployment Tool**

A GUI tool used to deploy a single JAR file (created by the Assemble Tool) on to a target machine. It creates the execution environment for the CR Adapter server, and installs the CR Adapter executable into it.

**DOM**

*Document Object Model*: An XML edit library (Fujitsu XML processor).

**Duplicate execution prevention function**

This function prevents accidental duplicate execution of processing for the same contents by an automatic execution application (application using the server-type API) or a control server.

**EAI**

*Enterprise Application Integration*: The purpose of EAI technology is to provide two-way linkages between different business systems within an enterprise and between systems at different enterprises, and to integrate processes and data. Fujitsu defines EAI as "the foundation for building a logical hub which is used to quickly link different departmental systems and systems in different enterprises, and thereby create new value". In this context, linkage means the ability to exchange electronic information, such as orders or accounting data via an open network or a closed network.

In January 2000, Fujitsu added the EAI tool Interstage CollaborationRing to the web-based Interstage development environment. An EAI tool is a software development kit that provides functions used to make linkages between the various application pairs in core business systems, ERP systems, and so on.

**EAP**

*Enterprise Application Pipelines*: See Business Linkage Functions (EAP).
ebXML


---

**EJB**

*Enterprise JavaBeans:* EJB adds, to the JavaBeans specifications used to construct application software by combining program parts created in Java language, functions required for server-side processing of network distributed type applications. EJB is part of the Enterprise Java specifications that group functions required at the server when Java is used in business applications, and is installed in Web servers, and similar.

---

**ERP**

*Enterprise Resource Planning:* ERP provides integrated management of the various administrative resources, including production and sales, inventory, purchasing, delivery, accounts, and personnel resources, within an enterprise to improve operation efficiency. ERP is an impressive solution for improving use of in-house administrative resources and administrative efficiency, and is also known as "management task integration" and "administrative resource integration management".

The ERP package is a business package that integrates the core business systems of an enterprise. Rather than creating separate computer systems for core tasks such as administration, production management, and personnel management, package groups prepared in advance are specified in order to build a total solution of linked systems.

The adapters provided by CollaborationRing PM can be used to make linkages with ERP systems such as SAP R/3 (R), and Oracle E-Business Suite.

---

**Event data**

Event data are elements in wide-area tracking. Event data are the information collected each time the control server invokes an activity.

---

**Event collection method**

The event collection method is one of the methods provided by CollaborationRing PM for wide-area tracking. In this method, one history collection server is constructed to manage history data in the units in which the history is collected. At regular intervals, each of the control servers transfers to the history collection server the history (event) data that are obtained each time an activity is invoked, then the database on the history collection server is used for centralized management. Users can send a reference request to the history collection server for a collective search (tracking) of the status of all business documents that match the specified conditions.

---

**Execution server (OTS link server)**

CollaborationRing PM is a server product, and the execution server (OTS link server) is one element in the server configuration. The execution server controls execution of automatic execution applications and helps prevent duplicate execution. The execution server can be on the same machine as the control server but can be mapped to multiple machines to distribute the load of each job. This is needed only when you execute the automatic execution application that uses the Interstage Application Server database linkage service (OTS).

---

**Failover**

If a fault occurs in one server in a cluster system, the failover function moves the applications being executed on that
server to another server such that continuous server resources are provided.

---

**FEDIT**

FEDIT is conversion software that translates formats. It can be the input-output for conversion of the following formats:

- UN/EDIFACT format
- User-specific formats
- XML documents

---

**Fixed type-b format (business document format)**

The fixed type-b format is one of the user-specific formats. This format is a fixed-length format, and consists of header records, detail records, and trailer records. Other user-specific formats are the fixed type-a format and the variable type format.

---

**Form**

The term for the information set that associates the tasks a user performs on a particular process with the client application that actually carries out the tasks.

---

**Form Output Support Function**

This function supports the development of a form output environment in the CollaborationRing application environment. By adding it as a subprocess to business processes on CollaborationRing, it communicates with Systemwalker/ListCREATOR to send voucher data.

---

**Format conversion function**

The format conversion service is one of the basic functions comprising CollaborationRing PM. This function carries out format conversions and character code conversions for data transmitted between integrated systems.

---

**Global tracking**

The Global tracking function searches for (tracks) business document and forms processes that span multiple servers. CollaborationRing stores information from each invocation of an activity in the database that manages status. Users can search this database to find out which process is currently being performed for a business document.

---

**GUI**

*Graphical User Interface*: A graphical tool for interfacing with a user.

---

**HA**

*High Availability*: A system that offers high level of reliability. This is a function that, in a multi-server system, allows
another system to take over processing instantly if the operational system fails for any reason. There are two standby modes: Operational and Mutual. CollaborationRing PM monitors computers by means of a twin server composition. When an abnormality is detected in one server, operations are switched instantly to the other.

---

**History collection server**

The history collection server is one component of wide-area tracking. The event data collected at each of the control servers is collected (accumulated) and managed as a batch on this server. The history collection server is one of the control server functions. Usually, the history collection server is located on a separate machine to the control server machine, but may be mapped to the same machine as the control server.

---

**IDL**

*Interface Definition Language*: IDL is the language used during program development to code the interface (external specifications) that enables the program parts called as objects to be used from other programs. IDL can be used to define methods (commands) that encompass objects, properties (attributes), and similar information.

---

**IIOP**

*Internet Inter-ORB Protocol*: This is the standard protocol used for communication between ORBs in CORBA 2.0. IIOP enables linkages between application objects in distributed systems comprised of a variety of hardware and operating systems. In addition, the applications do not require modification if hardware is added or if the system configuration is changed, and applications can be freely mapped. This enables flexible system construction that is independent of platforms and vendors.

---

**Interstage**

This is link-up software that provides development and runtime environments for implementing flexible distributed object environments above the operating system.

---

**Interstage Application Server**

This is foundation software that executes CORBA-based applications.

---

**Interstage CollaborationRing**

The CollaborationRing product group is intended for the integration of business applications and is classified as an Interstage Series Integration. This is the Fujitsu solution for quick achievement of Enterprise Application Integration (EAI), thereby integrating and continuing to use existing assets, packages, and components to create new value in the context of the business revolution.

---

**JCA**

*J2EE Connector Architecture*: A set of rules proposed by Sun Microsystems, Inc. governing interface standards from J2EE applications to enterprise information systems.
JDBC

*Java Database Connectivity*: A Java API for executing SQL syntax, comprising class and interface sets written in Java. It provides a standard API for developers of tools and databases, and enables the creation of database applications using Pure Java API.

---

Job-dependent information

Job-dependent information is one of the elements comprising a process document. This is a collection of data parts expressed in an XML document (really an XML fragment), and is referred to as the start sign data.

---

J2EE

*Java2 Enterprise Edition*: A Java component development standard specification proposed by Sun Microsystems, Inc. that governs and defines component architecture for distributed applications in Java.

---

Management information

Management information is one of the elements comprising process documents. The process flow controller is the part that manages processes. The management information includes process status, custom attribute 1 to 5, process title, consistent business document ID, and other data. These parts are not gathered together in one table, but are in the control information of individual activities and interfaces and are expressed indirectly in the data used for condition evaluation by the process flow controller.

---

Message ID

The message ID is a unique 28-digit number that the control server adds to data when it sends the data to an automatic execution application in order to prevent duplicate execution of that data by the automatic execution application.

After completion of the processing that returns data from the automatic execution application to the control server, the data may not reach the control server due to a communication error or other reason. In this case, the retry process at the control server may post the same data to the automatic execution application again. Applications can use the message ID to prevent duplicate execution of the same data.

---

Node

An element of a process definition that can represent items start points, end points, and junctions.

---

OLTP

*Online Transaction Processing*: The processing of online transactions.

---

PIP

*Partner Interface Process*: An electronic commerce specified on RosettaNet that defines the business processes
among the companies dealt with on the supply chain. It also defines the format for documents passed between companies, data conversion tools, and work flow.

**Pluggable Module**
Java class.

**Process**
Giving the initial data to a process definition and starting the work flow is called the start sign or process generation, and that which is generated is called the process or process instance. The process flow manager uses the client-type API library to generate the process by means of the start sign (CreateProcess).

**Process definition**
A process definition defines the processing flow in the workflow. Activity and business document data definitions are also included.

A process definition is identified from the process definition title. Thus, when the process definition tools create a process definition, a process definition title that is unique within the control server must be specified.

**Process Definition Add-in**
An add-in to the UML editor that enables process definitions to be described in the UML editor. With this tool, you can define UDDI access conditions and Web service execution conditions.

**Process Definition Tool**
The Process Definition Tool is the GUI tool used for entering process rules. This tool can be used to code business processes as process definitions. The process definitions can be executed after they are registered at the control server.

**Process document**
When each business process is generated from the process definitions, the business documents are managed separately by each of those processes. The business documents managed by each process are the business document bus embodiment. These business documents, when managed separately by each process, are called process documents. A process document is split into management information and job-dependent information (also called start sign data), and is managed in XML format.

**Process flow controller**
The process flow controller is one of the basic functions comprising the CollaborationRing PM business flow manager. This function is for managing execution of the process instances as indicated in the process definitions, and is usually on the control server.
**Process flow manager**

The process flow manager is the function that performs CollaborationRing PM process management and execution management. The control server (and its operation management tools), execution server (OTS link server), client-type API library, server-type API library, HTTP bridge, process definition tools, and so on, are included.

---

**Process rules**

In the Interstage CollaborationRing EAI model, the process rules are defined as the rules used to link multiple activities. The activity processing sequence, the conditions used to select activities, and the business document conversion format can be set in the process rules.

---

**Properties File**

Java specific information repository file.

---

**R/3 (R) System**

This is the name of the ERP (Enterprise Resource Planning) system provided by SAP AG. The R/3(R) configuration is a client server system consisting of three layers: databases, applications, and presentations (clients). These layers have open specifications that are independent of the hardware and operating system. In addition, the BAPI (business API) programming interface is provided to enable flexible addition of extension functions.

---

**Repository**

Store for adapter configuration information. Refers to the Repository Server and Repository database.

---

**RMI**

Java Remote Method Invocation.

---

**Role**

The person in charge of an activity. Indicates the person responsible for carrying out a UI activity task.

---

**RosettaNet**

A consortium to promote the efficiency of supply chains for computer industries and electronic component industries. By creating business processes and a common interface (PIP) for each system from a business viewpoint rather than from a system viewpoint, it eases the business information exchange among companies and aims to reduce the cost reduction for procurements and sales.

This uses a protocol that uses XML on a large scale instead of the traditional EDI messages.
Scenario definition

A scenario is an expression for the flow of processes of transactions between companies or of the tasks that are required (such as converting documents) when companies do business together. A scenario definition is a statement of that flow (often, this is in the form of a written contract). Occasionally referred to as a scenario model. Refer to the TPM Release Guide for more information.

Server-type API

The server-type API is one of the APIs provided by the process flow manager, and is used when an automatic execution application is started. This interface is called by the CORBA server objects in the XML document input and output. The interface has the specified type of IDL and a fixed method name, and can be created as an Interstage Application Server distributed object.

The process flow controller determines the start timing for this interface on the basis of the process definitions.

Session Bean

An object that communicates with clients by session. It is impermanent and only exists between sessions.

Skeleton

"Skeleton" is an abbreviated term for "skeleton file". In the computer field, a skeleton is often prepared in advance as a pre-coded file containing items that must be entered. Such a file is called a skeleton file, or simply a skeleton. In UNIX, for example, when a new user registers, the administrator places a file in the user's home directory. This file already contains entries for basic settings such as ".login" and ".chsrc". This type of file is a skeleton file.

Also, when you use the adapter kit (for general application use) in compliance with questions posed by the wizard, the basic source code that meets purpose for the program is created, which is also called a "Skeleton".

SOAP

Simple Object Access Protocol: Used as a communication protocol to exchange the information for XML base on distributed network environments.

Start sign

The part that provides the initial data to the process definitions and starts the workflow is the start sign. The process flow controller uses the client-type library to generate a process by means of the start sign (CreateProcess).

Subprocess

A subprocess is a process in a subordinate layer. The process definitions are layered and modular in the same way as functions and subroutines. Generation of processes on a subordinate layer from the parent layer is referred to as subprocess generation, and processes on the subordinate layer are known as subprocesses. Subprocess activities generate the subprocesses. SWAP is used to invoke subprocesses.
**Subprocess activity**

A subprocess activity is an activity that generates a subprocess. A subprocess is a process on a subordinate layer.

When workflow control is extended to subprocess activities, the data (input variables) specified in the subprocess activity definitions is extracted from the business document, and that data is used as the start sign initial data when the start sign starts the subprocess. Subprocess activities can be either synchronous or asynchronous.

---

**SSL**

*Secure Socket Layer.* Developed after a proposal by Netscape Communications Corporation, SSL is a network security method designed to prevent unauthorized viewing of data on the Internet. The method provides security by first verifying such details as the identity and password of users when they connect to the Web server before it allows those users to send and receive information.

---

**SWAP**

*Simple Workflow Access Protocol.* SWAP is a profile proposed by Netscape and others as the protocol to be used by workflow engines to exchange workflow information in an Internet environment. Workflow data coded in XML is translated into HTTP extended protocol.

---

**Systemwalker**

Systemwalker is an operation management product for information systems based on PSM (Policy-based Systems Management) concepts. PSM concepts have evolved from the question "How can information systems be best utilized in conjunction with administrative strategies?"

---

**TPM**

*Trading Partner Manager.* A function for implementing business transactions using the XML-based trading standards RosettaNet, ebXML, and CIDX.

---

**Trading analysis**

Trading analysis is one of the business process analyzer functions. When an inter-enterprise task that is defined in advance as a process flow is executed, this function supplies statistical information related to the workflow. This information can be used to determine problem points and make improvements to inter-enterprise tasks. This function is usually located on a separate machine to the history collection server machine, but can be on the same machine. Use of the trading analysis API enables analysis from a variety of angles in addition to the information provided as a default.

---

**Trading analysis function**

The trading analysis function is one of the functions for analyzing trading, and displays total processing times for forms exchanged between enterprises and statistical processing results. This function records the progress status of inter-enterprise tasks as trading logs and calculates statistical information related to the processing times for each process. These results can be used to find bottlenecks in the supply chain between the enterprises involved in the trading and determine the causes of delays, thereby optimizing the supply chain and improving the tasks.
Trading logs

The trading logs contain the data used for trading analysis. Parts of the management database data on the history collection server are incorporated. In practice, the main information in these logs is the date and time of forms transfers and the DUNS and other information obtained at the time of TPM trading.

Transaction

Before and after a process, the transaction processing guarantees that the information in the database is consistent. This processing is used if the database can be used by multiple users. OMG prescribes the operations, such as start, end, and cancellation of processing, performed by transaction processing as a common object service. Applications can use the APIs of ORB products that conform to these guidelines to avoid dependence on a specific vendor.

UDDI

Universal Description, Discovery and Integration: A global directory for registering and searching for Web services. Essential business information regarding the Web services is registered in XML format by the company offering the service. Companies wishing to use a Web service can search through the directory to find an appropriate service. The actual registering, querying, and updating are carried out using a SOAP-based API.

UI

User Initiative: When activities are classified on the basis of operation, UI activities are one of the activity types. UI activities are used for business document operations at any time that is convenient for the task. This type of activity waits for the operation completion instruction (complete) from the task before proceeding to the next task. These activities can wait for and receive user operations, or be used to search or update multiple business documents. Other activity types are the application automatic execution type and the subprocess type.

UML

Unified Modeling Language: A modeling language that combines the best methods of other modeling languages and is used to develop object-oriented systems.

UN/EDIFACT

United Nations/Electronic Data Interchange For Administration Commerce and Transport: These are the international rules for electronic data interchange in administration, commerce, and transport, and were developed by the trade procedure simplification task force of the United Nations ECE (European commercial association). UN/EDIFACT has been set as an international EDI standard under ISO9735 and ISO7372.

User element

User elements are the elements in the business data tags specified in the input variables and the output variables in the process definitions. The basic data that can be included directly in a user element is character data, but the document structure is not restricted.

Any XML documents (strictly speaking, XML fragments) can be used in user elements. The user element is the unit that corresponds to activity input and output when the process definition tools are used. Roughly speaking, if a user element (or variable) does not have an internal tag structure, it is the character string type, and user elements that have an internal tag structure are the XML document type. The parts that the process flow controller can target for evaluating conditions are character string type user elements and some variables (custom attribute, and so on) in the
management information.

---

**User processing wait function**

The user processing wait function stops processing partway through the workflow, then waits until an operator (or application program) performs user-lead access to the workflow, performs the task that waits for processing, and ends processing. The user processing wait function is expressed in a UI activity. When workflow control reaches the UI activity, the business documents stop as tasks requiring operator action. The operator (or application program) uses the client-type API to process the business documents that have stopped at this activity. The business documents do not advance beyond this activity until the operator processes them.

---

**Web server**

CollaborationRing PM is the main server product, and a Web server is one of its structural elements. The Web server receives requests from clients and performs communication between business processes. The Web server is usually on the same machine as the control server, but can be on a separate machine.

---

**Web service**

A service component available for use via the Internet. Various individual Web services can be combined to create newer, better Web services. Web services use SOAP as the communication protocol, WSDL as the language for writing interfaces, and can be registered and searched for using UDDI.

---

**Web Services Aggregator**

This function simplifies the construction of work systems to link with Web services. Web Services Aggregator components include functions to make CollaborationRing PM business processes available as Web services and to analyze Web service performance. Web Services Aggregator is occasionally referred to as WSA.

---

**WF-XML**

WF-XML is the protocol for workflow linkage between enterprises that was standardized by the current workflow standardization body, the Workflow Management Coalition (WFMC). WF-XML has greater functionality than previous workflow linkage standards, including search functions, query functions, and so on, and uses XML as the data format.

---

**Wizard**

A tool for gathering information from the user through a number of steps and generating other information (or files) from the given input.

---

**Worklist**

Within the tasks (activities) activated in a process, those that wait for operator tasks (UI) are grouped separately for each operator in lists known as worklists. Operators check these lists to determine which task to start.
**WSDL**

*Web Services Description Language:* An XML-based language for writing Web service interfaces. You can describe such information as the Web service URL, the protocol to use, and message formats.

---

**XML**

*Extensible Markup Language:* XML is a page coding language standardized by the Web Consortium (W3C). XML integrates the strong points from HTML, the current page coding language standard, and from the SGML (Standard Generalized Markup Language) document coding language that is controlled as an international standard.

---

**XSLT**

*XML Stylesheet Language Transformation:* An XSL file used for XML transformation, or an engine that executes the transformation.