

**uCosminexus Stream Data Platform -  
Application Framework  
Setup and Operation Guide**

3020-3-V02(E)

## ■ Relevant program products

P-2464-9B17 uCosminexus Stream Data Platform - Application Framework 01-00 (for Windows Server 2008)

## ■ Trademarks

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

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

Java is a registered trademark of Oracle and/or its affiliates.

JDK is either a registered trademark or a trademark of Oracle and/or its affiliates.

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

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

Sun is either a registered trademark or a trademark of Oracle and/or its affiliates.

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

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

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

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

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

This product includes software developed by Andy Clark.

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

This product includes software developed by Daisuke Okajima and Kohsuke Kawaguchi (<http://relaxngcc.sf.net/>).

This product includes software developed by IAIK of Graz University of Technology.

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

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

This product includes software developed by the Java Apache Project for use in the Apache JServ servlet engine project (<http://java.apache.org/>).

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

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

uCosminexus Stream Data Platform - Application Framework contains RSA(R) BSAFE™ software from EMC Corporation.

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

## ■ Microsoft product name abbreviations

This manual uses the following abbreviations for Microsoft product names.

Full name or meaning	Abbreviation
Microsoft(R) Windows Server(R) 2008 Enterprise	Windows Server 2008 or Windows
Microsoft(R) Windows Server(R) 2008 Enterprise 32-bit	
Microsoft(R) Windows Server(R) 2008 R2 Enterprise	
Microsoft(R) Windows Server(R) 2008 R2 Standard	
Microsoft(R) Windows Server(R) 2008 Standard	

Full name or meaning	Abbreviation
Microsoft(R) Windows Server(R) 2008 Standard 32-bit	
Microsoft(R) Windows(R) Internet Explorer(R)	Internet Explorer

■ **Restrictions**

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

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

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

Printed in Japan.

■ **Edition history**

Aug. 2011: 3020-3-V02(E)

■ **Copyright**

All Rights Reserved. Copyright (C) 2011, Hitachi, Ltd.



---

# Preface

---

This manual explains how to design, set up, and operate uCosminexus Stream Data Platform - Application Framework systems, and it provides details about functions that can be specified when a system is set up. This manual is intended to provide the user with the ability to analyze stream data through the design, setup, and operation of uCosminexus Stream Data Platform - Application Framework systems.

## Intended readers

This manual is intended for system administrators who design, set up, and operate uCosminexus Stream Data Platform - Application Framework systems.

Readers of this manual must have:

- A basic knowledge of operating systems
- A basic knowledge of Java (an understanding of the meanings of Java-related terms)
- A basic knowledge of XML

This manual also assumes that the reader is familiar with the manual *uCosminexus Stream Data Platform - Application Framework Description*, so we recommend that you first read this manual.

## Conventions: Fonts and symbols

The following table explains the fonts used in this manual:

Font	Convention
<b>Bold</b>	<b>Bold</b> type indicates text on a window, other than the window title. Such text includes menus, menu options, buttons, radio box options, or explanatory labels. For example: <ul style="list-style-type: none"><li>• From the <b>File</b> menu, choose <b>Open</b>.</li><li>• Click the <b>Cancel</b> button.</li><li>• In the <b>Enter name</b> entry box, type your name.</li></ul>
<i>Italics</i>	<i>Italics</i> are used to indicate a placeholder for some actual text to be provided by the user or system. For example: <ul style="list-style-type: none"><li>• Write the command as follows: <code>copy source-file target-file</code></li><li>• The following message appears: A file was not found. (file = <i>file-name</i>)</li></ul> <i>Italics</i> are also used for emphasis. For example: <ul style="list-style-type: none"><li>• Do <i>not</i> delete the configuration file.</li></ul>

Font	Convention
Code font	<p>A code font indicates text that the user enters without change, or text (such as messages) output by the system. For example:</p> <ul style="list-style-type: none"> <li>• At the prompt, enter <code>dir</code>.</li> <li>• Use the <code>send</code> command to send mail.</li> <li>• The following message is displayed: <code>The password is incorrect.</code></li> </ul>

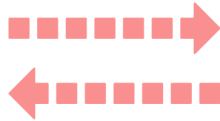
The following table explains the symbols used in this manual:

Symbol	Convention
	<p>In syntax explanations, a vertical bar separates multiple items, and has the meaning of OR. For example: <code>A B C</code> means A, or B, or C.</p>
{ }	<p>In syntax explanations, curly brackets indicate that only one of the enclosed items is to be selected. For example: <code>{A B C}</code> means only one of A, or B, or C.</p>
[ ]	<p>In syntax explanations, square brackets indicate that the enclosed item or items are optional. For example: <code>[A]</code> means that you can specify A or nothing. <code>[B C]</code> means that you can specify B, or C, or nothing.</p>
...	<p>In coding, an ellipsis (...) indicates that one or more lines of coding are not shown for purposes of brevity. In syntax explanations, an ellipsis indicates that the immediately preceding item can be repeated as many times as necessary. For example: <code>A, B, B, ...</code> means that, after you specify A, B, you can specify B as many times as necessary.</p>
< >	<p>Angle brackets (&lt; &gt;) indicate that the enclosed item (element, file, or other item) either is specified by the user or is output by the system. For example: <code>&lt;parameter name&gt;</code> means either that the user specifies the actual parameter name or that the parameter name is output by the system.</p>
x	Multiplication sign
/ or ÷	Division sign

## Conventions: Diagrams

This manual uses the following conventions in diagrams:

- Flow of stream data



## Conventions: Version numbers

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

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

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





---

# Contents

---

<b>Preface</b>	i
Intended readers .....	i
Conventions: Fonts and symbols.....	i
Conventions: Diagrams .....	iii
Conventions: Version numbers.....	iii

## **PART 1: Before You Start**

<b>1. Overview of the Configuration and Operation of Stream Data Platform - AF</b>	<b>1</b>
1.1 Flow from introduction to operation .....	2
1.2 Organization of this manual .....	5

## **PART 2: Design and Configuration**

<b>2. System Design</b>	<b>7</b>
2.1 Flow of design .....	8
2.2 Evaluating the system configuration .....	10
2.2.1 Program configuration in a stream data processing system .....	10
2.2.2 Components of a stream data processing system .....	11
2.2.3 Configuration when standard adaptors are used .....	16
2.2.4 Configuration when custom adaptors are used .....	20
2.3 Evaluating the stream data processing engine.....	21
2.3.1 Evaluating the number of SDP servers .....	21
2.3.2 Evaluating the SDP server's time control method.....	21
2.3.3 Evaluating adjustment of the timestamps in tuples .....	21
2.4 Evaluating the adaptor configuration .....	23
2.4.1 Evaluating the adaptors to be used .....	23
2.4.2 Evaluation items when the standard adaptors are used.....	23
2.4.3 Evaluation items when custom adaptors are used .....	24
2.5 Evaluating the processing to be performed by the standard adaptors .....	25
2.5.1 Evaluating the data input methods .....	26
2.5.2 Evaluating the data editing methods .....	26
2.5.3 Evaluating the data output methods .....	29
2.6 Evaluating the summary analysis scenarios .....	30
2.6.1 Evaluating the stream data summary analysis scenarios.....	30
2.6.2 Evaluating the configuration of the working directories for executing query groups.....	31

2.7	Estimating the memory requirements .....	33
2.7.1	Estimating the memory requirements for the stream data processing engine .....	34
2.7.2	Estimating the memory requirements for standard adaptors .....	38
<b>3.</b>	<b>System Configuration</b> .....	<b>45</b>
3.1	Flow of configuring .....	46
3.2	Directory structure .....	49
3.2.1	Structure of the installation directory .....	49
3.2.2	Structure of the working directories .....	52
3.3	Setting up an operating environment .....	55
3.3.1	Registering an administrator user .....	55
3.3.2	Creating a working directory .....	55
3.4	Creating the SDP server definition files .....	56
3.4.1	Settings that can be specified in the SDP server definition files .....	56
3.4.2	How to create the SDP server definition files .....	58
3.5	Creating the query definition files .....	59
3.5.1	Settings that can be specified in the query definition files .....	59
3.5.2	How to create the query definition files .....	59
3.6	Creating the adaptor definition files .....	60
3.6.1	Settings that can be specified in the adaptor definition files .....	60
3.6.2	How to create the adaptor definition files .....	60
3.7	Setting up the dashboard .....	62
3.7.1	Output settings for dashboard-display data .....	62
3.7.2	Setting up Dashboard Server .....	62
3.7.3	Setting up Dashboard Viewer .....	64

## **PART 3: Operation**

<b>4.</b>	<b>System Operation</b> .....	<b>67</b>
4.1	Flow of system operation .....	68
4.2	Starting the system .....	72
4.2.1	Starting the SDP server .....	72
4.2.2	Registering query groups .....	72
4.2.3	Starting query groups .....	73
4.2.4	Starting adaptors (standard adaptors) .....	73
4.2.5	Starting adaptors (custom adaptors) .....	74
4.2.6	Checking the status of adaptor groups .....	75
4.3	Using queries and query groups .....	78
4.3.1	Re-executing queries .....	78
4.3.2	Displaying the status of query groups .....	81
4.4	Shutting down the system .....	82
4.4.1	Terminating adaptors (standard adaptors) .....	82

4.4.2 Terminating adaptors (custom adaptors) .....	83
4.4.3 Terminating query groups .....	83
4.4.4 Shutting down the SDP server .....	85
4.5 Displaying analysis results on a dashboard .....	87
4.5.1 Starting Dashboard Server .....	87
4.5.2 Using Dashboard Viewer to display analysis results .....	87
4.5.3 Shutting down Dashboard Server .....	88

## **5. Modifying the System** 89

---

5.1 Overview of system modification .....	90
5.2 Modifying the stream data processing engine .....	91
5.3 Modifying query groups .....	92
5.3.1 Changing settings in the property files .....	92
5.3.2 Changing the query definition file .....	93
5.4 Modifying adaptors .....	94
5.5 Modifying the dashboard .....	95

## **6. Troubleshooting** 97

---

6.1 Error handling procedure .....	98
6.2 Data to be collected in the event of an error .....	99
6.3 Details of log files and trace files .....	106
6.3.1 Details of log files .....	106
6.3.2 Details of API trace information .....	110
6.3.3 Details of adaptor trace information .....	111
6.3.4 Details of tuple logs .....	118
6.3.5 Details of a thread dump .....	122
6.4 How to handle major problems .....	127
6.4.1 Problems during system operation .....	127

## **PART 4: Reference**

### **7. Commands** 133

---

Format of command explanations .....	134
List of commands .....	135
sdpcql (registers a query group) .....	136
sdpcqldel (deletes a query group) .....	138
sdpcqlstart (starts a query group) .....	139
sdpcqlstop (terminates a query group) .....	141
sdpls (displays the status of query groups) .....	143
sdpsetup (sets up an operating environment) .....	148
sdpstart (starts the SDP server) .....	150
sdpstartap (starts RMI-connection adaptors) .....	152
sdpstartinpro (starts in-process-connection adaptors) .....	155

sdpstop (stops the SDP server) .....	157
sdpstopap (terminates RMI-connection adaptors) .....	159
sdpstopinpro (terminates in-process-connection adaptors).....	161
sdptppls (displays tuple information) .....	163
sdptplput (reloads tuples).....	169
sdptrced (edits trace information) .....	173
<b>8. SDP Server Definition Files</b> .....	<b>177</b>
8.1 Format of SDP server definition file explanations .....	178
8.2 Notes about creating SDP server definition files .....	179
8.3 List of SDP server definition files .....	180
8.4 JavaVM options file for SDP servers (jvm_options.cfg).....	182
8.5 JavaVM options file for RMI connections (jvm_client_options.cfg).....	187
8.6 System configuration property file (system_config.properties) .....	192
8.6.1 Details of the system configuration property file (system_config.properties).....	192
8.6.2 Details of the parameters in the system configuration property file (system_config.properties).....	196
8.7 Query group property file .....	209
8.7.1 Details of the query group property file .....	209
8.7.2 Details of the parameters in the query group property file.....	214
8.7.3 Coding rules for conditional expressions .....	224
8.8 Stream property file .....	227
8.8.1 Details of the stream property file .....	227
8.8.2 Details of the parameters in the stream property file.....	231
8.9 In-process connection property file (user_app.adaptor-group-name-or-adaptor-name.properties).....	239
8.10 Log file output property file (logger.properties) .....	241
8.11 List of JavaVM options.....	242
<b>9. Adaptor Definition Files</b> .....	<b>249</b>
9.1 Format of adaptor definition file explanations .....	250
9.2 Notes about creating adaptor definition files .....	251
9.3 List of adaptor definition files .....	252
9.4 Adaptor command definition file (AgentManagerDefinition.xml) .....	253
9.5 Adaptor configuration definition file (AdaptorCompositionDefinition.xml).....	254
9.5.1 Overview of the adaptor configuration definition file (AdaptorCompositionDefinition.xml).....	254
9.5.2 Adaptor configuration definition file namespaces.....	258
9.6 Common definition in the adaptor configuration definition file.....	260
9.6.1 Common definition.....	260
9.6.2 Adaptor trace definition.....	260
9.7 Adaptor group definition in the adaptor configuration definition file.....	262
9.7.1 In-process group definition.....	262

9.7.2 RMI group definition .....	263
9.8 Adaptor definition in the adaptor configuration definition file .....	264
9.8.1 Input adaptor definition .....	264
9.8.2 Output adaptor definition .....	266
9.9 CB definition in the adaptor configuration definition file .....	268
9.9.1 CB definition for input .....	268
9.9.2 CB definition for output .....	270
9.9.3 CB definition for editing .....	271
9.9.4 CB definition for sending .....	273
9.9.5 CB definition for receiving .....	274
9.10 CB definitions for input and output in the adaptor configuration definition file .....	276
9.10.1 File input connector definition .....	276
9.10.2 HTTP packet input connector definition .....	282
9.10.3 File output connector definition .....	292
9.10.4 Dashboard output connector definition .....	296
9.11 CB definitions for data editing in the adaptor configuration definition file .....	301
9.11.1 Format conversion definition .....	301
9.11.2 Mapping definition .....	318
9.11.3 Filter definition .....	330
9.11.4 Record extraction definition .....	334
9.12 CB definitions for sending and receiving in the adaptor configuration definition file .....	344
9.12.1 Input stream definition .....	344
9.12.2 Output stream definition .....	345
9.13 Coding examples for an adaptor configuration definition file .....	346
9.13.1 Example 1 .....	346
9.13.2 Example 2 .....	352

## **10. Details About Definitions in the Definition Files** 365

---

10.1 Organization of this chapter .....	366
10.2 File input .....	367
10.2.1 Overview of file input .....	367
10.2.2 Flow of data processing during file input .....	367
10.2.3 File input settings .....	375
10.3 HTTP packet input .....	376
10.3.1 Overview of HTTP packet input .....	376
10.3.2 Flow of data processing during HTTP packet input .....	377
10.3.3 HTTP packet input settings .....	379
10.4 Record filtering .....	381
10.4.1 Overview of record filtering .....	381
10.4.2 Flow of data processing during record filtering .....	382
10.4.3 Record filtering settings .....	383
10.5 Record extraction .....	384
10.5.1 Overview of record extraction .....	384

10.5.2	Flow of data processing during record extraction .....	385
10.5.3	Record extraction settings .....	393
10.6	File output .....	394
10.6.1	Overview of file output .....	394
10.6.2	Flow of data processing during file output .....	394
10.6.3	File output settings .....	399
10.7	Dashboard output .....	400
10.7.1	Overview of dashboard output .....	400
10.7.2	Flow of data processing during dashboard output .....	401
10.7.3	Dashboard output settings .....	405
10.8	Timestamp adjustment for tuples .....	406
10.8.1	Scope of timestamp adjustment .....	406
10.8.2	Range of times to be adjusted .....	406
10.8.3	How to adjust the time .....	407
10.8.4	Processing after tuples have been input .....	415
10.8.5	Operation when the query group shuts down .....	416
10.8.6	Tuple retention period .....	416
10.8.7	Selecting tuples by filtering .....	417
10.8.8	Timestamp adjustment settings .....	417
<b>11.</b>	<b>Details of Flex Dashboard Settings</b> .....	<b>419</b>
11.1	Organization of this chapter .....	420
11.2	Dashboard Server internal settings file (usrconf.properties) .....	421
11.3	Dashboard Viewer window layout file .....	423
11.3.1	Configuration of Dashboard Viewer window .....	423
11.3.2	Details of a Dashboard Viewer window layout file .....	425
11.4	Example definitions for dashboard output .....	439
11.4.1	Output adaptor definition (displaying the most recent data) .....	439
11.4.2	Output adaptor definition (displaying data including historical data) .....	442
<b>Appendix</b>		<b>445</b>
A.	Reference Material for This Manual .....	446
A.1	Related publications .....	446
A.2	Conventions: Abbreviations for product names .....	446
A.3	Conventions: Acronyms .....	447
A.4	Conventions: KB, MB, GB, and TB .....	447
<b>Index</b>		<b>449</b>

## Chapter

---

# 1. Overview of the Configuration and Operation of Stream Data Platform - AF

---

This chapter discusses the flow from introduction to operation of Stream Data Platform - AF. You should be familiar with the organization of this manual before you begin reading the subsequent chapters.

- 1.1 Flow from introduction to operation
- 1.2 Organization of this manual

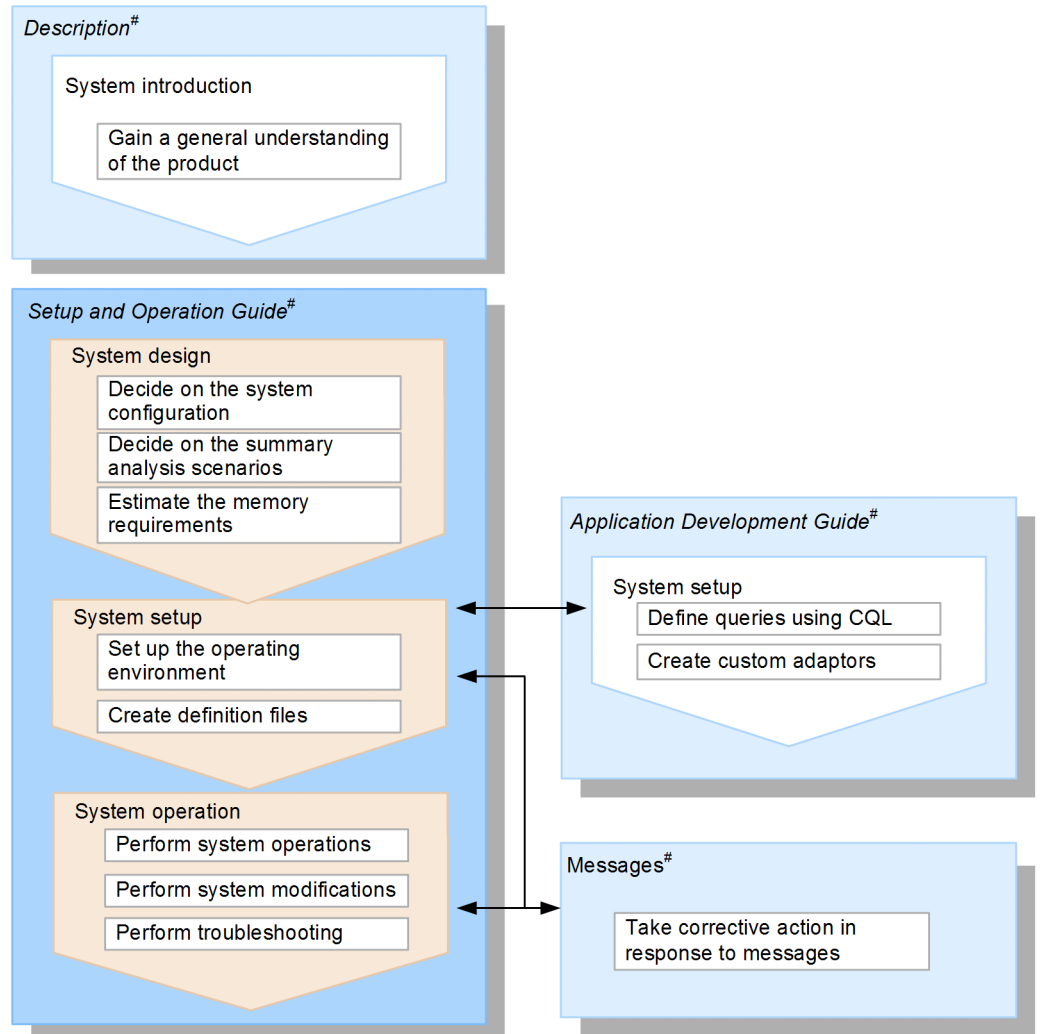
## **1.1 Flow from introduction to operation**

---

The figure below shows the relationship between the related manuals and the flow from introduction to operation of Stream Data Platform - AF.



Figure 1-1: Relationship between the related manuals and the flow from introduction to operation of Stream Data Platform - AF



Legend:

- : This manual
- : Other manuals in this series
- : Work phase
- : Task performed by the user
- : Refer to these items if needed

#: In the manual titles, the *uCosminexus Stream Data Platform - Application Framework* portion is omitted.

In the flow from introduction to operation, this manual discusses the stages of *System*

*design, System setup, and System operation.*

Before reading this manual, you should be familiar with the manual *uCosminexus Stream Data Platform - Application Framework Description*, which provides an overview of the Stream Data Platform - AF products.

Also, if necessary, you should consult the related manuals for each work stage. For example, at the stages of system design and system configuration, see the manual *uCosminexus Stream Data Platform - Application Framework Application Development Guide* for how to use CQL to define queries and how to create custom adaptors. At the stage of system operation, see the manual *uCosminexus Stream Data Platform - Application Framework Messages* for details of checking troubleshooting messages.

## 1.2 Organization of this manual

This section describes the organization of this manual. The manual consists of the following four parts and an appendix:

- Part 1: Before You Start

Part 1 discusses the flow from introduction to operation of Stream Data Platform - AF and the organization of this manual, which you should understand before you read this manual.

- Part 2: Design and Configuration

Part 2 discusses the design and configuration of a stream data processing system.

- Part 3: Operation

Part 3 discusses how to operate, modify, and troubleshoot a stream data processing system.

- Part 4: Reference

Part 4 provides details about the commands and definition files.

- Appendix A

Appendix A provides reference information for this manual.

The table below provides an overview of this manual's chapters and appendix.

*Table 1-1: Overview of chapters and appendix*

<b>Part</b>	<b>Chapter or appendix</b>	<b>Description</b>
<i>Part 1 Before You Start</i>	<i>Chapter 1. Overview of the Configuration and Operation of Stream Data Platform - AF</i>	Describes the flow from introduction to operation of Stream Data Platform - AF and the organization of this manual.
<i>Part 2 Design and Configuration</i>	<i>Chapter 2. System Design</i>	Describes the flow of system design and discusses items that should be evaluated at the design stage.
	<i>Chapter 3. System Configuration</i>	Describes the flow of system configuration, the structure of the Stream Data Platform - AF directories, and the settings required for configuring a system.
<i>Part 3 Operation</i>	<i>Chapter 4. System Operation</i>	Describes how to operate a configured system.
	<i>Chapter 5. Modifying the System</i>	Describes how to change system settings and the configuration during system operation.

1. Overview of the Configuration and Operation of Stream Data Platform - AF

<b>Part</b>	<b>Chapter or appendix</b>	<b>Description</b>
	<i>Chapter 6. Troubleshooting</i>	Describes the types of data to be collected in the event of a problem during system operation, how to interpret the data, and how to handle problems that arise.
<i>Part 4 Reference</i>	<i>Chapter 7. Commands</i>	Describes the commands used to operate the system.
	<i>Chapter 8. SDP Server Definition Files</i>	Describes the SDP server definition files.
	<i>Chapter 9. Adaptor Definition Files</i>	Describes the adaptor definition files.
	<i>Chapter 10. Details About Definitions in the Definition Files</i>	Provides details about the definitions in the SDP server definition files and adaptor definition files.
	<i>Chapter 11. Details of Flex Dashboard Settings</i>	Provides details about the Dashboard Server internal settings file and the Dashboard Viewer window layout file.
<i>Appendix</i>	<i>Appendix A. Reference Material for This Manual</i>	Provides reference information for this manual.

## Chapter

---

# 2. System Design

---

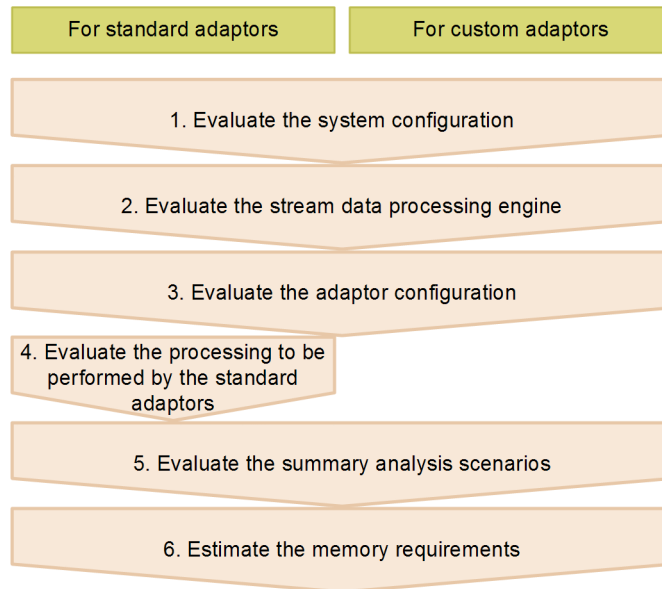
This chapter describes the flow of designing a stream data processing system and the items that should be evaluated at the design stage.

- 2.1 Flow of design
- 2.2 Evaluating the system configuration
- 2.3 Evaluating the stream data processing engine
- 2.4 Evaluating the adaptor configuration
- 2.5 Evaluating the processing to be performed by the standard adaptors
- 2.6 Evaluating the summary analysis scenarios
- 2.7 Estimating the memory requirements


## 2.1 Flow of design

The work required for designing a stream data processing system depends on the type of adaptors used. The following figure shows the flow of system design.

*Figure 2-1: Flow of system design*



Legend:

 : Required task

The following describes each task:

1. Evaluating the system configuration  
Evaluate the configuration of programs and the placement of system components, in accordance with the purpose of your stream data processing system. For details, see *2.2 Evaluating the system configuration*.
2. Evaluating the stream data processing engine  
Evaluate the number of SDP servers to be used in the stream data processing system and the functions to be used by those SDP servers. For details, see *2.3 Evaluating the stream data processing engine*.
3. Evaluating the adaptor configuration  
Evaluate the types of adaptors to be used in the stream data processing system, the

number of adaptors to be created, and the functions used by those adaptors. For details, see *2.4 Evaluating the adaptor configuration*.

4. Evaluating the processing to be performed by the standard adaptors  
If you use the standard adaptors, evaluate the processing to be performed during data input, data editing, and data output. For details, see *2.5 Evaluating the processing to be performed by the standard adaptors*.
5. Evaluating the summary analysis scenarios  
Evaluate the scenarios used to summarize and analyze stream data. For details, see *2.6 Evaluating the summary analysis scenarios*.
6. Estimating the memory requirements  
Estimate the amount of memory required for the stream data processing system. For details about how to estimate the memory requirements, see *2.7 Estimating the memory requirements*.

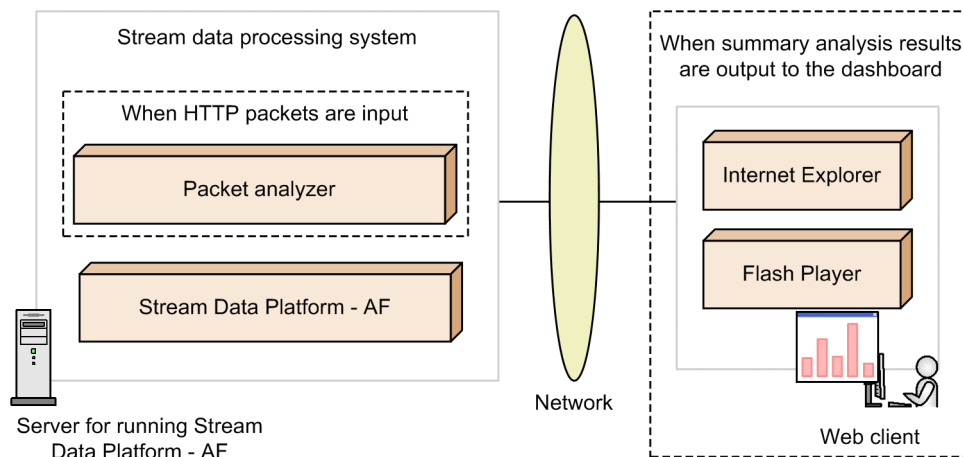
## 2.2 Evaluating the system configuration

This section discusses a program configuration, components, and adaptors for a stream data processing system.

### 2.2.1 Program configuration in a stream data processing system

The programs required for a stream data processing system will depend on the type of adaptors to be used and the functions to be used by the adaptors. The following figure shows a configuration of programs in a stream data processing system.

Figure 2-2: Configuration of programs in a stream data processing system



With a basic system configuration, the only program required for the server that runs Stream Data Platform - AF is Stream Data Platform - AF.

Note the prerequisite programs that are also required in the following cases:

- When a standard adaptor is used to input HTTP packets

A packet analyzer is required on the server that runs Stream Data Platform - AF. For details about the packet analyzers supported by Stream Data Platform - AF, see *10.3.1 Overview of HTTP packet input*.

- When a standard adaptor is used to output summary analysis results to the dashboard

To use a Web browser to view the summary analysis results of stream data that are to be output to a dashboard, you need the following programs for the Web client:

- Internet Explorer
- Flash Player



For the program versions, see the Release Notes for Stream Data Platform - AF.

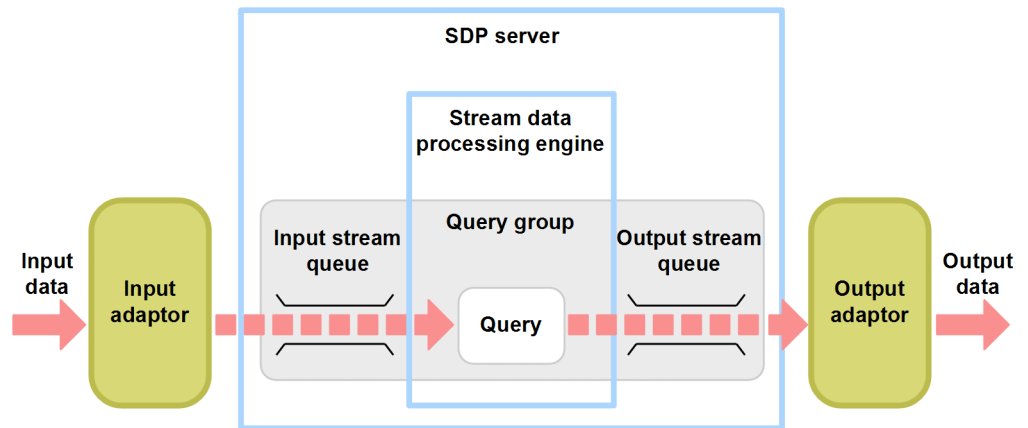
## 2.2.2 Components of a stream data processing system

This subsection describes the components of a stream data processing system. It also discusses the mode of connection between adaptors and the SDP server and the number of components that can be placed in the stream data processing system.

### (1) Components of a stream data processing system

The following figure shows the components of a stream data processing system.

Figure 2-3: Components of a stream data processing system



The following subsections describe the components shown in the figure:

- *Adaptors (input adaptor and output adaptor)*

The input adaptor converts input data to a format that can be processed by the stream data processing engine and then sends the data to the stream data processing engine. The output adaptor receives the data processed by the stream data processing engine, converts it to a specified format, and then outputs the data.

The two types of adaptors are the standard adaptors provided by Stream Data Platform - AF (*standard adaptors*) and adaptors created by the user using Java programming (*custom adaptors*).

Standard adaptors function as a group called an *adaptor group*. For details about adaptor groups, see 2.2.3(1) *Configuration of an adaptor group*.

For the custom adaptors, an input adaptor is called a *data transmission application* and an output adaptor is called a *data reception application*.

- *Stream data processing engine*

The stream data processing engine processes the data sent from an input adaptor

in accordance with a pre-registered query. The processed data is then sent to an output adaptor.

A server process that runs on the stream data processing engine and processes stream data is called an *SDP server*. The adaptors and the SDP server can all operate in the same process, or they can operate in separate processes. In the figure above, the adaptors and the SDP server are operating in separate processes. For details about the connection mode between adaptors and SDP server, see (2) *Connection mode between adaptors and SDP server*.

### ■ *Query group*

A query group constitutes the summary analysis scenarios for stream data. In the stream data processing system, the summary analysis scenarios for stream data, such as the types of data to be input and how that data is to be processed and output, are defined as query groups.

A query group consists of the following elements:

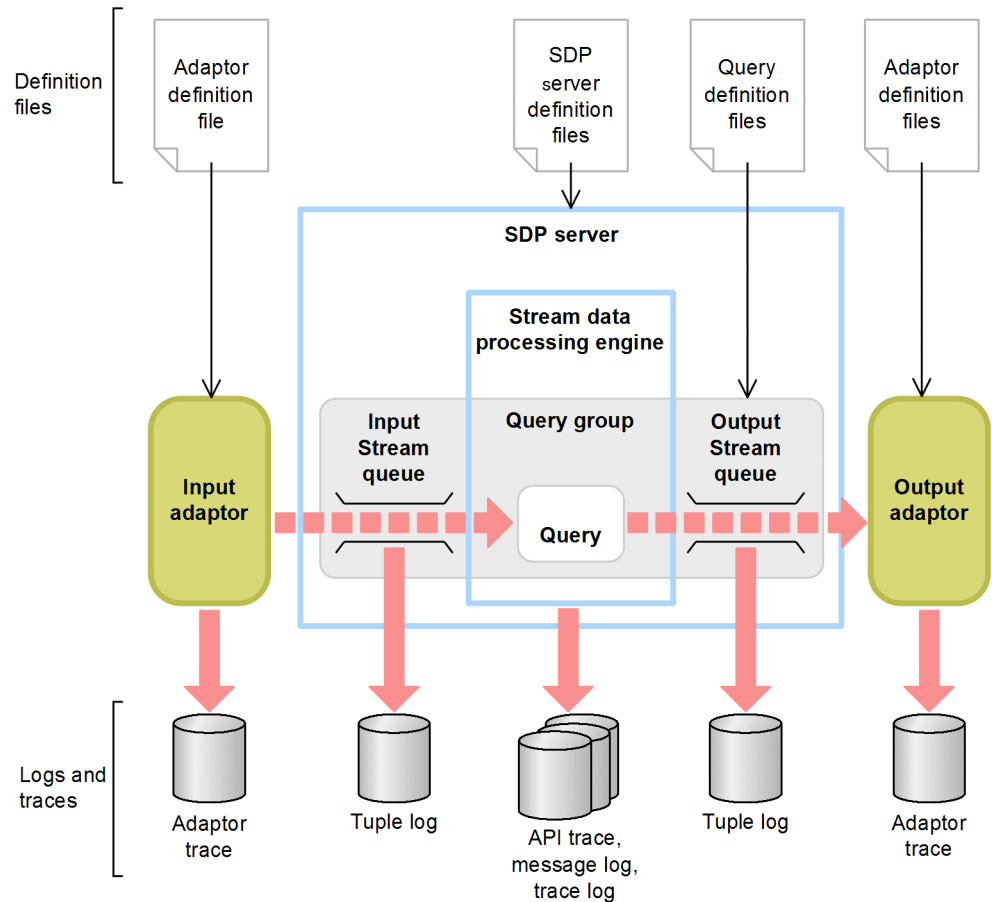
- *Input stream queue* (input stream)
- *Query*
- *Output stream queue* (output stream)

A query is a definition that specifies how stream data is to be processed. Stream data is sent to a query via an input stream queue. The result of query processing is converted to stream data and then output via an output stream queue. There is a one-to-one correspondence between a stream queue and a stream.

For details about query groups, see 2.6 *Evaluating the summary analysis scenarios*.

You use definition files to specify the operation of the adaptors and the SDP server. Files, such as log and trace files, are output by the adaptors and the SDP server. The following figure shows the relationships between the components of a stream data processing system and individual files.

Figure 2-4: Relationships between the components of a stream data processing system and individual files



The following describes the files shown in the figure:

- Definition files

The following explains the types of definition files:

- *SDP server definition files*

These definition files define the SDP server operations. For details about the SDP server definition files, see [3.4 Creating the SDP server definition files](#) and [8. SDP Server Definition Files](#).

- *Query definition files*

These definition files define summary analysis scenarios for stream data. For

details about the query definition files, see 3.5 *Creating the query definition files*.

- *Adaptor definition files*

These definition files define the operation of the standard adaptors. For details about the adaptor definition files, see 3.6 *Creating the adaptor definition files* and 9. *Adaptor Definition Files*.

- **Logs and traces**

The logs and traces include the adaptor traces that output the adaptor processing status, the tuple logs that output information about the input and output tuples, and the API traces that output query entry and exit information. For details about these files, see 6.2 *Data to be collected in the event of an error*.

## (2) **Connection mode between adaptors and SDP server**

The adaptors and the SDP server operate either all in the same process or in separate processes. The mode of connection that is used when you operate the adaptors and the SDP server in the same process is called an *in-process connection*, and the mode of connection that is used when you operate them in separate processes is called an *RMI connection*.

The table below describes the advantages and disadvantages of these two connection modes.

*Table 2-1: Advantages and disadvantages of the adaptor and SDP server connection modes*

No.	Connection mode	Advantage	Disadvantage
1	In-process connection	Cost of communication between SDP server and adaptors is low and the tuple processing latency can be minimized.	Memory usage must be monitored because all processing is performed in a single process.
2	RMI connection	The effects of an adaptor failure on the stream data processing engine can be minimized. Also, more memory can be allocated to adaptors.	The communication cost is higher than with in-process connection because process-to-process communication must be performed.

You must determine how to configure the SDP server and adaptors, taking into account the advantages and disadvantages of the connection modes.

The system configuration in each connection mode depends on whether standard adaptors or custom adaptors are used. For details about these types of system configurations, see 2.2.3 *Configuration when standard adaptors are used* and 2.2.4 *Configuration when custom adaptors are used*.

### (3) Number of components that can be placed in a stream data processing system

There are limitations to the number of adaptors that can be started by an SDP server and the number of adaptors that can be managed in an adaptor group. The number of components that can be placed in a stream data processing system depends on whether standard adaptors or custom adaptors are used. The table below shows the number of components that can be placed in a stream data processing system.

*Table 2-2:* Number of components that can be placed in a stream data processing system

No.	Item	Maximum value when standard adaptors are used	Maximum value when custom adaptors are used
1	Number of SDP servers that can be run in one working directory <sup>#1</sup>	1	1
2	Number of RMI connection adaptors that can be run by one SDP server	--	32 <sup>#2</sup>
3	Number of in-process-connection adaptors that can be registered in one SDP server	--	16 <sup>#2</sup>
4	Number of adaptor groups that can be executed by one SDP server	1	--
5	Number of adaptors that can be managed in one adaptor group	64 <sup>#2</sup>	--
6	Number of query groups that can be executed by one SDP server	8	8
7	Number of streams that can be registered in one SDP server	1,024 <sup>#3</sup> in total	1,024 <sup>#3</sup> in total
8	Number of queries that can be registered in one SDP server		

Legend:

--: Not applicable

#1

The *working directory* is a directory used for running the stream data processing system. For details about the configuration of the working directory, see 2.6.2 *Evaluating the configuration of the working directories for executing query groups*.

#2

This maximum value is a guideline. You should determine the number of adaptors on the basis of the performance of the server on which Stream Data Platform - AF is to be run and performance in the stream data processing system.

#3

Set the numbers of streams and queries so that their total number combined does not exceed 1,024.

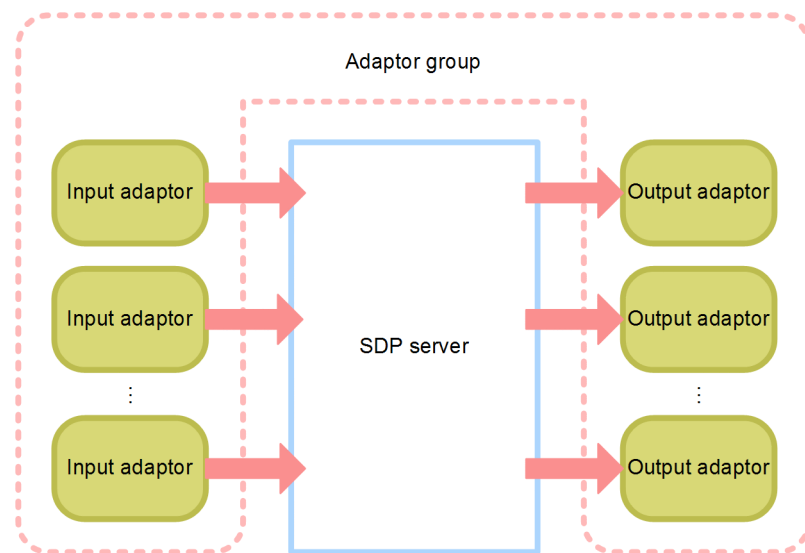
### 2.2.3 Configuration when standard adaptors are used

This subsection describes the configuration of an adaptor group when the standard adaptors are used and the mode of connection between the standard adaptors and the SDP server. It also discusses the relationship between the standard adaptors and streams.


#### (1) Configuration of an adaptor group

In the case of using standard adaptors, the group of adaptors is called an *adaptor group*. The following figure shows an example of a configuration of an adaptor group.

Figure 2-5: Example configuration of an adaptor group



Legend:

 : Adaptor group

Only one adaptor group can be run by a single SDP server. You run standard adaptors

in an adaptor group.

You can manage a maximum of 64 input adaptors and 64 output adaptors per adaptor group. In the adaptor group, a thread is assigned to each adaptor and all adaptors are run in a single process.

## **(2) Mode of connection between standard adaptors and SDP server**

As mentioned in 2.2.2(2) *Connection mode between adaptors and SDP server*, the modes of connection between adaptors and the SDP server are the in-process connection and the RMI connection.

Standard adaptors are connected with the SDP server in adaptor groups. An adaptor group used in the in-process connection mode is called an *in-process group*, and an adaptor group used in the RMI connection mode is called an *RMI group*.

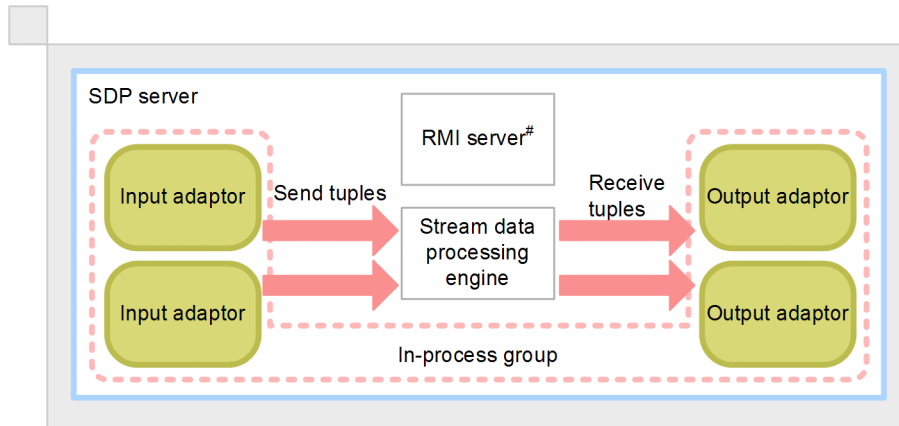
Note that adaptors in the in-process connection mode and adaptors in the RMI connection mode cannot be intermixed in the same adaptor group. For example, you cannot have a configuration in which the input adaptors form an RMI group and the output adaptors form an in-process group.

The following discusses the configurations of an in-process group and an RMI group.

### ■ *Configuration of an in-process group*

In the in-process connection mode, the adaptors and the SDP server run in the same process and send and receive tuples within the process. The following shows an example configuration of an in-process group.

Figure 2-6: Example configuration of an in-process group



Legend:



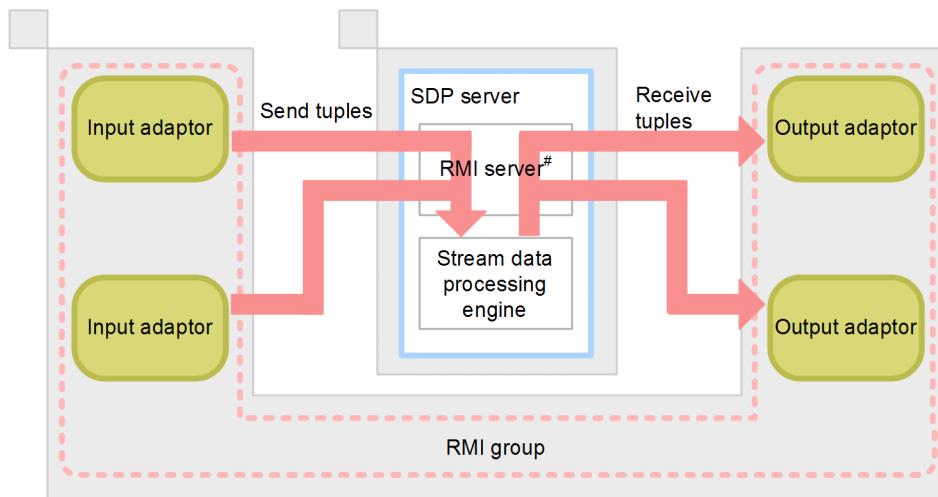
#: The RMI server, which runs on the SDP server, performs RMI communication processing. In the case of an in-process group, an RMI server is not needed.

■ *Configuration of an RMI group*

In the RMI connection mode, the adaptors and the SDP server run in separate processes and use Java RMI to send and receive tuples between the processes. The following shows an example configuration of an RMI group.



Figure 2-7: Example configuration of an RMI group



Legend:

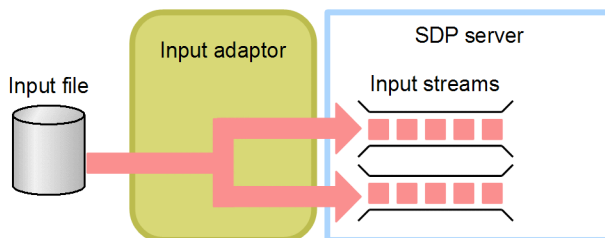


#: The RMI server, which runs on the SDP server, performs RMI communication processing.

**(3) Relationship between standard adaptors and stream queue (stream)**

With standard adaptors, there is a 1-to-*n* correspondence between an input adaptor and input stream queues (input streams). That is, one input adaptor can send stream data to multiple input stream queues. The following figure shows the relationship between an input adaptor and input streams.

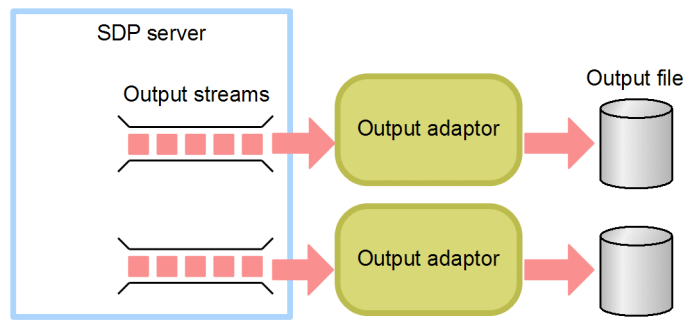
Figure 2-8: Relationship between input adaptor and input streams



On the other hand, there is a 1-to-1 correspondence between an output adaptor and an output stream queue (output stream). That is, one output adaptor receives stream data from a single output stream queue. This means that if there are multiple output streams, you must create an output adaptor for each output stream. The following figure shows

the relationship between output adaptors and output streams.

Figure 2-9: Relationship between output adaptors and output streams



## 2.2.4 Configuration when custom adaptors are used

As mentioned in 2.2.2(2) *Connection mode between adaptors and SDP server*, the modes of connection between adaptors and the SDP server are the in-process connection and the RMI connection. If you use custom adaptors, you can intermix adaptors that operate in the two connection modes. You would determine the placement of the SDP server and the adaptors, taking into account the size and purpose of your stream data processing system. For details about system configurations depending on the connection mode, see the manual *uCosminexus Stream Data Platform - Application Framework Application Development Guide*.

---

## 2.3 Evaluating the stream data processing engine

---

To help you evaluate the requirements of the stream data processing engine, this section discusses how to determine the number of SDP servers and the functions to be used by the SDP servers.

### 2.3.1 Evaluating the number of SDP servers

You can run only one SDP server in a single working directory. If there are multiple working directories, you must run an SDP server for each working directory.

If you use the standard adaptors, make sure there is a 1-to-1 correspondence between adaptor groups and SDP servers.

### 2.3.2 Evaluating the SDP server's time control method

You must evaluate which of the following timestamp modes is to be used to set the time in tuples.

- *Server mode*

This mode sets in a tuple the time at which the tuple arrives at the SDP server.

- *Data source mode*

This mode sets in a tuple the time at which data is created.

For details about the timestamp modes, see the manual *uCosminexus Stream Data Platform - Application Framework Description*.

To specify the timestamp mode, you use the system configuration property file (`system_config.properties`) of the SDP server definition files or the `stream.timestampMode` parameter in the query group property file. For details, see [8.6 System configuration property file \(`system\_config.properties`\)](#) or [8.7 Query group property file](#).

### 2.3.3 Evaluating adjustment of the timestamps in tuples

If you use the data source mode as the SDP server's time control method, the times set in the tuples need to be in ascending order of the input streams. If tuples are sent from multiple input adaptors, an error might occur between tuple times and arrival times, and a tuple that has arrived so that its time is out of sequence might be discarded by the SDP server. For example, if tuples arrive at the SDP server in the order 09:00:04, 09:00:05, and 09:00:02, the tuple with the time 09:00:02 would be discarded.

In such a case, you can use the timestamp adjustment function to set a tuple arrival time adjustment value, so that the SDP server can receive those tuples notwithstanding that the times are not in chronological order. In the above example, if the tuple arrival time adjustment value is set to 5, the tuple with timestamp 09:00:02 will not be discarded

## 2. System Design

even though it arrived after 09:00:05.

For details about the mechanism of the timestamp adjustment function and how to set up the function, see *10.8 Timestamp adjustment for tuples*.

---

## 2.4 Evaluating the adaptor configuration

---

To help you evaluate the requirements of the adaptor configuration, this section discusses evaluation of the adaptors to be used, and the evaluation items for each type of adaptor.

### 2.4.1 Evaluating the adaptors to be used

You must evaluate whether you will use the standard adaptors or custom adaptors as the adaptors for sending and receiving stream data.

Standard adaptors are provided by Stream Data Platform - AF. They support functions, such as data input from files and HTTP packets, data editing involving mapping and record extraction, and data output to files and a dashboard.

Custom adaptors, on the other hand, are created by the user using Java programming. You create custom adaptors in order to use functions that are not supported by the standard adaptors and to be able to use adaptors to perform special processing.

You determine whether the standard adaptors or custom adaptors are to be used as appropriate for the stream data summary analysis in your system.

### 2.4.2 Evaluation items when the standard adaptors are used

This subsection discusses the items to be evaluated when you use the standard adaptors.

- Number of adaptors to be created

You must create as many adaptors as there are data input sources and output destinations. As mentioned in 2.2.2(3) *Number of components that can be placed in a stream data processing system*, you can manage a maximum of 64 input adaptors and 64 output adaptors per adaptor group.

For details about the numbers of adaptors and streams, see 2.2.3(3) *Relationship between standard adaptors and stream queue (stream)*.

- Processing to be performed by the adaptors

For evaluation of the processing to be performed by the adaptors, see 2.5 *Evaluating the processing to be performed by the standard adaptors*.

- Number of adaptor groups to be created

There must be a 1-to-1 correspondence between adaptor groups and SDP servers.

- Mode of connection between adaptors and SDP server

You must evaluate whether the in-process connection mode or the RMI connection mode to use to connect adaptors and the SDP server. For details about

the connection modes, see *2.2.3(2) Mode of connection between standard adaptors and SDP server*.

You use the adaptor configuration definition file, which is one of the adaptor definition files, to specify detailed settings for the standard adaptors. For details about the adaptor configuration definition files, see *3.6 Creating the adaptor definition files* and *9. Adaptor Definition Files*.

### **2.4.3 Evaluation items when custom adaptors are used**

This subsection discusses the items to be evaluated when you use custom adaptors.

- Number of adaptors to be created

You must create as many adaptors as there are data input sources and output destinations. For details about the numbers of adaptors that can be created, see *2.2.2(3) Number of components that can be placed in a stream data processing system*. You determine the number of adaptors to be created on the basis of the performance of the server on which Stream Data Platform - AF is run and performance in the stream data processing system.

- Processing to be performed by the adaptors

You must evaluate the processing to be performed by the adaptors, such as how to input and output stream data. Based on that evaluation, you should further evaluate how those functions are to be implemented by using Java APIs provided by Stream Data Platform - AF.

- Mode of connection between adaptors and SDP server

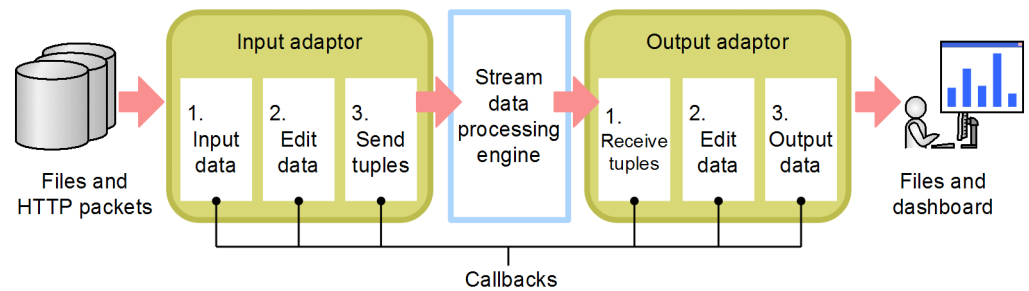
You must evaluate whether the in-process connection mode or the RMI connection mode is to be used to connect adaptors and the SDP server. For details about the connection modes, see *2.2.4 Configuration when custom adaptors are used*.

## 2.5 Evaluating the processing to be performed by the standard adaptors

This section discusses the evaluation of the processing that is to be performed by the standard adaptors.

The processing to be performed by the standard adaptors includes data input and output, data editing, and sending and receiving of tuples. The following figure shows the flow of processing performed by the standard adaptors.

Figure 2-10: Flow of processing performed by the standard adaptors



### Processing performed by input adaptors

1. Input data from files and HTTP packets
2. Edit data, such as mapping and record extraction
3. Send tuples to the stream data processing engine

### Processing performed by output adaptors

1. Receive tuples from the stream data processing engine
2. Edit data, such as mapping and filtering
3. Output data to files and dashboard

With the standard adaptors, each of the data input and output, data editing, and tuple exchange processing elements is called a *callback*. Callback processing is defined in each CB definition in the adaptor configuration definition file, one of the adaptor definition files.

For the callback processing performed by the standard adaptors, you must evaluate the nature of the processing that is to be performed as data input and output and data editing processing. Your evaluation will include how data is to be input, output, and edited in accordance with the purpose of stream data summary analysis in your system.

### 2.5.1 Evaluating the data input methods

You must evaluate which of the input methods provided by the standard adaptors is to be used to input data.

Data input is performed by an input adaptor. The standard adaptors enable you to use connectors for data input (input connectors) to input data from files and from HTTP packets. The table below lists and describes the data input methods.

*Table 2-3: Data input methods*

No.	Data input method	Description
1	File input	This method uses a <i>file input connector</i> to input files, such as error logs and access logs, as data for stream data summary analysis. For details about file input, see <i>10.2 File input</i> .
2	HTTP packet input	This method uses an <i>HTTP packet input connector</i> to input HTTP packets as data for stream data summary analysis. The HTTP packet input method inputs HTTP packets that were output from the packet analyzer by using an HTTP packet input connector. For details about the HTTP packet input, see <i>10.3 HTTP packet input</i> .

Note that you specify data input in the CB definition for input in the adaptor configuration definition file. For details about the CB definition in the adaptor configuration definition file, see *9. Adaptor Definition Files*.

### 2.5.2 Evaluating the data editing methods

You must evaluate which of the data editing methods provided by the standard adaptors is to be used to edit data.

Data editing is performed by the input or output adaptors. The standard adaptors support data editing, such as mapping and record extraction. The table below lists and describes the data editing methods.

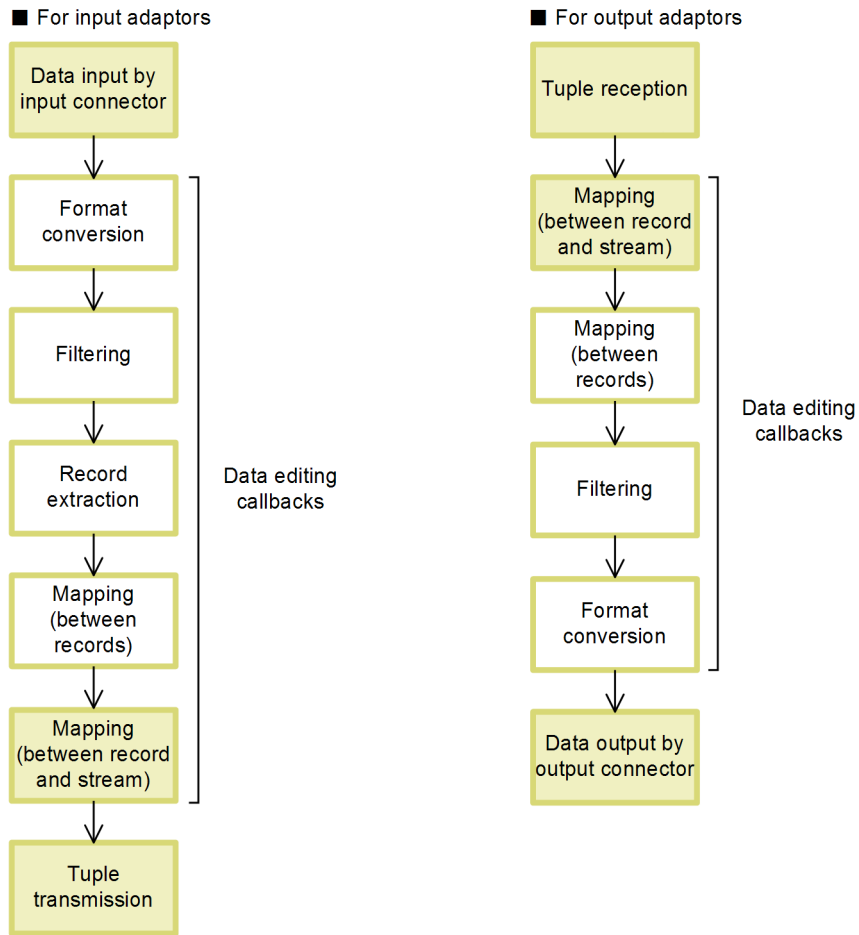


Table 2-4: Data editing methods

No.	Data editing method	Description
1	Format conversion	<p>You can use <i>format conversion</i> to convert from input or output records to common records, and vice versa.</p> <ul style="list-style-type: none"> <li>• Input records and output records: Each record consists of a row of data separated by commas (such as in CSV files).</li> <li>• Common records: Each record is a set of data consisting of a name (field name) and a value (field value).</li> </ul> <p>An input adaptor's format conversion involves conversion from an input record to a common record. An output adaptor's format conversion involves conversion from a common record to an output record. For details about each data format, see <i>10.2.2(1) Data formats handled by an input adaptor</i> and <i>10.6.2(1) Data formats handled by an output adaptor</i></p> <p>Note that format conversion is performed only in the following cases:</p> <ul style="list-style-type: none"> <li>• When the file input method is used to input data</li> <li>• When the file output method is used to output data</li> </ul> <p>For details about format conversion, see <i>10.2.2(3) Format conversion</i> or <i>10.6.2(4) Format conversion</i>.</p>
2	Mapping	<p>You can use <i>mapping</i> to associate the common record output in the callback that precedes the mapping and the common record input in the callback that follows the mapping.</p> <p>The two types of mapping are the mapping between records and the mapping between record and stream.</p> <p>For example, a mapping between record and stream associates a common record that is input or output by format conversion with a common record based on the input stream or output stream format.</p> <p>For details about mapping, see <i>10.2.2(4) Mapping</i> or <i>10.6.2(3) Mapping</i>.</p>
3	Record filtering	<p>You can use a <i>filter</i> to select records. The target of record filtering is common records.</p> <p>For details about record filtering, see <i>10.4 Record filtering</i>.</p>
4	Record extraction	<p>You can use <i>record extraction</i> to extract records that satisfy a specified condition and then join the extracted records to create a new record. The target of record extraction is common records.</p> <p>Note that record extraction is applicable only to input adaptors.</p> <p>For details about record extraction, see <i>10.5 Record extraction</i>.</p>

The following figure shows the timing of data editing.

Figure 2-11: Timing of data editing



Legend:

: Mandatory callback

: Optional callback

### Timing of format conversion

For input adaptors, format conversion is performed after data input by an input connector. For output adaptors, format conversion is performed before data output by an output connector.

### Timing of mapping<sup>#</sup>

Mapping between record and stream is performed before a tuple is sent for input adaptors, and after a tuple is received for output adaptors.

Mapping between records is performed after data input by input connector, or after format conversion for input adaptors and before data conversion or data output by output connector for output adaptors.

#### Timing of filtering<sup>#</sup>

For input adaptors, filtering is performed after data input by the input connector or format conversion. For output adaptors, filtering is performed before format conversion or data output by output connector.

#### Timing of record extraction<sup>#</sup>

Record extraction is performed after data input by input connector or format conversion.

#

You can perform filtering, record extraction, and mapping in any order.

Note that you specify data editing in the CB definition for editing in the adaptor configuration definition file. For details about the CB definition in the adaptor configuration definition file, see *9. Adaptor Definition Files*.

### 2.5.3 Evaluating the data output methods

You must evaluate which of the output methods provided by the standard adaptors is to be used to output data.

Data output is performed by an output adaptor. The standard adaptors enable you to use connectors for data output (output connectors) to output the results of stream data summary analysis to files and a dashboard. The table below lists and describes the data output methods.

*Table 2-5: Data output methods*

No.	Data output method	Description
1	File output	This method uses a <i>file output connector</i> to output the results of stream data summary analysis to files. For details about file output, see <i>10.6 File output</i> .
2	Dashboard output	This method uses a <i>dashboard output connector</i> to output the results of stream data summary analysis and to view the data using Dashboard Viewer of Flex Dashboard. You can edit the dashboard window displayed by Dashboard Viewer, such as the chart display layout. For details about dashboard output, see <i>10.7 Dashboard output</i> .

Note that you specify data output in the CB definition for output in the adaptor configuration definition file. For details about the CB definition in the adaptor configuration definition file, see *9. Adaptor Definition Files*.

---

## 2.6 Evaluating the summary analysis scenarios

---

This section discusses the evaluation of scenarios used for summarizing and analyzing stream data. It also discusses the configuration of the working directories for running the query groups that define scenarios.

### 2.6.1 Evaluating the stream data summary analysis scenarios

In the stream data processing system, you define as query groups the stream data summary analysis scenarios that specify the types of data to be input and how the data is to be processed and output.

A query group consists of input streams, queries, and an output stream. You must evaluate whether to define multiple query groups or to define multiple streams and queries in a single query group according to the stream data summary analysis processing to be performed.

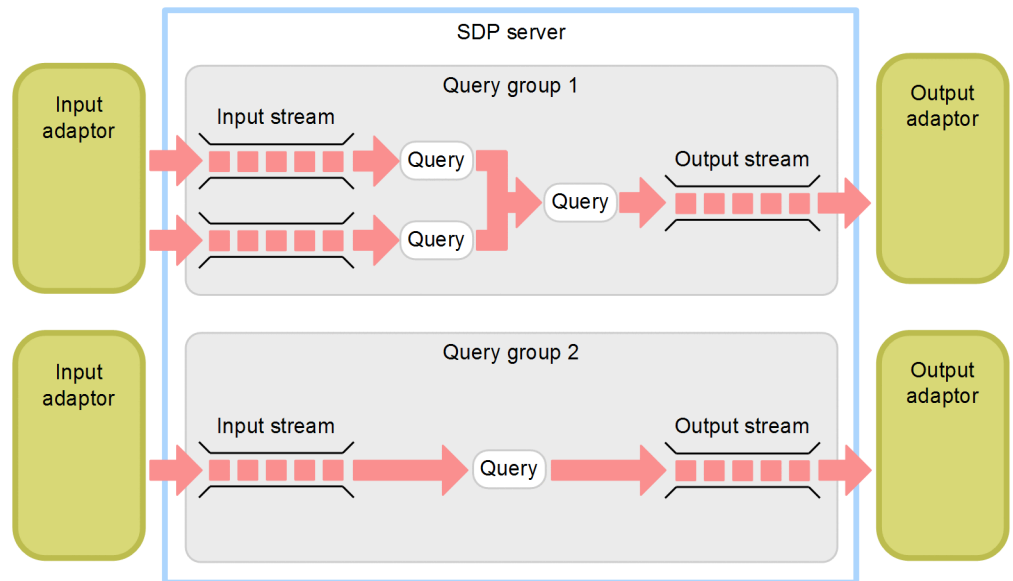
You define as many query groups as there are summary analysis scenarios. By executing multiple query groups, you can use a single SDP server to perform multiple summary analysis processes. One SDP server can execute a maximum of 8 query groups.

You define the input and output streams according to the configuration of the corresponding input and output adaptors.

A query definition depends on the type of summary analysis processing that is performed in a query group. If you wish to have a single summary analysis scenario perform multiple summary analysis processes, you can define multiple queries in a query group.

The following figure shows an example configuration of query groups.

Figure 2-12: Example configuration of query groups



In this example, two query groups (query groups 1 and 2) are registered in the SDP server, and each query group is connected to an input adaptor and an output adaptor. Query group 1 defines multiple queries to perform multiple summary analysis processes.

### 2.6.2 Evaluating the configuration of the working directories for executing query groups

If you execute multiple query groups, you must evaluate the number of working directories to be used to execute the query groups.

To execute multiple query groups, the working directories and query groups can be configured to have a 1-to-1 correspondence or a 1-to- $n$  correspondence. The table below describes each configuration.

Table 2-6: Configurations of query groups and working directories

No.	Configuration	Description
1	Configuration in which there is a 1-to-1 correspondence between a working directory and a query group	<p>In this configuration, you create one working directory for each query group and execute one query group in one working directory.</p> <p>This configuration has the following advantages:</p> <ul style="list-style-type: none"> <li>• Performance is improved because memory space is divided among servers, thereby reducing the frequency of garbage collection.</li> <li>• Failures can be localized because a failure in one SDP server has no effect on query execution by other SDP servers.</li> </ul> <p>Note that because you run only one SDP server per working directory, with this configuration you must have as many SDP servers as there are query groups.</p>
2	Configuration in which there is a 1-to- <i>n</i> correspondence between a working directory and multiple query groups	<p>In this configuration, you create only one working directory and execute multiple query groups in that working directory.</p> <p>This configuration has the following advantages:</p> <ul style="list-style-type: none"> <li>• You only have to configure an operating environment once, which involves creation of the working directory, definition files, and adaptors.</li> <li>• Multiple query groups can be executed with fewer resources, compared with the configuration in which each query group is executed in its own working directory.</li> </ul>

Note that if the memory requirement for a single SDP server exceeds 1.6 gigabytes, we recommend that you use the 1-to-1 configuration in which there is a working directory for each query group.

If you execute nine or more query groups, determine the number of working directories to be used to execute the query groups (creating as many 1:1 configurations as there are query groups or creating multiple 1: *n* configurations) by referencing 2.7 *Estimating the memory requirements*.

## 2.7 Estimating the memory requirements

This section discusses how to estimate the memory requirements for the stream data processing engine and standard adaptors. It also presents examples of estimating memory requirements.

The table below lists and describes the items for which memory requirements must be estimated and the subsections that describe how to estimate them.

*Table 2-7:* Items for which memory requirements must be estimated and the subsections that describe how to estimate them

No.	Item for which memory requirements must be estimated	Subsection
1	Stream data processing engine	Running an SDP server
2		Tuples
3		Retaining tuples
4		Changing the maximum number of elements to be used in a stream queue
5		Changing the API trace buffers
6		Acquiring tuple logs
7		Executing query groups
8		Timestamp adjustment
9	Standard adaptors	Running adaptors
10		Adaptor as an in-process group
11		Adaptor as an RMI group
12		Record processing (when a file input connector is used)
13		Record processing (when an HTTP packet input connector is used)
14		Record processing (when a file output connector or dashboard output connector is used)
15		Record accumulation (when record extraction or dashboard output connector is used)

### 2.7.1 Estimating the memory requirements for the stream data processing engine

This subsection explains how to estimate the memory requirements for the stream data processing engine.

To specify the memory requirements for the stream data processing engine, you use parameters in the JavaVM options file for SDP servers (`jvm_options.cfg`), one of the SDP server definition files, as shown in the table below.

*Table 2-8: Parameters used to specify the memory requirements for the stream data processing engine*

No.	Parameter	Description
1	SDP_INITIAL_MEM_SIZE	Specifies the initial size for the Java heap. The default value is 512 megabytes.
2	SDP_MAX_MEM_SIZE	Specifies the maximum size of the Java heap. The default value is 1,024 megabytes.

If the default memory size is not sufficient, change the parameter values as appropriate. For stream data processing, a fixed amount of memory is required for the stream data processing engine, as described below. To reduce the frequency of garbage collection, specify a sufficient value for each parameter based on the values obtained below.

For details about the parameters in the JavaVM options file for SDP servers, see *8.4 JavaVM options file for SDP servers (jvm\_options.cfg)*.

The following subsections explain each of the stream data processing engine memory requirement items.

#### (1) Memory required for running an SDP server

The memory required for running an SDP server is 145 megabytes.

#### (2) Memory required for one tuple

The memory required for one tuple is determined by the *size of one tuple*. You can use the following formula to determine the size of one tuple:

$\text{Size of one tuple (bytes)} = 500^{\#} + \text{size of user-defined data area}$
---

#

This is the size of a fixed area for a tuple (bytes).

In the formula above, *size of user-defined data area* is determined by the data types and number of data items defined by the user in the CQL schema. You can use the



following formula to determine the size of the user-defined data area:

$$\text{Size of user-defined data area (bytes) = } (256^{\#} \times \text{number of user-defined data items}) + (\text{length of the character string defined as string data} \times 2)$$

#

This is the size per data item (bytes).

### (3) Memory required for retaining tuples

As the sizes of the windows used in window operations increase, the number of tuples held by the SDP server increases, and the amount of memory required for retaining the tuples also increases. The size of a window is specified by the number of tuples (for ROWS window) or time (for RANGE window).

You can use the formulas below to determine the amount of memory required when window operations are used. In the case of a RANGE window, the formula depends on whether or not the time division function is specified.

For a ROWS window:

$$\text{Size of memory required (bytes) = } A \times \text{number of tuples specified in the ROWS window} + 50,000^{\#}$$

For a RANGE window (when time division function is not specified):

$$\text{Size of memory required (bytes) = } A \times \text{time specified in the RANGE window} \times \text{number of tuples input per unit of time} + 50,000^{\#}$$

For a RANGE window (when time division function is specified):

$$\text{Size of memory required (bytes) = } A \times \text{time specified in the RANGE window} / \text{mesh interval specified in the time division function} \times \text{number of tuples input per mesh interval} + 50,000^{\#}$$

Legend:

*A*: Size per tuple shown in (2) *Memory required for one tuple*

#

Size of a fixed area for the window (bytes)

Specification of the time division function is useful when it is difficult to predict the frequency of data input or when you want to limit the size of the available memory. Note that there are prerequisites and limitations to the specification of the time division function. For details about specification of the time division function, see the

**(4) Memory required for changing the maximum number of elements to be used in a stream queue**

You can change the maximum number of elements in a stream queue by using the `engine.maxQueueSize` parameter in the system configuration property file (`system_config.properties`), one of the SDP server definition files. Changing the maximum number of elements affects the size of memory used by the stream queue. If you change the maximum number of elements, you can use the following formula to determine the amount of memory required per stream queue:

Memory required per stream queue (bytes) = $A \times B$
---

Legend:

*A*: Value of the `engine.maxQueueSize` parameter (maximum number of elements in a stream queue)

*B*: Size per tuple shown in (2) *Memory required for one tuple*

For details about the parameters in the system configuration property file, see 8.6 *System configuration property file (system\_config.properties)*.

**(5) Memory required for changing the API trace buffers**

You can change the API trace buffers by using the following parameters in the system configuration property file (`system_config.properties`), one of the SDP server definition files:

- `trc.api.bufferSize` (specifies the maximum size API trace buffer size)
- `trc.api.bufferCount` (specifies the number of API trace buffer sectors)
- `trc.api.ioBufferSize` (specifies the maximum size of the API trace I/O buffer)

Changing any of these parameters affects the size of the memory used for API trace information. You can use the following formula to determine the memory used for API trace information when the API trace buffer are changed:

Memory required for API trace information (kilobytes) = $A \times B + C$
--

Legend:

*A*: Value of the `trc.api.bufferSize` parameter (maximum API trace buffer size)

*B*: Value of the `trc.api.bufferCount` parameter (number of API trace buffer sectors)

C: Value of the `trc.api.ioBufferSize` parameter (maximum size of API trace I/O buffer)

For details about the parameters in the system configuration property file, see 8.6 *System configuration property file (system\_config.properties)*.

### (6) Memory required for acquiring tuple logs

You can change the timing of outputting tuple logs to files by using the `tpl.outputTrigger` parameter in the system configuration property file (`system_config.properties`), one of the SDP server definition files.

Specifying `BUFFER` (default value) as the timing of outputting tuple logs to files in the `tpl.outputTrigger` parameter affects the size of the memory required for acquiring tuple logs per stream.

You can use the formula shown below to determine the size of the memory required for acquiring tuple logs per stream when you specify `BUFFER` as the timing of outputting tuple logs to files:

$\text{Memory required for acquiring tuple logs per stream (kilobytes)} = A \times B$
---

Legend:

A: Value of the `tpl.bufferSize` parameter in the system configuration property file (maximum tuple log buffer size)

B: Value of the `tpl.bufferCount` parameter in the system configuration property file (number of tuple log buffer sectors)

For details about the parameters in the system configuration property file, see 8.6 *System configuration property file (system\_config.properties)*.

### (7) Memory required for executing query groups

You can use the following formula to determine the size of the memory required per query group:

$\text{Memory required per query group (megabytes)} = 2^{\#1} \times \text{number of queries contained in the registered query group} \times 128^{\#2}$
---

#1

Size of memory required per query (megabytes)

#2

Size of query group execution area (megabytes)

### (8) Memory required for timestamp adjustment

You can use the following formula to determine the size of the memory required for

using the timestamp adjustment function:

$$\text{Memory required for timestamp adjustment (bytes)} = \text{number of input streams} \times \text{number of tuples per unit of time} \times \text{time adjustment range} \times A$$

Legend:

*A*: Size per tuple shown in (2) *Memory required for one tuple*

The values of *number of tuples per unit of time* and *time adjustment range* in the formula depend on the value of the `stream.timestampAccuracy` parameter in the system configuration property file (`system_config.properties`), query group property file, or stream property file, all of which are SDP server definition files. The table below shows the relationship between the `stream.timestampAccuracy` parameter value and the value in the formula.

<b>stream.timestampAccuracy parameter value</b>	<b>Number of tuples per unit of time</b>	<b>Time adjustment range</b>
unuse	0	0
Other than unuse	Number of tuples per unit time	Time adjustment range <sup>#</sup> + 1

#

This is the `stream.timestampAccuracy` parameter value.

For details about the parameters in each file, see 8.6 *System configuration property file* (`system_config.properties`), 8.7 *Query group property file*, and 8.8 *Stream property file*.

### 2.7.2 Estimating the memory requirements for standard adaptors

This subsection explains how to estimate the memory requirements for the standard adaptors.

To specify the memory requirements for the standard adaptors, you use parameters in the JavaVM options file for SDP servers (`jvm_options.cfg`), one of the SDP server definition files, or the JavaVM options file for RMI connections (`jvm_client_options.cfg`), as shown in the table below.

*Table 2-9:* Parameters used to specify the memory requirements for the standard adaptors (JavaVM options file for SDP servers (`jvm_options.cfg`))

<b>No.</b>	<b>Parameter</b>	<b>Description</b>
1	<code>SDP_INITIAL_MEM_SIZE</code>	Specifies the initial size for the Java heap. The default value is 512 megabytes.

No.	Parameter	Description
2	SDP_MAX_MEM_SIZE	Specifies the maximum size of the Java heap. The default value is 1,024 megabytes.

*Table 2-10:* Parameters used to specify the memory requirement for the standard adaptors (JavaVM options file for RMI connections (jvm\_client\_options.cfg))

No.	Parameter	Description
1	SDP_INITIAL_MEM_SIZE	Specifies the initial size for the Java heap. The default value is 256 megabytes.
2	SDP_MAX_MEM_SIZE	Specifies the maximum size of the Java heap. The default value is 512 megabytes.

If the default memory size is not sufficient, change the parameter values as appropriate. To run the standard adaptors, you need the amount of memory estimated for the stream data processing engine in *2.7.1 Estimating the memory requirements for the stream data processing engine* plus a fixed amount of memory for the standard adaptors, as described below. To reduce the frequency of garbage collection, specify a sufficient value for each parameter based on the values obtained below.

For details about the parameters in each file, see *8.4 JavaVM options file for SDP servers (jvm\_options.cfg)* or *8.5 JavaVM options file for RMI connections (jvm\_client\_options.cfg)*.

The following subsections explain each of the standard adaptor memory requirement items.

**(1) Memory required for running adaptors**

The memory required for running adaptors is 5 megabytes (fixed).

**(2) Memory required for one adaptor configured as an in-process group**

The memory required for one adaptor configured as an in-process group is 10 megabytes (fixed).

**(3) Memory required for one adaptor configured as an RMI group**

The memory required for one adaptor configured as an RMI group is 20 megabytes (fixed).

**(4) Memory required for record processing (when a file input connector is used)**

When a file input connector is used, you can use the following formula to determine the size of the memory required for record processing:

Memory required for record processing (bytes) = <i>(memory required for processing one record x number of records processed at one time)</i> + <i>(A x number of records processed at one time)</i>
---

Legend:

*A*: Size per tuple shown in 2.7.1(2) *Memory required for one tuple*

#### Memory required for processing one record

In the formula above, the *memory required for processing one record* can be obtained from the following formula:

Memory required for processing one record (bytes) = <i>(500 x number of field definitions in the format conversion definition<sup>#</sup>)</i> + <i>(maximum length of common record)</i>
---

#

This is the number of field definitions (`field` tags) in the format conversion definition (`FormatDefinition` tags) in the adaptor configuration definition file, one of the adaptor definition files.

#### Number of records processed at one time

In the above format, the *number of records processed at one time* can be obtained from the following formula:

Number of records processed at one time (bytes) = <i>number of records read by a file input connector at one time<sup>#</sup></i>
--

#

This is the `readUnit` attribute value in the input definition (`input` tag) in the file input connector definition (`FileInputConnectorDefinition`) in the adaptor configuration definition file, one of the adaptor definition files.

For details about the definitions in the adaptor configuration definition file, see 9.5 *Adaptor configuration definition file (AdaptorCompositionDefinition.xml)*.

### **(5) Memory required for record processing (when an HTTP packet input connector is used)**

If you use an HTTP packet input connector, you can use the following formula to determine the size of memory required for record processing:

Memory required for record processing (bytes) = <i>(memory required for processing one record x number of records processed at one time)</i> + <i>(A x number of records processed at one time)</i>
---

Legend:

*A*: Size per tuple shown in 2.7.1(2) *Memory required for one tuple*

Memory required for processing one record

In the formula above, *memory required for processing one record* can be obtained from the following formula:

<p>Memory required for processing one record (bytes) =  <i>(500 x number of field definitions in the HTTP packet input connector definition<sup>#</sup>)</i>  <i>+ (maximum length of common record)</i></p>
--

#

This is the number of field definitions (`field` tags) in the HTTP packet input connector definition (`HttpPacketInputConnectorDefinition` tag) in the adaptor configuration definition file, one of the adaptor definition files.

Number of records processed at one time:

In the formula above, *number of records processed at one time* can be obtained from the following formula:

<p>Number of records processed at one time (bytes) =  <i>number of records that are read by the HTTP packet input connector at one time<sup>#</sup></i></p>
---

#

This is the `unit` attribute value in the output definition (`output` tag) in the HTTP packet input connector definition (`HttpPacketInputConnectorDefinition` tag) in the adaptor configuration definition file, one of the adaptor definition files.

For details about the definitions in the adaptor configuration definition file, see 9.5 *Adaptor configuration definition file (AdaptorCompositionDefinition.xml)*.

### **(6) Memory required for record processing (when a file output connector or dashboard output connector is used)**

If you use a file output connector or a dashboard output connector, you can use the following formula to determine the amount of memory required for record processing:

<p>Memory required for record processing (bytes) =  <i>(memory required for processing one record x number of records processed at one time)</i>  <i>+ (A x number of records processed at one time)</i></p>
--

Legend:

*A*: Size per tuple shown in 2.7.1(2) *Memory required for one tuple*

### Memory required for processing one record

In the formula above, *memory required for processing one record* can be obtained from the following formula:

<p>Memory required for processing one record (bytes) =  <math>(500 \times \text{number of mapping information definitions in the mapping definition}^{\#})</math>  <math>+ (\text{maximum length of common record})</math></p>
--

#

This is the number of mapping information definitions (map tags) of mappings between records and streams in the mapping definition (`MappingDefinition` tag) of the adaptor configuration definition file, one of the adaptor definition files.

For details about the definitions in the adaptor configuration definition file, see *9.5 Adaptor configuration definition file (AdaptorCompositionDefinition.xml)*.

### Number of records processed at one time

In the formula above, *number of records processed at one time* is 1,024 (maximum number of tuples that can be received).

### **(7) Memory required for record accumulation (when record extraction or dashboard output connector is used)**

If you use record extraction or a dashboard output connector, you can use the following formula to determine the memory required for record accumulation:

<p>Memory required for record accumulation (bytes) =  <math>(\text{memory required for processing one record} \times \text{number of records to be accumulated})</math></p>
---

### Memory required for processing one record

In formula above, *memory required for processing one record* depends on the connector being used, as described below.

- When a file input connector is used to perform record extraction  
 See *memory required for processing one record* in (4) *Memory required for record processing (when a file input connector is used)*.
- When an HTTP packet input connector is used to perform record extraction  
 See *memory required for processing one record* in (5) *Memory required for record processing (when an HTTP packet input connector is used)*.
- When a dashboard output connector is used  
 See *memory required for processing one record* in (6) *Memory required for record processing (when a file output connector or dashboard output connector is used)*.



### Number of records to be accumulated

In the formula above, *number of records to be accumulated* depends on whether record extraction or a dashboard output connector is used, as described below.

- When record extraction is used  
Use the `size` attribute value in the extraction condition group definition (`extractions` tag) in the record extraction definition (`RecordExtractionDefinition` tag) in the adaptor configuration definition file, one of the adaptor definition files.
- When a dashboard output connector is used  
Use the `MaxNum` attribute value in the dashboard output connector definition (`DashboardOutputConnectorDefinition` tag) in the adaptor configuration definition file, one of the adaptor definition files.

For details about the definitions in the adaptor configuration definition file, see [9.5 Adaptor configuration definition file \(\*AdaptorCompositionDefinition.xml\*\)](#).



## Chapter

---

# 3. System Configuration

---

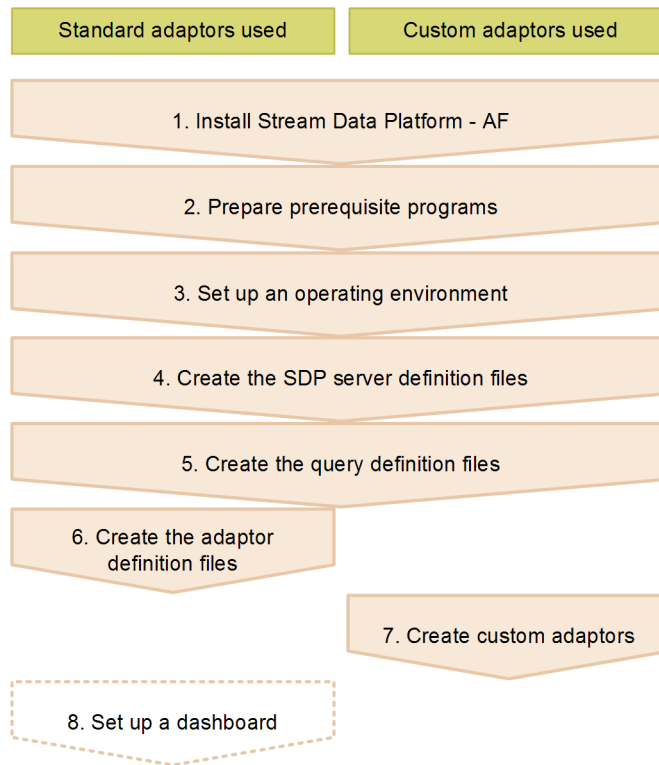
This chapter describes the flow of configuring a stream data processing system, the structure of the Stream Data Platform - AF directories, and the settings required for system configuration.

- 3.1 Flow of configuring
- 3.2 Directory structure
- 3.3 Setting up an operating environment
- 3.4 Creating the SDP server definition files
- 3.5 Creating the query definition files
- 3.6 Creating the adaptor definition files
- 3.7 Setting up the dashboard

## 3.1 Flow of configuring

The procedure for configuring a stream data processing system depends on the mode of connection between the adaptors and the SDP server and the functions used by the adaptors. The following figure shows a flow of system configuration.

*Figure 3-1:* Flow of system configuration



Legend:

 : Required step  : Optional step

The following describes the steps.

### 1. Installing Stream Data Platform - AF

Install Stream Data Platform - AF. For details about the installation procedure, see the Release Notes for Stream Data Platform - AF.

### 2. Preparing prerequisite programs

Prerequisite programs are required in order to input HTTP packets and output data to a dashboard. For details about the prerequisite programs, see *2.2.1 Program configuration in a stream data processing system*.

Install and set up the prerequisite programs by referencing each program's documentation.

3. Setting up an operating environment
 

Register an administrator user and create the working directories.

For details about the setup method, see *3.3 Setting up an operating environment*.
4. Creating the SDP server definition files
 

Create the SDP server definition files and set up the SDP server operation.

For details about how to create these files, see *3.4 Creating the SDP server definition files*.
5. Creating the query definition files
 

Create the query definition files and define the stream data summary analysis scenarios.

For details about how to create these files, see *3.5 Creating the query definition files*.
6. Creating the adaptor definition files
 

If you use the standard adaptors, you must create the adaptor definition files and set up the operation of the standard adaptors. If you use custom adaptors, there is no need to create the adaptor definition files.

For details about how to create these files, see *3.6 Creating the adaptor definition files*.
7. Creating custom adaptors
 

Create custom adaptors using the data communication API provided by Stream Data Platform - AF. If you use the standard adaptors, there is no need to create custom adaptors.

For details about how to create custom adaptors, see the manual *uCosminexus Stream Data Platform - Application Framework Application Development Guide*.
8. Setting up a dashboard
 

If you use the standard adaptors to output the stream data summary analysis results to a dashboard, you must set up the dashboard. If you do not output the results to a dashboard, there is no need to perform this setup.

For details about the setup method, see *3.7 Setting up the dashboard*.

### 3. System Configuration

For details about how to change the settings of a system that has already been configured, see *5. Modifying the System*.

---

## 3.2 Directory structure

---

This section discusses the structure of the installation directory and the working directories for Stream Data Platform - AF.

*Reference note:*

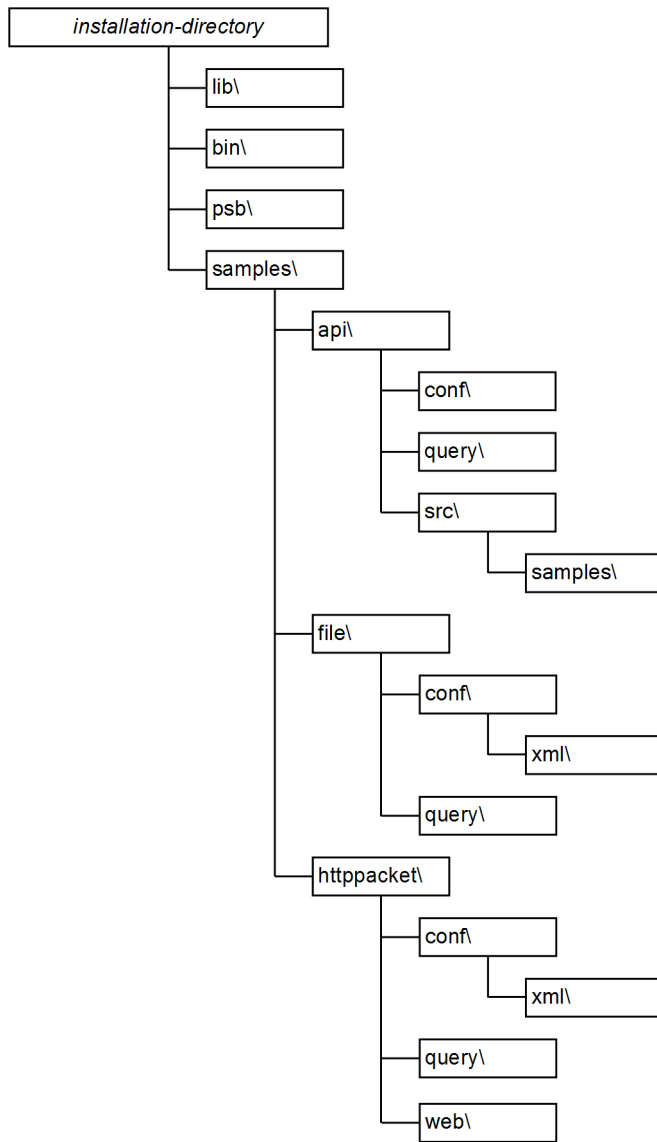
When you install Stream Data Platform - AF, the directory for the Hitachi trace common library (default installation directory is *OS-installation-drive:\Program Files\Hitachi\HNTRLib2\*) is also created. This manual does not discuss this directory because it will not be used by the user.

### 3.2.1 Structure of the installation directory

This subsection describes the installation directory for Stream Data Platform - AF.

The following figure shows the structure of the Stream Data Platform - AF installation directory.

Figure 3-2: Structure of the installation directory





*Reference note:*

- To specify the installation directory, you can also use the variable `%SDP_INSTALL_DIR%` which represents the installation directory.
- The default installation directory is *OS-installation-drive*: `\Program Files\Hitachi\SDP`.

The table below lists and describes the directories in the installation directory. For details about the files in the directories, see the sections indicated in the table.

*Table 3-1:* Descriptions of directories in the installation directory

Directory		Description	Section	
lib\		Directory for storing the Stream Data Platform - AF libraries	--	
bin\		Directory for storing the <code>sdpsetup</code> command that is used when a working directory is created. Once you have created a working directory, execute the commands under <i>working-directory</i> \bin\.	Chapter 7	
psb\		Directory for storing the Web server used by Stream Data Platform - AF	--	
samples\	--	Directory for storing sample files. Three sets of sample files are provided for different types of adaptors and callbacks. They are stored in the directories <code>api</code> , <code>file</code> , and <code>httppacket</code> .	--	
	api\	--	Directory for storing sample files for using custom adaptors	
		conf\	Directory for storing sample files for the SDP server definition files	3.4, Chapter 8
		query\	Directory for storing sample files for the query definition files	3.5
		src\	samples\	Directory for storing sample programs for custom adaptors
file\	--	Directory for storing sample files for enabling the standard adaptors to use the following callbacks: <ul style="list-style-type: none"> <li>• File input connector</li> <li>• File output connector</li> </ul>	--	

Directory			Description	Section	
		conf\ --	Directory for storing sample files for the SDP server definition files	3.4, Chapter 8	
		xml\ --	Directory for storing sample files for the adaptor definition files	3.6, Chapter 9	
		query\ --	Directory for storing sample files for the query definition files	3.5	
	httppacket\ --	--		Directory for storing sample files for enabling the standard adaptors to use the following callbacks: <ul style="list-style-type: none"> <li>• HTTP packet input connector</li> <li>• Filter</li> <li>• Record extraction</li> <li>• Dashboard output connector</li> </ul>	--
		conf\ --	Directory for storing sample files for the SDP server definition files	3.4, Chapter 8	
		xml\ --	Directory for storing sample files for the adaptor definition files	3.6, Chapter 9	
		query\ --	Directory for storing sample files for the query definition files	3.5	
	web\ --	Directory for storing sample files for using the dashboard output connector's callback	3.7		

Legend:

--: Not applicable.

#

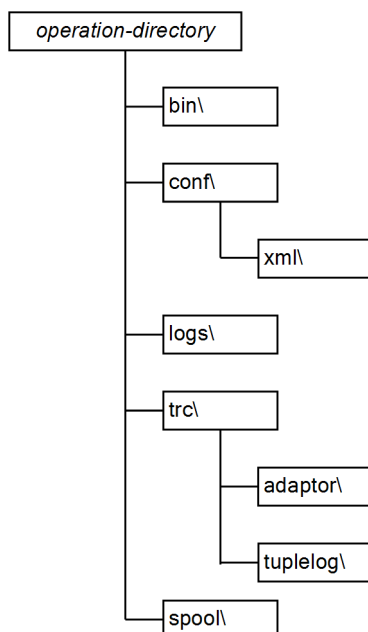
For details about these sample programs, see the manual *uCosminexus Stream Data Platform - Application Framework Application Development Guide*.

### 3.2.2 Structure of the working directories

A *working directory* is used for the operation of the stream data processing system. The user creates operation directories after installing Stream Data Platform - AF. For details about how to create working directories, see 3.3.2 *Creating a working directory*.

The following figure shows the structure of a working directory.

Figure 3-3: Structure of a working directory



The table below lists and describes the directories in a working directory. For details about the files in the directories, see the sections indicated in the table.

Table 3-2: Descriptions of directories in a working directory

Directory		Description	Section
bin\		Directory for storing the commands used during operation	Chapter 7
conf\	--	Directory for storing the SDP server definition files. Store in this directory the SDP server definition files created by the user.	3.4, Chapter 8
	xml\	Directory for storing the adaptor definition files. Store in this directory the adaptor definition files created by the user.	3.6, Chapter 9
logs\		Directory to which log files are output	6.3.1
trc\	--	Directory to which various traces (such as API traces) are output	6.3.2
	adaptor\	Directory to which adaptor trace information is output	6.3.3
	tuplelog\	Directory to which tuple logs are output	6.3.4
spool\		Work directory for Stream Data Platform - AF	--

### 3. System Configuration

#### Legend:

--: Not applicable.

---

## 3.3 Setting up an operating environment

---

After you have installed Stream Data Platform - AF, you must register an administrator user and then create the working directories.

### 3.3.1 Registering an administrator user

An *administrator user* configures and runs the stream data processing system. This user has the necessary permissions to execute the Stream Data Platform - AF commands and is the owner of the working directories.

Add the user account of the administrator user by choosing **User Accounts** from **Control Panel**. For details about how to add user accounts, see Windows Help.

### 3.3.2 Creating a working directory

A *working directory* is used to configure and run the stream data processing system. An administrator user registered in 3.3.1 *Registering an administrator user* creates a working directory by executing the `sdpsetup` command.

```
installation-directory\bin\sdpsetup working-directory
```

For *working-directory*, specify the absolute path of the working directory.

For a working directory, set the Full Control permissions for all administrator users. Note that the Stream Data Platform - AF installation directory cannot be a working directory.

If you execute multiple query groups, you may create as many working directories as there are query groups or you may execute multiple query groups using a single working directory. For the advantages and disadvantages of each method, see 2.6.2 *Evaluating the configuration of the working directories for executing query groups*.

For details about the `sdpsetup` command, see *sdpsetup (sets up an operating environment)* in 7. *Commands*.

For details about the structure of a working directory, see 3.2.2 *Structure of the working directories*.

## 3.4 Creating the SDP server definition files

You must create the SDP server definition files and set up an operating environment for the SDP server.

### 3.4.1 Settings that can be specified in the SDP server definition files

The settings to be specified in the SDP server definition files include the JavaVM start options for executing the SDP server and adaptors, SDP server port number, and details of API trace information and tuple logs to be collected. The properties of query groups and streams are also specified.

The required files depend on the mode of connection between adaptors and SDP server and the units of property settings. The table below shows whether or not each SDP server definition file needs to be created.

*Table 3-3: Whether or not the SDP server definition files need to be created*

No.	SDP server definition file	Whether or not necessary to create	
1	JavaVM options file for SDP servers ( <code>jvm_options.cfg</code> )	Y	You must create this file for each working directory. Specify in this file the JavaVM options for running the SDP server. Note that in-process-connection adaptors operate according to the options specified in this file.
2	JavaVM options file for RMI connections ( <code>jvm_client_options.cfg</code> )	O	If you use the RMI connection mode, you must create this file for each working directory or each adaptor. Specify in this file the JavaVM options for running the RMI-connection adaptors.

No.	SDP server definition file	Whether or not necessary to create	
3	System configuration property file ( <i>system_config.properties</i> )	Y	You must create this file for each working directory. Specify in this file such settings as the port number used by the SDP server and the details of API trace information and tuple logs to be collected.
4	Query group property file	Y	You must create this file for each query group. Specify in this file the paths of the query definition files and the tuning parameters used for executing the query group.
5	Stream property file	O	If you wish to specify tuning parameters for a stream in a query group, you must create this file for that stream. If you do not create this file, the settings in the query group property file are used.
6	In-process connection property file ( <i>user_app.adaptor-group-name-or-adaptor-name.properties</i> )	O	When you use in-process connection, you must create this file for each adaptor group if you use the standard adaptors, and for each adaptor if you use custom adaptors. Specify in this file the class names of the in-process connection adaptors and the paths of the jar files.

No.	SDP server definition file	Whether or not necessary to create	
7	Log file output property file ( <code>logger.properties</code> )	O	You must create this file for each working directory. If you do not create this file, the default values are used. Specify in this file the numbers and sizes of message logs and trace logs.

Legend:

Y: File creation is mandatory

O: File creation is optional

For details about the SDP server definition files, see 8. *SDP Server Definition Files*.

### 3.4.2 How to create the SDP server definition files

Except for the stream property file, we recommend that you create the SDP server definition files by using the sample files provided during installation. Copy the sample files from the directory shown below and then edit the sample files.

- Sample files for the standard adaptors

*installation-directory*\samples\file-or-httppacket#\conf\

- Sample files for custom adaptors

*installation-directory*\samples\api\conf\

#

For details about the directories for storing the sample files, see 3.2.1 *Structure of the installation directory*.

Note that the values set in the sample files might differ from the default values for the files discussed in 8. *SDP Server Definition Files*.

The storage location for each created SDP server definition file depends on the file. For details about the storage locations for the SDP server definition files, see 8. *SDP Server Definition Files*.



---

## 3.5 Creating the query definition files

---

You must create the query definition files and define stream data summary analysis scenarios.

### 3.5.1 Settings that can be specified in the query definition files

The settings to be specified in the query definition files include registration of streams and queries in the stream data processing system and operations on the stream data. You must create the query definition files for each query group.

You specify the query definition files in CQL. For details about CQL and how to create the query definition files, see the manual *uCosminexus Stream Data Platform - Application Framework Application Development Guide*.

### 3.5.2 How to create the query definition files

We recommend that you create the query definition files by using the sample files provided during installation. Copy the sample files from the directory shown below and then edit the sample files.

- Sample files for the standard adaptors  
*installation-directory*\samples\file-or-httppacket#\query\
- Sample files for custom adaptors  
*installation-directory*\samples\api\query\  
#

For details about the directories for storing the sample files, see *3.2.1 Structure of the installation directory*.

For details about the sample files for the query definition files, see the manual *uCosminexus Stream Data Platform - Application Framework Application Development Guide*.

Store the created query definition files at any desired location and then specify the file path and file name in the `querygroup.cqlFilePath` parameter in the query group property file. For details about the query group property file, see *8.7 Query group property file*.

## 3.6 Creating the adaptor definition files

When you use the standard adaptors, you must create the adaptor definition files and define the operation of the standard adaptors. If you use custom adaptors, there is no need to create the adaptor definition files.

### 3.6.1 Settings that can be specified in the adaptor definition files

The settings to be specified in the adaptor definition files include the port number used by the RMI connection adaptors, configuration of adaptor groups, and callback processing performed by the standard adaptors.

The required files depend on the mode of connection between the adaptors and the SDP server. The table below shows whether or not each adaptor definition file needs to be created.

*Table 3-4: Whether or not the adaptor definition files need to be created*

No.	Adaptor definition file	Whether or not necessary to create	
1	Adaptor command definition file (AgentManagerDefinition.xml)	O	If you use the RMI connection mode, you must create this file for each working directory. Specify in this file the port numbers used by the RMI-connection adaptors.
2	Adaptor configuration definition file (AdaptorCompositionDefinition.xml)	Y	You must create this file for each working directory. Specify in this file settings such as the configuration of adaptor groups and the callback processing to be executed by the standard adaptors.

Legend:

Y: File creation is mandatory

O: File creation is optional

For details about the adaptor definition files, see *9. Adaptor Definition Files*.

### 3.6.2 How to create the adaptor definition files

Create the adaptor definition files by referencing the sample files presented in *9. Adaptor Definition Files*. We recommend that you create the adaptor definition files by using the sample files provided during installation. Copy the sample files from the directory shown below and then edit the sample files.

`installation-directory\samples\file-or-httppacket#\conf\xml\`

#

For details about the directories for storing the sample files, see *3.2.1 Structure of the installation directory*.

Note that the values set in the sample files might differ from the default values for the files discussed in *9. Adaptor Definition Files*.

Store the created adaptor definition files in the following directory:  
*working-directory*\conf\xml\

---

## 3.7 Setting up the dashboard

---

When you use the standard adaptors to output stream data summary analysis results to a dashboard, you must set up the dashboard. If you do not output summary analysis results to a dashboard, there is no need to perform dashboard setup.

To output data to a dashboard, you must specify the settings that enable the stream data summary analysis results to be output as dashboard-display data. To display the output dashboard-display data on Flex Dashboard, you must set up Flex Dashboard's Dashboard Server and Dashboard Viewer. This section describes how to set up each of them.

For details about output to a dashboard, see *10.7 Dashboard output*.

### 3.7.1 Output settings for dashboard-display data

You specify the output settings for dashboard-display data in the adaptor configuration definition file, one of the adaptor definition files. Use the output adaptor to specify the dashboard output connector definition.

For details about the creation of an adaptor configuration definition file, see *3.6 Creating the adaptor definition files* and *9. Adaptor Definition Files*.

### 3.7.2 Setting up Dashboard Server

Dashboard Server is a Flex Dashboard application for acquiring dashboard-display data output by an output connector. Dashboard Server accepts requests from Dashboard Viewer and sends the dashboard-display data acquired from the dashboard output connector to Dashboard Viewer.

This subsection describes how to set up Dashboard Server.

#### (1) *Registering Dashboard Server*

To use Dashboard Server, you must first set up the Web server and the Web container server. The setup methods for these two servers are described below.

- Setting up the Web server

You execute the command shown below to register the Web server (Hitachi Web Server) as a Windows service. The service name to be registered is `Hitachi Web Server for uCSDPAF` (fixed value).

```
installation-directory\psb\httpsd\httpsd.exe -k install -n "Hitachi Web Server for uCSDPAF"
```

- Setting up the Web container server

You execute the command shown below to set up the Web container server that

runs Dashboard Server. The server name to be set up is uCSDPAF\_Server (fixed value).

```
installation-directory\psb\CC\web\bin\cjwebsetup.exe uCSDPAF_Server
```

When setup is completed, the following directory is created:  
*installation-directory\psb\CC\web\containers\uCSDPAF\_Server\*

The following shows an example of the commands that set up the Web server and Web container server:

```
C:\> installation-directory\psb\httpsd\httpsd.exe -k install -n "Hitachi Web Server for
uCSDPAF"

Installing the Hitachi Web Server for uCSDPAF service
The Hitachi Web Server for uCSDPAF service is successfully installed.
C:\> installation-directory\psb\CC\web\bin\cjwebsetup.exe uCSDPAF_Server

KDJE41800-I The setup for the Web container server has finished successfully. Server
name = uCSDPAF_Server
```

## (2) Specifying an environment variable

Specify the following environment variable:

- Name of the environment variable

PRFSPOOL

- Setting

*installation-directory\psb\CC\web\redirector\logs*

## (3) Preparing the files to be used by Dashboard Server

Prepare the files that are to be used by Dashboard Server.

- Editing a file used by Dashboard Server

Edit the following file:

- File to be edited

*installation-directory\psb\httpsd\conf\httpsd.conf*

- Editing

Add the following line at the end of the file:

```
Include ..\CC\web\redirector\mod_jk.conf
```

- Copying and editing the files used by Dashboard Server

Copy each source file to its target directory, as shown in the table below.

Source file	Target directory	Whether or not file editing is required after copying
<i>installation-directory</i> \samples\httppacket\web\redirector\workers.properties	<i>installation-directory</i> \psb\CC\web\redirector\	N
<i>installation-directory</i> \samples\httppacket\web\redirector\mod_jk.conf	<i>installation-directory</i> \psb\CC\web\redirector\	N
<i>installation-directory</i> \samples\httppacket\web\containers\uCSDPAF_Server\usrconf\usrconf.cfg	<i>installation-directory</i> \psb\CC\web\containers\uCSDPAF_Server\usrconf\	N
<i>installation-directory</i> \samples\httppacket\web\containers\uCSDPAF_Server\usrconf\usrconf.properties	<i>installation-directory</i> \psb\CC\web\containers\uCSDPAF_Server\usrconf\	Y
<i>installation-directory</i> \lib\dashboard.war	<i>installation-directory</i> \psb\CC\web\containers\uCSDPAF_Server\webapps\	N

Legend:

Y: File must be edited after being copied.

N: There is no need to edit the file after it is copied.

Among the copied files, you must specify in the Dashboard Server internal settings file (*usrconf.properties*) the interval at which Dashboard Server is to access the dashboard output connector and refresh the dashboard-display data; you must also specify a retry count.

For details about editing the server internal settings file, see *11.2 Dashboard Server internal settings file (usrconf.properties)*.

### 3.7.3 Setting up Dashboard Viewer

Dashboard Viewer is a Flex Dashboard application for viewing a dashboard using a Web browser. Dashboard Viewer accesses Dashboard Server periodically to refresh the dashboard-display data and displays the real-time stream data summary analysis results.

This subsection describes how to set up Dashboard Viewer.

**(1) Editing the Dashboard Viewer window**

The user can edit the dashboard window that is displayed by Dashboard Viewer. Editing the dashboard window enables you to create and display charts from specific data and to change the layout of charts.

For details about creation of a Dashboard Viewer window layout file, see *11.3.2 Details of a Dashboard Viewer window layout file*.

To edit the dashboard window, you must be familiar with the configuration and components of the dashboard windows that are displayed by Dashboard Viewer. For details about the dashboard windows, see *11.3.1 Configuration of Dashboard Viewer window*.

**(2) Notifying the URL of the Dashboard Viewer**

After you have finished editing the Dashboard Viewer window, you must provide the URL of Dashboard Viewer to any user who will be viewing the dashboard window. The URL of the Dashboard Viewer is as follows:

```
http://host-name:port-number/dashboard/viewer.swf?layout=window-name#
```

#

This is the window name specified as the file name (*window-name.xml*) in the Dashboard Viewer window layout file.





## Chapter

---

# 4. System Operation

---

This chapter discusses the flow of operations in the stream data processing system, how to start and shut down the system, and use of queries and query groups. It also describes how to display summary analysis results on a dashboard.

- 4.1 Flow of system operation
- 4.2 Starting the system
- 4.3 Using queries and query groups
- 4.4 Shutting down the system
- 4.5 Displaying analysis results on a dashboard

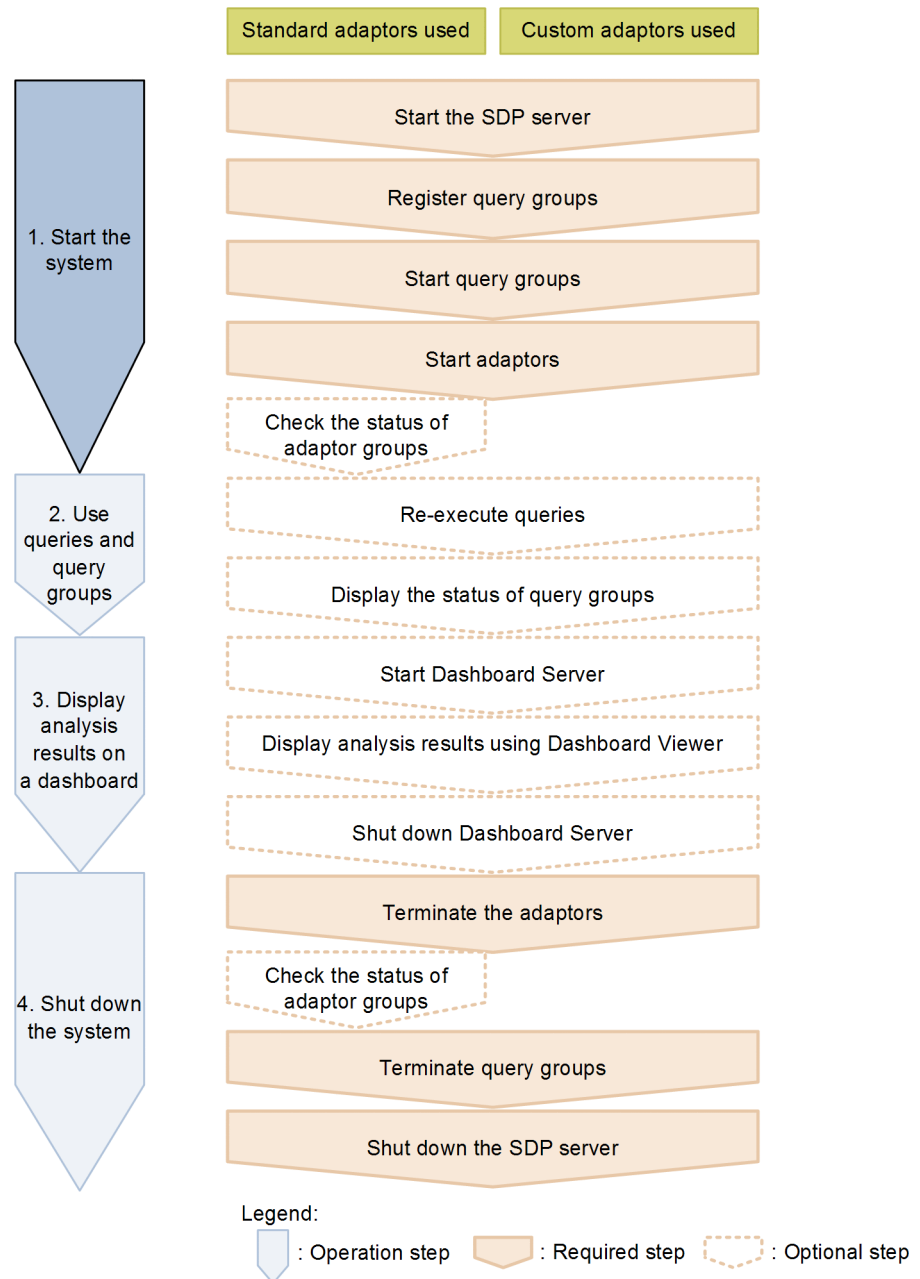
---

## 4.1 Flow of system operation

---

System operation includes starting and shutting down the system and using query groups. The following shows the flow of operations in a stream data processing system.

Figure 4-1: Flow of system operation



The following subsections explain the steps.

### **(1) Starting the system**

You start the system by starting the SDP server, query groups, and adaptors in the following order:

1. Start the SDP server

Start the SDP server. For details about how to start the SDP server, see *4.2.1 Starting the SDP server*.

2. Register query groups

Register query groups into the SDP server. For details about how to register query groups, see *4.2.2 Registering query groups*.

3. Start query groups

Start the query groups that have been registered into the SDP server. For details about how to start query groups, see *4.2.3 Starting query groups*.

4. Start adaptors

Start the standard adaptors or custom adaptors. For details about how to start the standard adaptors, see *4.2.4 Starting adaptors (standard adaptors)*; for details about how to start custom adaptors, see *4.2.5 Starting adaptors (custom adaptors)*.

5. Check the status of adaptor groups

If you are using the standard adaptors, check to see if the adaptor groups are running normally. For details about how to check adaptor group status, see *4.2.6 Checking the status of adaptor groups*.

### **(2) Using queries and query groups**

If necessary, you can use queries and query groups by performing the operations explained below.

1. Re-execute queries

Re-execute queries that have already been executed. For details about how to re-execute queries, see *4.3.1 Re-executing queries*.

2. Display the status of query groups

Display the status of query groups and the accumulation status of input and output streams. For details about how to display the status of query groups, see *4.3.2 Displaying the status of query groups*.

### **(3) Displaying analysis results on a dashboard**

The following describes how to use the standard adaptors to output stream data summary analysis results to a dashboard:

1. Start Dashboard Server

Start the Web server, PRF daemon, and Web container server; then start Dashboard Server. For details about how to start Dashboard Server, see *4.5.1 Starting Dashboard Server*.

2. Use Dashboard Viewer to display analysis results

Use Dashboard Viewer to open the dashboard window and display analysis results. For details about how to use Dashboard Viewer to display analysis results, see *4.5.2 Using Dashboard Viewer to display analysis results*.

3. Shut down Dashboard Server.

Shut down the Web container server, PRF daemon, and Web server, and then shut down Dashboard Server. For details about how to shut down Dashboard Server, see *4.5.3 Shutting down Dashboard Server*.

#### **(4) Shutting down the system**

Terminate the adaptors, query groups, and SDP server in the order described below, and then shut down the system.

1. Terminate the adaptors

Terminate the standard or custom adaptors. If you are using custom adaptors in the RMI connection mode, you must implement a termination process within the custom adaptors because Stream Data Platform - AF commands cannot be used to terminate them.

For details about how to terminate the standard adaptors, see *4.4.1 Terminating adaptors (standard adaptors)*; for details about how to terminate custom adaptors, see *4.4.2 Terminating adaptors (custom adaptors)*.

2. Check the status of adaptor groups

Check that the adaptor groups have terminated normally. For details about how to check the status of adaptor groups, see *4.2.6 Checking the status of adaptor groups*.

3. Terminate the query groups

Terminate the query groups. For details about how to terminate the query groups, see *4.4.3 Terminating query groups*.

4. Shut down the SDP server

Shut down the SDP server. For details about how to shut down the SDP server, see *4.4.4 Shutting down the SDP server*.

---

## 4.2 Starting the system

---

This section discusses how to start the system. The system startup steps must be performed in the order described below.

### 4.2.1 Starting the SDP server

Use the `sdpstart` command to start the SDP server. The following shows an example of command execution:

```
working-directory\bin\sdpstart
```

For details about the `sdpstart` command, see *sdpstart (starts the SDP server)* in 7. *Commands*.

### 4.2.2 Registering query groups

Register query groups into the SDP server. To register query groups, execute the `sdpcql` command with the definition files created during configuration specified. For details about creation of definition files, see 3.4 *Creating the SDP server definition files* and 3.5 *Creating the query definition files*.

The files to be specified when query groups are registered are listed below; registration of the stream property file is optional:

- Query definition files
- Query group property file
- Stream property file

An example of the command to be executed is shown below (in this example, `QueryGroupSample` is the name of the query group property file):

```
working-directory\bin\sdpcql QueryGroupSample
```

You can register a query group and start it at the same time by executing the `sdpcql` command with the `-autostart` option specified.

The example shown below simultaneously registers and starts a query group. In this example, `QueryGroupSample` is the name of the query group property file.

```
working-directory\bin\sdpcql -autostart QueryGroupSample
```

For details about the `sdpcql` command, see *sdpcql (registers a query group)* in 7. *Commands*.

### 4.2.3 Starting query groups

Start a query group that has already been registered into the SDP server.

When a query group starts, it receives send data from adaptors and then executes queries.

To start a query group, execute the `sdpcqlstart` command. An example of command execution is shown below. In this example, `QueryGroupSample` is the name of the query group property file.

```
working-directory\bin\sdpcqlstart QueryGroupSample
```

To register and start a query group at the same time, execute the `sdpcql` command with the `-autostart` option specified.

For details about the `sdpcqlstart` or `sdpcql` command, see *sdpcqlstart (starts a query group)* or *sdpcql (registers a query group)* in 7. Commands.

### 4.2.4 Starting adaptors (standard adaptors)

Start the standard adaptors. The query groups must have been registered and started before you start the adaptors.

The command to be used depends on the connection mode (in-process connection or RMI connection). This subsection describes how to start the adaptors in both connection modes.

#### (1) When in-process connection is used

Execute the `sdpstartinpro` command to start the standard adaptors.

In the command's argument, specify an adaptor group name for the `name` attribute in the in-process group definition in the adaptor configuration definition file. For details about the in-process group definition, see 9.7.1 *In-process group definition*.

An example of command execution is shown below. In this example, `InprocessAPSample` is the adaptor group name.

```
working-directory\bin\sdpstartinpro InprocessAPSample
```

For details about the `sdpstartinpro` command, see *sdpstartinpro (starts in-process-connection adaptors)* in 7. Commands.

#### (2) When RMI connection is used

Execute the `sdpstartap` command to start the adaptors.

In the command's argument, specify the adaptor group name specified for the `name` attribute in the RMI group definition in the adaptor configuration definition file. For details about the RMI group definition, see 9.7.2 *RMI group definition*.

An example of command execution is shown below. In this example, `RMIGroupSample` is the adaptor group name.

```
working-directory\bin\sdpstartap jp.co.Hitachi.soft.sdp.adaptor.AdaptorManager  
RMIGroupSample
```

For details about the `sdpstartap` command, see *sdpstartap (starts RMI-connection adaptors)* in 7. *Commands*.

### (3) Adaptor status change

When you start the standard adaptors, the adaptor group is placed in online status and data transmission with the SDP server becomes available. There are two statuses for an adaptor group:

- Inactive status

The adaptors are not active. You can create and edit definition files while the adaptors are in this status. Adaptors are placed in inactive status when you terminate an adaptor group by executing the `sdpstopinpro` command for adaptors in the in-process connection mode or the `sdpstopap` command for adaptors in the RMI connection mode.

- Online status

The adaptors are active and data can be input and output. Adaptors are placed in online status when you start an adaptor group by executing the `sdpstartinpro` command for adaptors in the in-process connection mode or the `sdpstartap` command for adaptors in the RMI connection mode.

Adaptors are also placed in inactive status when record input processing during file input is completed in the batch processing mode or when an unresumable failure occurs. In such a case, make sure that you execute the `sdpstopinpro` command for adaptors in the in-process connection mode.

## 4.2.5 Starting adaptors (custom adaptors)

Start the custom adaptors. The query groups must have been registered and started before you start the adaptors.

The command to be used depends on the connection mode (in-process connection or RMI connection). This subsection describes how to start the adaptors in both connection modes.

### (1) When in-process connection is used

Execute the `sdpstartinpro` command to start an adaptor.

In the command's argument, specify a custom adaptor name specified as the name of the in-process connection property file. For details about the in-process connection property file, see 8.9 *In-process connection property file*



(*user\_app.adaptor-group-name-or-adaptor-name.properties*).

An example of command execution is shown below. In this example, `InproAppSample` is the custom adaptor name.

```
working-directory\bin\sdpstartinpro InproAppSample
```

For details about the `sdpstartinpro` command, see *sdpstartinpro (starts in-process-connection adaptors)* in *7. Commands*.

## (2) When RMI connection is used

Execute the `sdpstartap` command to start an adaptor.

In the command's arguments, specify the necessary information, such as the path name of the JavaVM options file for RMI connections, class name of the Java application, and arguments to be passed to the `main` method. For details about the JavaVM options file for RMI connections, see *8.5 JavaVM options file for RMI connections (jvm\_client\_options.cfg)*.

An example of command execution is shown below. In this example, `AppSample` is the class name of the Java application.

```
working-directory\bin\sdpstartap -clientcfg conf\jvm_client_options.cfg AppSample
```

For details about the `sdpstartap` command, see *sdpstartap (starts RMI-connection adaptors)* in *7. Commands*.

## 4.2.6 Checking the status of adaptor groups

After you have started or terminated the standard adaptors, check the status of the adaptor group and the status of data processing to determine whether the standard adaptors are running normally or have terminated normally and whether data transmission is being performed normally.

This subsection describes how to check the status (active or inactive) of adaptor groups and data processing.

### (1) Checking whether an adaptor group is active or inactive

After you have started or terminated the standard adaptors, check whether the standard adaptors have started or terminated normally. You use error messages and the command's return value to determine the status of adaptor groups.

#### ■ Checking for error messages

Check the items listed below for any E (Error) or W (Warning) messages:

- Console where you executed the command to start or terminate the standard adaptors

- Adaptors' log files output to *working-directory*\logs

If the KFSP56101-E message is output, an XML schema verification error has occurred in the adaptor configuration definition file. In such a case, identify the cause of the error on the basis of the error identifier and message displayed in the message details. For a list of identifiers, see *Part 1: Structures Appendices C* in the W3C XML Schema specifications.

For example, if the name attribute is omitted from the `InprocessGroupDefinition` tag in the in-process group definition, the KFSP56101-E message with the following message details is displayed:

```
KECX06032-E cvc-complex-type.4: Attribute 'name' must appear on element 'adp:InprocessGroupDefinition'.
```

In this case, the identifier is `cvc-complex-type.4`. This is a violation of rule 4 in *Validation Rule: Element Locally Valid (Complex Type)*.

- Checking the command's return value

Execute the following command at the console where you executed the command for starting or terminating the standard adaptors to make sure that the result is 0:

```
echo %ERRORLEVEL%
```

## (2) Checking the result of data transmission

When you have started the standard adaptors, check the items listed below to determine whether data is being transmitted successfully between the SDP server and adaptors.

- Checking for error messages

Check the items listed below for any E (Error) or W (Warning) messages:

- Console where you executed the command to start the standard adaptors or the SDP server
- Adaptor and SDP server message log files output to *working-directory*\logs

For details about the log files, see *6.2(5) Log files* and *6.3.1 Details of log files*.

- Checking the result of data transmission

Check the input source file and the tuple log information to make sure that the tuples sent from the input adaptors were sent to the SDP server. Also check the queries and the data obtained from file output to make sure that the expected results have been obtained.

For details about the tuple logs, see 6.2(8) *Tuple logs* and 6.3.4 *Details of tuple logs*.

---

## 4.3 Using queries and query groups

---

You re-execute queries and display the status of query groups. You can perform these operations at any time.

### 4.3.1 Re-executing queries

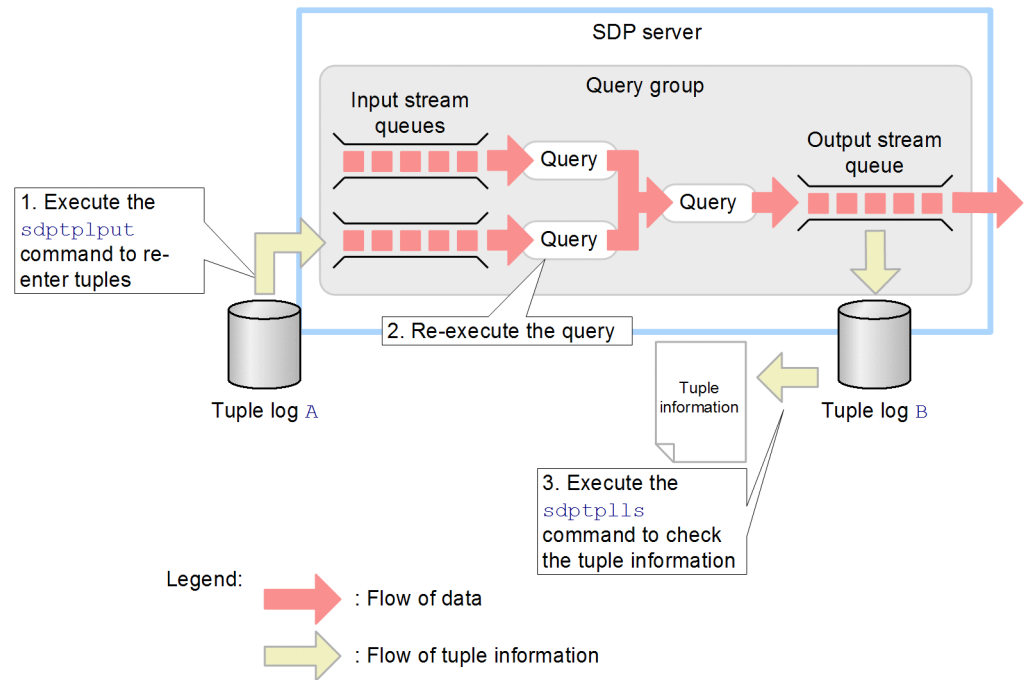
If you want to reproduce a problem event that has occurred, or re-check summary analysis results, you re-execute queries by reloading the data for a specific time period.

To re-execute queries, you use the `sdptplput` command. For the target data to be re-processed, use information about the input tuples collected in the tuple log file. When you execute the command, tuples are reloaded to the input stream queues from the tuple information collected in the tuple log file. After reloading of tuples is completed, the `putEnd` method is executed on all input stream queues, and the query group is initialized.

You can check the result of query re-execution by using the `sdptpls` command to display the tuple information collected from the output stream queue to the tuple log.

The following figure shows the flow of data when queries are re-executed to check execution results.

Figure 4-2: Flow of data during query re-execution



1. Execute the `sdptp1put` command (with tuple log A specified) to reload tuples.
2. Re-execute the query on the loaded tuples.
3. Execute the `sdptp1ls` command (with tuple log B specified) to check the tuple information.

The following subsections discuss the permitted range of query re-execution, query re-execution procedure, and notes about re-executing queries.

### (1) Permitted range of query re-execution

The permitted range of query re-execution differs between the server mode and the data source mode. The permitted range is as follows:

- In the server mode
  - Query execution
- In the data source mode
  - From timestamp adjustment to query execution

For details about the timestamp adjustment function, see *10.8 Timestamp adjustment for tuples*.

## (2) How to re-execute queries

You use a tuple log file that has been acquired to re-execute a query. A query can be re-executed in an environment that is different from the environment used to acquire the tuple logs, except that the default character encoding must be the same in both environments.

This example assumes that the environment in which queries are re-executed is different from the environment where tuple logs were acquired.

To re-execute queries:

1. Copy the required files to the environment where queries are to be re-executed.  
Copy the query definition files for the query group to be re-executed, query group property file, and tuple log file from the environment where tuples were acquired to the environment where the queries are to be re-executed.
2. Specify the settings required for acquiring tuple logs.  
Specify `BUFFER` in the `tpl.outputTrigger` parameter in the query group property file to acquire tuple logs from the output stream queue. If you use a tuple log file acquired in the server mode, specify `true` in the `stream.tupleLogMode` parameter.
3. Register the query group to be re-executed.  
Use the `sdpcql` command to register the query group to be re-executed.
4. Start the query group to be re-executed.  
Use the `sdpcqlstart` command to start the query group to be re-executed.
5. Re-execute the queries.  
Use the `sdptlput` command to reload tuples from the tuple log file and then re-execute the queries.
6. Check the results.  
Use the `sdptplls` command to display the contents of the tuple log file in the output stream queue and check the results.

For details about the `sdpcql`, `sdpcqlstart`, `sdptlput`, and `sdptplls` commands, see *sdpcql (registers a query group)*, *sdpcqlstart (starts a query group)*, *sdptlput (reloads tuples)*, and *sdptplls (displays tuple information)* in 7. Commands.

## (3) Notes about re-executing queries

- If you re-execute queries by loading tuples starting with any tuple other than the first tuple loaded by an adaptor, the results might differ from those obtained when the queries were executed using the adaptor.
- Because the `sdptlput` command does not reproduce the tuple loading interval,

the input stream queue might result in an overflow. If overflow occurs, the operation depends on whether the server mode or the data source mode is used:

- In the server mode
 

A tuple whose reload failed is reloaded at the interval specified in the `-interval` option. In this case, the `KFSP42005-E` and `KFSP52003-E` messages are displayed, but they have no effect on query re-execution.
- In data source mode
 

If queue overflow occurs, the query group is shut down, resulting in a query re-execution error. Therefore, use the `-interval` and `-count` options to adjust the interval at which tuples are reloaded and the number of tuples to be reloaded.
- Do not start adaptors while the `sdptplput` command is executing. If adaptors are started at such a time, the following occurs:
  - When input adaptors are started
 

The reloaded tuples are intermixed with the tuple loaded by the input adaptors, thereby causing the results obtained from query re-execution to be different from the results obtained when queries were executed by using the input adaptors.
  - When output adaptors are started
 

Overflow might occur in the output stream queue, resulting in shutdown of the query group.

### 4.3.2 Displaying the status of query groups

You can use the `sdpls` command to obtain the status of query groups registered in the SDP server. When you execute the `sdpls` command, the command displays the status of query groups and the accumulation status of input and output stream queues.

An example of command execution is shown below. In this example, `QueryGroupSample` is the query group property file.

```
working-directory\bin\sdpls QueryGroupSample
```

For details about the `sdpls` command, see *sdpls (displays the status of query groups)* in 7. *Commands*.

---

## 4.4 Shutting down the system

---

This section discusses system shutdown. For details about how to check the status of adaptor groups, see *4.2.6(1) Checking whether an adaptor group is active or inactive*.

### 4.4.1 Terminating adaptors (standard adaptors)

You must terminate the standard adaptors. When you terminate the adaptors, the adaptor group is placed in inactive status. For details about the status of adaptor groups, see *4.2.4(3) Adaptor status change*.

The command used for in-process connection differs from the command used for RMI connection. The following subsections describe the termination methods for both connection modes.

#### (1) When in-process connection is used

Use the `sdpstopinpro` command to terminate the adaptors.

In the command's argument, specify the adaptor group name that was specified for the name attribute in the in-process group definition in the adaptor configuration definition file. For details about the in-process group definition, see *9.7.1 In-process group definition*.

An example of command execution is shown below. In this example, `InprocessAPSample` is the adaptor group name.

```
working-directory\bin\sdpstopinpro InprocessAPSample
```

For details about the `sdpstopinpro` command, see *sdpstopinpro (terminates in-process-connection adaptors)* in *7. Commands*.

#### (2) When RMI connection is used

Use the `sdpstopap` command to terminate the adaptors.

In the command's argument, specify the adaptor group name that was specified for the name attribute in the RMI group definition in the adaptor configuration definition file. For details about the RMI group definition, see *9.7.2 RMI group definition*.

An example of command execution is shown below. In this example, `RMIGroupSample` is the adaptor group name.

```
working-directory\bin\sdpstopap RMIGroupSample
```

For details about the `sdpstopap` command, see *sdpstopap (terminates RMI-connection adaptors)* in *7. Commands*.



## 4.4.2 Terminating adaptors (custom adaptors)

You must terminate custom adaptors. The method used for in-process connection differs from that used for RMI connection. The following subsections describe the termination methods for both connection modes.

### (1) When in-process connection is used

Use the `sdpstopinpro` command to terminate an adaptor.

In the command's argument, specify the custom adaptor name specified as the name of the in-process connection property file.

For details about the in-process connection property file, see *8.9 In-process connection property file (user\_app.adaptor-group-name-or-adaptor-name.properties)*.

An example of command execution is shown below. In this example, `InproAppSample` is the custom adaptor name.

```
working-directory\bin\sdpstopinpro InproAppSample
```

For details about the `sdpstopinpro` command, see *sdpstopinpro (terminates in-process-connection adaptors)* in *7. Commands*.

### (2) When RMI connection is used

When you are using RMI connection, you must implement a termination process within the custom adaptors because Stream Data Platform - AF commands cannot be used to terminate them.

## 4.4.3 Terminating query groups

You must terminate the query groups registered in the SDP server. Note that termination of adaptors must be completed before you terminate query groups.

Perform this procedure when you terminate a query group that is run only for a set period of time each day or when you need to perform maintenance on a query group that normally runs continuously. When a query group has been terminated, send data from adaptors is no longer accepted.

There are two ways to terminate query groups:

- *Normal termination*
- *Forced termination*

This subsection provides the details of normal termination and forced termination and discusses relation discarding during query group termination.

### (1) Normal termination of query groups

When you terminate a query group during daily operation, you use the normal

termination method. When you execute normal termination, the query group is terminated after the tuple processing currently in the input stream queue has been completed.

To terminate query groups normally, execute the `sdpcqlstop` command. An example of command execution is shown below. In this example, `QueryGroupSample` is the query group property file.

```
working-directory\bin\sdpcqlstop QueryGroupSample
```

For details about the `sdpcqlstop` command, see *sdpcqlstop (terminates a query group)* in *7. Commands*.

### **(2) Forced termination of query groups**

If a problem occurs while a query group is executing, you use the forced termination method to terminate the query group immediately. When you execute forced termination, tuples currently in the input stream queue are discarded and the query group is terminated immediately.

To terminate query groups forcibly, execute the `sdpcqlstop` command with the `-force` option specified. An example of command execution is shown below. In this example, `QueryGroupSample` is the query group property file.

```
working-directory\bin\sdpcqlstop -force QueryGroupSample
```

For details about the `sdpcqlstop` command, see *sdpcqlstop (terminates a query group)* in *7. Commands*.

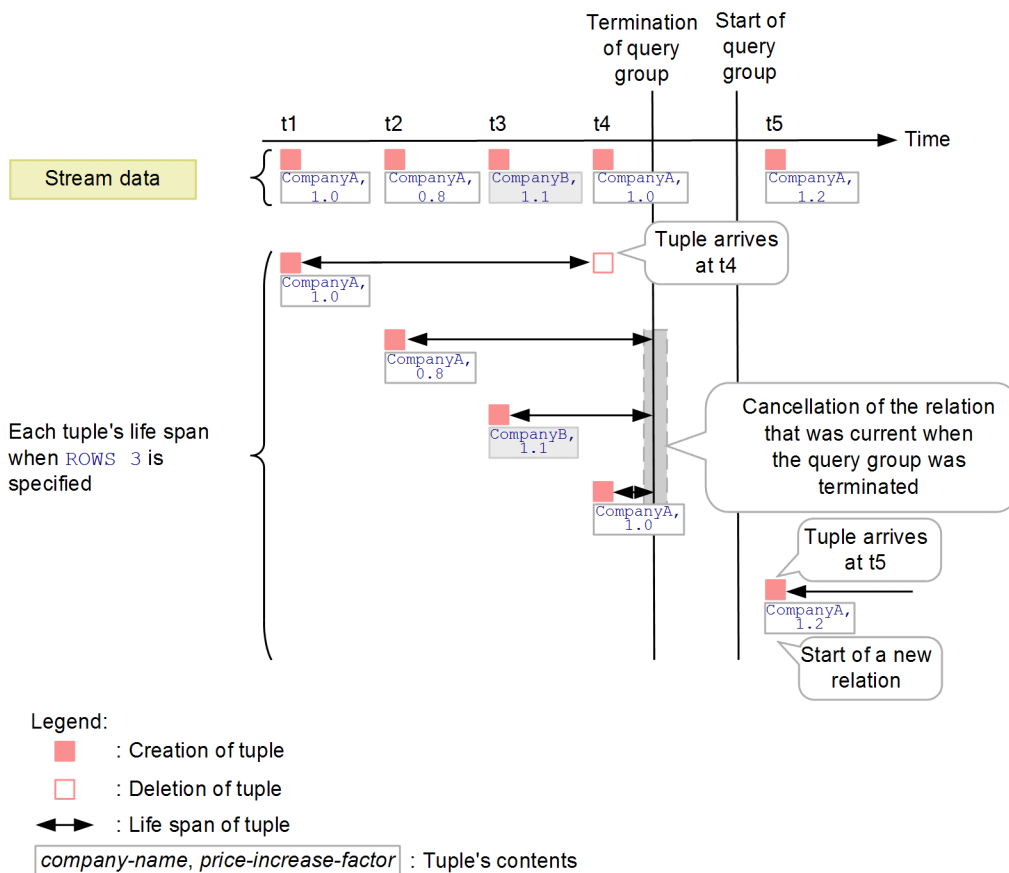
### **(3) Relation discarding**

A *relation* is a set of tuples retrieved by a window operation. Such a set of tuples becomes the target of data manipulation operations. A window operation refers to an operation that specifies a range subject to summary analysis; CQL is used to define such an operation. For details about the window operations that can be defined with CQL, see the manual *uCosminexus Stream Data Platform - Application Framework Application Development Guide*.

When a query group is terminated, any remaining input relation is discarded. The next time a query group is started, a new input relation starts without inheriting the previous input relation.

The following figure shows the status of an input relation when a query group is terminated.

Figure 4-3: Status of input relation when a query group is terminated



This figure shows the status of an input relation when the query group is terminated with [ROWS 3] specified for the window operation. [ROWS 3] means that there are three tuples in existence at the same time in the input relation. The abscissa indicates the time axis. The time progresses from left to right.  $t_1$  through  $t_5$  indicate the arrival times of tuples. The format of each tuple is *company-name, price-increase-factor*.

When this query group is terminated, the input relation containing *CompanyA, 0.8*, *CompanyB, 1.1*, and *CompanyA, 1.0* is discarded. The tuple *CompanyA, 1.2* that arrived at  $t_5$ , which is after the query group was started, is then created and a new input relation begins.

#### 4.4.4 Shutting down the SDP server

You use the `sdpstop` command to shut down the SDP server.

There are two ways to shut down the SDP server:

- *Normal termination*
- *Forced termination*

### **(1) Normal termination of SDP server**

This termination method terminates the SDP server after all query groups currently executing and all custom adaptors in the in-process connection mode have terminated normally.

If a custom adaptor in the in-process connection mode is still running, the SDP server calls the `stop` method in the class with the `StreamInprocessUP` interface implemented and performs termination processing on the custom adaptor in the in-process connection mode.

To terminate the SDP server normally, execute the `sdpstop` command. The following shows an example of command execution:

```
working-directory\bin\sdpstop
```

For details about the `sdpstop` command, see *sdpstop (stops the SDP server)* in 7. *Commands*.

### **(2) Forced termination of SDP server**

This termination method terminates the SDP server immediately without waiting for termination of the query groups currently executing or the custom adaptors in the in-process connection mode. All tuples in the input stream queue are discarded.

To terminate the SDP server forcibly, execute the `sdpstop` command with the `-force` option specified. The following shows an example of command execution:

```
working-directory\bin\sdpstop -force
```

For details about the `sdpstop` command, see *sdpstop (stops the SDP server)* in 7. *Commands*.

## 4.5 Displaying analysis results on a dashboard

This section describes how to display stream data summary analysis results on a dashboard when the standard adaptors are used.

### 4.5.1 Starting Dashboard Server

To start Dashboard Server, you must start the Web server, PRF daemon, and Web container server in the order described below.

To start Dashboard Server:

1. Execute the `httpsd` command to start the Web server.

You can also start the Web server with the following command:

```
net start "Hitachi Web Server for uCSDPAF"
```

2. Execute the `cprfstart` command to start the PRF daemon. The PRF daemon is a process that outputs trace information.
3. Execute the `cjstartweb` command to start the Web container server.

Note that once you execute this command, control is not returned to the command prompt until the Web container server is shut down.

The following shows an example of command execution:

```
installation-directory\psb\httpsd\httpsd.exe -k start -n "Hitachi Web Server for uCSDPAF"

Starting the Hitachi Web Server for uCSDPAF service
The Hitachi Web Server for uCSDPAF service is running.

installation-directory\psb\PRF\bin\cprfstart.exe

Wed Feb 24 23:54:53 2010:KFCT73410-I 380 2380:now starting cprfd.
Wed Feb 24 23:54:54 2010:KFCT73412-I 380 2380:cprfd is now online.

installation-directory\psb\CC\web\bin\cjstartweb.exe uCSDPAF_Server

KDJE39001-I The web container is now starting. (server name = uCSDPAF_Server)
KDJE39278-W The default value is applied to the server ID appended to the session ID.
(default value = CtF3Hx9H)
KDJE39219-I The ClassLoader for the web application was initialized. (context root =
/dashboard, initialized time = 2010/02/25 00:26:47.625)
KDJE39003-I The web container started. (server name = uCSDPAF_Server)
```

### 4.5.2 Using Dashboard Viewer to display analysis results

To use Dashboard Viewer to display analysis results:

1. Open the dashboard window.

Enter the appropriate URL in the Web browser and display the saved dashboard window. For details about the URL, see 3.7.3(2) *Notifying the URL of the Dashboard Viewer*.

2. View the data.

View the analysis results displayed in the dashboard window. You can display other windows as necessary from the analysis results window. For details about the dashboard window, see 11.3.1 *Configuration of Dashboard Viewer window*.

### 4.5.3 Shutting down Dashboard Server

To shut down Dashboard Server, you must terminate the Web container server, PRF daemon, and Web server in the order described below.

To shut down Dashboard Server:

1. Execute the `cjstopweb` command to terminate the Web container server.

When you execute this command, the following messages are displayed at the command prompt from which the Web container server was started, and control is returned:

```
KDJE39002-I The web container is now stopping. (server name = uCSDPAF_Server)
KDJE39004-I The web container stopped. (server name = uCSDPAF_Server)
```

2. Execute the `cprfstop` command to terminate the PRF daemon.
3. Execute the `httpsd` command to terminate the Web server.

You can also terminate the Web server with the following command:

```
net stop "Hitachi Web Server for uCSDPAF"
```

The following shows an example of command execution:

```
installation-directory\psb\CC\web\bin\cjstopweb.exe uCSDPAF_Server

installation-directory\psb\PRF\bin\cprfstop.exe

Thu Feb 25 01:21:48 2010:KFCT73413-I 2256 980:now terminating cprfd. terminate type =
NORMAL STOP
Thu Feb 25 01:21:50 2010:KFCT73001-I 2256 980:prf tracing service stopped. ID:PRF_ID
Thu Feb 25 01:21:50 2010:KFCT73414-I 2256 980:CPRFD stop.
installation-directory\installation-directory\psb\httpsd\httpsd.exe -k stop -n "Hitachi Web Server
for uCSDPAF"

The Hitachi Web Server for uCSDPAF service is stopping.
The Hitachi Web Server for HRTM uCSDPAF service has stopped.
```

## Chapter

---

# 5. Modifying the System

---

This chapter describes how to change system settings and modify the configuration during system operation.

- 5.1 Overview of system modification
- 5.2 Modifying the stream data processing engine
- 5.3 Modifying query groups
- 5.4 Modifying adaptors
- 5.5 Modifying the dashboard

---

## 5.1 Overview of system modification

---

You can make changes to the system during operation by using definition files and commands. The system modifications that can be made are as follows:

- **Modifying the stream data processing engine**

You can change information about the stream data processing engine that is being used for current operations.

- **Modifying query groups**

You can change information about the query groups that are being used for current operations.

- **Modifying adaptors**

You can change information about the adaptors that are being used for current operations.

- **Modifying the dashboard**

If you use a dashboard to display stream data summary analysis results, you can change settings, such as the configuration of the Dashboard Viewer windows. You can also cancel the registration of Dashboard Server.



---

## 5.2 Modifying the stream data processing engine

---

You can modify the stream data processing engine that is used for current operations by changing the contents of the definition files shown below.

Definition file to be changed	Section
JavaVM options file for SDP servers ( <code>jvm_options.cfg</code> )	8.4
System configuration property file ( <code>system_config.properties</code> )	8.6
Log file output property file ( <code>logger.properties</code> )	8.10

To change a definition file:

1. Use the `sdpstop` command to shut down the SDP server.  
For details about how to shut down the SDP server, see *4.4.4 Shutting down the SDP server*.
2. Change the contents of the definition files.
3. Use the `sdpstart` command to start the SDP server.  
For details about how to start the SDP server, see *4.2.1 Starting the SDP server*.

---

## 5.3 Modifying query groups

---

You modify the query groups in order to change the query groups used for current operations or change their definitions.

The table below lists the query group definition files that can be changed and the sections to be referenced for the details of changing their contents.

Query group definition file name	Section
Query group property file	8.7
Stream property file	8.8
Query definition file	Manual <i>uCosminexus Stream Data Platform - Application Framework Application Development Guide</i>

The procedure for changing settings in the query group property file and stream property file differs from the procedure for changing settings in the query definition files. This section discusses how to change settings in the property files separately from how to change settings in the query definition file.

### 5.3.1 Changing settings in the property files

To change the properties of query groups or streams:

1. Use the `sdpstopinpro` or `sdpstopap` command to terminate the adaptor group.  
For details about how to terminate adaptor groups, see *4.4.1 Terminating adaptors (standard adaptors)*.
2. Use the `sdpcqlstop` command to terminate the query group.  
For details about how to terminate query groups, see *4.4.3 Terminating query groups*.
3. Change the contents of the property file for the query group or stream.  
For details about the property files for query groups and streams, see *8.7 Query group property file* or *8.8 Stream property file*.
4. Execute the `sdpcqlstart` command with the `-reload` option specified to start the query group.  
For details about starting query groups, see *4.2.3 Starting query groups*.

Note that when you have started a query group using the `sdpcqlstart` command with the `-reload` option specified, the settings cannot be changed for some

parameters in the property files. To change the settings of such parameters, you must first delete the query group and then register and start a new query group. For details about the parameters whose settings cannot be changed, see *8.7.1 Details of the query group property file* or *8.8.1 Details of the stream property file*.

### 5.3.2 Changing the query definition file

This subsection describes how to delete a query group and change a query definition file when you wish to change CQL statements defined in the query definition file.

#### (1) Deleting a query group

You can delete a query group from the SDP server by executing the `sdpcqldel` command. Once a query group is deleted, operations on the query group or data transmission from adaptors are no longer available.

An example of deletion of a query group is shown below. In this example, `QueryGroupSample` is the name of the query group property file.

```
working-directory\bin\sdpcqldel QueryGroupSample
```

For details about the `sdpcqldel` command, see *sdpcqldel (deletes a query group)* in *7. Commands*.

#### (2) Changing a query definition file

To change the contents of a query definition file:

1. Use the `sdpstopinpro` or `sdpstopap` command to terminate the adaptor group.  
For details about how to terminate adaptor groups, see *4.4.1 Terminating adaptors (standard adaptors)*.
2. Use the `sdpcqlstop` command to terminate the query group.  
For details about how to terminate query groups, see *4.4.3 Terminating query groups*.
3. Use the `sdpcqldel` command to delete the query group.  
For details about deleting query groups, see *(1) Deleting a query group*.
4. Change the contents of the query definition file.  
For details about query definition files, see the manual *uCosminexus Stream Data Platform - Application Framework Application Development Guide*.
5. Use the `sdpcql` command to register the query group.  
For details about registering query groups, see *4.2.2 Registering query groups*.

---

## 5.4 Modifying adaptors

---

To make changes to the standard adaptors, you change the contents of the adaptor definition files.

The table below lists the names of adaptor definition files to be changed and the sections to be referenced.

Adaptor definition file name	Section
Adaptor configuration definition file	9.5
Adaptor command definition file	9.4

To change the contents of a definition file:

1. Use the `sdpstopinpro` or `sdpstopap` command to terminate the adaptor group.  
For details about how to terminate adaptor groups, see *4.4.1 Terminating adaptors (standard adaptors)*.
2. Change the contents of the definition file.  
For details about adaptor configuration definition files, see *9.5 Adaptor configuration definition file (AdaptorCompositionDefinition.xml)*; for details about Adaptor command definition files, see *9.4 Adaptor command definition file (AgentManagerDefinition.xml)*.
3. Use the `sdpstartinpro` or `sdpstartap` command to start the adaptor group.  
For details about starting adaptor groups, see *4.2.4 Starting adaptors (standard adaptors)*.

---

## 5.5 Modifying the dashboard

---

This section describes how to make changes to Dashboard Viewer windows while stream data summary analysis results are being displayed on the dashboard and how to cancel the registration of Dashboard Server.

### ■ Changing a Dashboard Viewer window

If you wish to make changes to a Dashboard Viewer window, such as changing the layout of charts displayed in the dashboard window, use the following procedure to edit the Dashboard Viewer window layout file:

1. Use the Web browser to close the dashboard window.
2. Shut down Dashboard Server.

For details, see *4.5.3 Shutting down Dashboard Server*.

3. Edit the Dashboard Viewer window layout file.

For details, see *3.7.3(1) Editing the Dashboard Viewer window*.

4. Restart Dashboard Server.

For details, see *4.5.1 Starting Dashboard Server*.

5. Use the Web browser to open the dashboard window.

For details, see *4.5.2 Using Dashboard Viewer to display analysis results*.

### ■ Canceling registration of Dashboard Server

When you no longer have need to use a dashboard to display data or you are going to uninstall Stream Data Platform - AF, you must cancel the registration of Dashboard Server.

When you cancel Dashboard Server's registration, you must use the following procedure to also cancel the setup of Web container server and then cancel registration of the Web server (Hitachi Web Server) service:

1. Cancel the setup of the Web container server.

Execute the following command to cancel the setup of the Web container server that communicates with Dashboard Server:

```
installation-directory\psb\CC\web\bin\cjwebsetup.exe -d uCSDPAF_Server
```

2. Cancel the registration of Web server.

Execute the following command to cancel the registration of Web server (Hitachi Web Server) service:

## 5. Modifying the System

```
installation-directory\psb\httpsd\httpsd.exe -k uninstall -n "Hitachi Web Server for  
uCSDPAF"
```

## Chapter

---

# 6. Troubleshooting

---

This chapter describes the types of data to be collected in the event a problem arises during system operation, how to interpret the data, and how to resolve problems.

- 6.1 Error handling procedure
- 6.2 Data to be collected in the event of an error
- 6.3 Details of log files and trace files
- 6.4 How to handle major problems

---

## 6.1 Error handling procedure

---

To handle errors during system operation:

1. Check if a message has been issued.

In the event of an error, one or more error messages should have been output to the console or message log.

2. Check the error event and take appropriate action.

Check error messages that have been output. Correct the cause of the error according to the error handling procedures described in the manual *uCosminexus Stream Data Platform - Application Framework Messages*. For details about how to handle major problems, see *6.4 How to handle major problems*.

If the error cannot be resolved, contact the customer engineer (i.e., contact Hitachi customer service according to the contract you entered into at the time you purchased the product).

3. Collect the data needed for handling the error

Collect the data needed to determine the cause of the error by referencing *6.2 Data to be collected in the event of an error*.



## 6.2 Data to be collected in the event of an error

This section discusses the data that should be collected in the event of an error. You must collect data when any of the following events occurs:

- Invalid timeout

This includes when a timeout occurs because query processing is taking too long while the CPU usage rate does not indicate any heavy load on the system.

- Abnormal server shutdown

This includes when the server has shut down due to an internal system error or illegal runtime exception.

- Server process shutdown

This includes a sudden shutdown of the JavaVM process on the SDP server that is not accompanied by output of a message.

The table below lists the data that needs to be collected for each event.

*Table 6-1:* Data that needs to be collected for each event

No.	Data	Event		
		Invalid timeout	Abnormal server shutdown	Server process shutdown
1	Standard output, standard error output	Y	Y	Y
2	Error report file	--	--	Y
3	SDP server definition files	Y	Y	Y
4	Adaptor definition files	Y	Y	Y
5	Log files	Y	Y	Y
6	Trace files	Y <sup>#1</sup>	Y	Y
7	Adaptor trace information	Y <sup>#2</sup>	Y	Y
8	Tuple logs	Y <sup>#3</sup>	Y	Y
9	Thread dump	Y	Y	Y
10	OS status	Y	--	--
11	OS statistics	Y	Y	Y

No.	Data	Event		
		Invalid timeout	Abnormal server shutdown	Server process shutdown
12	Hitachi Web Server error log file <sup>#4</sup>	Y	Y	Y

Legend:

Y: Must be collected

--: Need not be collected

#1

You can collect this data after you have shut down the SDP server. For details about shutting down the SDP server in the event of an invalid timeout, see *6.4.1(5) A timeout occurred during query processing.*

#2

You can collect this data after you have terminated the adaptors.

#3

You can collect this data by terminating query groups.

#4

You can collect the Hitachi Web Server error log file in the event of a Dashboard Server error.

The table below describes how to collect the data.

*Table 6-2: How to collect the data*

No.	Data	How to collect
1	Standard output, standard error output	Make a note of (or copy) the messages output to the standard output and standard error output
2	Error report file	Copy the files under the following directory: <i>working-directory\</i>
3	SDP server definition files	Copy the files under the following directory: <i>working-directory\conf\</i>
4	Adaptor definition files	Copy the files under the following directory: <i>working-directory\conf\xml\</i>

No.	Data	How to collect
5	Log files	Copy the files under the following directory: <i>working-directory</i> \logs\
6	Trace files	Copy the files under the following directory: <i>working-directory</i> \trc\
7	Adaptor trace information	Copy the files under the following directory: <i>working-directory</i> \trc\adaptor\
8	Tuple logs	Copy the files under the following directory: <i>working-directory</i> \trc\tuplelog\
9	Thread dump	Collect data by executing the <code>jheapprof</code> command
10	OS status	Collect data by executing the <code>netstat</code> command
11	OS statistics	Collect performance monitor logs
12	Hitachi Web Server error log file	Collect the Hitachi Web Server error log files under the following directory: <i>installation-directory</i> \psb\CC\web\containers\uCSDPAF_Server\logs\

The details about how to collect each type of data are described below.

### (1) **Standard output and standard error output**

Make a note of (or copy) the messages output to the *standard output* and *standard error output*.

Note that all messages related to startup and termination of the SDP server and adaptors are output to the standard output or the standard error output. Some of these messages are also output to log files, but others are not.

### (2) **Error report file**

An *error report file* is output in the event of abnormal termination of JavaVM. You can use this file to determine the location and cause of a process shutdown.

The error report file is output directly under the working directory and has the following file name:

`hs_err_pidprocess-ID.log`

### (3) **SDP server definition files**

The *SDP server definition files* include the following files:

- JavaVM options file for SDP servers (`jvm_options.cfg`)
- JavaVM options file for RMI connections (`jvm_client_options.cfg`)
- System configuration property file (`system_config.properties`)

- Query group property files
- Stream property files
- In-process connection property files  
(`user_app.adapter-group-name-or-adapter-name.properties`)
- Log file output property file (`logger.properties`)

Collect these files under the following directory:

`working-directory\conf\`

If the JavaVM options file for RMI connections is not stored under this directory, also collect that file.

#### **(4) Adaptor definition files**

The *adaptor definition files* include the following files:

- Adaptor configuration definition file  
(`AdaptorCompositionDefinition.xml`)
- Adaptor command definition file (`AgentManagerDefinition.xml`)

Collect these files under the following directory:

`working-directory\conf\xml\`

#### **(5) Log files**

The *log files* include the following:

- Message log
- Trace log
- Hitachi JavaVM log file

Collect all the files under the following directory:

`working-directory\logs\`

For details about the message logs and trace logs, see *6.3.1 Details of log files*.

The Hitachi JavaVM log file is the log file to which Hitachi-specific JavaVM logs are output; it is obtained by using extended options added by Hitachi to the standard JavaVM. This file provides more troubleshooting information than the standard JavaVM.

#### **(6) Trace files**

The *trace files* include various trace files, such as API trace information.

Collect all the files under the following directory:

`working-directory\trc\`

For details about the API trace information, see 6.3.2 *Details of API trace information*.

### (7) **Adaptor trace information**

*Adaptor trace information* is provided as trace files to which the adaptor processing status is output.

Collect the files under the following directory:

*working-directory*\trc\adaptor\

For details about the adaptor trace information, see 6.3.3 *Details of adaptor trace information*.

### (8) **Tuple logs**

*Tuple logs* are the log files to which information about the input and output tuples is output.

Collect the files under the following directory:

*working-directory*\trc\tuplelog\

For details about the tuple logs, see 6.3.4 *Details of tuple logs*.

### (9) **Thread dump**

A *thread dump* is a file to which information about the threads running inside Java processes is output.

You use the `jheapprof` command to collect a thread dump.

For details about how to collect a thread dump and details about the `jheapprof` command, see 6.3.5 *Details of a thread dump*.

### (10) **OS status**

Collect as the *OS status* the network information and environment variables. To collect this information, execute the `netstat` command as follows:

```
netstat -e > netstat_e.txt
netstat -s > netstat_s.txt
netstat -an > netstat_an.txt

set > set.txt
```

### (11) **OS statistics**

The *OS statistics* includes information that enables you to check the status, such as system loading and system performance. You use the OS's performance monitor to collect this information.

While the SDP server is executing, collect at a regular interval the performance monitor logs shown in the table below.

Table 6-3: Performance monitor logs

Object	Instance	Counter
Processor	_Total	%Processor Time
		%Privileged Time
		%User Time
Memory	--	Cache Bytes
		Cache Faults/sec
		Page Faults/sec
		Transition Faults/sec
		Transition Faults/sec
Process	_Total	Handle Count
		Page Faults/sec
		Private Bytes
		Virtual Bytes
		Working Set Bytes
	java <sup>#</sup>	%Processor Time
		%Privileged Time
		%User Time
		Page Faults/sec
		Thread Count
		Private Bytes
		Virtual Bytes
		Working Set Bytes

## Legend:

--: Not applicable

#

This information can be collected only if collection of performance logs has already been started in advance.

We recommend setting 60 seconds as the log collection interval; however, you should determine the appropriate interval on the basis of disk capacity. Setting the collection interval to a large value can reduce the adverse effects of OS statistics collection on performance, but the accuracy of the OS statistics might be compromised.

For details about how to specify the setting, see the OS documentation.

### **(12) Hitachi Web Server error log file**

The *Hitachi Web Server error log file* is used to check and monitor the Web container operating status. Error, warning, and information messages are output to this file.

If you output stream data summary analysis results to a dashboard using the standard adaptors, collect the following log file in the event of a Dashboard Server error (the question mark (?) in the log file name represents a 1- or 2-digit integer):  
*installation-directory\psb\CC\web\containers\uCSDPAF\_Server\logs\user\_err?.log*

---

## 6.3 Details of log files and trace files

---

This section provides the details of the following log and trace files and describes how to collect them:

- Log files
- API trace information
- Adaptor trace information
- Tuple logs
- Thread dump

### 6.3.1 Details of log files

Stream Data Platform - AF outputs the following log files:

- *Message log* (`SDPServerMessageserial-number#.log`)

This log file is used to check and monitor the server's operating status. It stores the error, warning, and information messages that are issued by the SDP server.

- *Trace log* (`SDPServerTraceserial-number#.log`)

This log file is used to check problems that have occurred in Stream Data Platform - AF. It stores the stack trace output by the SDP server and maintenance information.

#

This is a serial number for the log file. If there are 16 files, the value that is set will be in the range of 1 to 16.

Note that all messages output to these files are issued by the SDP server. These files do not contain messages issued by commands or adaptors.

#### (1) Overview of log files

The table below provides size values for the log files as well as describing the timing of switching output files.

Item	Description
File size	The file size is specified in the log file output property file ( <code>logger.properties</code> ). When a file's size is approaching the maximum value, the specified size might be exceeded depending on the size of the next message that is output.



Item	Description
Timing of switching output files	File switching occurs when the size of the current file exceeds the specified value. You can change the number of output files in the log file output property file ( <code>logger.properties</code> ).
File size during switching	When file switching occurs, the size of the target file is set to 0 (all rows deleted).
Operation during file switching	When file switching occurs, all rows are deleted so that data can be output starting from the beginning of the file.
Selection of output file during process restart	During a process restart, data is added to the file with the most recent timestamp.

For details about the log file output property file, see *8.10 Log file output property file (logger.properties)*.

## (2) Output format of log files

This subsection describes the output format of log files. A log file is output in the following format:

```
number date time application-name pid tid message-ID message-type message-text
```

Legend:

*number*: Serial number (4 digits) in the trace record

*date*: Trace collection date (*yyyy/mm/dd*)

*time*: Trace collection time (*hh:mm:ss.sss*)

*application-name*: `SDPServer vrrss`

(*vrrss* indicates the version number, revision number, and a special code)

*pid*: Process ID

*tid*: Thread identifier

*message-ID*: `KFSPnnnnn-x`

*message-type*: One of the following values is displayed:

EC: Exception occurred

ER: Error message

No value: Other than EC or ER

*message-text*: Message that corresponds to the message ID

For details about the messages, see the manual *uCosminexus Stream Data Platform -*

*Application Framework Messages.*

The following shows an example of output to a log file:

```
0000 2010/10/15 18:46:07.723      SDPServer 0100      01642BD6 0179F36B KFSP81002-I The
server has started.
0001 2010/10/15 18:46:21.614      SDPServer 0100      01642BD6 0179F36B KFSP81003-I The
server will stop now.
0002 2010/10/15 18:46:21.630      SDPServer 0100      01642BD6 0179F36B KFSP81004-I The
server has stopped.
```

**(3) Messages that are output to the standard output**

Messages related to startup and termination of the SDP server and adaptors are output to the standard output (or the standard error output). Some of those messages are also output to the log files.

Note that the following messages, which are output before initialization of log file output processing, are all output to the standard output or the standard error output:

- KFSP81001-I (message: The server will start now)
- Property file and duplicate startup error messages

Some messages output to the standard output or the standard error output are also output to a log file. The tables below show whether or not messages related to startup and termination of the SDP server and adaptors are also output to a log file. For details about the messages, see the manual *uCosminexus Stream Data Platform - Application Framework Messages*.

*Table 6-4:* Whether or not messages output to the standard output are also output to a log file (messages related to startup and termination of SDP server)

No.	Message ID	Whether output to log file
1	KFSP41001-E	N
2	KFSP41002-E	N
3	KFSP41003-E	N
4	KFSP51001-W	Y
5	KFSP51002-W	Y
6	KFSP51003-W	Y
7	KFSP51004-E	Y
8	KFSP61004-E	Y
9	KFSP61005-E	Y <sup>#</sup>

No.	Message ID	Whether output to log file
10	KFSP61006-E	N
11	KFSP61007-E	Y
12	KFSP61008-E	N
13	KFSP61009-E	N
14	KFSP62000-E	Y
15	KFSP81001-I	N
16	KFSP81002-I	Y
17	KFSP81003-I	Y
18	KFSP81004-I	Y
19	KFSP81005-I	Y
20	KFSP81006-I	Y

Legend:

Y: Output to log file

N: Not output to log file

#

If logs have not been initialized, this message is output only to the standard output or the standard error output.

*Table 6-5:* Whether or not messages output to the standard output are also output to a log file (messages related to startup and termination of adaptors)

No.	Message ID	Whether output to log file
1	KFSP46002-W	N
2	KFSP46101-E	N
3	KFSP46115-E	N
4	KFSP56003-E	N
5	KFSP56004-I	N
6	KFSP56101-E	N
7	KFSP86001-I	Y

No.	Message ID	Whether output to log file
8	KFSP86002-I	Y
9	KFSP86003-I	Y
10	KFSP86004-I	Y
11	KFSP86005-I	Y
12	KFSP86006-I	Y
13	KFSP86007-I	Y
14	KFSP86008-I	Y
15	KFSP86009-I	Y
16	KFSP86011-I	Y
17	KFSP86018-E	Y
18	KFSP86019-I	Y
19	KFSP86020-I	Y
20	KFSP86022-E	N
21	KFSP86101-I	N
22	KFSP86102-I	N

Legend:

Y: Output to log file

N: Not output to log file

#### **(4) Error during log file processing**

If an error occurs during log file initialization processing, an error message is output. If an error occurs during message output processing, the log file output destination is changed to the standard output or the standard error output and then the message is output.

In such a case, the SDP server is shut down based on the default settings.

### **6.3.2 Details of API trace information**

API trace information is provided as a trace file to which query entry and exit information is output. You can use this information to determine the startup and termination of adaptors and queries.

You can use the `sdptrced` command to edit and output the API trace information. The

figure below shows an example of trace information that has been output as a result of executing the `sdptrced` command. This example outputs trace information on queries Q1 through Q3 that are in a mutually dependent relationship (Q1's output tuples are input to Q2, and Q2's output tuples are input to Q3).

*Figure 6-1: Example output of API trace information*

Date	Time	nSec	Current	Parent	Root	EventID	Rc	Data	
2010/02/20	12:16:47	61542000	trace-serial-number reserved-area root-trace-serial-number			0x00020000	0	Q1	1.
2010/02/20	12:16:47	61700000	trace-serial-number reserved-area root-trace-serial-number			0x00020001	0	Q1	2.
2010/02/20	12:16:47	61774000	trace-serial-number reserved-area root-trace-serial-number			0x00020000	0	Q2	
2010/02/20	12:16:47	61921000	trace-serial-number reserved-area root-trace-serial-number			0x00020001	0	Q2	
2010/02/20	12:16:47	61995000	trace-serial-number reserved-area root-trace-serial-number			0x00020000	0	Q3	
2010/02/20	12:16:47	62139000	trace-serial-number reserved-area root-trace-serial-number			0x00020001	0	Q3	

*Notes:*

In this figure, the version information for the Stream Data Platform - AF that output the trace file, the trace file type (API), and the path (absolute path) of the API trace file are output at the top.

A 30-digit value is set for each of *trace-serial-number*, *reserved-area*, and *root-trace-serial-number*.

From this example, you can obtain the following information (the numbers below correspond to the numbers on the right side of the figure):

1. The value of `Data` is Q1 and the value of `EventID` is 0x00020000 (input of a tuple to the query).

This trace information means that a tuple was input to query Q1 at the indicated time.

2. The value of `Data` is Q1 and the value of `EventID` is 0x00020001 (output of a tuple from the query).

This trace information means that a tuple was output from query Q1 at the indicated time.

The information in `EventID` and `Data` on the subsequent lines shows that input to and output from query Q2 and input to and output from query Q3 were processed in this order.

You can obtain the time of an input or output operation in nanoseconds from the value of `nSec`.

For details about the `sdptrced` command, see *sdptrced (edits trace information)* in 7. *Commands*.

### 6.3.3 Details of adaptor trace information

Adaptor trace information is provided as trace files to which the processing status of

adaptors is output. This information is used at the stage of testing the operation of Stream Data Platform - AF to check the processing status of each callback operation.

The default is that adaptor trace information is not output because its output affects adaptor throughput. To make this trace information available, specify `ON` for the `trace` attribute in the adaptor trace definition. Before actual operations begin, set this attribute so that adaptor trace information will not be output. For details about the adaptor trace definition, see *9.6.2 Adaptor trace definition*.

Adaptor trace information is created for each adaptor, with two megabytes of trace information output to each file in the wraparound mode.

### (1) Name of adaptor trace information

Adaptor trace information is output under the following name:  
`adp_type-adaptor-group-name-adaptor-name_serial-number`

The following describes each component of the file name:

- *type*

One of the following is set:

`IN`: This value is set for an input adaptor.

`OUT`: This value is set for an output adaptor.

- *adaptor-group-name*

This is the value specified in the `name` attribute in the adaptor group definition.

- *adaptor-name*

This is the value specified in the `name` attribute in the adaptor definition

- *serial-number*

This is a serial number for the file. A value in the range from `001` to `004` is set.

If the adaptor trace file already exists when the adaptor starts, its file name is changed to *file-name*.`bk`.

For details about the adaptor definition files, see *9. Adaptor Definition Files*.

### (2) Output format of adaptor trace information

Adaptor trace information is output in CSV format, as shown below:

<i>number, date, time, thread, event</i>
--

The table below describes each output item.

Table 6-6: Items output as adaptor trace information

No.	Item	Description
1	<i>number</i>	4-digit serial number (when the value reaches 9999, it initializes to 0001)
2	<i>date</i>	Date the trace information was collected, in the format <i>YYYY/MM/DD</i>
3	<i>time</i>	Time the trace information was collected, in the format <i>hh:mm:ss.fff</i>
4	<i>thread</i>	Adaptor name (value specified in the <i>name</i> attribute in the adaptor definition)
5	<i>event</i>	<p>Callback name and detailed information, in the following format:  <i>callback-name</i> <math>\Delta</math> {<i>entry</i> <i>exit</i>}</p> <ul style="list-style-type: none"> <li>• <i>callback-name</i>: Value specified in the <i>name</i> attribute in the CB definition</li> <li>• <math>\Delta</math>: Single-byte space</li> <li>• {<i>entry</i> <i>exit</i>}:  <i>entry</i>: Callback processing was started  <i>exit</i>: Callback processing was terminated</li> </ul>

An example of the output is shown below. In this example, the first line is a header and the subsequent lines are records.

```
No,Date,Time,Thread,Event
0001,2010/10/15,02:08:41.173,OutputAdaptor1,receiver entry
0002,2010/10/15,02:08:41.173,OutputAdaptor1,receiver exit
0003,2010/10/15,02:08:41.188,OutputAdaptor1,editor1 entry
0004,2010/10/15,02:08:41.188,OutputAdaptor1,editor1 exit
0005,2010/10/15,02:08:41.188,OutputAdaptor1,editor3 entry
0006,2010/10/15,02:08:41.204,OutputAdaptor1,editor3 exit
0007,2010/10/15,02:08:41.204,OutputAdaptor1,outputer entry
0008,2010/10/15,02:08:41.204,OutputAdaptor1,outputer exit
0009,2010/10/15,02:08:41.204,OutputAdaptor1,receiver entry
0010,2010/10/15,02:08:41.204,OutputAdaptor1,receiver exit
0011,2010/10/15,02:08:41.204,OutputAdaptor1,editor1 entry
0012,2010/10/15,02:08:41.204,OutputAdaptor1,editor1 exit
0013,2010/10/15,02:08:41.204,OutputAdaptor1,editor3 entry
0014,2010/10/15,02:08:41.204,OutputAdaptor1,editor3 exit
0015,2010/10/15,02:08:41.204,OutputAdaptor1,outputer entry
0016,2010/10/15,02:08:41.204,OutputAdaptor1,outputer exit
:
```

### (3) Example of output of adaptor trace information

This subsection describes the adaptor trace information that is output when records are discarded during callback processing or packets are discarded during HTTP packet input connector processing.

- When records are discarded during callback processing

Each time callback is executed, the total number of records discarded immediately before the callback processing terminated is output to the adaptor

trace file.

In this case, the *event* output item in Table 6-6 *Items output as adaptor trace information* is output in the following format:

*callback-name*  $\Delta$  *execute delete*  $\Delta$  *detailed-information*

- *callback-name*: Value specified in the name attribute in the CB definition
- $\Delta$ : Single-byte space
- *execute delete*: Type of operation
- *detailed-information*: Detailed information for the callback, as shown in the table below.

Table 6-7: Detailed information in the adaptor trace information for a callback (applicable to discarding of records)

No.	Callback type	Detailed information
1	HTTP packet input connector	<ul style="list-style-type: none"> <li>• When records were discarded because the number of HTTP packets in the output buffer reached the maximum value:<sup>#1</sup> <i>record</i> <math>\Delta</math> <i>overflow</i> <math>\Delta</math> <i>discarded-records-count</i></li> </ul>
2	Filter	<ul style="list-style-type: none"> <li>• When records were discarded because a field condition was not satisfied:<sup>#2</sup> <i>condition</i> <math>\Delta</math> <i>record-name</i> <math>\Delta</math> <i>discarded-records-count</i></li> </ul>
3	Record extraction	<ul style="list-style-type: none"> <li>• When records were discarded because an extraction target condition was not satisfied:<sup>#3</sup> <i>condition</i> <math>\Delta</math> <i>record-name</i><sup>#4</sup> <math>\Delta</math> <i>discarded-records-count</i></li> <li>• When records were discarded due to record duplication: <i>same-record</i> <math>\Delta</math> <i>name-of-record-to-be-extracted</i> <math>\Delta</math> <i>discarded-records-count</i></li> <li>• When records were discarded due to a timeout:<sup>#5</sup> <i>timeout</i> <math>\Delta</math> <i>extraction-condition-name</i> <math>\Delta</math> <i>discarded-records-count</i></li> <li>• When records were discarded because the maximum number of records that can be retained had been reached:<sup>#6</sup> <i>overflow</i> <math>\Delta</math> <i>discarded-records-count</i></li> <li>• When records were discarded due to a record out-of-sequence time: <i>time-reverse</i> <math>\Delta</math> <i>discarded-records-count</i></li> </ul>
4	Dashboard output connector	<ul style="list-style-type: none"> <li>• When records were discarded because the record retention period<sup>#7</sup> had elapsed: <i>timeout</i> <math>\Delta</math> <i>discarded-records-count</i></li> <li>• When records were discarded because the maximum number of records that can be retained had been reached:<sup>#8</sup> <i>overflow</i> <math>\Delta</math> <i>discarded-records-count</i></li> <li>• When records were discarded due to deletion of collected records:<sup>#9</sup> <i>read</i> <math>\Delta</math> <i>discarded-records-count</i></li> </ul>



#1

The maximum number of HTTP packets that can be stored in the output buffer is specified in the `bufferSize` attribute of the `input` tag in the HTTP packet input connector definition. For details about the `bufferSize` attribute, see [9.10.2 HTTP packet input connector definition](#).

#2

A field condition is specified in the `field` tag in the filter definition. For details about the `field` tag, see [9.11.3 Filter definition](#).

#3

An extraction target condition is specified in the `targetRecord` tag in the record extraction definition. For details about the `targetRecord` tag, see [9.11.4 Record extraction definition](#).

#4

If none of the extraction target conditions is satisfied, the name of the record being discarded is output; if an extraction condition is not satisfied, the name of the extraction target record is output.

#5

The timeout value is specified in the `timeLimit` attribute of the `extraction` tag in the record extraction definition. For details about the `timeLimit` attribute, see [9.11.4 Record extraction definition](#).

#6

The maximum number of records that can be retained is specified in the `size` attribute of the `extractions` tag in the record extraction definition. For details about the `size` attribute, see [9.11.4 Record extraction definition](#).

#7

The record retention period is specified in the `RecordHoldTime` tag in the dashboard output connector definition. For details about the `RecordHoldTime` tag, see [9.10.4 Dashboard output connector definition](#).

#8

The maximum number of records that can be retained is specified in the `MaxNum` attribute of the `DashboardOutputConnectorDefinition` tag in the dashboard output connector definition. For details about the `MaxNum` attribute, see [9.10.4 Dashboard output connector definition](#).

#9

Deletion of collected records is specified in the `ReadRecordRemoveFlag` attribute of the `DashboardOutputConnectorDefinition` tag in the

dashboard output connector definition. For details about the `ReadRecordRemoveFlag` attribute, see *9.10.4 Dashboard output connector definition*.

The following shows examples of output of adaptor trace information for callbacks:

#### 1. HTTP packet input connector

This example discarded 50 excess records because the number of records to be stored in the output buffer for the HTTP packet input connector (callback name: `inputer`) exceeded the maximum value.

```
3288,2010/10/15,10:16:11.223,InputAdaptor1,inputer entry
3289,2010/10/15,10:16:11.223,InputAdaptor1,inputer execute delete record overflow 50
3290,2010/10/15,10:16:11.223,InputAdaptor1,inputer exit
```

#### 2. Filter

This example discarded 100 records (`R1`) that did not satisfy a field condition for filtering (callback name: `filter1`):

```
3291,2010/10/15,10:16:11.223,InputAdaptor1,filter1 entry
3292,2010/10/15,10:16:11.223,InputAdaptor1,filter1 execute delete condition R1 100
3293,2010/10/15,10:16:11.223,InputAdaptor1,filter1 exit
```

#### 3. Record extraction

This example discarded 10 records that did not satisfy an extraction target record condition (`conditionE2`) for record extraction (callback name: `extract1`):

```
3294,2010/10/15,10:16:11.223,InputAdaptor1,extract1 entry
3295,2010/10/15,10:16:11.223,InputAdaptor1,extract1 execute delete condition
conditionE2 10
3296,2010/10/15,10:16:11.223,InputAdaptor1,extract1 exit
```

#### 4. Dashboard output connector

This example discarded 20 records because the number of records to be retained by the dashboard output connector (`outputer`) reached the maximum value:

```
3288,2010/10/15,10:16:11.223,OutputAdaptor1,outputer entry
3289,2010/10/15,10:16:11.223,OutputAdaptor1,outputer execute delete numoverflow 20
3290,2010/10/15,10:16:11.223,OutputAdaptor1,outputer exit
```

- When HTTP packets or messages are discarded by the HTTP packet input connector

Each time HTTP packet input connector processing is performed, the total number of HTTP packets or HTTP messages discarded before the processing terminated is output to the adaptor trace file.

In this case, the *event* output item in Table 6-6 *Items output as adaptor trace information* is output in the following format:

*callback-name*  $\Delta$  *execute delete*  $\Delta$  *detailed-information*

- *callback-name*: Value specified in the *name* attribute in the CB definition
- $\Delta$ : Single-byte space
- *execute delete*: Type of operation
- *detailed-information*: Detailed information

Table 6-8: Detailed information in the adaptor trace information (applicable to discarding of packets or HTTP messages)

Callback type	Detailed information
HTTP packet input connector	<ul style="list-style-type: none"> <li>• When HTTP packets were discarded because the number of HTTP packets in the input buffer reached the maximum value:<sup>#1</sup> <i>packet</i> <math>\Delta</math> <i>overflow</i> <math>\Delta</math> <i>discarded-packets-or-HTTP-messages-count</i></li> <li>• When HTTP messages were discarded because the packet segment assembly time<sup>#2</sup> resulted in a timeout: <i>httpmessage</i> <math>\Delta</math> <i>timeout</i> <math>\Delta</math> <i>discarded-packets-or-HTTP-messages-count</i></li> </ul>

#1

The maximum number of HTTP packets that can be stored in the input buffer is specified in the *buffersize* attribute of the *input* tag in the HTTP packet input connector definition. For details about the *buffersize* attribute, see 9.10.2 *HTTP packet input connector definition*.

#2

The packet segment assembly time is specified in the *assemblingtime* attribute of the *input* tag in the HTTP packet input connector definition. For details about the *assemblingtime* attribute, see 9.10.2 *HTTP packet input connector definition*.

The following shows examples of output of adaptor trace information:

- This example discarded 20 excess packets because the number of packets to be stored in the input buffer exceeded the maximum value:

```
3288,2010/10/15,10:16:11.223,InputAdaptor1,input entry
3289,2010/10/15,10:16:11.223,InputAdaptor1,input execute delete packet overflow 20
3291,2010/10/15,10:16:11.223,InputAdaptor1,input exit
```

- This example discarded 5 HTTP messages because the packet segment assembly time resulted in a timeout:

```
3288,2010/10/15,10:16:11.223,InputAdaptor1,inputer entry
3290,2010/10/15,10:16:11.223,InputAdaptor1,inputer execute delete httpmessage timeout
5
3291,2010/10/15,10:16:11.223,InputAdaptor1,inputer exit
```

### 6.3.4 Details of tuple logs

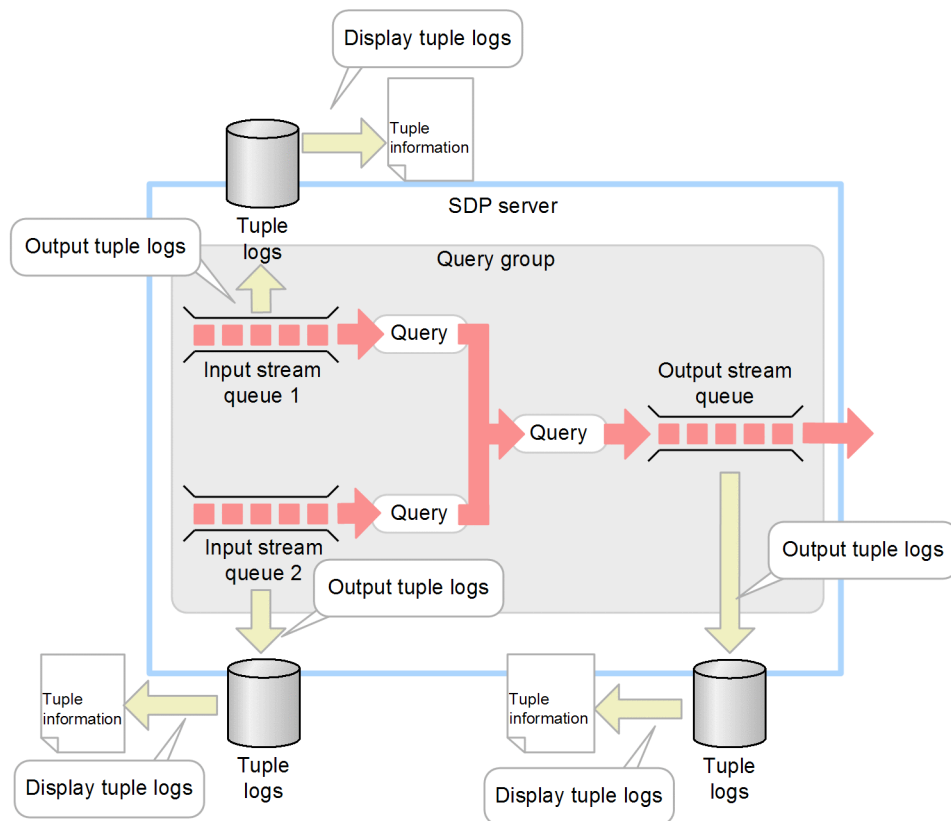
Tuple logs are log files to which information about input and output tuples is output. A tuple log is output for each stream queue. You can use tuple logs for the following purposes:

- To check the tuples that have arrived at the SDP server
- To re-execute queries and check their results


For details about re-execution of queries, see *4.3.1 Re-executing queries*.


The following figure shows a flow of output and display of tuple logs.

Figure 6-2: Flow of output and display of tuple logs



Legend:

 : Flow of data

 : Flow of tuple information

- **Outputting tuple logs**  
Input and output tuples are output to tuple log files. One tuple log file is output for each stream queue.
- **Displaying tuple logs**  
You can use the `sdptpls` command to display tuple log files.  
You can confirm from the contents of the tuple log files the tuples that have arrived at the SDP server.  
For details about the `sdptpls` command, see *sdptpls (displays tuple*

information) in 7. *Commands*.

### (1) **Outputting tuple logs**

You specify whether or not tuple logs are to be output by using parameters that begin with `tpl.` in the files listed below:

- `system_config.properties` (system configuration property file)
- Query group property file
- Stream property file

For details about each file, see 8. *SDP Server Definition Files*.

Tuple logs are placed in a tuple buffer provided for each stream queue and then are output to file. The table below describes the tuple log output timing.

*Table 6-9: Tuple log output timing*

No.	Tuple log output timing	Tuple log that is output to file
1	When the specified <code>tpl.bufferSize</code> parameter value is exceeded	Tuple log in the buffer
2	When the <code>putEnd</code> method is executed on an input stream queue	All tuple logs in the buffer for the input stream queue
3	When the <code>putEnd</code> method is executed on all input stream queues in a query group	All tuple logs in the buffer for the output stream queue in the query group
4	When a query group is terminated	All tuple logs in the buffers for the stream queues in the query group
5	When the SDP server is shut down	All tuple logs in the buffers for the stream queues on the SDP server

Tuple logs are output in the wraparound mode to as many files as the value specified in the `tpl.fileCount` parameter. The following table describes the timing of switching tuple log files.

*Table 6-10: Timing of switching tuple log files*

No.	Timing of tuple log file switching	Tuple log file to be switched
1	When the specified <code>tpl.fileSize</code> parameter value is exceeded	Tuple log file that has become full
2	When the <code>putEnd</code> method is executed on all input stream queues in a query group	Tuple log files for the stream queues in the query group

Tuple logs are not output in the following cases:

- When an exception occurs in the `put` method

If an exception occurs in a `put` method for which multiple tuples are specified in the `ArrayList` parameter, only the tuples that were input before the exception are output. You can use the `sdptpls` command to determine the tuples that were output.

For details about the `sdptpls` command, see *sdptpls (displays tuple information)* in 7. *Commands*.

- When a query group is terminated forcibly or the SDP server is shut down forcibly while a tuple is being input

The tuple being input might not be collected.

## (2) Names of tuple logs

The files listed below are output to the tuple log output destination directory. Note that the file access permissions depend on the user permissions used to execute the `sdpstart` command.

- Tuple log files
- Backup files of tuple log files

These files are output under the following names:

Tuple log files:

*tpl\_query-group-name-stream-name\_file-serial-number*

Backup files of tuple log files:

*tpl\_query-group-name-stream-name\_file-serial-number.bkbackup-generation-number*

The following describes each component of the file names:

- *query-group-name*

This is the name of the query group to which the target stream belongs.

- *stream-name*

This is the stream name for the stream queue that is subject to collection of tuple logs.

- *file-serial-number*

This is a file serial number (a 3-digit number specified in the `tpl.fileCount` parameter, beginning with 001).

- *backup-generation-number*

This is the generation number for a backup file (a 2-digit number specified in the `tpl.backupFileCount` parameter, beginning with 01).

**(3) Displaying tuple logs**

You can check the output tuple log information by executing the `sdptp11s` command.

For details about the `sdptp11s` command, see *sdptp11s (displays tuple information)* in 7. *Commands*.

**6.3.5 Details of a thread dump**

A thread dump is a file to which information about the threads running within a Java process are output.

This subsection describes how to collect a thread dump and discusses the `jheapprof` command.

**(1) How to collect a thread dump**

To collect a thread dump:

1. Execute the following command to determine the server process running in the working directory:

```
installation-directory\psb\jdk\bin\jps -v
```

An example of the result of executing this command is shown below. The line containing `SDPManager` and *working-directory* indicates the SDP server. In this example, that SDP server's process ID is 4616.

```
3980 Program -Xms40m -Xmx256m
5216 Jps -Dapplication.home=installation-directory\psb\jdk -Xms8m
4616 SDPManager -Dsdp.home=working-directory -Dsdp.serverLogging
```

2. Execute the following command to collect a thread dump for the SDP server's process ID obtained in step 1:

```
installation-directory\psb\jdk\jre\bin\jheapprof -f -p 4616
```

This command creates under the working directory a thread dump file named `javacoreprocess-ID.date-and-time.txt`.

We recommend that you collect around 10 thread dumps at an interval of 3 seconds. By collecting multiple thread dumps, you can check the method processes in chronological order to identify a process that is causing a processing slowdown or is not responding.

When you collect multiple thread dumps, make sure that output of one thread dump is completed before collecting a subsequent thread dump.



**(2) Details of *jheapprof* (outputs extended thread dump with Hitachi statistics by class)**

Format:

```
jheapprof [-i|-f] [-class class-name] [-explicit|-noexplicit]
[-fullgc|-copygc|-nogc] -p process-ID
```

Function:

The *jheapprof* command outputs an extended thread dump containing Hitachi statistics by class. The Hitachi statistics by class enable you to obtain the sizes of all instances under the members that belong to the instances of each class.

Execution permissions:

None

Prerequisites for command execution:

None

Storage directory:

```
installation-directory\psb\jdk\jre\bin\
```

Arguments:

**-i**

Specifies that the user is to be asked to confirm that this command is to be executed on the process with the specified process ID.

Even if this specification is omitted, this option is effective unless the **-f** option is specified.

**-f**

Specifies that the user is not to be asked to confirm that this command is to be executed on the process with the specified process ID.

**-class *class-name***

Specifies that the output to the thread dump is to be as a list member of the structure of the class (instance) specified in *class-name*. You must enclose the package name of the class in double-quotation marks (").

**-explicit**

Specifies that an `Explicit` heap is to be included as a target of the instance statistics function. If this option is specified together with the `-noexplicit` option, the last option specified takes effect. Note that there is no need to specify this option for the SDP server.

`-noexplicit`

Specifies that an `Explicit` heap is not to be included as a target of the instance statistics function. If this option is specified together with the `-explicit` option, the last option specified takes effect. Note that there is no need to specify this option for the SDP server.

`-fullgc`

Specifies that a full garbage collection is to be executed before the statistical information is output.

If this option is specified together with the `-copygc` or `-nogc` option, the last option specified takes effect.

`-copygc`

Specifies that a copy garbage collection is to be executed before the statistical information is output.

If this option is specified together with the `-fullgc` or `-nogc` option, the last option specified takes effect.

`-nogc`

Specifies that no garbage collection is to be executed before the statistical information is output.

If this option is specified together with the `-fullgc` or `-copygc` option, the last option specified takes effect.

`-p process-ID`

Specifies the process ID of the Java program for which Hitachi statistics by class are to be output.

*Notes:*

- This command cannot be executed more than once for the same Java process. If you want to execute the command more than once for the same Java process, wait until the statistics by class have been output to an extended thread dump by the `jheapprof` command before executing the command again.
- When a Java process starts, it uses MailSlot to initialize communication. If this initialization fails, the Java process outputs a message and cancels processing.
- This command can be executed by a user who is not the owner of the Java process whose process ID is specified in the argument.

Return value:

Value	Description
0	Command terminated normally.
1	Command resulted in an error.
2	Termination of output processing for the Hitachi statistics by class was not received within the specified amount of time.

Error messages:

When any of the messages listed below is displayed, the extended thread dump with Hitachi statistics by class was not output.

No.	Error message	Description
1	usage: jheapprof [-f -i] [-class classname] [-explicit -noexplicit] [-fullgc -copygc -nogc] [-garbage -nogarbage] [-rootobjectinfo -norootobjectinfo] [-rootobjectinfo size] -p process-id jheapprof	A specified command argument is invalid.
2	jheapprof: illegal option -- <i>option</i>	<i>option</i> specified in the jheapprof command is invalid.
3	<i>process-ID</i> : Now processing previous request, this request canceled	The process with the specified <i>process-ID</i> is currently outputting statistics by class.
4	0: Not owner	A value of 0 is specified for the <i>process-ID</i> argument in the jheapprof command.
5	jheapprof: can't create work file at temporary directory, this request canceled	Because the user does not have permissions to reference or write the directory for temporary files, the user cannot output an extended thread dump with Hitachi statistics by class. The request to output an extended thread dump with Hitachi statistics by class is canceled.
6	jheapprof: can't get temporary directory, this request canceled	The directory for temporary files could not be retrieved, so the extended thread dump with Hitachi statistics by class could not be output. The request to output an extended thread dump with Hitachi statistics by class is canceled.
7	jheapprof: please delete <i>name-of-file-whose-deletion-failed</i> in <i>full-path-of-file-whose-deletion-failed</i>	The jheapprof command was unable to delete an internal file when it terminated. Delete the file at the indicated full path whose deletion failed.

No.	Error message	Description
8	jheapprof: unexpected error occurred: <i>cause-of-error</i>	An unexpected error occurred during execution of the jheapprof command. <i>cause-of-error</i> displays the reason for the error, such as the following: <ul style="list-style-type: none"> <li>Allocation of work memory failed: malloc syscall fail (errno=Y)</li> <li>Object close processing resulted in an error: close syscall fail (errno=Y)</li> </ul>
9	jheapprof: can't communicate with process <i>process-ID</i>	Communication could not be performed due to a communication error in the process with the specified <i>process-ID</i> , or there was no process with the specified <i>process-ID</i> .
10	<i>process-ID</i> : Timeout occurred. Java process not responding	Completion of Hitachi statistics by class output processing for the process with the specified <i>process-ID</i> was not received within the specified amount of time.

**Examples:**

This example collects an extended thread dump with Hitachi statistics by class from the Java program with process ID 8662:

```
jheapprof -p 8662
```

When this command is executed, a confirmation message is displayed asking whether or not the extended thread dump with Hitachi statistics by class is to be output:

```
Force VM to output HitachiJavaHeapProfile: ? (y/n)
```

To output the extended thread dump with Hitachi statistics by class, enter **Y** or **y**. If you enter any other character, the command terminates without outputting the extended thread dump with Hitachi statistics by class.

```
Force VM to output HitachiJavaHeapProfile: ? (y/n)y
```

When an extended thread dump with Hitachi statistics by class is output, the following message is displayed in the Java program that is executing:

```
Writing Java core to javacore8662.030806215140.txt... OK
```

An extended thread dump with the following file name is output to the current directory of the Java program and then execution of the Java program continues:

```
javacoreprocess-ID.date-and-time.txt
```

---

## 6.4 How to handle major problems

---

This section discusses how to handle major problems in Stream Data Platform - AF.

### 6.4.1 Problems during system operation

This subsection discusses problems that might occur during system operation and how to handle them.

#### **(1) Data cannot be sent or received between the stream data processing engine and adaptors**

The following describes the action to be taken when data cannot be sent or received between the stream data processing engine and adaptors:

1. Check the message log for an error message and correct the cause of the error, if necessary.

If an error message has been output, take appropriate action to eliminate the cause of the error by referencing the manual *uCosminexus Stream Data Platform - Application Framework Messages*.

2. Check the following:
  - Whether the path of the input source or output target is specified correctly
  - Whether the specification method for format conversion is correct

If you have specified IGNORE in the `unmatchedFormat` tag in the format conversion definition in the adaptor configuration definition file, specify WARNING in the `unmatchedFormat` tag to check the operation so that an invalid format can be detected.

#### **(2) An overflow occurred in the input stream queue for the stream data processing engine**

If an overflow has occurred in the input stream queue for the stream data processing engine, control the volume of input adaptor processing. By changing the number of records that are read at one time or the interval at which records are sent to the SDP server, you can reduce the amount of memory used per unit of time.

You can change the number of records by using the `readUnit` attribute of the `input` tag in file input connector definition in the adaptor configuration definition file.

You can change the transmission interval by using the `interval` attribute in the CB definition for input.

#### **(3) An internal conflict message was output**

If an interval conflict message is output for a query group or the SDP server, collect

data by referencing *6.2 Data to be collected in the event of an error*, and then contact the customer engineer.

#### **(4) A query group was shut down**

If a failure occurs while a query group is executing, Stream Data Platform - AF shuts down the query group. If a query group is shut down, the tuples accumulated in the stream queues are discarded.

The following describes how to handle query group shutdown for each cause.

When the cause is a stream queue overflow:

If an overflow occurs in a query group's stream queue, the following messages are output to the message log:

```
KFSP42005-E The upper limit value of the stream queue was exceeded. Stream
name = ..., number of elements = ..., upper limit value = ...
KFSP82205-I A query group was blocked. Query group name=...
```

When these messages are output, check the settings in the SDP server definition files, such as the JavaVM heap size, queue maximum value, and maximum number of tuples that can be retained. If necessary, adjust values and restart the query group.

For details about changing settings in the SDP server definition files, see *5.3.1 Changing settings in the property files*. For details about restarting query groups, see *sdpcqlstart (starts a query group)* in *7. Commands*.

When the cause is input of invalid data:

If a division-by-zero error or an invalid type conversion error (value obtained after conversion is not a type supported by the target) occurs during query execution, the following message is output to the message log:

```
KFSP420**-E =...
KFSP82205-I A query group was blocked. Query group name=...
```

When this message is output, check the query definition files for an error. If there is an error in the query definition files, correct it. If there is no error in the query definition files but there is a problem in the send data, correct the contents of the input adaptor, and then restart the query group.

For details about changing query definition files, see *5.3.2 Changing the query definition file*. For details about restarting query groups, see *sdpcqlstart (starts a query group)* in *7. Commands*.

When the cause is that the number of tuples to be retained exceeded the maximum value while the timestamp adjustment function was being used:

If the number of tuples to be retained exceeded the maximum value for the

timestamp adjustment function, the following message is output to the message log:

```
KFSP42301-E The number of tuples held by the timestamp adjustment function
exceeded the upper limit value...
KFSP82205-I A query group was blocked. Query group name=...
```

When this message is output, check and revise (if necessary) the maximum number of tuples that can be retained or the units and range of times to be adjusted. For details of the tuple retention period, see *10.8.6 Tuple retention period*.

#### (5) A timeout occurred during query processing

If query processing required too much time and a timeout resulted, see *6.2 Data to be collected in the event of an error*, collect the required data, and determine the cause of the error. If necessary, take appropriate action, such as forcing termination of the SDP server.

For details about forced termination of the SDP server, see *sdpstop (stops the SDP server)* in *7. Commands*.

#### (6) Adaptor processing has slowed

If adaptor processing has slowed, check to see if full garbage collections have been occurring frequently. All adaptor processing stops while a full garbage collection is underway, resulting in a slowdown of processing speed. You can obtain information about full garbage collection from the following file, as applicable:

In the in-process connection mode:

```
working-directory\logs\SDPServerVMinteger.log
```

In the RMI connection mode:

```
working-directory\logs\SDPClientVMinteger.log
```

Note that *integer* in the file name is a number in the range from 01 to 04 (because a maximum of four files can be used). The output destination and names of these files are set to the default values.

The following shows an example of output of a full garbage collection:

```
[VGC]<Fri Jan 08 17:19:46.159 2010>[Full GC 54330K->45174K(519296K),
0.1150830 secs] [DefNew::Eden: 9156K->0K(164608K)]...
```

If the underlined `Full GC` is output frequently or garbage collection takes several seconds, JavaVM assumes that not enough memory is allocated to the heap area. In such a case, consider taking one of the corrective actions described below:

- Change the maximum size of JavaVM's heap area.

You can change the maximum size in the applicable file shown below. For details, see 8. *SDP Server Definition Files*.

- JavaVM options file for SDP servers (`jvm_options.cfg`)
- JavaVM options file for RMI connections (`jvm_client_options.cfg`)
- Control the volume of input adaptor processing.

For details about how to control the volume of input adaptor processing, see (2) *An overflow occurred in the input stream queue for the stream data processing engine*.

### **(7) Adaptor processing has stopped due to unmatched data types**

If the data type specified in the `type` attribute of the `field` tag in the format conversion definition in the adaptor configuration definition file does not match the data type in the corresponding CQL, an error occurs during tuple transmission and the adaptor processing stops. In such a case, check and revise (if necessary) the data type.

For details about the `type` attribute of the `field` tag in the format conversion definition, see 9.11.1 *Format conversion definition*.

### **(8) Some tuple logs have been lost**

If all tuple log buffers become full, some tuple logs will be lost.

If tuple logs have been lost, the result of query re-execution by the `sdptplput` command might not match the result obtained by executing queries using adaptors. Therefore, you must specify the appropriate number and sizes of buffers so that a sufficient number of tuples can be acquired.

The formula for estimating the tuple log buffers is shown below. Using this formula, determine the number and sizes of tuple log buffers that you need for storing tuples.

Size of one tuple log buffer (bytes) =  
*size of one tuple x number of tuples to be stored*

For the size of one tuple, see 2.7.1(2) *Memory required for one tuple*.

### **(9) The analysis results cannot be displayed on the dashboard**

If the analysis results cannot be displayed on the dashboard, check the following:

- Check whether or not Hitachi Web Server is running
  - If Hitachi Web Server is not running, start it.
- Check the version of the installed Flash Player.
  - If the version of Flash Player is earlier than 9.0.124.0, install the most recent version.



- Check the contents of the following files for any errors:
  - Dashboard output connector definition in the adaptor configuration definition file
  - Dashboard Server window layout file

If the files contain errors, correct them.

- Check whether the dashboard output connector is running.

If the dashboard output connector is not running, start the dashboard output connector that has the data to be received.

If an error occurs on the dashboard, an error dialog box is displayed in the dashboard window. This dialog box displays the following information:

Displayed name	Description
"ErrID"	Displays an error message ID
"ErrMsg"	Displays the error message

For details about the messages, see the manual *uCosminexus Stream Data Platform - Application Framework Messages*.



## Chapter

---

# 7. Commands

---

This chapter describes the commands used to run a stream data processing system.

- Format of command explanations

- List of commands

- `sdpcql` (registers a query group)

- `sdpcqldel` (deletes a query group)

- `sdpcqlstart` (starts a query group)

- `sdpcqlstop` (terminates a query group)

- `sdpls` (displays the status of query groups)

- `sdpsetup` (sets up an operating environment)

- `sdpstart` (starts the SDP server)

- `sdpstartap` (starts RMI-connection adaptors)

- `sdpstartinpro` (starts in-process-connection adaptors)

- `sdpstop` (stops the SDP server)

- `sdpstopap` (terminates RMI-connection adaptors)

- `sdpstopinpro` (terminates in-process-connection adaptors)

- `sdptpils` (displays tuple information)

- `sdptpinput` (reloads tuples)

- `sdptrced` (edits trace information)

## **Format of command explanations**

---

This section describes the format used to explain the commands.

The following items are provided for the command explanations. Note that some items are not provided for all commands.

### **Format**

Shows the command's specification format.

### **Function**

Describes the command's function.

### **Execution permissions**

Describes the user permissions needed to execute the command.

### **Prerequisites for command execution**

Describes any prerequisites required for command execution.

### **Storage directory**

Shows the command's storage directory.

### **Arguments**

Describes each of the command's arguments.

### **Notes**

Provides additional information about the command.

### **Return value**

Describes the command's return values.

### **Output format**

Shows examples of the output from execution of the command.

---

## List of commands

---

The table below lists and describes the commands used to configure and run the system.

*Table 7-1:* List of commands used to configure and run the system

No.	Command name	Function
1	sdpcql	Registers a query group into the SDP server.
2	sdpcqldel	Deletes a query group on inactive or shutdown status that is registered in the SDP server.
3	sdpcqlstart	Starts a query group on inactive or shutdown status that is registered in the SDP server.
4	sdpcqlstop	Terminates a query group that is running on the SDP server.
5	sdpls	Displays information about query groups registered in the SDP server.
6	sdpsetup	Sets up an operating environment.
7	sdpstart	Starts the SDP server.
8	sdpstartap	Starts specified standard adaptors or a specified custom adaptor to run in the RMI connection mode.
9	sdpstartinpro	Starts specified standard adaptors or a specified custom adaptor to run in the in-process connection mode.
10	sdpstop	Shuts down the SDP server.
11	sdpstopap	Terminates specified standard adaptors running in the RMI connection mode.
12	sdpstopinpro	Terminates specified standard adaptors or a specified custom adaptor running in the in-process connection mode.
13	sdptpls	Displays information about the tuples collected in tuple log files.
14	sdptplput	Reloads tuples collected in the tuple log files to the input stream queue.
15	sdptrced	Edits and outputs API trace information.

---

## sdpcql (registers a query group)

---

### Format

```
sdpcql { [-autostart] query-group-property-file-name | -help }
```

### Function

This command registers a specified query group into the SDP server.

The command performs syntax analysis of all queries in the query definition file specified in the query group property file. If the command detects an error in the file, it outputs a syntax error message to the log file and console, then resumes syntax analysis of the queries in the file.

Note that the registered query group is not retained after the SDP server is shut down.

### Execution permissions

Administrator user

### Prerequisites for command execution

To execute this command, you need the following files:

- System configuration property file (*system\_config.properties*)
- JavaVM options file for SDP servers (*jvm\_options.cfg*)
- Query group property file
- Stream property file
- Query definition file

### Storage directory

*working-directory*\bin\

### Arguments

- *-autostart*

After a query group is registered, this option starts executing the query group and accepting input tuples.

If this option is omitted, the command places the query group in inactive status after it is registered.

- *query-group-property-file-name*

Specifies the name of the query group property file for the query group that is to be registered.

For details about the query group property file, see 8.7 *Query group property file*.

- -help

Displays how to use the command.

If this option is specified, the command terminates without registering the query group or checking the options for errors.

## Notes

- The command returns the value 0 if there are no syntax errors in any of the CQL statements specified in the query definition file, and it is able to register the query group successfully.
- The command registers the name of the query group property file as the query group name. Therefore, a query group property file with the same name as a registered query group cannot be registered.
- If an error occurs during analysis of the query group property file, stream property file, or query definition file, the defined streams and queries are not registered.
- A stream or query cannot be registered if it has the same name as a stream or query that has already been registered in the system.
- The query group is registered even if the message `A query group cannot be started` is displayed as a result of executing the command with the `-autostart` option specified. In such a case, eliminate the cause of the error and then start the query group by executing `sdpcqlstart` command.
- If the command's execution is canceled by pressing **Ctrl + C**, the message `KFSP52205-E (internal conflict)` might be displayed the next time the `sdpcql` command is executed.

## Return value

Value	Meaning
0	The command terminated normally.
1	The command resulted in an error.

---

## sdpcqldel (deletes a query group)

---

### Format

sdpcqldel {*query-group-name*|-help}

### Function

This command deletes a query group on inactive or shutdown status that is registered in the SDP server.

### Execution permissions

Administrator user

### Prerequisites for command execution

To execute this command, you need the following files:

- System configuration property file (*system\_config.properties*)
- JavaVM options file for SDP servers (*jvm\_options.cfg*)

### Storage directory

*working-directory*\bin\

### Arguments

- *query-group-name*

Specifies the name of the query group that is to be deleted.

- -help

Displays how to use the command.

If this option is specified, the command terminates without deleting the query group or checking the options for errors.

### Notes

None

### Return value

Value	Meaning
0	The command terminated normally.
1	The command resulted in an error.



---

## sdpcqlstart (starts a query group)

---

### Format

```
sdpcqlstart {[-clear] [-reload] query-group-name|-help}
```

### Function

This command starts a query group on inactive or shutdown status that is registered in the SDP server.

### Execution permissions

Administrator user

### Prerequisites for command execution

To execute this command, you need the following files:

- System configuration property file (`system_config.properties`)
- JavaVM options file for SDP servers (`jvm_options.cfg`)

### Storage directory

*working-directory*\bin\

### Arguments

- `-clear`

Discards any tuples that are in the output stream queue before the command starts the query group.

If this option is omitted, the command starts the query group without discarding tuples that were stored in the output stream queue during the previous execution.

- `-reload`

Reloads the contents of the query group property file and stream property file, applies the settings, and then starts the query group.

You use this option when you want to change the property values of the query group that were specified when the query group was registered.

For details about the query group property file and the property values that can be changed when the query group is started, see [8.7 Query group property file](#). For details about the stream property file and the property values that can be changed when the query group is started, see [8.8 Stream property file](#).

- *query-group-name*

Specifies the name of the query group to be started.

sdpcqlstart (starts a query group)

- `-help`

Displays how to use the command.

If this option is specified, the command terminates without starting the query group or checking the options for errors.

## Notes

- If the `-reload` option is specified and the definition analysis of the query group property file results in an error, the query group is not started.
- If a query group is started with the `-reload` option specified, but startup processing results in an error, the settings in the query group property file and stream property file do not take effect.
- If connection is established from the input adaptor to the input stream and this command is executed with the `-clear` option specified, tuples in the data reception application connection queue in the input stream are also discarded.

## Return value

Value	Meaning
0	The command terminated normally.
1	The command resulted in an error.

---

## sdpcqlstop (terminates a query group)

---

### Format

```
sdpcqlstop { [-force] query-group-name | -help }
```

### Function

This command terminates a query group that is running on the SDP server. Unless the `-force` option is specified, the command processes any tuples that are in the input stream queue when the command is entered, and then terminates the query group.

### Execution permissions

Administrator user

### Prerequisites for command execution

To execute this command, you need the following files:

- System configuration property file (`system_config.properties`)
- JavaVM options file for SDP servers (`jvm_options.cfg`)

### Storage directory

*working-directory*\bin\

### Arguments

- `-force`

Terminates the query group without processing tuples in the input stream queue. Tuples that are in the input stream queue are discarded.

- *query-group-name*

Specifies the name of the query group to be terminated.

- `-help`

Displays how to use the command.

If this option is specified, the command terminates without terminating the query group or checking the options for errors.

### Notes

- If tuple send processing (including by the `putEnd` method) is executed while normal termination processing by this command is underway, the send processing might be placed on hold until the termination processing is completed.
- If you execute this command more than once, the command might not return

sdpcqlstop (terminates a query group)

control until the current termination processing is completed.

- If this command is executed without the `-force` option specified while tuple send processing is underway, normal termination processing is executed after tuple send processing is completed. If this command is executed with the `-force` option specified while tuple send processing is underway, the tuples under send processing might remain in the input stream queue. Tuples remaining in the input stream queue will be discarded when the next query group is started.

### Return value

Value	Meaning
0	The command terminated normally.
1	The command resulted in an error.

---

## sdpls (displays the status of query groups)

---

### Format

```
sdpls {{-all| query-group-name [query-group-name...]}|-help}
```

### Function

This command displays information about query groups registered in the SDP server.

### Execution permissions

Administrator user

### Prerequisites for command execution

To execute this command, you need the following files:

- System configuration property file (`system_config.properties`)
- JavaVM options file for SDP servers (`jvm_options.cfg`)

### Storage directory

*working-directory*\bin\

### Arguments

- `-all`

Displays information about all query groups registered in the SDP server.

- *query-group-name* [*query-group-name...*]

Specifies the names of the query groups whose information is to be displayed. If you specify multiple query group names, delimit their names with a single-byte space.

- `-help`

Displays how to use the command.

If this option is specified, the command terminates without displaying stream and query group information or checking the options for errors.

### Notes

- If display of all query groups specified in the argument is successful, the command returns the value 0.
- If you execute this command with multiple query group names specified and an error occurs while the command is acquiring information about one of the query groups, the command continues displaying information about the query groups that can be processed successfully. In such a case, the command returns the value

sdpls (displays the status of query groups)

1.
  - For query groups that are being registered, the command displays ----/--/--  
-- : -- : -- as the start time. For query groups that are executing, the command displays ----/--/-- -- : -- : -- as the termination time.
  - Once the total number of tuples stored in the stream queue exceeds the maximum value (9,223,372,036,854,775,807), the value will no longer be updated.
  - Even if a query group's name is specified more than once in the argument, information about that query group is output only once.
  - The total number of tuples stored in the stream queue and the maximum number of tuples retained in the stream queue are values from start to termination of the query group; these values are reset when the query group is terminated and then re-started. These values are also reset when the `putEnd` method is issued for all input streams in the query group. In such a case, the following values are set:
    - Total number of tuples stored in the stream queue: 0
    - Maximum number of tuples that can be retained in the stream queue:  
Number of tuples retained at the start
    - Maximum number of tuples held by the timestamp adjustment function: 0
    - Number of tuples discarded by the timestamp adjustment function: 0
  - The total number of tuples stored in the stream queue includes tuples for control that are created internally. Therefore, the number of tuples displayed might be greater than the number of tuples that were sent in the following cases:
    - The `putEnd` method was issued from a custom adaptor.
    - The query group terminated normally.
  - When connection is established from the input adaptor to the input stream, the number of tuples retained in the stream from the input adaptor might be displayed as the maximum and current numbers of tuples retained in the stream queue. Therefore, the displayed value might be greater than the actual tuples retained in the SDP server, such as when a retrieval of tuples from the input adaptor is delayed.

### Return value

Value	Meaning
0	The command terminated normally.
1	The command resulted in an error.

## Output format

```

*** Total Information ***
Total Query Group Count : aaaaaaaaaa
Total Stream Count      : bbbbbbbbbb
Total Query Count       : ccccccccc
-----
*** Query Group Information ***
Query Group Name       : dd...dd
Time Stamp Mode        : ee...ee
Tuple Log Mode         : ff...ff
Stream Count           : gggggggggg
Query Count            : hhhhhhhhhh
Status                 : ii...ii
Start Time              : jjjj/jj/jj jj:jj:jj
End Time                : kkkk/kk/kk kk:kk:kk

**** Stream Information ****
Stream Name            : ll...ll
Data Count             : mm...mm
Queue Count            : nnnnnnnnnn
Max Count              : oooooooooo
Accuracy Adjust        : pp...pp
Tuple Count            : qq qq qq qq qq
Max Tuple Count        : rrrrrrrrrr
Reject Tuple Count     : ssssssssss

**** Query Information ****
Query Name             : tt...tt
Data Count             : uu...uu
Queue Count            : vvvvvvvvvv
Max Count              : wwwwwwwwww

```

### Legend:

- 1: Displayed only when the -a11 option is specified.
- 2: Displayed for each query group.
- 3: Displayed for each stream registered in the query group.
- 4: Displayed only when the timestamp mode for the query group is DATASOURCE.
- 5: Displayed for each query registered in the query group.

The table below explains the variables used in the output format.

Information category	Variable	Meaning
Information output when the -a11 option is specified	aaaaaaaaaa	Total number of query groups (10-digit decimal number)

sdpls (displays the status of query groups)

Information category	Variable	Meaning
	<i>bbbbbbbbbb</i>	Total number of streams (10-digit decimal number)
	<i>cccccccccc</i>	Total number of queries (10-digit decimal number)
Query group information	<i>dd...dd</i>	Query group name
	<i>ee...ee</i>	Timestamp mode: <ul style="list-style-type: none"> <li>• <code>DataSource</code>: Data source mode</li> <li>• <code>Server</code>: Server mode</li> </ul>
	<i>ff...ff</i>	Whether or not the command can be executed in the server mode: <ul style="list-style-type: none"> <li>• <code>true</code>: Executable</li> <li>• <code>false</code>: Not executable</li> </ul>
	<i>gggggggggg</i>	Number of streams that belong to the group (10-digit decimal number)
	<i>hhhhhhhhhh</i>	Number of queries that belong to the group (10-digit decimal number)
	<i>ii...ii</i>	Status of the query group: <ul style="list-style-type: none"> <li>• <code>Stop</code>: Query group has terminated</li> <li>• <code>Running</code>: Query group is executing</li> <li>• <code>Hold</code>: Query group has shut down due to a runtime error or queue overflow</li> <li>• <code>FatalHold</code>: Query group is in an unresumable status</li> <li>• <code>Stopping</code>: Query group is under termination processing</li> <li>• <code>PreparingForStop</code>: Tuples in the input queue are being processed or the current tuple processing is continuing, due to a query group termination request</li> <li>• <code>Starting</code>: Query group is starting</li> <li>• <code>Deleting</code>: Query group is being deleted</li> <li>• <code>Holding</code>: Query group is being shut down</li> </ul>
	<i>jjj/ij/jj Δjj:jj:jj</i>	Query group start time
	<i>kkkk/kk/kk Δkk:kk:kk</i>	Query group termination time
Stream information	<i>ll...ll</i>	Stream name
	<i>mm...mm</i>	Total number of tuples stored in the stream queue (19-digit decimal number)
	<i>nnnnnnnnnn</i>	Number of tuples retained in the current stream queue (10-digit decimal number)



Information category	Variable	Meaning
	<i>oooooooooooo</i>	Maximum number of tuples retained in the stream queue (10-digit decimal number)
	<i>pp...pp</i>	Status of the timestamp adjustment function: <ul style="list-style-type: none"> <li>• <code>stop</code>: Timestamp adjustment function is inactive</li> <li>• <code>running</code>: Timestamp adjustment function is running</li> </ul>
	<i>qqqqqqqqqq</i>	Number of tuples held by the timestamp adjustment function (10-digit decimal number)
	<i>rrrrrrrrrr</i>	Maximum number of tuples retained by the timestamp adjustment function (10-digit decimal number)
	<i>sssssssss</i>	Number of tuples discarded by the timestamp adjustment function (10-digit decimal number)
Query information	<i>tt...tt</i>	Query name
	<i>uu...uu</i>	Total number of tuples stored in the output stream queue (19-digit decimal number)
	<i>vvvvvvvvv</i>	Number of tuples retained in the current output stream queue (10-digit decimal number)
	<i>wwwwwwwwwww</i>	Maximum number of tuples retained in the output stream queue (10-digit decimal number)

Legend:

Δ: Single-byte space

---

## **sdpsetup (sets up an operating environment)**

---

### **Format**

`sdpsetup working-directory`

### **Function**

This command set up an operating environment.

### **Execution permissions**

Administrator user

### **Prerequisites for command execution**

None

### **Storage directory**

`installation-directory\bin\`

### **Arguments**

#### ■ *working-directory*

Specifies the absolute path of the working directory to be used to run the system. Make sure that the specified directory can be referenced by the administrator user.

- If the specified directory is not found  
The command creates a new directory.
- If the specified directory already exists

The command overwrites the `bin` directory in the specified directory.

If the specified directory contains any directories other than `bin`, those directories are not overwritten. The command creates a new directory only if the specified directory contains only a `bin` directory.

For details about the directory structure after setup, see *3.2 Directory structure*.

The access permissions for the working directory and its subdirectories and files are set when this command executes. The command does not change access permissions for directories and files already in existence at the time of setup.

If the argument is omitted or an error is detected in the specified argument, the command displays the `Usage` message and terminates.

If the argument that is specified is valid, the command displays the following message:

```

KFSP91030-I === Setup information:
KFSP91032-I Setup directory: working-directory
KFSP91033-Q Do you want to continue? (y/n)

```

If you choose **y**, the command starts setup.

If you choose **n**, the command cancels setup.

## Notes

- If you specify the argument more than once, the command accepts only the first argument specified, ignores the other specifications, and resumes setup.
- None of the installation directory's subdirectories can be specified. If such a subdirectory is specified, an error results.
- The path name of the working directory must consist of only the following characters:

Single-byte alphanumeric characters, backslash (\), period (.), underscore (\_), colon (:)

## Return value

Value	Meaning
0	Setup was completed successfully.
1	An error occurred during setup.
2	Setup was canceled by the user.

---

## sdpstart (starts the SDP server)

---

### Format

`sdpstart`

### Function

This command starts the SDP server.

### Execution permissions

Administrator user

### Prerequisites for command execution

To execute this command, you need the following files:

- System configuration property file (`system_config.properties`)
- JavaVM options file for SDP servers (`jvm_options.cfg`)

### Storage directory

`working-directory\bin\`

### Arguments

None

### Notes

- When this command executes, it moves the current directory to the working directory and then starts the server. Therefore, if you specify a relative path for the class path in the JavaVM options file for SDP servers, you must specify the correct path relative to the working directory.
- If the port number is being used, an error results. In such a case, you must change the port number in specified `system_config.properties`.
- Only one SDP server can be run in a working directory.
- Once the SDP server starts, control will not be returned to this command until the SDP server is shut down.
- If JavaVM terminates abnormally, JavaVM's return value is set.
- If the `exit` method is executed in a process that is implemented by the user using a custom adaptor in the in-process connection mode, this command returns the value specified by the `exit` method as its return value.

**Return value**

<b>Value</b>	<b>Meaning</b>
0	The server terminated normally.
1	The server shut down because the system detected an error or because the user forcibly shut it down.

---

## sdpstartap (starts RMI-connection adaptors)

---

### Format

**For starting the standard adaptors in the RMI connection mode:**

```
sdpstartap jp.co.Hitachi.soft.sdp.adaptor.AdaptorManager  
adaptor-group-name
```

**For starting a custom adaptor in the RMI connection mode:**

```
sdpstartap [-clientcfg path-of-jvm_client_options.cfg]  
Java-application-class-name  
[ arguments-passed-to-main-method... ]
```

### Function

This command starts specified standard adaptors or a custom adaptor in the RMI connection mode.

### Execution permissions

Administrator user

### Prerequisites for command execution

To execute this command, you need the following files:

- System configuration property file (*system\_config.properties*)
- JavaVM options file for SDP servers (*jvm\_options.cfg*)

### Storage directory

*working-directory*\bin\

### Arguments

- *adaptor-group-name*

Specifies the name of the RMI group to be started. This must be an adaptor group name specified in the name attribute in the RMI group definition (<RMIGroupDefinition> tag) in the adaptor configuration definition file.

- *-clientcfg path-of-jvm\_client\_options.cfg*

If a separate JavaVM options file for RMI connections is used for the Java application to perform send/receive operations, this argument specifies the absolute or relative path of that file. If you specify a relative path, specify the path that is relative to the directory at which the command is entered.

If this option is omitted, the command uses `working-directory\conf\jvm_client_options.cfg`.

- *Java-application-class-name*

Specifies the name of the Java application class to be started. A Java application class name that begins with - (hyphen) cannot be specified.

- *arguments-passed-to-main-method*

Specifies arguments that are to be passed to the `main` method of the Java application class.

## Notes

- If the arguments are omitted or an erroneous argument is specified, the command displays a message that shows how to use the command and then terminates. If you are starting standard adaptors in the RMI connection mode, `Class` is replaced in the displayed message with `jp.co.Hitachi.soft.sdp.adaptor.AdaptorManager` and `[Args...]` is replaced with adaptor group name.

- If this command is executed for a custom adaptor, it executes the `main` method in the main class of the Java application. The `main` method must be declared as `public static void main(String[])`.

For details about creating custom adaptors in the RMI connection mode, see the manual *uCosminexus Stream Data Platform - Application Framework Application Development Guide*.

- The value `1` must not be used as the termination code for a custom adaptor.

For details about creating custom adaptors in the RMI connection mode, see the manual *uCosminexus Stream Data Platform - Application Framework Application Development Guide*.

- In the case of a custom adaptor, if the main class is stored in a `jar` file, you must specify the class path of the `jar` file in `jvm_client_options.cfg` as shown below:

```
SDP_CLASS_PATH=.\home\abc\sdpclient.jar
```

Note that the command does not use the current directory as the default class path. To add the current directory to the class path to be used, specify the following in `jvm_client_options.cfg`:

```
SDP_CLASS_PATH=.
```

For details about how to specify the settings, see *8.5 JavaVM options file for RMI connections (jvm\_client\_options.cfg)*.

- In order to start multiple custom adaptors, you must provide multiple JavaVM option files for RMI connection so that the prefix name of the Java log files is not

sdpstartap (starts RMI-connection adaptors)

duplicated.

For details about how to specify the settings, see *8.5 JavaVM options file for RMI connections (jvm\_client\_options.cfg)*.

### Return value

Value	Meaning
1	A specified argument is invalid.
Other than 1	A termination code of the Java application is returned.



---

## sdpstartinpro (starts in-process-connection adaptors)

---

### Format

**For starting the standard adaptors in the in-process connection mode:**

```
sdpstartinpro {adaptor-group-name|-help}
```

**For starting a custom adaptor in the in-process connection mode:**

```
sdpstartinpro {name-of-in-process-connection-custom-adaptor|-help}
```

### Function

This command starts specified standard adaptors or a specified custom adaptor in the in-process connection mode.

If you execute this command with a custom adaptor in the in-process connection mode specified, the command loads the main class specified in the in-process connection property file, registers it into the SDP server, and then executes the `execute` method in the `StreamInprocessUP` interface implementation class.

### Execution permissions

Administrator user

### Prerequisites for command execution

To execute this command, you need the following files:

- System configuration property file (`system_config.properties`)
- JavaVM options file for SDP servers (`jvm_options.cfg`)

### Storage directory

*working-directory*\bin\

### Arguments

- *adaptor-group-name*

Specifies the name of the in-process group to be started. This must be an adaptor group name specified in the `name` attribute in the in-process group definition (`<InprocessGroupDefinition>` tag) in the adaptor configuration definition file (`AdaptorCompositionDefinition.xml`).

- *name-of-in-process-connection-custom-adaptor*

Specifies the name of the custom adaptor in the in-process connection mode that is to be started. This must be a name specified in `user_app.name-of-in-process-connection-custom-adaptor.properties`.

- -help

Displays how to use the command.

If this option is specified, the command terminates without starting the standard adaptors in the in-process connection mode or the custom adaptor in the in-process connection mode, or checking the options for errors.

## Notes

- If the arguments are omitted or an erroneous argument is specified, the command displays a message that shows how to use the command and then terminates. If you are starting the standard adaptors in the in-process connection mode, `InprocessApName` is replaced in the displayed message with `adaptor group name`.
- You can execute this command only while the SDP server is running.
- If you execute the command for a custom adaptor in the in-process connection mode that is running, an error results.
- If `user_app.adaptor-name.properties` is not found or the analysis of `user_app.adaptor-name.properties` results in an error, the custom adaptor in the in-process connection mode is not started.
- If you execute this command with a custom adaptor in the in-process connection mode specified, the command starts the `execute` method with the `StreamInprocessUP` interface and then returns control without waiting for completion of the method. Therefore, the command returns the value 0 even if an error occurs during processing within the `execute` method.
- Once you use this command to start a custom adaptor in the in-process connection mode, the custom adaptor cannot be restarted until the `sdpstopinpro` command has terminated even if the `execute` method implemented by an application has terminated.

## Return value

Value	Meaning
0	The command terminated normally.
1	The command resulted in an error.

---

## sdpstop (stops the SDP server)

---

### Format

```
sdpstop { [-force] | -help }
```

### Function

This command shuts down the SDP server. If you execute the command with no argument specified, the command first terminates all registered query groups and adaptors in the in-process connection mode, and then shuts down the SDP server.

### Execution permissions

Administrator user

### Prerequisites for command execution

To execute this command, you need the following files:

- System configuration property file (`system_config.properties`)
- JavaVM options file for SDP servers (`jvm_options.cfg`)

### Storage directory

*working-directory*\bin\

### Arguments

- `-force`

Forcibly shuts down the SDP server. You specify this option when the SDP server cannot be shut down by this command with no argument specified.

- `-help`

Displays how to use the command.

If this option is specified, the command terminates without shutting down the SDP server or checking the options for errors.

### Notes

- When `-force` is specified, the timeout value for termination processing on each query execution thread is 30 seconds.  
If this command is executed with `-force` specified and a timeout occurs during termination processing on a query execution thread, some of the trace information for the query execution thread resulting in the timeout might not be collected.
- If this command is executed with no argument specified and a query group or in-process-connection adaptor is still running, the command will not be able to

sdpstop (stops the SDP server)

terminate normally; in such a case, the command will return the value 1.

- If an error occurs while a specific query group is being terminated, the command continues with the termination processing on other query groups and the system. If the system is shut down, the command returns the value 0.

### Return value

Value	Meaning
0	The command terminated normally.
1	The command resulted in an error.

---

## sdpstopap (terminates RMI-connection adaptors)

---

### Format

```
sdpstopap { [-force] adaptor-group-name | -help }
```

### Function

This command terminates specified standard adaptors in the RMI connection mode.

### Execution permissions

Administrator user

### Prerequisites for command execution

None

### Storage directory

*working-directory*\bin\

### Arguments

■ *-force*

Forcibly terminates the adaptors that belong to the specified RMI group. The command discards all data being processed and terminates the adaptor group immediately without executing callback termination processing.

■ *adaptor-group-name*

Specifies the name of the RMI group to be terminated. This must be an adaptor group name specified in the `name` attribute in the RMI group definition (`<RMIGroupDefinition>` tag) in the adaptor configuration definition file (`AdaptorCompositionDefinition.xml`).

■ *-help*

Displays how to use the command.

If this option is specified, the command terminates without terminating the RMI-connection adaptors or checking the options for errors.

### Notes

- You can execute this command only for standard adaptors running in the RMI connection mode. If you attempt to execute the command while no standard adaptors are running in the RMI connection mode, an error results.
- If you execute the command with *-force* specified, an error message might be displayed depending on the command's execution timing. This message has no

sdpstopap (terminates RMI-connection adaptors)

effect on the operation.

### Return value

Value	Meaning
0	The command terminated normally.
1	The command resulted in an error.

---

## sdpstopinpro (terminates in-process-connection adaptors)

---

### Format

**For terminating standard adaptors in the in-process connection mode:**

```
sdpstopinpro {adaptor-group-name | -help}
```

**For terminating custom adaptors in the in-process connection mode:**

```
sdpstopinpro {name-of-in-process-connection-custom-adaptor | -help}
```

### Function

This command terminates specified standard adaptors or a specified custom adaptor in the in-process connection mode.

If you execute this command with the name of a custom adaptor in the in-process connection mode specified, the command executes the `stop` method with the `StreamInprocessUP` interface that is registered in the SDP server.

### Execution permissions

Administrator user

### Prerequisites for command execution

To execute this command, you need the following files:

- System configuration property file (`system_config.properties`)
- JavaVM options file for SDP servers (`jvm_options.cfg`)

### Storage directory

*working-directory*\bin\

### Arguments

- *adaptor-group-name*

Specifies the name of the in-process group to be terminated. This must be an adaptor group name specified in the `name` attribute in the in-process group definition (`<InprocessGroupDefinition>` tag) in the adaptor configuration definition file (`AdaptorCompositionDefinition.xml`).

- *name-of-in-process-connection-custom-adaptor*

Specifies the name of the custom adaptor in the in-process connection mode that is to be terminated. This must be a name specified in `user_app.adaptor-name.properties`.

- `-help`

Displays how to use the command.

If this option is specified, the command terminates without terminating the standard adaptors or the custom adaptor in the in-process connection mode or checking the options for errors.

## Notes

- In the case of a custom adaptor, make sure that you terminate within the `stop` method the user thread that was started by the `execute` method with the `StreamInprocessUP` interface. If the `stop` method does not contain termination processing or the user thread contains processing that cannot be terminated by the `stop` method (such as an infinite loop), the user thread might remain without being terminated.
- In the case of a custom adaptor, if the `execute` method with the `StreamInprocessUP` interface contains processing such as an infinite loop, an execution thread of the `execute` method itself might remain.
- This command does not return control until the `stop` method has terminated. Therefore, if you specify processing in the `stop` method that is time-consuming, the command might not return control.
- Even if the `stop` method results in an error, the termination processing continues.

## Return value

Value	Meaning
0	The command terminated normally.
1	The command resulted in an error.



---

## sdptpls (displays tuple information)

---

### Format

```
sdptpls {-file tuple-log-file [ tuple-log-file...]  
        [-time [start-time] [, end-time]] [-info {file|tuple}]  
        [-csv] [-data] |-file tuple-log-file  
        [ tuple-log-file... ] -check|-help}
```

### Function

This command displays information about the tuples collected in tuple log files.

### Execution permissions

Administrator user

### Prerequisites for command execution

None

### Storage directory

*working-directory*\bin\

### Arguments

- -file *tuple-log-file*[ *tuple-log-file*...]

Specifies the paths of the tuple log files to be displayed.

If you specify multiple tuple log files, delimit the files with a single-byte space. In such a case, information about the specified files is displayed in the order specified.

- -time [*start-time*] [, *end-time*]

Specifies in the following format a start time and end time for the tuples that are to be displayed:

*hhmmsslll* [*MMDD* [*YYYY*]]

*hh*: Hour (00 ≤ *hh* ≤ 23)

*mm*: Minute (00 ≤ *mm* ≤ 59)

*ss*: Second (00 ≤ *ss* ≤ 59)

*lll*: Millisecond (000 ≤ *lll* ≤ 999)

*MM*: Month (01 ≤ *MM* ≤ 12)

*DD*: Day (01 ≤ *DD* ≤ 31)

*YYYY*: Year (4-digit calendar year)

sdptp1ls (displays tuple information)

- Specify this argument in single-byte numeric characters.
- If the year is omitted from the start or end time, the command assumes the specified month, day, and time in the current year.
- If you omit specification of year, month, and day, the command assumes the specified time on the current day, in the current month, in the current year.
- You cannot omit only the month and day, or only the month, or only the day. If this rule is violated, an option error results.
- If you want to omit the month or day, omit all of year, month, and day.

The *start-time* and *end-time* values are the start and end times of tuple collection. You can use this command to check the tuple collection times.

Specify *start-time* and *end-time* delimited with a comma and without any spaces.

The following describes the tuples that are displayed according to the specified time:

- When *start-time* is omitted:  
The command displays all tuples whose time precedes the *end-time*.
- When *end-time* is omitted:  
The command displays all tuples whose time follows the *start-time*.
- When *start-time* and *end-time* are both omitted:  
The command displays all tuples.
- When the same value is specified in *start-time* and *end-time*:  
The command displays only those tuples whose collection time is the specified time.
- When the specified *start-time* is subsequent to the *end-time*:  
An option error occurs.

Note that if you specify `file` in the `-info` option, this argument is ignored.

- `-info {file|tuple}`  
Specifies the type of information to be displayed.
  - `file`: Displays file information.
  - `tuple`: Displays tuple information.If you omit this argument, the command displays tuple information.
- `-csv`  
Displays the information in the CSV file format.

If you specify this option, the command treats the date and time in time information (such as a timestamp) as a single data item.

- `-data`

Displays the contents of the tuple data objects.

If you specify `file` in the `-info` option, this option is ignored.

If you specify the `-csv` option, a tuple's data object that is displayed by this option is treated as a single data item.

- `-check`

Checks the tuple log files for any erroneous tuples.

- `-help`

Displays how to use the command.

If this option is specified, the command terminates without displaying tuple information or checking the options for errors.

## Notes

If a specified tuple log file has been acquired from an environment that uses a non-default character encoding, the character string information in the tuples might not be restored correctly.

## Return value

Value	Meaning
0	The command terminated normally.
1	The command resulted in an error.

## Output format

- When the `-info file` option is specified:

When the `-csv` option is omitted:

```
File Name       : aa...aa
Version        : bb-bb-bb
Query Group Name : cc...cc
Stream Name    : dd...dd
Time Stamp Mode : ee...ee
Stream Type    : ff..ff
Base Time      : gggg/gg/gg gg:gg:gg ggg
Update Time   : hhhh/hh/hh hh:hh:hh hhh
```

... (repeated for each file)

sdptp1ls (displays tuple information)

When the `-csv` option is specified:

File Name, Version, Query Group Name, Stream Name, Time Stamp  
 Mode, Stream Type, Base Time, Update  
 Time *aa...aa, bb-bb-bb, cc...cc, dd...dd, ee...ee, ff...ff, gggg/gg/gg gg:gg:gg  
 ggg, hhhh/hh/hh hh:hh:hh hhh...* (repeated for each file)

The table below explains the variables used in the output format.

Variable	Meaning
<i>aa...aa</i>	Path of the tuple log file (absolute path)
<i>bb-bb-bb</i>	Information about the Stream Data Platform - AF version that acquired the tuple logs
<i>cc...cc</i>	Name of the query group that acquired the tuple logs (string of 1 to 64 characters)
<i>dd...dd</i>	Name of the stream that acquired the tuple logs (string of 1 to 100 characters)
<i>ee...ee</i>	Timestamp mode of the query group that acquired the tuple logs: <ul style="list-style-type: none"> <li>• DataSource: Data source mode</li> <li>• Server: Server mode</li> </ul>
<i>ff...ff</i>	Type of stream that acquired the tuple logs: <ul style="list-style-type: none"> <li>• Input: Input stream</li> <li>• Output: Output stream</li> </ul>
<i>gggg/gg/gg gg:gg:gg ggg</i>	<p>Tuple log collection reference time (year/month/day hour:minute:second millisecond)</p> <p>The tuple log collection reference time is the time information that is output to the tuple log file when the query group is initialized. All tuple log files with the same tuple log collection reference time become the tuple logs required for query re-execution.</p>
<i>hhhh/hh/hh hh:hh:hh hhh</i>	Update start time for the tuple log file (year/month/day hour:minute:second millisecond)

■ When the `-info tuple` option is specified:

When the `-csv` and `-data` options are both omitted:

File Name : *ii...ii*  
 Record No Record Time Time Stamp Status  
*jj...jj kkkk/kk/kk kk:kk:kk kkk llll/ll/ll ll:ll:ll lll mm...mm*  
*...* (repeated for each tuple)  
*...* (repeated for each file)

When the `-csv` option is omitted and the `-data` option is specified:

```
File Name : ii...ii
Record No Record Time           Time Stamp           Status
Data
jj...jj   kkkk/kk/kk kk:kk:kk kkk lll/ll/ll ll:ll:ll lll mm...mm nn...nn
... (repeated for each tuple)

... (repeated for each file)
```

When the `-csv` option is specified and the `-data` option is omitted:

```
File Name, Record No, Record Time, Time Stamp, Status
ii...ii, jj...jj, kkkk/kk/kk kk:kk:kk kkk, lll/ll/ll ll:ll:ll lll, mm...mm
... (repeated for each tuple)

... (repeated for each file)
```

When the `-csv` and `-data` options are both specified:

```
File Name, Record No, Record Time, Time Stamp, Status, Data
ii...ii, jj...jj, kkkk/kk/kk kk:kk:kk kkk, lll/ll/ll ll:ll:ll lll, mm...mm, nn...nn
... (repeated for each tuple)

... (repeated for each file)
```

The table below explains the variables used in the output format.

variable	Meaning
<i>ii...ii</i>	Path of the tuple log file (absolute path)
<i>jj...jj</i>	Serial number assigned to a tuple log (19-digit decimal number)
<i>kkkk/kk/kk kk:kk:kk kkk</i>	Time the tuple log was acquired (year/month/day hour:minute:second millisecond)
<i>llll/ll/ll ll:ll:ll lll</i>	Timestamp in the tuple (year/month/day hour:minute:second millisecond)
<i>mm...mm</i>	Tuple's status: <ul style="list-style-type: none"> <li>Put: Normal</li> <li>TimeBack: Discarded due to out-of-sequence time</li> </ul>
<i>nn...nn</i>	Contents of data object ( <i>[column-1   column-2   . . .]</i> ) This information is displayed only when the <code>-data</code> option is specified.

■ When the `-check` option is specified:

```
File Name : oo...oo
First No Last No
```

sdptp11s (displays tuple information)

*pp...pp qq...qq*  
Lost Norr...rr-rr...rr  
... (repeated for each missing range)  
  
... (repeated for each file)

The table below explains the variables used in the output format.

<b>Variable</b>	<b>Meaning</b>
<i>oo...oo</i>	Path of the tuple log file (absolute path)
<i>pp...pp</i>	First serial number in the tuple log file (19-digit decimal number)
<i>qq...qq</i>	Last serial number in the tuple log file (19-digit decimal number)
<i>rr...rr-rr...rr</i>	Range of missing tuples ( <i>serial-number-of-first-missing-tuple-serial-number-of-last-missing-tuple</i> )

---

## sdptplput (reloads tuples)

---

### Format

```
sdptplput {-file tuple-log-file [tuple-log-file...]  
           [-time [start-time] [, end-time]] [-interval loading-interval]  
           [-count number-of-tuples-to-be-loaded] | -help}
```

### Function

This command reloads tuples collected in tuple log files into the input stream queue.

### Execution permissions

Administrator user

### Prerequisites for command execution

To execute this command, you need the following files:

- System configuration property file (*system\_config.properties*)
- JavaVM options file for SDP servers (*jvm\_options.cfg*)

### Storage directory

*working-directory*\bin\

### Arguments

- -file *tuple-log-file* [*tuple-log-file*...]

Specifies the paths of the tuple log files to be reloaded.

If you specify multiple tuple log files, delimit the files with a single-byte space.

If you specify multiple files, note the following:

- If you specify at the same time multiple tuple log files that do not all have the same tuple log collection reference time, a file specification error results.

The tuple log collection reference time is the time information that is output when a query group is initialized. A series of tuple logs required for query re-execution is those tuple log files with the same tuple log collection reference time.

You can use the `sdptpls` command to obtain the tuple log collection reference time of each tuple log file.

- If tuple logs were output to multiple files or are to be reloaded to a query group that has multiple input streams, no error results even if not all the collected tuple log files are specified. The command uses only the specified files to reload tuples.
- If the same tuple log file is specified more than once, a file specification error

results.

- If multiple tuple log files in the same stream queue are specified, the tuple log files are reloaded in chronological order of the file update times.

You can use the `sdptpls` command to obtain the file update time of each tuple log file.

- `-time [start-time] [, end-time]`

Specifies in the following format a start time and end time for the tuples that are to be reloaded:

`hhmmsslll [MMDD [YYYY] ]`

*hh*: Hour ( $00 \leq hh \leq 23$ )

*mm*: Minute ( $00 \leq mm \leq 59$ )

*ss*: Second ( $00 \leq ss \leq 59$ )

*lll*: Millisecond ( $000 \leq lll \leq 999$ )

*MM*: Month ( $01 \leq MM \leq 12$ )

*DD*: Day ( $01 \leq DD \leq 31$ )

*YYYY*: Year (4-digit calendar year)

- Specify this argument in single-byte numeric characters.
- If the year is omitted from the start or end time, the command assumes the specified month, day, and time in the current year.
- If you omit specification of year, month, and day, the command assumes the specified time on the current day, in the current month, in the current year.
- You cannot omit only the month and day, or only the month, or only the day. If this rule is violated, an option error results.
- If you want to omit the month or day, omit all of year, month, and day.

The *start-time* and *end-time* values are the start and end times of tuple collection. You can use the `sdptpls` command to check the tuple collection times.

Specify *start-time* and *end-time* delimited with a comma and without any spaces.

The following describes the tuples that are reloaded according to the specified time:

- When *start-time* is omitted:

The command reloads all tuples whose time precedes the *end-time*.

- When *end-time* is omitted:

The command reloads all tuples whose time follows the *start-time*.



- When *start-time* and *end-time* are both omitted:  
The command reloads all tuples.
- When the same value is specified in *start-time* and *end-time*:  
The command reloads only those tuples whose collection time is the specified time.
- When the specified *start-time* is subsequent to the *end-time*:  
An option error occurs.
- `-interval loading-interval` (1 to 600000)  
Specifies in milliseconds an interval value for tuple reloading. If this argument is omitted, the command assumes 100 milliseconds.  
The command reloads as many tuples as specified in the `-count` option and then resumes reloading tuples after the interval specified here. If an overflow occurs in the input stream queue, the command reloads after the specified interval the tuples whose reloading failed.  
If the `-count` option is omitted, the loading interval specified in the `-interval` option is applicable only when an overflow occurs in the input stream queue.
- `-count number-of-tuples-to-be-loaded` (1 to 1048576)  
Specifies the number of tuples to be reloaded at one time.  
The command reloads as many tuples as specified in this argument and then waits for the interval specified in the `-interval` option before resuming reloading of tuples.  
If this argument is omitted, the command reloads all tuples at once without pause.
- `-help`  
Displays how to use the command.  
If this option is specified, the command terminates without reloading tuples or checking the options for errors.

## Notes

- This command cannot reproduce a timeout based on queuing of tuples in multiple input stream queues.
- If a specified tuple log file has been acquired from an environment that uses a non-default character encoding, the character string information in the tuples might not be restored correctly.
- When this command is executed, tuple logs are collected in the same manner as when the input adaptors are used to load the tuples. Before you reload tuples, you should first make a backup from the working directory of the tuple log files to be

used and then use that backup copy to execute the command.

- This command cannot reproduce an interval from reloading of the last input tuple to execution of the `putEnd` method. Therefore, when tuple log files collected in the server mode are used to re-execute queries, the result obtained from the time the last input tuple is loaded to the time the `putEnd` method is executed might not match the result obtained by executing the queries using the input adaptors.
- This command reloads tuples within the SDP server. Therefore, if you forcibly terminate the command process, the command continues to reload tuples. To cancel reloading of tuples, you must use the `sdpcqlstop` command to terminate the query group.
- When you execute this command with the `-interval` and `-count` options specified, any change to the query group status is detected at the interval specified in the `-interval` option. Therefore, if you terminate the query group during command execution, the reloading process might not be canceled immediately.

### Return value

Value	Meaning
0	The command terminated normally.
1	The command resulted in an error.

---

## sdptrced (edits trace information)

---

### Format

```
sdptrced {-file trace-file-name [ trace-file-name... ]
          [-time [start-time] [, end-time]]
          [-threadid thread-ID [ thread-ID...]] [-csv] | -help}
```

### Function

This command edits and outputs API trace information.

### Execution permissions

Administrator user

### Prerequisites for command execution

To execute this command, you need the following files:

- System configuration property file (`system_config.properties`)
- JavaVM options file for SDP servers (`jvm_options.cfg`)

### Storage directory

*working-directory*\bin\

### Arguments

- `-file trace-file-name [ trace-file-name... ]`

Specifies the absolute path names of the trace files to be edited. When you specify multiple trace file names, delimit the file names with a single-byte space.

- `-time [start-time] [, end-time]`

Specifies in the following format a start time and end time for the trace information that is to be output:

*hhmmsslll*[*MMDD*[*YYYY*]]

*hh*: Hour (00 ≤ *hh* ≤ 23)

*mm*: Minute (00 ≤ *mm* ≤ 59)

*ss*: Second (00 ≤ *ss* ≤ 59)

*lll*: Millisecond (000 ≤ *lll* ≤ 999)

*MM*: Month (01 ≤ *MM* ≤ 12)

*DD*: Day (01 ≤ *DD* ≤ 31)

YYYY: Year (4-digit calendar year)

- Specify this argument in single-byte numeric characters.
- If the year is omitted from the start or end time, the command assumes the specified month, day, and time in the current year.
- If you omit specification of year, month, and day, the command assumes the specified time on the current day, in the current month, in the current year.
- You cannot omit only the month and day, or only the month, or only the day. If this rule is violated, an option error results.
- If you want to omit the month or day, omit all of year, month, and day.

Specify *start-time* and *end-time* delimited with a comma and without any spaces.

The following describes the trace information that is displayed according to the specified time:

- When *start-time* is omitted:

The command outputs all trace information whose time precedes the *end-time*.

- When *end-time* is omitted:

The command outputs all trace information whose time follows the *start-time*.

- When *start-time* and *end-time* are both omitted:

The command displays all trace information.

- When the same value is specified in *start-time* and *end-time*:

The command outputs only the trace information with the specified time.

- When the specified *start-time* is subsequent to the *end-time*:

An option error occurs.

- `-threadid thread-ID [ thread-ID . . . ]`

Outputs only that trace information that was output by threads with the specified thread IDs.

Express a thread ID in single-byte numeric characters. The permitted value range is from 0 to 9223372036854775807. To specify multiple thread IDs, delimit them with the single-byte space.

- `-csv`

Displays the trace information editing results in the CSV file format.

- `-help`

Displays how to use the command.

If this option is specified, the command terminates without outputting trace information or checking the options for errors.

## Notes

- The trace information output as API trace information is not sorted in chronological order.
- If the size of the trace file specified in the `-file` option is large, the editing process might require an extended amount of time.

## Return value

Value	Meaning
0	The command terminated normally.
1	The command terminated abnormally.

## Output format

The output format is shown below. For an example of the output, see [6.3.2 Details of API trace information](#).

When the `-csv` option is omitted:

```
Ver      Kind  File
aa-aa-aa bbb  cc...cc
Date    Time  nSec  Current  Parent  Root
EventID
dddd/dd/dd ee:ee:ee ffffffff gg...gg-hh...hh
ii...ii-jj...jj kk...kk-ll...ll 0xmmmmmmmm
Rc      Data
noo...oo pp...pp
```

When the `-csv` option is specified:

```
Ver,Kind,File
aa-aa-aa,bbb,cc...cc
Date,Time,nSec,Current,Parent,Root,EventID,Rc,Data
dddd/dd/
dd,ee:ee:ee,fffffff,gg...gg-hh...hh,ii...ii-jj...jj,kk...
kk-ll...ll,0xmmmmmmmm,
noo...oo,pp...pp
```

The table below explains the variables used in the output format.

Variable	Meaning
<code>aa-aa-aa</code>	Information about the Stream Data Platform - AF version used to output the trace files

Variable	Meaning
<i>bbb</i>	Type of trace files: <ul style="list-style-type: none"> <li>• API: API trace information files</li> <li>• None: Other than Stream Data Platform - AF trace files</li> </ul>
<i>cc...cc</i>	Path (absolute path) of the trace file
<i>dddd/dd/dd</i>	Trace output date (year/month/day)
<i>ee:ee:ee</i>	Trace output time (hour:minute:second)
<i>fffffff</i>	Trace output time nanosecond (9-digit decimal number)
<i>gg...gg-hh...hh</i>	Serial number assigned to this trace (thread ID (19-digit decimal number) -processing serial number (10-digit decimal number))
<i>ii...ii-jj...jj<sup>#</sup></i>	Reserved area. This information is output in the format <i>19-digit-decimal-number-10-digit-decimal-number</i> .
<i>kk...kk-ll...ll</i>	Serial number assigned to the trace start point (thread ID (19-digit decimal number) -processing serial number (10-digit decimal number)) The trace start point is the time at which the tuple handled by the process that collected this trace was input (or generated within a query). Traces with the same serial number constitute a series of processing on the same tuple.
<i>mmmmmmmm</i>	Event ID (8-digit hexadecimal number): <ul style="list-style-type: none"> <li>• 0x00020000: Query processing (input of tuples)</li> <li>• 0x00020001: Query processing (output of tuples)</li> </ul>
<i>noo...oo<sup>#</sup></i>	Reserved area <i>n</i> indicates the sign part. This information is output as an 11-digit decimal number. A single-byte space is displayed in <i>n</i> .
<i>pp...pp</i>	Name of query that collected trace information (first 32 bytes of the query name)

#

In Stream Data Platform - AF 01-00, a single-byte space plus 10 digits of zeros are output.

## Chapter

---

# 8. SDP Server Definition Files

---

This chapter describes the SDP server definition files.

- 8.1 Format of SDP server definition file explanations
- 8.2 Notes about creating SDP server definition files
- 8.3 List of SDP server definition files
- 8.4 JavaVM options file for SDP servers (jvm\_options.cfg)
- 8.5 JavaVM options file for RMI connections (jvm\_client\_options.cfg)
- 8.6 System configuration property file (system\_config.properties)
- 8.7 Query group property file
- 8.8 Stream property file
- 8.9 In-process connection property file  
(user\_app.adaptor-group-name-or-adaptor-name.properties)
- 8.10 Log file output property file (logger.properties)
- 8.11 List of JavaVM options

---

## 8.1 Format of SDP server definition file explanations

---

This section describes the format used to explain the SDP server definition files. The following items are provided to explain each file:

**(1) Format**

Shows the file's specification format

**(2) File name**

Indicates the file name

**(3) File storage location**

Shows the file's storage location

**(4) Description**

Describes the usage of the file

**(5) Specifiable parameters**

Describes the parameters that can be specified in the file<sup>#</sup>

#

For some files, such as the system configuration property file (`system_config.properties`), the details of the parameters are provided in a separate subsection.

**(6) Notes**

Provides additional information that should be noted when you create the definition file

*Reference note:*

Note that some items are not provided for all files.



---

## 8.2 Notes about creating SDP server definition files

---

The following points about creating SDP server definition files should be noted.

- A line beginning with a hash mark (#) is treated as a comment line.
- Use \\ as the file separator when you specify file paths.

Example:

```
SDP_CLASS_PATH=C:\\sdp\\AP
```

- The contents of the following files must not be changed once you have started the system:
  - JavaVM options file for SDP servers (`jvm_options.cfg`)
  - System configuration property file (`system_config.properties`)
  - Log file output property file (`logger.properties`)

To change the contents of these files, see *5.3.1 Changing settings in the property files*.

- Specify the SDP server definition file according to the `java.util.Properties` class specifications.

### 8.3 List of SDP server definition files

The table below lists and describes the SDP server definition files. For details about each file, see the section indicated in the table.

*Table 8-1:* List of SDP server definition files

No.	Classification	Definition file	Overview	Section
1	JavaVM start option settings	JavaVM options file for SDP servers ( <code>jvm_options.cfg</code> )	You must create this file for each working directory. Specify in this file the JavaVM options to be used to run the SDP server. In-process-connection adaptors are run according to the options specified in this file.	8.4
2		JavaVM options file for RMI connections ( <code>jvm_client_options.cfg</code> )	Only if you use RMI connection, you must create this file for each working directory or adaptor. Specify in this file the JavaVM options to be used to run the RMI-connection adaptors.	8.5
3	Operating environment property settings	System configuration property file ( <code>system_config.properties</code> )	You must create this file for each working directory. Specify in this file the port number used by the SDP server and details of the API trace information and tuple logs that are to be collected.	8.6
4		Query group property file	You must create this file for each query group. Specify in this file the path of the query definition file and tuning parameters to be used to run the query group.	8.7
5		Stream property file	If you need to specify tuning parameters for a stream in a query group, create this file for each such stream. If this file is not created for a stream, the settings in the query group property file are used.	8.8

No.	Classification	Definition file	Overview	Section
6		In-process connection property file ( <i>user_app.adaptor-group-name-or-adaptor-name.properties</i> )	Only if you use in-process connection, you must create this file for each adaptor group in the case of using standard adaptors and for each adaptor in the case of using custom adaptors. Specify in this file the class name of the in-process-connection adaptor(s) and the path of the jar file.	8.9
7		Log file output property file ( <i>logger.properties</i> )	You create this file for each working directory. If this file is not created, the default values are used. Specify in this file the numbers and sizes of message and trace log files.	8.10

---

## 8.4 JavaVM options file for SDP servers (jvm\_options.cfg)

---

### (1) Format

Specify each parameter in the following format:

<i>parameter-name=JavaVM-option</i>
-------------------------------------

- Spaces cannot be specified except in comments. If you need to specify as a JavaVM option a path that includes a space, you can do so by enclosing the path in double-quotation marks ("), as shown in the example below.

Example:

```
SDP_JVM_LOG=-XX:HitachiJavaLog:"D:\\a b c\\SDPServerVM"
```

- You can specify only one JavaVM option per parameter name.

### (2) File name

```
jvm_options.cfg
```

### (3) File storage location

This file must be stored in the following directory:

```
working-directory\conf\
```

### (4) Description

This file specifies the JavaVM options to be used when the SDP server is started by the `sdpstart` command or when Stream Data Platform - AF commands are executed. In-process-connection adaptors operate based on the `SDP_USER_OPT` parameter specified in this file because they run on the same JavaVM as the SDP server. You must create this file for each working directory.

The JavaVM option specified in each parameter in this file is specified on the Java command line as shown below.

For the `SDP_CLASS_PATH` parameter:

The JavaVM options are delimited by the semicolon and specified in the `-classpath` option of a Java command.

For other parameters:

The JavaVM options are delimited by the single-byte space and specified as an argument of a Java command.

### (5) Specifiable parameters

The table below lists and describes the parameters that can be specified and their

default values. For details about the JavaVM options, see *8.11 List of JavaVM options*.

*Table 8-2: Specifiable parameters and their default values (jvm\_options.cfg)*

No.	Parameter name	Description	Default value
1	SDP_CLASS_PATH	Specifies the jar file that is used by the in-process-connection adaptors. This file depends on the system configuration. <sup>#</sup> If you need to specify multiple jar files, you must specify this parameter for each jar file. The following shows an example: Example: SDP_CLASS_PATH= SDP_CLASS_PATH=C:\\sdp\\AP If you specify a relative path for a jar file, make sure that the specified path is relative to the working directory.	None
2	SDP_CLASSLIB_TRACE	Specifies whether a stack trace is to be output for the class library.	-XX:-HitachiJavaClassLibTrace
3	SDP_CLASSLIB_TRACE_LINESIZE	Specifies the number of characters per line of stack trace for the class library.	-XX:HitachiJavaClassLibTraceLineSize=1024
4	SDP_GC	Specifies whether extended verbosegc information is to be output when garbage collection occurs.	-XX:-HitachiVerboseGC
5	SDP_GC_PRINT_CAUSE	Specifies whether the cause of garbage collection is to be output.	-XX:+HitachiVerboseGCPrintCause
6	SDP_INITIAL_MEM_SIZE	Specifies the initial Java heap size. You specify in this parameter the value obtained in <i>2.7.1 Estimating the memory requirements for the stream data processing engine</i> and <i>2.7.2 Estimating the memory requirements for standard adaptors</i> .	-Xms2048k
7	SDP_JVM_LOG	Specifies a prefix for the log file name.	-XX:HitachiJavaLog:javaLog
8	SDP_JVM_LOG_FILE_SIZE	Specifies the maximum size of a single log file.	-XX:HitachiJavaLogFileSize=256k
9	SDP_LOCALS_IN_STACK_TRACE	Specifies whether the local variable information is to be output to the stack trace during a thread dump output.	-XX:-HitachiLocalsInStackTrace

8. SDP Server Definition Files

No.	Parameter name	Description	Default value
10	SDP_LOCALS_SIMPLE_FORMAT	Specifies whether the simple format is to be used to output local variable information.	-XX:-HitachiLocalsSimpleFormat
11	SDP_MAX_MEM_SIZE	Specifies the maximum Java heap size. You specify in this parameter the value obtained in <i>2.7.1 Estimating the memory requirements for the stream data processing engine</i> and <i>2.7.2 Estimating the memory requirements for standard adaptors</i> .	-Xmx64m
12	SDP_MAX_PERM_SIZE	Specifies the maximum size of the Permanent area.	-XX:MaxPermSize=64m
13	SDP_NEW_RATIO	Specifies the ratio of the Tenured area to the DefNew area.	-XX:NewRatio=2
14	SDP_OOM_STACK_TRACE	Specifies whether a stack trace is to be output in the event of an OutOfMemoryError.	-XX:-HitachiOutOfMemoryStackTrace
15	SDP_OUTPUT_MILLI_TIME	Specifies whether the time is to be output in milliseconds.	-XX:-HitachiOutputMilliTime
16	SDP_PERM_SIZE	Specifies the initial size of the Permanent area.	-XX:PermSize=16m
17	SDP_SYS_OPT	Specifies the JavaVM option required for execution of Stream Data Platform - AF commands. Specify this parameter if you need to specify the JavaVM option in a command depending on the system environment. This parameter takes effect only when a Stream Data Platform - AF command is executed.	None
18	SDP_THRD_DUMP	Specifies whether a thread dump is to be output to the standard output.	-XX:+HitachiThreadDumpToStdout
19	SDP_TRUE_TYPE_IN_LOCALS	Specifies whether the actual type name of a local variable object is to be output as a character string when the local variable information is output.	-XX:-HitachiTrueTypeInfoLocals

No.	Parameter name	Description	Default value
20	SDP_USER_OPT	<p>Specifies a JavaVM option in either of the following cases:</p> <ul style="list-style-type: none"> <li>• The administrator user wishes to add a JavaVM option.</li> <li>• A JavaVM option for running in-process-connection adaptors is to be specified.</li> </ul> <p>To specify multiple JavaVM options, you must specify this parameter for each of the options. The following shows an example: Example: SDP_USER_OPT=-Dxxx=www SDP_USER_OPT=-Dyyy=zzz</p> <p>If you specify the same option more than once, the last option specified (that is the closest to the end of the file) takes effect.</p>	None

#

For the standard adaptors, the values are fixed. Specify the following values:

```
SDP_CLASS_PATH=installation-directory\psb\jaxp\lib\csmjaxb.jar
```

```
SDP_CLASS_PATH=installation-directory\psb\jaxp\lib\csmjaxp.jar
```

```
SDP_CLASS_PATH=installation-directory\psb\jaxp\lib\csmstax.jar
```

```
SDP_CLASS_PATH=installation-directory\psb\CC\lib\hitj2ee.jar
```

## (6) Notes

- Garbage collection is performed periodically on references to remote objects. You can specify a garbage collection interval in milliseconds in the `sun.rmi.dgc.client.gcInterval` and `sun.rmi.dgc.server.gcInterval` properties that are specified when RMI connection is used. The default value is 60,000 milliseconds (60 seconds). To change the garbage collection interval, specify the `sun.rmi.dgc.client.gcInterval` and `sun.rmi.dgc.server.gcInterval` properties in the `SDP_USER_OPT` parameter and specify the desired interval. The permitted value range is from 1 to 9,223,372,036,854,775,806. The following shows an example.

Example:

```
SDP_USER_OPT=-Dsun.rmi.dgc.client.gcInterval=100000000
```

## 8. SDP Server Definition Files

```
SDP_USER_OPT=-Dsun.rmi.dgc.server.gcInterval=1000000000
```



---

## 8.5 JavaVM options file for RMI connections (jvm\_client\_options.cfg)

---

### (1) Format

Specify each parameter in the following format:

<i>parameter-name=JavaVM-option</i>
-------------------------------------

- Spaces cannot be specified except in comments. If you need to specify as a JavaVM option a path that includes a space, you can do so by enclosing the path in double-quotation marks ("), as shown in the example below.

Example:

```
SDP_JVM_LOG=-XX:HitachiJavaLog:"D:\\a b c\\SDPServerVM"
```

- You can specify only one JavaVM option per parameter name.

### (2) File name

jvm\_client\_options.cfg

### (3) File storage location

You can store this file at any location. You use an argument of the `sdpstartap` command that starts RMI-connection adaptors to specify the file storage location. If you omit specification of the argument, *working-directory\conf\* is assumed.

For details about the `sdpstartap` command, see *sdpstartap (starts RMI-connection adaptors)* in 7. *Commands*.

### (4) Description

This file specifies the JavaVM options to be used to start RMI-connection adaptors. Only if you use RMI connection, you must create this file for each working directory or adaptor.

The JavaVM option specified in each parameter in this file is specified on the Java command line as shown below.

For the `SDP_CLASS_PATH` parameter:

The JavaVM options are delimited by the semicolon and specified in the `-classpath` option of a Java command.

For other parameters:

The JavaVM options are delimited by the single-byte space and specified as an argument of a Java command.

**(5) Specifiable parameters**

The table below lists and describes the parameters that can be specified and their default values. For details about the JavaVM options, see *8.11 List of JavaVM options*.

*Table 8-3: Specifiable parameters and their default values  
(jvm\_client\_options.cfg)*

No.	Parameter name	Description	Default value
1	SDP_CLASS_PATH	Specifies the <code>jar</code> file that is used by the RMI-connection adaptors. This file depends on the system configuration. <sup>#</sup> If you need to specify multiple <code>jar</code> files, you must specify this parameter for each <code>jar</code> file. The following shows an example: Example: SDP_CLASS_PATH= . SDP_CLASS_PATH=C:\\sdp\\AP If you specify a relative path for a <code>jar</code> file, make sure that the specified path is relative to the working directory.	None
2	SDP_CLASSLIB_TRACE	Specifies whether a stack trace is to be output for the class library.	-XX:-HitachiJavaClassLibTrace
3	SDP_CLASSLIB_TRACE_LINE_SIZE	Specifies the number of characters per line of stack trace for the class library.	-XX:HitachiJavaClassLibTraceLineSize=1024
4	SDP_GC	Specifies whether extended <code>verbosegc</code> information is to be output when garbage collection occurs.	-XX:-HitachiVerboseGC
5	SDP_GC_PRINT_CAUSE	Specifies whether the cause of garbage collection is to be output.	-XX:+HitachiVerboseGCPrintCause
6	SDP_INITIAL_MEM_SIZE	Specifies the initial Java heap size. You specify in this parameter the value obtained in <i>2.7.1 Estimating the memory requirements for the stream data processing engine</i> and <i>2.7.2 Estimating the memory requirements for standard adaptors</i> .	-Xms2048k
7	SDP_JVM_LOG	Specifies a prefix for the log file name.	-XX:HitachiJavaLog:javalog
8	SDP_JVM_LOG_FILE_SIZE	Specifies the maximum size of a single log file.	-XX:HitachiJavaLogFile=256k

No.	Parameter name	Description	Default value
9	SDP_LOCALS_IN_STACK_TRACE	Specifies whether the local variable information is to be output to the stack trace during a thread dump output.	-XX:-HitachiLocalsInStackTrace
10	SDP_LOCALS_SIMPLE_FORMAT	Specifies whether the simple format is to be used to output local variable information.	-XX:-HitachiLocalsSimpleFormat
11	SDP_MAX_MEM_SIZE	Specifies the maximum Java heap size. You specify in this parameter the value obtained in <i>2.7.1 Estimating the memory requirements for the stream data processing engine</i> and <i>2.7.2 Estimating the memory requirements for standard adaptors</i> .	-Xmx64m
12	SDP_MAX_PERM_SIZE	Specifies the maximum size of the Permanent area.	-XX:MaxPermSize=64m
13	SDP_NEW_RATIO	Specifies the ratio of the Tenured area to the DefNew area.	-XX:NewRatio=2
14	SDP_OOM_STACK_TRACE	Specifies whether a stack trace is to be output in the event of an OutOfMemoryError.	-XX:-HitachiOutOfMemoryStackTrace
15	SDP_OUTPUT_MILLI_TIME	Specifies whether the time is to be output in milliseconds.	-XX:-HitachiOutputMilliTime
16	SDP_PERM_SIZE	Specifies the initial size of the Permanent area.	-XX:PermSize=16m
17	SDP_THRD_DUMP	Specifies whether a thread dump is to be output to the standard output.	-XX:+HitachiThreadDumpToStdout
18	SDP_TRUE_TYPE_IN_LOCALS	Specifies whether the actual type name of a local variable object is to be output as a character string when the local variable information is output.	-XX:-HitachiTrueTypeInLocals

No.	Parameter name	Description	Default value
19	SDP_USER_OPT	<p>Specifies a JavaVM option when the administrator user wishes to add a JavaVM option.</p> <p>To specify multiple JavaVM options, you must specify this parameter for each of the options. The following shows an example:</p> <p>Example:</p> <pre>SDP_USER_OPT=-Dxxx=www SDP_USER_OPT=-Dyyy=zzz</pre> <p>If you specify the same option more than once, the last option specified (that is the closest to the end of the file) takes effect.</p>	None

#

For standard adaptors, the values are fixed. Specify the following values:

```
SDP_CLASS_PATH=installation-directory\psb\jaxp\lib\csmjaxb.jar
```

```
SDP_CLASS_PATH=installation-directory\psb\jaxp\lib\csmjaxp.jar
```

```
SDP_CLASS_PATH=installation-directory\psb\jaxp\lib\csmstax.jar
```

```
SDP_CLASS_PATH=installation-directory\psb\CC\lib\hitj2ee.jar
```

#### (6) Notes

- Garbage collection is performed periodically on references to remote objects. You can specify a garbage collection interval in milliseconds in the `sun.rmi.dgc.client.gcInterval` and `sun.rmi.dgc.server.gcInterval` properties that are specified when RMI connection is used. The default value is 60,000 milliseconds (60 seconds). To change the garbage collection interval, specify the `sun.rmi.dgc.client.gcInterval` and `sun.rmi.dgc.server.gcInterval` properties in the `SDP_USER_OPT` parameter and specify the desired interval. The permitted value range is from 1 to 9,223,372,036,854,775,806. The following shows an example.

Example:

```
SDP_USER_OPT=-Dsun.rmi.dgc.client.gcInterval=1000000000
```

```
SDP_USER_OPT=-Dsun.rmi.dgc.server.gcInterval=1000000000
```

- We recommend that you use different JavaVM log output destinations for input adaptors and output adaptors. If you wish to specify a JavaVM log output

destination and a Java heap size for each adaptor, create a JavaVM options file for RMI connections for each adaptor by following the procedure described below.

1. Prepare a JavaVM options file for RMI connections.

Copy the provided sample file for the JavaVM options file for RMI connections to the desired directory. For details about the sample file storage directory, see *3.2.1 Structure of the installation directory*.

2. Edit the copy of the JavaVM options file for RMI connections.

For a JavaVM log file name, a 2-digit number (01 to the number of log files) is added immediately after the prefix specified in the `SDP_JVM_LOG` parameter. If you use different log output destinations for input and output adaptors, make sure that their prefixes are different.

Example:

Specification of a JavaVM log output destination for input adaptors:

```
SDP_JVM_LOG=-XX:HitachiJavaLog:C:\\sdp\\logs\\SDPReceiverClientVM
```

Specification of a JavaVM log output destination for output adaptors:

```
SDP_JVM_LOG=-XX:HitachiJavaLog:C:\\sdp\\logs\\SDPSenderClientVM
```

3. When you start an adaptor, specify the correct JavaVM options file for RMI connections in the `-clientcfg` option of the `sdpstartap` command.

## 8.6 System configuration property file (system\_config.properties)

This section provides the details of the system configuration property file (`system_config.properties`) and the parameters that are specified in the file.

### 8.6.1 Details of the system configuration property file (system\_config.properties)

#### (1) Format

Specify each parameter in the following format:

```
parameter-name=value
```

- The value cannot be followed by spaces or a character string (such as a comment). Anything that follows the value will be considered part of the value.
- A line containing only a parameter name and no value will be ignored.

#### (2) File name

```
system_config.properties
```

#### (3) File storage location

This file must be stored in the following directory:

```
working-directory\conf\
```

#### (4) Description

This file specifies the port number used by the SDP server and details of the API trace information and tuple logs that are to be collected. You must create this file for each working directory.

#### (5) Specifiable parameters

The table below lists and describes the parameters that can be specified and their default values. For details about the parameters, see 8.6.2 *Details of the parameters in the system configuration property file (system\_config.properties)*.

Table 8-4: Specifiable parameters and their default values (system\_config.properties)

No.	Parameter name	Description	Default value	Permitted value range
1	engine.initialQueueSize	Specifies the initial number of elements used in the stream queue.	1024	Integer from 1 to 1048576

No.	Parameter name	Description	Default value	Permitted value range
2	<code>engine.maxQueueSize</code>	Specifies the maximum number of elements used in the stream queue.	4096	Integer from 1 to 1048576
3	<code>engine.watchQueueSize.threshold</code>	Specifies a threshold value (%) for monitoring the number of elements used in the stream queue.	None	Integer from 1 to 99
4	<code>logger.console.abnormal.enabled</code>	Specifies whether the SDP server start processing is to be continued in the event of a failure during log file initialization: <ul style="list-style-type: none"> <li><code>true</code>: Outputs a message to the console and then starts the SDP server.</li> <li><code>false</code>: Shuts down the SDP server.</li> </ul>	<code>false</code>	<code>true</code> or <code>false</code>
5	<code>logger.console.enabled</code>	Specifies whether messages issued by the SDP server are to be output to the console: <ul style="list-style-type: none"> <li><code>true</code>: Outputs messages to both log file and console.</li> <li><code>false</code>: Does not output messages to the console.</li> </ul>	<code>false</code>	<code>true</code> or <code>false</code>
6	<code>mon.process_exp_time</code>	Specifies (in milliseconds) the processing time for determining a timeout.	30000	Integer from 1 to 2147483647
7	<code>query.decimalMaxPrecision</code>	Specifies the maximum precision for an arithmetic operation when the result of a query is the <code>DECIMAL</code> type.	38	Integer from 1 to 38
8	<code>query.decimalRoundingMode</code>	Specifies the rounding operation to be performed in the event the precision specified in the <code>query.decimalMaxPrecision</code> parameter is exceeded when the result of a query is the <code>DECIMAL</code> type.	<code>HALF_UP</code>	<code>HALF_UP</code> OR <code>DOWN</code>
9	<code>querygroup.sleepOnOverStore</code>	Specifies the period of time that execution of a query group is to be placed in the sleep mode when the SDP server's checking determines that there is no available space in the output stream queue.	100	1 to 2147483647

8. SDP Server Definition Files

No.	Parameter name	Description	Default value	Permitted value range
10	<code>querygroup.sleepOnOverStoreRetryCount</code>	Specifies the number of times the SDP server is to check the output stream queue for available space before it loads to the output stream queue the tuples obtained from query execution.	0	0 to 2147483647
11	<code>rmi.serverPort</code>	Specifies a port number for the SDP server.	20400	Integer from 1 to 65535
12	<code>stream.freeInputQueueSizeThreshold</code>	Specifies a threshold value (%) for the available size with respect to the maximum number of elements used in the input stream queue.	None	Integer from 1 to 99
13	<code>stream.freeInputQueueSizeThresholdOutputMessage</code>	Specifies whether a warning message is to be output to the SDP server's message log: <ul style="list-style-type: none"> <li>• <code>true</code>: Outputs warning message.</li> <li>• <code>false</code>: Does not output warning message.</li> </ul>	<code>false</code>	<code>true</code> or <code>false</code>
14	<code>stream.maxKeepTupleCount</code>	Specifies the maximum number of tuples that can be retained by the timestamp adjustment function.	125828	Integer from 1 to 1048576
15	<code>stream.timestampAccuracy</code>	Specifies the time units and time adjustment range for the timestamp adjustment function.	None	See 8.6.2(15) <i>stream.timestampAccuracy = {sec msec usec}, time-adjustment-range unuse}</i> .
16	<code>stream.timestampMode</code>	Specifies the timestamp mode used to timestamp tuples: <ul style="list-style-type: none"> <li>• <code>Server</code>: Uses the server mode.</li> <li>• <code>DataSource</code>: Uses the data source mode.</li> </ul>	<code>Server</code>	<code>Server</code> or <code>DataSource</code>
17	<code>stream.timestampPosition</code>	Specifies the name of the time-data column in tuples.	None	See 8.6.2(17) <i>stream.timestampPosition = time-data-column-name</i> .



No.	Parameter name	Description	Default value	Permitted value range
18	<code>stream.tupleLogMode</code>	Specifies whether execution of the <code>sdptplput</code> command is to be enabled: <ul style="list-style-type: none"> <li>• <code>true</code>: <code>sdptplput</code> command is to be executed.</li> <li>• <code>false</code>: Execution of <code>sdptplput</code> command is to be disabled.</li> </ul>	<code>false</code>	<code>true</code> or <code>false</code>
19	<code>tpl.backupFileCount</code>	Specifies the maximum number of backup generations to be retained for the tuple log file.	1	Integer from 0 to 10
20	<code>tpl.bufferCount</code>	Specifies the number of tuple log buffers.	5	Integer from 3 to 512
21	<code>tpl.bufferSize</code>	Specifies the maximum size of a tuple log buffer (in kilobytes).	1024	Integer from 1 to 2048000
22	<code>tpl.fileCount</code>	Specifies the maximum number of tuple log files.	3	Integer from 3 to 512
23	<code>tpl.fileSize</code>	Specifies the maximum size of a tuple log file (in megabytes).	100	Integer from 1 to 2048
24	<code>tpl.outputLevel</code>	Specifies the tuple log output level: <ul style="list-style-type: none"> <li>• 1: Outputs tuple logs for the tuples that are stored in the stream queue.</li> <li>• 2: Outputs tuple logs for the tuples that are discarded due to out-of-sequence time.</li> <li>• 3: Outputs tuple logs for tuples with levels 1 and 2.</li> </ul>	3	Integer from 1 to 3
25	<code>tpl.outputTrigger</code>	Specifies the tuple log file output timing: <ul style="list-style-type: none"> <li>• <code>BUFFER</code>: Outputs tuple logs to a file when the buffer being used to collect the current tuple logs in the target stream becomes full.</li> <li>• <code>NONE</code>: Does not output tuple logs to a file or perform buffering of tuple logs.</li> </ul>	<ul style="list-style-type: none"> <li>• Input stream queue: <code>BUFFER</code></li> <li>• Output stream queue: <code>NONE</code></li> </ul>	<code>BUFFER</code> or <code>NONE</code>
26	<code>tpl.useOverwrite</code>	Specifies whether the buffer is to be overwritten in the event of a full tuple log buffer: <ul style="list-style-type: none"> <li>• <code>true</code>: Overwrites the tuple log buffer.</li> <li>• <code>false</code>: Does not overwrite the tuple log buffer.</li> </ul>	<code>true</code>	<code>true</code> or <code>false</code>

No.	Parameter name	Description	Default value	Permitted value range
27	<code>trc.api.bufferCount</code>	Specifies the number of API trace buffers.	3	Integer from 3 to 512
28	<code>trc.api.bufferSize</code>	Specifies the maximum size of an API trace buffer (in kilobytes).	1024	Integer from 1 to 2048000
29	<code>trc.api.fileCount</code>	Specifies the maximum number of files to which API trace information is to be output.	3	Integer from 3 to 512
30	<code>trc.api.fileSize</code>	Specifies the maximum size of an API trace file (in megabytes).	1024	Integer from 1 to 2048
31	<code>trc.api.ioBufferSize</code>	Specifies the maximum size of an API trace I/O buffer (in kilobytes).	2048	Integer from 1 to 2048000
32	<code>trc.api.outputTrigger</code>	Specifies the timing for output of API trace information to a file: <ul style="list-style-type: none"> <li><code>BUFFER</code>: Outputs API trace information to a file when the I/O thread buffer becomes full.</li> <li><code>NONE</code>: Does not output API trace information to a file or perform buffering of API trace information.</li> </ul>	<code>BUFFER</code>	<code>BUFFER</code> or <code>NONE</code>
33	<code>trc.api.useOverwrite</code>	Specifies whether the API trace buffer is to be overwritten when the buffer becomes full: <ul style="list-style-type: none"> <li><code>true</code>: Overwrites the API trace buffer.</li> <li><code>false</code>: Does not overwrite the API trace buffer.</li> </ul>	<code>true</code>	<code>true</code> or <code>false</code>

### 8.6.2 Details of the parameters in the system configuration property file (`system_config.properties`)

This subsection provides the details of the parameters in the system configuration property file (`system_config.properties`) shown in 8.6.1(5) *Specifiable parameters*.

#### (1) `engine.initialQueueSize=initial-number-of-elements-used-in-stream-queue`

This parameter specifies as an integer from 1 to 1048576 the initial number of elements used in the stream queue. The default value is 1024.

If you attempt to register more elements in the stream queue than specified here, the stream queue is extended up to the maximum value specified in the `engine.maxQueueSize` parameter.

Note that the initial number of elements specified here becomes the initial value for each stream queue.

**(2) *engine.maxQueueSize=maximum-number-of-elements-used-in-stream-queue***

This parameter specifies as an integer from 1 to 1048576 the maximum number of elements used in the stream queue. The default is 4096.

Make sure that the maximum number of elements used in the stream queue is no greater than the initial number of elements specified in the `engine.initialQueueSize` parameter.

Note that the maximum number of elements specified here becomes the maximum value for each stream queue.

**(3) *engine.watchQueueSize.threshold=threshold-value-for-monitoring-number-of-elements-used-in-stream-queue***

This parameter specifies the threshold value (%) for monitoring the number of elements used in the stream queue as an integer from 1 to 99. If this parameter is omitted, the number of elements is not monitored.

If the value obtained from the formula shown below is greater than the value of this parameter, a warning message is output.

$\text{Number of elements} \div \text{engine.maxQueueSize parameter value} \times 100$
--

Once this warning message is output, it will not be output again until the next time the number of elements exceeds the threshold value after it dropped below the threshold value.

**(4) *logger.console.abnormal.enabled={true|false}***

This parameter specifies, as `true` or `false`, whether the SDP server start processing is to be continued in the event of a failure during log file initialization. This value is not case sensitive. The default value is `false`.

`true`

Outputs a message to the console and then starts the SDP server.

`false`

Shuts down the SDP server.

**(5) *logger.console.enabled={true|false}***

This parameter specifies, as `true` or `false`, whether messages issued by the SDP server are to be output to the console. This value is not case sensitive. The default value is `false`.

true

Outputs messages to both log file and console.

false

Does not output messages to the console.

**(6) *mon.process\_exp\_time=timeout-processing-time***

This parameter specifies (in milliseconds) the amount of time used for processing before a timeout occurs, as an integer from 1 to 2147483647. The default value is 30000.

If use of the default value results in frequent output of a timeout detection message, specify an appropriate value on the basis of the actual processing time that is needed, as obtained from the API trace information.

**(7) *query.decimalMaxPrecision=maximum-precision-of-query-arithmetic-operation***

This parameter specifies as an integer from 1 to 38 the maximum precision for an arithmetic operation when the result of a query is the DECIMAL type (including the NUMERIC type). The default value is 38.

The following arithmetic operations result in the DECIMAL type:

- Four arithmetic operations (binary operations) that contain the DECIMAL type in operands
- Aggregate functions AVG and SUM whose argument is the DECIMAL type
- Casting of non-DECIMAL type data to DECIMAL type data

If the result of an arithmetic operation does not exceed the precision specified in this parameter, precision processing is not processed. If the result of an arithmetic operation exceeds the precision specified in this parameter, the result is rounded to the precision specified in this parameter. You use the `query.decimalRoundingMode` parameter to specify the rounding operation to be applied.

**(8) *query.decimalRoundingMode={HALF\_UP|DOWN}***

This parameter specifies the rounding operation to be performed in the event the precision specified in the `query.decimalMaxPrecision` parameter is exceeded when the result of a query is the DECIMAL type (including the NUMERIC type). The default value is HALF\_UP.

HALF\_UP

Rounds off on the basis of the first decimal place to be discarded.

DOWN

Rounds down the digit that immediately precedes the first decimal place to be

discarded.

**(9) *querygroup.sleepOnOverStore=query-group-execution-sleep-time***

This parameter specifies as an integer from 1 to 2147483647 the period of time (in milliseconds) that execution of a query group is to be placed in the sleep mode when the SDP server's checking determines that there is no available space in the output stream queue. The default value is 100.

This parameter takes effect if 1 or a greater value is specified in the `querygroup.sleepOnOverStoreRetryCount` parameter.

You can perform the following operations on a query group that has been placed in the sleep mode by this parameter:

- Loading of tuples into the input stream queue by the input adaptor
- Collection of tuples from the output stream queue by the output adaptor

You can specify this parameter in the system configuration property file and the query group property file. If the parameter is duplicated or omitted, the specified value takes effect in the following order (1 > 2):

1. Query group property file
2. System configuration property file

**(10) *querygroup.sleepOnOverStoreRetryCount=number-of-times-to-check-output-stream-queue-for-available-space***

This parameter specifies as an integer from 0 to 2147483647 the number of times the SDP server is to check the output stream queue for available space before it loads the tuples obtained from query execution to the output stream queue. The default value is 0.

If the value 0 is specified, the SDP server loads the tuples obtained from query execution to the output stream queue without checking the output stream queue for available space.

If 1 or a greater value is specified, the SDP server performs the following processing as many times as specified:

1. The SDP server checks whether there is room in the output stream queue.
2. If there is no room in the output stream queue, the SDP server places execution of the query group that uses the output stream queue in the sleep mode. The length of the sleep time is specified in the `querygroup.sleepOnOverStore` parameter.

When the sleep period expires, if this processing has not been performed as many times as specified in this parameter, the SDP server returns to step 1.

3. If there is room in the output stream queue, the SDP server loads the tuples

obtained from query execution into the output stream queue.

You can specify this parameter in the system configuration property file and the query group property file. If the parameter is duplicated or omitted, the specified value takes effect in the following order (1 > 2):

1. Query group property file
2. System configuration property file

**(11) *rmi.serverPort=port-number-for-SDP-server***

This parameter specifies a port number for the SDP server, as an integer from 1 to 65535. The default value is 20400.

If the default port number (20400) is being used as a port number in another system, you must specify a different port number. We recommend that you do not use port numbers 1 through 1023 (because they are commonly-used port numbers).

**(12) *stream.freeInputQueueSizeThreshold=threshold-value-for-available-size-to-maximum-size-of-input-stream-queue***

This parameter specifies a threshold value (%) for the available size with respect to the maximum number of elements used in the input stream queue as an integer from 1 to 99 (the `engine.maxQueueSize` parameter value).

When the condition shown below is satisfied, a `SDPClientFreeInputQueueSizeThresholdOverException` exception is thrown from the `put(StreamTuple tuple)` method or the `put(ArrayList<StreamTuple> tuple_list)` method. In such a case, loading of tuples into the input stream queue has been successful.

$\text{This parameter's value} \geq ((\text{available size of input stream queue} \div \text{maximum size of input stream queue}) \times 100)$
--

If this parameter is omitted, an exception by threshold value checking will not occur.

Note that the `engine.watchQueueSize.threshold` parameter specifies a threshold value for the usage size compared to the maximum size of input and output stream queues. It is different from the `stream.freeInputQueueSizeThreshold` parameter. For details, see the `engine.watchQueueSize.threshold` parameter.

When the `sdptplput` command is used to load tuples to the input stream queue, this parameter is ignored.

You can specify this parameter in the system configuration property file, the query group property file, and the stream property file. If the parameter is duplicated or omitted, the specified value takes effect in the following order (1 > 2 > 3):

1. Stream property file
2. Query group property file

## 3. System configuration property file

**(13) *stream.freeInputQueueSizeThresholdOutputMessage={true|false}***

This parameter specifies, as `true` or `false`, whether a warning message (the KFSP42032-W message) is to be output to the SDP server's message log. This value is not case sensitive. The default value is `false`.

`true`

Outputs the warning message.

`false`

Does not output the warning message.

This parameter takes effect only when the `stream.freeInputQueueSizeThreshold` parameter is specified.

Note that the warning message is output only when `true` is specified in this parameter and the following condition is satisfied:

*stream.freeInputQueueSizeThreshold* parameter value  $\geq ((\text{available size of input stream queue} \div \text{maximum size of input stream queue}) \times 100)$

You can specify this parameter in the system configuration property file, the query group property file, and the stream property file. If the parameter is duplicated or omitted, the specified value takes effect in the following order (1 > 2 > 3):

1. Stream property file
2. Query group property file
3. System configuration property file

**(14) *stream.maxKeepTupleCount=maximum-number-of-tuples-retained-by-time-stamp-adjustment-function***

This parameter specifies as an integer from 1 to 1048576 the maximum number of tuples that can be retained by the timestamp adjustment function. The default value is 125828.

The number of tuples specified in this parameter is used as the maximum value for the timestamp adjustment function for each input stream.

When the number of tuples exceeds the value specified in this parameter, the query group is shut down.

You can specify this parameter in the system configuration property file, the query group property file, and the stream property file. If the parameter is duplicated or omitted, the specified value takes effect in the following order (1 > 2 > 3):

1. Stream property file

2. Query group property file
3. System configuration property file

**(15) `stream.timestampAccuracy={{sec|msec|usec},time-adjustment-range|unuse}`**

This parameter specifies the time units and the time adjustment range for the timestamp adjustment function. This value is not case sensitive.

You must specify this parameter when you specify `DataSource` in the `stream.timestampMode` parameter. If you specify `Server` in the `stream.timestampMode` parameter, this parameter is ignored, but its format is checked.

`{sec|msec|usec}, time-adjustment-range`

Specifies the time units and time adjustment range. There must be no spaces or tabs preceding or following the single-byte comma (,) between the time units (`sec`, `msec`, or `usec`) and the time adjustment range. If you violate this rule, an error results. If you specify `sec`, `msec`, or `usec` as the time units and 0 as the time adjustment range, the time adjustment range is applied only to the reference time.

The meaning of each value is as follows:

`sec`

Specifies that seconds are to be used as the time units.

`msec`

Specifies that milliseconds are to be used as the time units.

`usec`

Specifies that microseconds are to be used as the time units.

*time-adjustment-range*

Specifies as an integer the range of times to be adjusted for timestamp adjustment. The permitted value range depends on the time units, as shown in the table below.

Time units	Permitted value range
<code>sec</code> (seconds)	Integer from 0 to 59
<code>msec</code> (milliseconds)	Integer from 0 to 999
<code>usec</code> (microseconds)	Integer from 0 to 999

`unuse`

Specifies that time adjustment is not to be performed.



You can specify this parameter in the system configuration property file, the query group property file, and the stream property file. If the parameter is duplicated or omitted, the specified value takes effect in the following order (1 > 2 > 3):

1. Stream property file
2. Query group property file
3. System configuration property file

**(16) *stream.timestampMode={Server|DataSource}***

This parameter specifies `Server` or `DataSource` as the timestamp mode to be used in timestamping tuples. This value is not case sensitive. The default value is `Server`.

`Server`

Uses the server mode.

`DataSource`

Uses the data source mode.

If you specify `DataSource`, you must specify the `stream.timestampAccuracy` and `stream.timestampPosition` parameters.

You can specify this parameter in the system configuration property file and the query group property file. If the parameter is duplicated or omitted, the specified value takes effect in the following order (1 > 2):

1. Query group property file
2. System configuration property file

**(17) *stream.timestampPosition=time-data-column-name***

This parameter specifies the name of the time-data column in tuples. This value is not case sensitive.

If you specify `DataSource` in the `stream.timestampMode` parameter, you must also specify this parameter. If you specify `Server` in the `stream.timestampMode` parameter, this parameter is ignored, but its format is checked.

The only data type that can be specified as time data is the `TIMESTAMP` type.

The permitted range of time data is from `1970/01/01 00:00:00.000000000` to `2261/12/31 23:59:59.999999999` in GMT (Greenwich Mean Time). If a specified time is outside this range, an exception occurs when the stream data is sent.

You can specify this parameter in the system configuration property file, the query group property file, and the stream property file. If the parameter is duplicated or omitted, the specified value takes effect in the following order (1 > 2 > 3):

1. Stream property file
2. Query group property file
3. System configuration property file

**(18) *stream.tupleLogMode={true|false}***

This parameter specifies whether execution of the `sdptplput` command is to be enabled. This value is not case sensitive. The default value is `false`.

`true`

`sdptplput` command is to be executed.

`false`

Execution of the `sdptplput` command is to be disabled.

If you specify `Server` in the `stream.timestampMode` parameter and `true` in this parameter, an `SDPClientException` exception will occur if you then send stream data using the `put` method of the `StreamInput` interface.

You can specify this parameter in the system configuration property file and the query group property file. If the parameter is duplicated or omitted, the specified value takes effect in the following order (1 > 2):

1. Query group property file
2. System configuration property file

**(19) *tpl.backupFileCount=maximum-number-of-backup-generations-to-be-retained-for-tuple-log-file***

This parameter specifies as an integer from 0 to 10 the maximum number of backup generations to be retained for the tuple log file. The default value is 1.

You can specify this parameter in the system configuration property file, the query group property file, and the stream property file. If the parameter is duplicated or omitted, the specified value takes effect in the following order (1 > 2 > 3):

1. Stream property file
2. Query group property file
3. System configuration property file

**(20) *tpl.bufferCount=tuple-log-buffers-count***

This parameter specifies the number of tuple log buffers, as an integer from 3 to 512. The default value is 5.

You can specify this parameter in the system configuration property file, the query group property file, and the stream property file. If the parameter is duplicated or omitted, the specified value takes effect in the following order (1 > 2 > 3):

1. Stream property file
2. Query group property file
3. System configuration property file

**(21) *tpl.bufferSize=maximum-tuple-log-buffer-size***

This parameter specifies as an integer from 1 to 2048000 the maximum size of a tuple log buffer (in kilobytes). The default value is 1024.

You can specify this parameter in the system configuration property file, the query group property file, and the stream property file. If the parameter is duplicated or omitted, the specified value takes effect in the following order (1 > 2 > 3):

1. Stream property file
2. Query group property file
3. System configuration property file

**(22) *tpl.fileCount=maximum-tuple-log-files-count***

This parameter specifies the maximum number of tuple log files, as an integer from 3 to 512. The default value is 3.

You can specify this parameter in the system configuration property file, the query group property file, and the stream property file. If the parameter is duplicated or omitted, the specified value takes effect in the following order (1 > 2 > 3):

1. Stream property file
2. Query group property file
3. System configuration property file

**(23) *tpl.fileSize=maximum-tuple-log-file-size***

This parameter specifies the maximum size of a tuple log file (in megabytes), as an integer from 1 to 2048. The default value is 100.

You can specify this parameter in the system configuration property file, the query group property file, and the stream property file. If the parameter is duplicated or omitted, the specified value takes effect in the following order (1 > 2 > 3):

1. Stream property file
2. Query group property file
3. System configuration property file

**(24) *tpl.outputLevel=tuple-log-output-level***

This parameter specifies the tuple log output level, as an integer from 1 to 3. The default value is 3.

1

Outputs tuple logs for the tuples that are stored in the stream queue.

2

Outputs tuple logs for the tuples that are discarded due to an out-of-sequence time.

3

Outputs tuple logs for tuples with levels 1 and 2.

You can specify this parameter in the system configuration property file, the query group property file, and the stream property file. If the parameter is duplicated or omitted, the specified value takes effect in the following order (1 > 2 > 3):

1. Stream property file
2. Query group property file
3. System configuration property file

**(25) *tpl.outputTrigger*={*BUFFER*|*NONE*}**

This parameter specifies `BUFFER` or `NONE` as the tuple log file output timing. This value is not case sensitive. The default value is `BUFFER` for input stream queues and `NONE` for output stream queues.

`BUFFER`

Outputs tuple logs to a file when the buffer being used to collect the current tuple logs in the target stream becomes full.

`NONE`

Does not output tuple logs to a file or perform buffering of tuple logs.

You can specify this parameter in the system configuration property file, the query group property file, and the stream property file. If the parameter is duplicated or omitted, the specified value takes effect in the following order (1 > 2 > 3):

1. Stream property file
2. Query group property file
3. System configuration property file

**(26) *tpl.useOverwrite*={*true*|*false*}**

This parameter specifies, as `true` or `false`, whether a tuple log buffer is to be overwritten in the event it becomes full. This value is not case sensitive. The default value is `true`.

`true`

Overwrites the tuple log buffer.

`false`

Does not overwrite the tuple log buffer.

You can specify this parameter in the system configuration property file, the query group property file, and the stream property file. If the parameter is duplicated or omitted, the specified value takes effect in the following order (1 > 2 > 3):

1. Stream property file
2. Query group property file
3. System configuration property file

**(27) *trc.api.bufferCount=API-trace-buffers-count***

This parameter specifies the number of API trace buffers, as an integer from 3 to 512. The default value is 3.

**(28) *trc.api.bufferSize=maximum-API-trace-buffer-size***

This parameter specifies the maximum size of an API trace buffer (in kilobytes), as an integer from 1 to 2048000. The default value is 1024.

**(29) *trc.api.fileCount=maximum-API-trace-files-count***

This parameter specifies as an integer from 3 to 512 the maximum number of files to which API trace information is to be output. The default value is 3.

**(30) *trc.api.fileSize=maximum-API-trace-file-size***

This parameter specifies the maximum size of an API trace file (in megabytes), as an integer from 1 to 2048. The default value is 1024.

**(31) *trc.api.ioBufferSize=maximum-API-trace-I/O-buffer-size***

This parameter specifies the maximum size of an API trace I/O buffer (in kilobytes), as an integer from 1 to 2048000. The default value is 2048.

**(32) *trc.api.outputTrigger={BUFFER|NONE}***

This parameter specifies `BUFFER` or `NONE` as the timing for output to file of API trace information. This value is not case sensitive. The default value is `BUFFER`.

`BUFFER`

Outputs API trace information to a file when the I/O thread buffer becomes full.

`NONE`

Does not output API trace information to a file or perform buffering of API trace information.

**(33) *trc.api.useOverwrite={true|false}***

This parameter specifies, as `true` or `false`, whether the API trace buffer is to be

overwritten when the buffer becomes full. This value is not case sensitive. The default value is `true`.

`true`

Overwrites the API trace buffer.

`false`

Does not overwrite the API trace buffer.

---

## 8.7 Query group property file

---

This section provides the details of the query group property file and the parameters that are specified in the file.

### 8.7.1 Details of the query group property file

#### (1) Format

Specify each parameter in the following format:

<i>parameter-name=value</i>
-----------------------------

- The value cannot be followed by spaces or a character string (such as a comment). Anything that follows the value will be considered part of the value.
- A line containing only a parameter name and no value will be ignored.

#### (2) File name

*query-group-name*

Specify a query group name as a string of 1 to 64 alphanumeric characters (0 to 9, a to z, A to Z) and the underscore (\_). A query group name must begin with a single-byte alphabetic character.

#### (3) File storage location

This file must be stored in the following directory:

*working-directory*\conf\

#### (4) Description

This file specifies the path of the query definition file and tuning parameters to be used to run the query group. You create a separate file for each query group.

#### (5) Specifiable parameters

The table below lists and describes the parameters that can be specified and their default values. For details about the parameters, see *8.7.2 Details of the parameters in the query group property file*.

*Table 8-5: Specifiable parameters and their default values (query group property file)*

No.	Parameter name	Description	Default value	Permitted value range	Whether or not changeable during restart <sup>#1</sup>
1	querygroup.cqlFilePath	Specifies the path of the query definition file that defines the query group.	None	Absolute path or path relative to the working directory	N
2	querygroup.sleepOnOverStore	Specifies the period of time that execution of the query group is to be placed in the sleep mode when the SDP server's checking determines that there is no available space in the output stream queue.	Value specified in system_config.properties <sup>#2</sup>	1 to 2147483647	Y
3	querygroup.sleepOnOverStoreRetryCount	Specifies the number of times the SDP server is to check the output stream queue for available space before it loads to the output stream queue the tuples obtained from query execution.	Value specified in system_config.properties <sup>#2</sup>	0 to 2147483647	Y
4	stream.filterCondition	Specifies a conditional expression for using the timestamp adjustment function to filter tuples.	None	See 8.7.2(4) <i>stream.filterCondition=conditional-expression</i> .	Y
5	stream.filterMode	Specifies whether the timestamp adjustment function is to be used to filter tuples: <ul style="list-style-type: none"> <li>• unuse: Does not filter tuples.</li> <li>• condition: Filters tuples.</li> </ul>	unuse	unuse or condition	Y



No.	Parameter name	Description	Default value	Permitted value range	Whether or not changeable during restart <sup>#1</sup>
6	<code>stream.freeInputQueueSizeThreshold</code>	Specifies a threshold value (%) for the available size with respect to the maximum number of elements used in the input stream queue.	Value specified in <code>system_config.properties</code> <sup>#2</sup>	Integer from 1 to 99	Y
7	<code>stream.freeInputQueueSizeThresholdOutputMessage</code>	Specifies whether a warning message is to be output to the SDP server's message log: <ul style="list-style-type: none"> <li><code>true</code>: Outputs warning message.</li> <li><code>false</code>: Does not output warning message.</li> </ul>	Value specified in <code>system_config.properties</code> <sup>#2</sup>	<code>true</code> or <code>false</code>	Y
8	<code>stream.maxKeepTupleCount</code>	Specifies the maximum number of tuples that can be retained by the timestamp adjustment function.	Value specified in <code>system_config.properties</code> <sup>#2</sup>	Integer from 1 to 1048576	Y
9	<code>stream.propertyFiles</code>	Specifies the names of the stream property files when properties are specified for individual streams.	None	--	Y
10	<code>stream.timestampAccuracy</code>	Specifies the time units and time adjustment range for the timestamp adjustment function.	Value specified in <code>system_config.properties</code> <sup>#2</sup>	See 8.7.2(10) <code>stream.timestampAccuracy = {{sec msec usec},time-adjustment-range unuse}</code> .	Y
11	<code>stream.timestampMode</code>	Specifies the timestamp mode used to timestamp tuples: <ul style="list-style-type: none"> <li><code>Server</code>: Uses the server mode.</li> <li><code>DataSource</code>: Uses the data source mode.</li> </ul>	Value specified in <code>system_config.properties</code> <sup>#2</sup>	Server or DataSource	N

8. SDP Server Definition Files

No.	Parameter name	Description	Default value	Permitted value range	Whether or not changeable during restart <sup>#1</sup>
12	stream.timestampPosition	Specifies the name of the time-data column in tuples.	Value specified in system_config.properties <sup>#2</sup>	--	Y
13	stream.tupleLogMode	Specifies whether execution of the sdptplput command is to be enabled: <ul style="list-style-type: none"> <li>• true: sdptplput command is to be executed.</li> <li>• false: Execution of sdptplput command is to be disabled.</li> </ul>	Value specified in system_config.properties <sup>#2</sup>	true or false	N
14	tpl.backupFileCount	Specifies the maximum number of backup generations to be retained for the tuple log file.	Value specified in system_config.properties <sup>#2</sup>	Integer from 0 to 10	Y
15	tpl.bufferCount	Specifies the number of tuple log buffers.	Value specified in system_config.properties <sup>#2</sup>	Integer from 3 to 512	Y
16	tpl.bufferSize	Specifies the maximum size of a tuple log buffer (in kilobytes).	Value specified in system_config.properties <sup>#2</sup>	Integer from 1 to 2048000	Y
17	tpl.fileCount	Specifies the maximum number of tuple log files.	Value specified in system_config.properties <sup>#2</sup>	Integer from 3 to 512	Y
18	tpl.fileSize	Specifies the maximum size of a tuple log file (in megabytes).	Value specified in system_config.properties <sup>#2</sup>	Integer from 1 to 2048	Y

No.	Parameter name	Description	Default value	Permitted value range	Whether or not changeable during restart <sup>#1</sup>
19	<code>tpl.outputLevel</code>	Specifies the tuple log output level: <ul style="list-style-type: none"> <li>1: Outputs tuple logs for the tuples that are stored in the stream queue.</li> <li>2: Outputs tuple logs for the tuples that are discarded due to out-of-sequence time.</li> <li>3: Outputs tuple logs for tuples with levels 1 and 2.</li> </ul>	Value specified in <code>system_config.properties</code> <sup>#2</sup>	Integer from 1 to 3	Y
20	<code>tpl.outputTrigger</code>	Specifies the tuple log file output timing: <ul style="list-style-type: none"> <li><code>BUFFER</code>: Outputs tuple logs to a file when the buffer being used to collect the current tuple logs in the target stream becomes full.</li> <li><code>NONE</code>: Does not output tuple logs to a file or perform buffering of tuple logs.</li> </ul>	Value specified in <code>system_config.properties</code> <sup>#2</sup>	<code>BUFFER</code> or <code>NONE</code>	Y
21	<code>tpl.useOverwrite</code>	Specifies whether the buffer is to be overwritten in the event of a full tuple log buffer: <ul style="list-style-type: none"> <li><code>true</code>: Overwrites the tuple log buffer.</li> <li><code>false</code>: Does not overwrite the tuple log buffer.</li> </ul>	Value specified in <code>system_config.properties</code> <sup>#2</sup>	<code>true</code> or <code>false</code>	Y

## Legend:

Y: Changeable

N: Not changeable

--: Not applicable

#1

Indicates whether (Y) or not (N) the parameter's setting can be changed when the query group is started by the `sdpcqlstart` command with the `-reload` option specified.

#2

If no value is specified in the system configuration property file (`system_config.properties`), the default value for the system configuration property file is used.

**(6) Notes**

- You must use the following procedure to change the contents of a query group property file after you have registered the query group:

1. Delete the registered query group.
2. Edit the property file.
3. Re-register the query group.

However, you do not need to delete the query group to change the values of the parameters for which Y is shown in the column *Whether or not changeable during restart* in the table in (5) *Specifiable parameters*; instead, you can follow the procedure below to change any of these parameter values:

1. Terminate the query group.
2. Edit the property file.
3. Restart the query group with the `sdpcqlstart` command with the `-reload` option specified.

- If you change the value of a parameter for which N is shown in the column *Whether or not changeable during restart* in the table in (5) *Specifiable parameters* and then you restart the query group with the `sdpcqlstart` command with the `-reload` option specified, the SDP server will ignore the change to the parameter value and continue processing.

**8.7.2 Details of the parameters in the query group property file**

This subsection provides the details of the parameters in the query group property file shown in 8.7.1(5) *Specifiable parameters*.

**(1) `querygroup.cqlFilePath=query-group-definition-file-path`**

This parameter specifies the path of the query definition file that defines the query group, expressed as an absolute path or a path relative to the working directory. If this parameter is omitted, the query group will not be registered.

**(2) *querygroup.sleepOnOverStore=query-group-execution-sleep-time***

This parameter specifies as an integer from 1 to 2147483647 the period of time (in milliseconds) that execution of the query group is to be placed in the sleep mode when the SDP server's checking determines that there is no available space in the output stream queue.

This parameter takes effect if 1 or a greater value is specified in the `querygroup.sleepOnOverStoreRetryCount` parameter.

You can perform the following operations on the query group while it is in the sleep mode:

- Loading of tuples into the input stream queue by the input adaptor
- Collection of tuples from the output stream queue by the output adaptor

You can specify this parameter in the system configuration property file and the query group property file. If the parameter is duplicated or omitted, the specified value takes effect in the following order (1 > 2):

1. Query group property file
2. System configuration property file

**(3) *querygroup.sleepOnOverStoreRetryCount=number-of-times-to-check-output-stream-queue-for-available-space***

This parameter specifies as an integer from 0 to 2147483647 the number of times the SDP server is to check the output stream queue for available space before it loads the tuples obtained from query execution to the output stream queue.

If the value 0 is specified, the SDP server loads the tuples obtained from query execution to the output stream queue without checking the output stream queue for available space.

If 1 or a greater value is specified, the SDP server performs the following processing as many times as specified:

1. The SDP server checks whether there is room in the output stream queue.
2. If there is no room in the output stream queue, the SDP server places execution of the query group that uses the output stream queue in the sleep mode. The length of the sleep time is specified in the `querygroup.sleepOnOverStore` parameter.

When the sleep period expires, if this processing has not been performed as many times as specified in this parameter, the SDP server returns to step 1.

3. If there is room in the output stream queue, the SDP server loads the tuples obtained from query execution into the output stream queue.

You can specify this parameter in the system configuration property file and the query

group property file. If the parameter is duplicated or omitted, the specified value takes effect in the following order (1 > 2):

1. Query group property file
2. System configuration property file

**(4) `stream.filterCondition=conditional-expression`**

This parameter specifies a conditional expression for filtering in order to use the timestamp adjustment function to filter tuples. A specified conditional operation is performed on the contents of the column whose name is specified in the conditional expression and on a specified constant. As a result of the conditional operation, the timestamp adjustment function retains tuples satisfying the condition and discards tuples that do not satisfy the condition.

For details about how to specify the conditional expression, see *8.7.3 Coding rules for conditional expressions*.

You must specify this parameter when you specify `condition` in the `stream.filterMode` parameter. If you specify `unuse` in the `stream.filterMode` parameter, this parameter is ignored, but its format is checked.

You can specify this parameter in the query group property file and the stream property file. If the parameter is duplicated or omitted, the specified value takes effect in the following order (1 > 2):

1. Stream property file
2. Query group property file

**(5) `stream.filterMode={unuse|condition}`**

This parameter specifies, as `unuse` or `condition`, whether the timestamp adjustment function is to be used to filter tuples. This value is not case sensitive. The default value is `unuse`.

`unuse`

Does not filter tuples.

`condition`

Filters tuples.

When you specify `condition`, you must also specify the `stream.filterCondition` parameter.

If you specify `unuse` in the `stream.timestampAccuracy` parameter, this parameter is ignored, but its format is checked.

You can specify this parameter in the query group property file and the stream property file. If the parameter is duplicated or omitted, the specified value takes effect in the

following order (1 > 2):

1. Stream property file
2. Query group property file

**(6) `stream.freeInputQueueSizeThreshold=threshold-value-for-available-size-to-maximum-size-of-input-stream-queue`**

This parameter specifies a threshold value (%) for the available size with respect to the maximum number of elements used in the input stream queue as an integer from 1 to 99 (the `engine.maxQueueSize` parameter value specified in the system configuration property file).

When the condition shown below is satisfied, a `SDPClientFreeInputQueueSizeThresholdOverException` exception is thrown from the `put(StreamTuple tuple)` method or the `put(ArrayList<StreamTuple> tuple_list)` method. In such a case, loading of tuples into the input stream queue has been successful.

*This parameter's value  $\geq ((\text{available size of input stream queue} \div \text{maximum size of the input stream queue}) \times 100)$*

If this parameter is omitted, an exception by threshold value checking will not occur.

When the `sdptplput` command is used to load tuples to the input stream queue, this parameter is ignored.

You can specify this parameter in the system configuration property file, the query group property file, and the stream property file. If the parameter is duplicated or omitted, the specified value takes effect in the following order (1 > 2 > 3):

1. Stream property file
2. Query group property file
3. System configuration property file

**(7) `stream.freeInputQueueSizeThresholdOutputMessage={true|false}`**

This parameter specifies, as `true` or `false`, whether a warning messages (the `KFSP42032-W` message) is to be output to the SDP server's message log. This value is not case sensitive.

`true`

Outputs the warning message.

`false`

Does not output the warning message.

This parameter takes effect only when the

`stream.freeInputQueueSizeThreshold` parameter is specified.

Note that the warning message is output only when `true` is specified in this parameter and the following condition is satisfied:

*`stream.freeInputQueueSizeThreshold` parameter value  $\geq ((\text{available size of input stream queue} \div \text{maximum size of input stream queue}) \times 100)$*

You can specify this parameter in the system configuration property file, the query group property file, and the stream property file. If the parameter is duplicated or omitted, the specified value takes effect in the following order (1 > 2 > 3):

1. Stream property file
2. Query group property file
3. System configuration property file

**(8) `stream.maxKeepTupleCount=maximum-number-of-tuples-retained-by-times tamp-adjustment-function`**

This parameter specifies as an integer from 1 to 1048576 the maximum number of tuples that can be retained by the timestamp adjustment function.

The number of tuples specified in this parameter is used as the maximum value for the timestamp adjustment function for each input stream.

When the number of tuples exceeds the value specified in this parameter, the query group is shut down.

You can specify this parameter in the system configuration property file, the query group property file, and the stream property file. If the parameter is duplicated or omitted, the specified value takes effect in the following order (1 > 2 > 3):

1. Stream property file
2. Query group property file
3. System configuration property file

**(9) `stream.propertyFiles=stream-property-file-names`**

This parameter specifies the names of the stream property files when you specify properties for individual streams.

In specifying a file name in this parameter, do not include the path name. If you specify a path name, it will be handled as part of the file name.

To specify multiple file names, delimit them with the single-byte comma (,). An error results in the following cases:

- There is no file name preceding or following a comma.



- A delimiter is preceded or followed by a space or tab.
- More than one stream property file is specified for the same stream.

If a stream property file is specified for a stream that does not exist in the query group, the specified file is still analyzed, but the definitions contained in it are ignored.

**(10) *stream.timestampAccuracy={{sec|msec|usec},time-adjustment-range|unuse}***

This parameter specifies the time units and time adjustment range for the timestamp adjustment function. This value is not case sensitive.

You must specify this parameter when you specify `DataSource` in the `stream.timestampMode` parameter. If you specify `Server` in the `stream.timestampMode` parameter, this parameter is ignored, but its format is checked.

`{sec|msec|usec}, time-adjustment-range`

Specifies the time units and time adjustment range. There must be no spaces or tabs preceding or following the single-byte comma (,) between the time units (`sec`, `msec`, or `usec`) and the time adjustment range. If you violate this rule, an error results. If you specify `sec`, `msec`, or `usec` as the time units and 0 as the time adjustment range, the time adjustment range is applied only to the reference time.

The meaning of each value is as follows:

`sec`

Specifies that seconds are to be used as the time units.

`msec`

Specifies that milliseconds are to be used as the time units.

`usec`

Specifies that microseconds are to be used as the time units.

*time-adjustment-range*

Specifies as an integer the range of times to be adjusted for timestamp adjustment. The permitted value range depends on the time units, as shown in the table below.

Time units	Permitted value range
<code>sec</code> (seconds)	Integer from 0 to 59
<code>msec</code> (milliseconds)	Integer from 0 to 999
<code>usec</code> (microseconds)	Integer from 0 to 999

`unuse`

Specifies that time adjustment is not to be performed.

You can specify this parameter in the system configuration property file, the query group property file, and the stream property file. If the parameter is duplicated or omitted, the specified value takes effect in the following order (1 > 2 > 3):

1. Stream property file
2. Query group property file
3. System configuration property file

**(11) `stream.timestampMode={Server|DataSource}`**

This parameter specifies `Server` or `DataSource` as the timestamp mode to be used in timestamping tuples. This value is not case sensitive.

`Server`

Uses the server mode.

`DataSource`

Uses the data source mode.

If you specify `DataSource`, you must specify the `stream.timestampAccuracy` and `stream.timestampPosition` parameters.

You can specify this parameter in the system configuration property file and the query group property file. If the parameter is duplicated or omitted, the specified value takes effect in the following order (1 > 2):

1. Query group property file
2. System configuration property file

**(12) `stream.timestampPosition=time-data-column-name`**

This parameter specifies the name of the time-data column in tuples. This value is not case sensitive.

If you specify `DataSource` in the `stream.timestampMode` parameter, you must also specify this parameter. If you specify `Server` in the `stream.timestampMode` parameter, this parameter is ignored, but its format is checked.

The only data type that can be specified as time data is the `TIMESTAMP` type.

The permitted range of time data is from `1970/01/01 00:00:00.000000000` to `2261/12/31 23:59:59.999999999` in GMT (Greenwich Mean Time). If a specified time is outside this range, an exception occurs when the stream data is sent.

You can specify this parameter in the system configuration property file, the query

group property file, and the stream property file. If the parameter is duplicated or omitted, the specified value takes effect in the following order (1 > 2 > 3):

1. Stream property file
2. Query group property file
3. System configuration property file

**(13) *stream.tupleLogMode={true|false}***

This parameter specifies, as `true` or `false`, whether the `sdptplput` command is to be enabled. This value is not case sensitive.

`true`

`sdptplput` command is to be executed.

`false`

Execution of the `sdptplput` command is to be disabled.

If you specify `Server` in the `stream.timestampMode` parameter and `true` in this parameter, an `SDPClientException` exception will occur if you then send stream data using the `put` method of the `StreamInput` interface.

You can specify this parameter in the system configuration property file and the query group property file. If the parameter is duplicated or omitted, the specified value takes effect in the following order (1 > 2):

1. Query group property file
2. System configuration property file

**(14) *tpl.backupFileCount=maximum-number-of-backup-generations-retained-for-tuple-log-file***

This parameter specifies as an integer from 0 to 10 the maximum number of backup generations to be retained for the tuple log file.

You can specify this parameter in the system configuration property file, the query group property file, and the stream property file. If the parameter is duplicated or omitted, the specified value takes effect in the following order (1 > 2 > 3):

1. Stream property file
2. Query group property file
3. System configuration property file

**(15) *tpl.bufferCount=tuple-log-buffers-count***

This parameter specifies the number of tuple log buffers, as an integer from 3 to 512.

You can specify this parameter in the system configuration property file, the query

group property file, and the stream property file. If the parameter is duplicated or omitted, the specified value takes effect in the following order (1 > 2 > 3):

1. Stream property file
2. Query group property file
3. System configuration property file

**(16) *tpl.bufferSize=maximum-tuple-log-buffer-size***

This parameter specifies as an integer from 1 to 2048000 the maximum size of a tuple log buffer (in kilobytes).

You can specify this parameter in the system configuration property file, the query group property file, and the stream property file. If the parameter is duplicated or omitted, the specified value takes effect in the following order (1 > 2 > 3):

1. Stream property file
2. Query group property file
3. System configuration property file

**(17) *tpl.fileCount=maximum-tuple-log-files-count***

This parameter specifies the maximum number of tuple log files, as an integer from 3 to 512.

You can specify this parameter in the system configuration property file, the query group property file, and the stream property file. If the parameter is duplicated or omitted, the specified value takes effect in the following order (1 > 2 > 3):

1. Stream property file
2. Query group property file
3. System configuration property file

**(18) *tpl.fileSize=maximum-tuple-log-file-size***

This parameter specifies the maximum size of a tuple log file (in megabytes), as an integer from 1 to 2048.

You can specify this parameter in the system configuration property file, the query group property file, and the stream property file. If the parameter is duplicated or omitted, the specified value takes effect in the following order (1 > 2 > 3):

1. Stream property file
2. Query group property file
3. System configuration property file

**(19) *tpl.outputLevel=tuple-log-output-level***

This parameter specifies the tuple log output level, as an integer from 1 to 3.

1

Outputs tuple logs for the tuples that are stored in the stream queue.

2

Outputs tuple logs for the tuples that are discarded due to an out-of-sequence time.

3

Outputs tuple logs for tuples with levels 1 and 2.

You can specify this parameter in the system configuration property file, the query group property file, and the stream property file. If the parameter is duplicated or omitted, the specified value takes effect in the following order (1 > 2 > 3):

1. Stream property file
2. Query group property file
3. System configuration property file

**(20) *tpl.outputTrigger={BUFFER|NONE}***

This parameter specifies `BUFFER` or `NONE` as the tuple log file output timing. This value is not case sensitive.

`BUFFER`

Outputs tuple logs to a file when the buffer being used to collect the current tuple logs in the target stream becomes full.

`NONE`

Does not output tuple logs to a file or perform buffering of tuple logs.

You can specify this parameter in the system configuration property file, the query group property file, and the stream property file. If the parameter is duplicated or omitted, the specified value takes effect in the following order (1 > 2 > 3):

1. Stream property file
2. Query group property file
3. System configuration property file

**(21) *tpl.useOverwrite={true|false}***

This parameter specifies, as `true` or `false`, whether a tuple log buffer is to be overwritten in the event it becomes full. This value is not case sensitive.

true

Overwrites the tuple log buffer.

false

Does not overwrite the tuple log buffer.

You can specify this parameter in the system configuration property file, the query group property file, and the stream property file. If the parameter is duplicated or omitted, the specified value takes effect in the following order (1 > 2 > 3):

1. Stream property file
2. Query group property file
3. System configuration property file

### 8.7.3 Coding rules for conditional expressions

This subsection discusses the coding rules for a conditional expression specified in the `stream.filterCondition` parameter.

#### (1) Format of conditional expression

Specify a conditional expression in the following format:

```
' ('column-name comparison-operator constant')' [{AND | OR}' ('column-name comparison-operator constant')'
{AND | OR}' ('column-name comparison-operator constant')'...]
```

To combine multiple comparison operations, use AND or OR to join the expressions. You can specify a maximum of ten expressions.

The following describes the elements of a conditional expression.

#### *column-name*

Specifies the name of the column that is to be subject to the comparison operation. This column name must be specified as a schema specification character string in the stream definition (`register stream` clause).

#### *comparison-operator*

Specifies the operator to be used in evaluating the condition. The table below lists and describes the comparison operators that can be specified.

Comparison operator	Usage example	Meaning
<=	A <= B	A is equal to or less than B
>=	A >= B	A is equal to or greater than B
<	A < B	A is less than B

Comparison operator	Usage example	Meaning
>	A > B	A is greater than B
=	A = B	A is equal to B
!=	A != B	A is not equal to B

*constant*

Specifies the value on which the comparison operation is to be performed, expressed as an integer constant or a character-string constant.

- Integer constant

If you use a numeric value for the comparison operation, specify an integer constant. The permitted value is an integer in the range from -9223372036854775808 to 9223372036854775807.

- Character-string constant

If you use a character string for the comparison operation, specify a character-string constant enclosed in single quotation marks ('). The permitted value is a string of single-byte and double-byte characters with a maximum length of 100 characters.

**(2) Example of conditional expression**

The following shows an example of a conditional expression:

```
stream.filterCondition=(xxx='abc') AND (zzz>18) AND (zzz<60)
```

This example retains only those tuples whose column *xxx* is *abc* and column *zzz* is greater than 18 and smaller than 60.

**(3) Notes on conditional expressions**

- AND and OR cannot be intermixed in the same expression.
- AND, OR, and column names are not case sensitive.
- If the column whose name is specified has a character data type (CHAR or VARCHAR), the only permitted comparison operators are = and !=.
- When an integer constant is specified, the data types permitted for the column name are those that are permitted in CQL, as shown below:
  - INT
  - SMALLINT
  - TINYINT

- BIGINT
- DEC
- NUMERIC
- REAL
- FLOAT
- DOUBLE

For details about the data types permitted in CQL, see the manual *uCosminexus Stream Data Platform - Application Framework Application Development Guide*.

- When you use an integer constant, the data type of the column whose name is specified and whether or not the specified integer constant is within the permissible value range are not checked.
- A character-string constant must be enclosed in single quotation marks ( ' ). If you need to use a single quotation mark as part of the constant, specify two single quotation marks in succession to represent the one single quotation mark. For example, to specify the constant `abc'abc`, specify `'abc''abc'`.
- When the character string in the column is longer than the character string specified as the character-string constant, only the leading part of the column's character string equivalent to the specified length is used for the comparison operation. For example, if the character string specified as the character-string constant is `'abc'` (three characters) and the character string contained in the corresponding column is `'abcde'` (five characters) only the first three characters `'abc'` are used for the comparison. In this example, the character string in the column would be considered to match the specified character-string constant.
- The null character (nothing specified between the two single quotation marks indicating the beginning and end of a character string) cannot be specified as a character-string constant. You can specify space and tabs before and after parentheses, comparison operators, AND, and OR, but such spaces and tab are ignored. Note that there must be no spaces or tabs following the closing parenthesis of the final expression.
- When a character-string constant is specified, the data types permitted for the column name are those that are permitted in CQL, as shown below:
  - CHAR
  - VARCHAR

For details about the data types permitted in CQL, see the manual *uCosminexus Stream Data Platform - Application Framework Application Development Guide*.



---

## 8.8 Stream property file

---

This section provides the details of the stream property file and the parameters that are specified in the file.

### 8.8.1 Details of the stream property file

#### (1) **Format**

Specify each parameter in the following format:

<i>parameter-name=value</i>
-----------------------------

- The value cannot be followed by spaces or a character string (such as a comment). Anything that follows the value will be considered part of the value.
- A line containing only a parameter name and no value will be ignored.

#### (2) **File name**

You can specify any file name. This file name must be specified in the `stream.propertyFiles` parameter in the applicable query group property file.

#### (3) **File storage location**

This file must be stored in the following directory:  
*working-directory*\conf\

#### (4) **Description**

This file specifies tuning parameters for a stream in a query group. You create this file for a stream only if you want to specify tuning parameters specifically for the stream.

#### (5) **Specifiable parameters**

The table below lists and describes the parameters that can be specified and their default values. For details about the parameters, see *8.8.2 Details of the parameters in the stream property file*.

Table 8-6: Specifiable parameters and their default values (stream property file)

No.	Parameter name	Description	Default value	Permitted value range	Whether or not changeable during restart <sup>#1</sup>
1	<code>stream.filterCondition</code>	Specifies a conditional expression for using the timestamp adjustment function to filter tuples.	Value specified in the query group property file <sup>#2</sup>	--	Y
2	<code>stream.filterMode</code>	Specifies whether the timestamp adjustment function is to be used to filter tuples: unuse: Does not filter tuples. condition: Filters tuples.	Value specified in the query group property file <sup>#2</sup>	unuse or condition	Y
3	<code>stream.freeInputQueueSizeThreshold</code>	Specifies a threshold value (%) for the available size with respect to the maximum number of elements used in the input stream queue.	Value specified in the query group property file <sup>#2</sup>	Integer from 1 to 99	Y
4	<code>stream.freeInputQueueSizeThresholdOutputMessage</code>	Specifies whether a warning message is to be output to the SDP server's message log: true: Outputs warning message. false: Does not output warning message.	Value specified in the query group property file <sup>#2</sup>	true or false	Y
5	<code>stream.maxKeepTupleCount</code>	Specifies the maximum number of tuples that can be retained by the timestamp adjustment function.	Value specified in the query group property file <sup>#2</sup>	Integer from 1 to 1048576	Y
6	<code>stream.streamName</code>	Specifies the stream name.	None	--	Y

No.	Parameter name	Description	Default value	Permitted value range	Whether or not changeable during restart <sup>#1</sup>
7	<code>stream.timestampAccuracy</code>	Specifies the time units and time adjustment range for the timestamp adjustment function.	Value specified in the query group property file <sup>#2</sup>	See 8.8.2(7) <i>stream.timestampAccuracy</i> = <code>{{sec msec usec},time-adjustment-range unuse}</code> .	Y
8	<code>stream.timestampPosition</code>	Specifies the name of the time-data column in tuples.	Value specified in the query group property file <sup>#2</sup>	See 8.8.2(8) <i>stream.timestampPosition</i> = <code>time-data-column-name</code> .	Y
9	<code>tpl.backupFileCount</code>	Specifies the maximum number of backup generations to be retained for the tuple log file.	Value specified in the query group property file <sup>#2</sup>	Integer from 0 to 10	Y
10	<code>tpl.bufferCount</code>	Specifies the number of tuple log buffers.	Value specified in the query group property file <sup>#2</sup>	Integer from 3 to 512	Y
11	<code>tpl.bufferSize</code>	Specifies the maximum size of a tuple log buffer (in kilobytes).	Value specified in the query group property file <sup>#2</sup>	Integer from 1 to 2048000	Y
12	<code>tpl.fileCount</code>	Specifies the maximum number of tuple log files.	Value specified in the query group property file <sup>#2</sup>	Integer from 3 to 512	Y
13	<code>tpl.fileSize</code>	Specifies the maximum size of a tuple log file (in megabytes).	Value specified in the query group property file <sup>#2</sup>	Integer from 1 to 2048	Y

No.	Parameter name	Description	Default value	Permitted value range	Whether or not changeable during restart <sup>#1</sup>
14	<code>tpl.outputLevel</code>	Specifies the tuple log output level: <ul style="list-style-type: none"> <li>• 1: Outputs tuple logs for the tuples that are stored in the stream queue.</li> <li>• 2: Outputs tuple logs for the tuples that are discarded due to out-of-sequence time.</li> <li>• 3: Outputs tuple logs for tuples with levels 1 and 2.</li> </ul>	Value specified in the query group property file <sup>#2</sup>	Integer from 1 to 3	Y
15	<code>tpl.outputTrigger</code>	Specifies the tuple log file output timing: <b>BUFFER</b> : Outputs tuple logs to a file when the buffer being used to collect the current tuple logs in the target stream becomes full. <b>NONE</b> : Does not output tuple logs to a file or perform buffering of tuple logs.	Value specified in the query group property file <sup>#2</sup>	BUFFER or NONE	Y
16	<code>tpl.useOverwrite</code>	Specifies whether the buffer is to be overwritten in the event of a full tuple log buffer: <ul style="list-style-type: none"> <li>• <code>true</code>: Overwrites the tuple log buffer.</li> <li>• <code>false</code>: Does not overwrite the tuple log buffer.</li> </ul>	Value specified in the query group property file <sup>#2</sup>	<code>true</code> or <code>false</code>	Y

Legend:

--: Not applicable

Y: Changeable

#1

Indicates whether (Y) or not (N) the parameter's setting can be changed when the

query group is started by the `sdpcqlstart` command with the `-reload` option specified.

#2

If no value is specified in the query group property file, the default value for the query group property file is used.

### (6) Notes

- To change contents of a stream property file after you have registered a query group, use the following procedure:

1. Delete the registered query group.
2. Edit the property file.
3. Re-register the query group.

However, you do not need to delete the query group to change the values of the parameters for which Y is shown in the column *Whether or not changeable during restart* in the table in (5) *Specifiable parameters*; instead, you can follow the procedure below to change any of these parameter values:

1. Terminate the query group.
2. Edit the property file.
3. Restart the query group with the `sdpcqlstart` command with the `-reload` option specified.

## 8.8.2 Details of the parameters in the stream property file

This subsection provides the details of the parameters in the stream property file shown in 8.8.1(5) *Specifiable parameters*.

### (1) *stream.filterCondition=conditional-expression*

This parameter specifies a conditional expression for filtering in order to use the timestamp adjustment function to filter tuples. A specified conditional operation is performed on the contents of the column whose name is specified in the conditional expression and on a specified constant. As a result of the conditional operation, the timestamp adjustment function retains tuples satisfying the condition and discards tuples that do not satisfy the condition.

For details about how to specify the conditional expression, see 8.7.3 *Coding rules for conditional expressions*.

You must specify this parameter when you specify `condition` in the `stream.filterMode` parameter. If you specify `unuse` in the `stream.filterMode` parameter, this parameter is ignored, but its format is checked.

You can specify this parameter in the query group property file and the stream property

file. If the parameter is duplicated or omitted, the specified value takes effect in the following order (1 > 2):

1. Stream property file
2. Query group property file

**(2) `stream.filterMode={unuse|condition}`**

This parameter specifies, as `unuse` or `condition`, whether the timestamp adjustment function is to be used to filter tuples. This value is not case sensitive.

`unuse`

Does not filter tuples.

`condition`

Filters tuples.

When you specify `condition`, you must also specify the `stream.filterCondition` parameter.

If you specify `unuse` in the `stream.timestampAccuracy` parameter, this parameter is ignored, but its format is checked.

You can specify this parameter in the query group property file and the stream property file. If the parameter is duplicated or omitted, the specified value takes effect in the following order (1 > 2):

1. Stream property file
2. Query group property file

**(3) `stream.freeInputQueueSizeThreshold=threshold-value-for-available-size-to-maximum-size-of-input-stream-queue`**

This parameter specifies a threshold value (%) for the available size with respect to the maximum number of elements used in the input stream queue as an integer from 1 to 99 (the `engine.maxQueueSize` parameter value specified in the system configuration property file).

When the condition shown below is satisfied, a `SDPClientFreeInputQueueSizeThresholdOverException` exception is thrown from the `put(StreamTuple tuple)` method or the `put(ArrayList<StreamTuple> tuple_list)` method. In such a case, loading of tuples into the input stream queue has been successful.

*This parameter's value  $\geq ((\text{available size of input stream queue} \div \text{maximum size of the input stream queue}) \times 100)$*

If this parameter is omitted, an exception by threshold value checking will not occur.

When the `sdptplput` command is used to input tuples to the input stream queue, this parameter is ignored.

You can specify this parameter in the system configuration property file, the query group property file, and the stream property file. If the parameter is duplicated or omitted, the specified value takes effect in the following order (1 > 2 > 3):

1. Stream property file
2. Query group property file
3. System configuration property file

**(4) `stream.freeInputQueueSizeThresholdOutputMessage={true|false}`**

This parameter specifies, as `true` or `false`, whether a warning message (the `KFSP42032-W` message) is to be output to the SDP server's message log. This value is not case sensitive.

`true`

Outputs the warning message.

`false`

Does not output the warning message.

This parameter takes effect only when the `stream.freeInputQueueSizeThreshold` parameter is specified.

Note that the warning message is output only when `true` is specified in this parameter and the following condition is satisfied:

$$\text{stream.freeInputQueueSizeThreshold parameter value} \geq ((\text{available size of input stream queue} \div \text{maximum size of input stream queue}) \times 100)$$

You can specify this parameter in the system configuration property file, the query group property file, and the stream property file. If the parameter is duplicated or omitted, the specified value takes effect in the following order (1 > 2 > 3):

1. Stream property file
2. Query group property file
3. System configuration property file

**(5) `stream.maxKeepTupleCount=maximum-number-of-tuples-retained-by-times tamp-adjustment-function`**

This parameter specifies as an integer from 1 to 1048576 the maximum number of tuples that can be retained by the timestamp adjustment function.

The number of tuples specified in this parameter is used as the maximum value for the

timestamp adjustment function for the input stream.

If the number of tuples exceeds the maximum value specified in this parameter, the query group is shut down.

You can specify this parameter in the system configuration property file, the query group property file, and the stream property file. If the parameter is duplicated or omitted, the specified value takes effect in the following order (1 > 2 > 3):

1. Stream property file
2. Query group property file
3. System configuration property file

**(6) *stream.streamName=stream-name***

This parameter specifies the name of the stream to which the definitions in this property file are to be applied.

The specified stream name is treated as being all upper-case letters. If this parameter is omitted, an error results.

**(7) *stream.timestampAccuracy={{sec|msec|usec},time-adjustment-range|unuse}***  
**}**

This parameter specifies the time units and time adjustment range for the timestamp adjustment function. This value is not case sensitive.

*{ sec | msec | usec } , time-adjustment-range*

Specifies the time units and time adjustment range. There must be no spaces or tabs preceding or following the single-byte comma ( , ) between the time units (*sec*, *msec*, or *usec*) and the time adjustment range. If you violate this rule, an error results. If you specify *sec*, *msec*, or *usec* as the time units and 0 as the time adjustment range, the time adjustment range is applied only to the reference time.

The meaning of each value is as follows:

*sec*

Specifies that seconds are to be used as the time units.

*msec*

Specifies that milliseconds are to be used as the time units.

*usec*

Specifies that microseconds are to be used as the time units.

*time-adjustment-range*

Specifies as an integer the range of times to be adjusted for timestamp adjustment. The permitted value range depends on the time units, as shown



in the table below.

Time units	Permitted value range
sec (seconds)	Integer from 0 to 59
msec (milliseconds)	Integer from 0 to 999
usec (microseconds)	Integer from 0 to 999

unuse

Specifies that time adjustment is not to be performed.

You can specify this parameter in the system configuration property file, the query group property file, and the stream property file. If the parameter is duplicated or omitted, the specified value takes effect in the following order (1 > 2 > 3):

1. Stream property file
2. Query group property file
3. System configuration property file

**(8) *stream.timestampPosition=time-data-column-name***

This parameter specifies the name of the time-data column in tuples. This value is not case sensitive.

The only data type that can be specified as time data is the `TIMESTAMP` type.

The permitted range of time data is from 1970/01/01 00:00:00.000000000 to 2261/12/31 23:59:59.999999999 in GMT (Greenwich Mean Time). If a specified time is outside this range, an exception occurs when the stream data is sent.

You can specify this parameter in the system configuration property file, the query group property file, and the stream property file. If the parameter is duplicated or omitted, the specified value takes effect in the following order (1 > 2 > 3):

1. Stream property file
2. Query group property file
3. System configuration property file

**(9) *tpl.backupFileCount=maximum-number-of-backup-generations-retained-for-tuple-log-file***

This parameter specifies as an integer from 0 to 10 the maximum number of backup generations to be retained for the tuple log file.

You can specify this parameter in the system configuration property file, the query group property file, and the stream property file. If the parameter is duplicated or omitted, the specified value takes effect in the following order (1 > 2 > 3):

1. Stream property file
2. Query group property file
3. System configuration property file

**(10) *tpl.bufferCount=tuple-log-buffers-count***

This parameter specifies the number of tuple log buffers, as an integer from 3 to 512.

You can specify this parameter in the system configuration property file, the query group property file, and the stream property file. If the parameter is duplicated or omitted, the specified value takes effect in the following order (1 > 2 > 3):

1. Stream property file
2. Query group property file
3. System configuration property file

**(11) *tpl.bufferSize=maximum-tuple-log-buffer-size***

This parameter specifies as an integer from 1 to 2048000 the maximum size of a tuple log buffer (in kilobytes).

You can specify this parameter in the system configuration property file, the query group property file, and the stream property file. If the parameter is duplicated or omitted, the specified value takes effect in the following order (1 > 2 > 3):

1. Stream property file
2. Query group property file
3. System configuration property file

**(12) *tpl.fileCount=maximum-tuple-log-files-count***

This parameter specifies the maximum number of tuple log files, as an integer from 3 to 512.

You can specify this parameter in the system configuration property file, the query group property file, and the stream property file. If the parameter is duplicated or omitted, the specified value takes effect in the following order (1 > 2 > 3):

1. Stream property file
2. Query group property file
3. System configuration property file

**(13) *tpl.fileSize=maximum-tuple-log-file-size***

This parameter specifies the maximum size of a tuple log file (in megabytes), as an integer from 1 to 2048.

You can specify this parameter in the system configuration property file, the query

group property file, and the stream property file. If the parameter is duplicated or omitted, the specified value takes effect in the following order (1 > 2 > 3):

1. Stream property file
2. Query group property file
3. System configuration property file

**(14) *tpl.outputLevel=tuple-log-output-level***

This parameter specifies the tuple log output level, as an integer from 1 to 3.

1

Outputs tuple logs for the tuples that are stored in the stream queue.

2

Outputs tuple logs for the tuples that are discarded due to an out-of-sequence time.

3

Outputs tuple logs for tuples with levels 1 and 2.

You can specify this parameter in the system configuration property file, the query group property file, and the stream property file. If the parameter is duplicated or omitted, the specified value takes effect in the following order (1 > 2 > 3):

1. Stream property file
2. Query group property file
3. System configuration property file

**(15) *tpl.outputTrigger={BUFFER|NONE}***

This parameter specifies BUFFER or NONE as the tuple log file output timing. This value is not case sensitive.

BUFFER

Outputs tuple logs to a file when the buffer being used to collect the current tuple logs in the target stream becomes full.

NONE

Does not output tuple logs to a file or perform buffering of tuple logs.

You can specify this parameter in the system configuration property file, the query group property file, and the stream property file. If the parameter is duplicated or omitted, the specified value takes effect in the following order (1 > 2 > 3):

1. Stream property file
2. Query group property file

3. System configuration property file

**(16) *tpl.useOverwrite={true|false}***

This parameter specifies, as `true` or `false`, whether a tuple log buffer is to be overwritten in the event it becomes full. This value is not case sensitive.

`true`

Overwrites the tuple log buffer.

`false`

Does not overwrite the tuple log buffer.

You can specify this parameter in the system configuration property file, the query group property file, and the stream property file. If the parameter is duplicated or omitted, the specified value takes effect in the following order (1 > 2 > 3):

1. Stream property file
2. Query group property file
3. System configuration property file

---

## 8.9 In-process connection property file (`user_app.adaptor-group-name-or-adaptor-name.properties`)

---

### (1) Format

Specify each parameter in the following format:

<i>parameter-name=value</i>
-----------------------------

- The value cannot be followed by spaces or a character string (such as a comment). Anything that follows the value will be considered part of the value.
- A line containing only a parameter name and no value will be ignored.

### (2) File name

`user_app.adaptor-group-name-or-adaptor-name.properties`

Specify an adaptor group name for standard adaptors or an adaptor name for a custom adaptor, expressed as alphanumeric characters (0 to 9, a to z, A to Z) and the underscore (`_`). The permitted length is from 1 to 32 characters. Note that for an adaptor group or adaptor that uses in-process connection, the adaptor group name or adaptor name must begin with a single-byte alphabetic character.

### (3) File storage location

This file must be stored in the following directory:  
`working-directory\conf\`

### (4) Description

This file specifies the class name of in-process-connection adaptors or the path of a `jar` file. When you use in-process connection, you must create this file for each adaptor group when using the standard adaptors and for each adaptor when using custom adaptors.

### (5) Specifiable parameters

The table below lists and describes the parameters that can be specified and their default values.

*Table 8-7: Specifiable parameters and their default values (in-process connection property file)*

No.	Parameter name	Description	Default value
1	<code>user_app.classname</code>	Specifies the main class name for an adaptor group or an adaptor that uses in-process connection. <sup>#1</sup> This class is implemented with the system-provided interface class. When you use in-process connection, this parameter is mandatory; if this parameter is omitted in such a case, an error results.	None
2	<code>user_app.classpath_dir</code>	Specifies the path of the directory storing the class file for the class specified in the <code>user_app.classname</code> parameter or the path of the jar file. <sup>#1, #2</sup> For the path, specify an absolute path or a path relative to the working directory. You can specify only one path in this parameter. When you use in-process connection, this parameter is mandatory; if this parameter is omitted in such a case, an error results.	None

#1

For standard adaptors, the value is fixed. Specify the following value:

```
user_app.classname=jp.co.Hitachi.soft.sdp.adaptor.AdaptorManager
user_app.classpath_dir=installation-directory\\lib\\sdp.jar
```

#2

If the path specified in the `SDP_CLASS_PATH` parameter in the JavaVM options file for SDP servers contains a jar file, that jar file takes precedence.

If a user-created in-process-connection adaptor is to reference a path other than the class path specified in this parameter, specify that path in the `SDP_CLASS_PATH` parameter in the JavaVM options file for SDP servers. Note that the path specified in the `SDP_CLASS_PATH` parameter in the JavaVM options file for SDP servers is read only when the SDP server starts. Therefore, if you make a change to the path value or the contents of the jar file after the SDP server has started, that change will not take effect.

## 8.10 Log file output property file (logger.properties)

### (1) Format

Specify each parameter in the following format:

```
parameter-name=value
```

- The value cannot be followed by spaces or a character string (such as a comment). Anything that follows the value will be considered part of the value.
- A line containing only a parameter name and no value will be ignored.

### (2) File name

```
logger.properties
```

### (3) File storage location

This file must be stored in the following directory:

```
working-directory\conf\
```

### (4) Description

This file specifies the numbers and sizes of message and trace log files. You create this file for each working directory. If this file is not created, the default values are used.

### (5) Specifiable parameters

The table below lists and describes the parameters that can be specified and their default values.

Table 8-8: Specifiable parameters and their default values (logger.properties)

No.	Parameter name	Description	Default value	Permitted value range
1	logger.serverMessageFileCount	Specifies the maximum number of message log files.	3	Integer from 2 to 16
2	logger.serverMessageMaxFileSize	Specifies the maximum size (in bytes) of a message log file.	1048576	Integer from 4096 to 2147483647
3	logger.serverTraceFileCount	Specifies the maximum number of trace log files.	3	Integer from 2 to 16
4	logger.serverTraceMaxFileSize	Specifies the maximum size (in bytes) of a trace log file.	1048576	Integer from 4096 to 2147483647

## 8.11 List of JavaVM options

This section discusses Hitachi JavaVM options. You can specify Hitachi JavaVM options in the following files:

- JavaVM options file for SDP servers (`jvm_options.cfg`)

For details about how to specify the JavaVM options in this file and their default values in this system, see 8.4 JavaVM options file for SDP servers (`jvm_options.cfg`).

- JavaVM options file for RMI connections (`jvm_client_options.cfg`)

For details about how to specify the JavaVM options in this file and their default values in this system, see 8.5 *JavaVM options file for RMI connections* (`jvm_client_options.cfg`).

The table below lists and describes the JavaVM options. In this table, the default value is the value that is assumed when the corresponding JavaVM option is omitted from the JavaVM options file for SDP servers (`jvm_options.cfg`) or the JavaVM options file for RMI connections (`jvm_client_options.cfg`).

Table 8-9: List of JavaVM options

No.	Classification	Option name	Description
1	Options for specifying the size and ratio of JavaVM memory space	<code>-XmsSize<sup>#1</sup></code>	Specifies the initial Java heap size.
		<code>-XmxSize<sup>#1</sup></code>	Specifies the maximum Java heap size.
		<code>-XX:NewRatio=value</code>	Specifies the ratio of the Tenured area to the DefNew area. If <i>value</i> is 2, the ratio of the Tenured area to the DefNew area is 1:2.
		<code>-XX:PermSize=size<sup>#1</sup></code>	Specifies the initial size of the Permanent area.
		<code>-XX:MaxPermSize=size<sup>#1</sup></code>	Specifies the maximum size of the Permanent area.



No.	Classification	Option name	Description
2	Extended thread dump function option	-XX:[+ -]HitachiThreadDumpToStdout	<p>Specifies whether a thread dump is to be output to the standard output:</p> <ul style="list-style-type: none"> <li>-XX:+HitachiThreadDumpToStdout: Outputs an extended thread dump to the standard output and Hitachi JavaVM log file.</li> <li>-XX:-HitachiThreadDumpToStdout: Outputs an extended thread dump only to the Hitachi JavaVM log file, not to the standard output.</li> </ul> <p>The default value is -XX:+HitachiThreadDumpToStdout.</p>
3	Hitachi JavaVM log file options	-XX:HitachiJavaLog:character-string	<p>Specifies a prefix for the log file name. A log file name is created in the format <i>character-string</i>xx.log (xx: serial number, 01 to 99).</p> <p>For example, if you specify <i>samp</i> for <i>character-string</i>, the log file name would be <i>samp01.log</i>.</p> <p>For <i>character-string</i>, you can also specify a path. If you specify a directory for <i>character-string</i>, a log file is created with the default name in the specified directory.</p> <p>The default value is <code>javalog</code>.</p>
		-XX:HitachiJavaLogFileSize= <i>integer-value</i>	<p>Specifies the maximum size of one file (in kilobytes), as an integer from 1024 to the value of <code>INT_MAX</code>. If the specified value exceeds the maximum size, nothing will be output to that file.</p> <p>The default value is 256.</p> <p>If the specified value is not within the permitted value range, 1024 is assumed.</p>
		-XX:HitachiJavaLogNumberOfFile= <i>integer-value</i>	<p>Specifies the maximum number of files that can be created (in order to limit the number of log files), as an integer from 1 to 99. When the maximum number of files is reached, data is output again to the first file that was created.</p> <p>The default value is 4.</p> <p>If a value greater than 99 is specified, 99 is assumed. If 0 or a smaller value is specified, 1 is assumed.</p>

8. SDP Server Definition Files

No.	Classification	Option name	Description
4	Detailed time output option	-XX: [+ -]HitachiOutputMilliTime	<p>Specifies whether the time is to be output in milliseconds:</p> <ul style="list-style-type: none"> <li>-XX:+HitachiOutputMilliTime: Outputs the time to the Hitachi JavaVM log file in milliseconds.</li> <li>-XX:-HitachiOutputMilliTime: Outputs the time to the Hitachi JavaVM log file in seconds.</li> </ul> <p>The default value is -XX:-HitachiOutputMilliTime.</p>
5	Extended verbosegc function option	-XX: [+ -]HitachiVerboseGC <sup>#2</sup>	<p>Specifies whether the extended verbosegc information is to be output when a garbage collection occurs:</p> <ul style="list-style-type: none"> <li>-XX:+HitachiVerboseGC: Outputs the extended verbosegc information to the Hitachi JavaVM log file when a garbage collection occurs.</li> </ul> <p>The extended verbosegc information includes information about the Eden, Survivor, Tenured, and Perm types, which are internal areas for garbage collection.</p> <ul style="list-style-type: none"> <li>-XX:-HitachiVerboseGC: Does not output the extended verbosegc information to the Hitachi JavaVM log file when a garbage collection occurs.</li> </ul> <p>The default value is -XX:-HitachiVerboseGC.</p>
		-XX: [+ -]HitachiVerboseGCPrintCause <sup>#3</sup>	<p>Specifies whether the cause of garbage collection is to be output.</p> <ul style="list-style-type: none"> <li>-XX:+HitachiVerboseGCPrintCause: Outputs the cause of garbage collection at the end of the extended verbosegc information.</li> <li>-XX:-HitachiVerboseGCPrintCause: Outputs the extended verbosegc information in the regular format.</li> </ul> <p>The default value is -XX:+HitachiVerboseGCPrintCause.</p>

No.	Classification	Option name	Description
6	Extended function option in the event of OutOfMemory Error	-XX: [+   -]HitachiOutOfMemoryStackTrace <sup>#2</sup>	<p>Specifies whether a stack trace is to be output in the event of OutOfMemoryError.</p> <ul style="list-style-type: none"> <li>-XX: +HitachiOutOfMemoryStackTrace: Outputs the exception information and stack trace to the Hitachi JavaVM log file in the event of OutOfMemoryError. For the stack trace, each stack is placed in buffer and then output after code conversion. Because a stack trace is output each time OutOfMemoryError is thrown, it might be output more than once when OutOfMemoryError is thrown again. Note that if OutOfMemoryError occurs during thread creation, stack trace is not output.</li> <li>-XX: -HitachiOutOfMemoryStackTrace: Does not output a stack trace to the Hitachi JavaVM log file in the event of OutOfMemoryError.</li> </ul> <p>The default value is -XX: -HitachiOutOfMemoryStackTrace.</p>
7	Class library trace function options	-XX: [+   -]HitachiJavaClassLibTrace <sup>#2</sup>	<p>Specifies whether a stack trace is to be output for the class library.</p> <ul style="list-style-type: none"> <li>-XX: +HitachiJavaClassLibTrace: Outputs a stack trace of the class library.</li> <li>-XX: -HitachiJavaClassLibTrace: Does not output a stack trace of the class library.</li> </ul> <p>The default value is -XX: -HitachiJavaClassLibTrace.</p>
		-XX: HitachiJavaClassLibTraceLineSize= <i>integer-value</i>	<p>Specifies the number of characters (in bytes) per line of class library stack trace, as an integer from 1024 to the value of INT_MAX. If there are more characters than specified, the leading part of the character string that follows at is deleted and as many characters as are specified are output.</p> <p>The default value is 1024.</p> <p>If the specified value is not within the permitted value range, 1024 is assumed.</p>

No.	Classification	Option name	Description
8	Local variable information output function options	<code>-XX: [+ -]HitachiLocalsInStackTrace</code>	<p>Specifies whether the local variable information is to be output to the stack trace during a thread dump output.</p> <ul style="list-style-type: none"> <li><code>-XX:+HitachiLocalsInStackTrace</code>: Outputs the local variable information to a stack trace when a thread dump is output.</li> <li><code>-XX:-HitachiLocalsInStackTrace</code>: Does not output the local variable information to a stack trace when a thread dump is output.</li> </ul> <p>The default value is <code>-XX:-HitachiLocalsInStackTrace</code>.</p>
		<code>-XX: [+ -]HitachiLocalsSimpleFormat</code>	<p>Specifies whether the simple format is to be used to output local variable information.</p> <ul style="list-style-type: none"> <li><code>-XX:+HitachiLocalsSimpleFormat</code>: Outputs the local variable information in the simple format.</li> <li><code>-XX:-HitachiLocalsSimpleFormat</code>: Outputs the local variable information in the regular format.</li> </ul> <p>The default value is <code>-XX:-HitachiLocalsSimpleFormat</code>.</p>
		<code>-XX: [+ -]HitachiTrueTypeInLocals</code>	<p>Specifies whether the actual type name of a local variable object is to be output as a character string when the local variable information is output.</p> <ul style="list-style-type: none"> <li><code>-XX:+HitachiTrueTypeInLocals</code>: Outputs the actual object type name to the local variable information.</li> <li><code>-XX:-HitachiTrueTypeInLocals</code>: Does not output the actual object type name to the local variable information.</li> </ul> <p>The default value is <code>-XX:-HitachiTrueTypeInLocals</code>.</p>

#1

Units for *size* are bytes.

You can specify the value in kilobytes by adding `k` or `K`, or in megabytes by adding `m` or `M`.

Example:

`-Xms6291456`: 6,291,456 bytes

`-Xms6144k`: 6,144 kilobytes

`-Xms6m`: 6 megabytes

#2

If you specify these options, a Hitachi JavaVM log file is output.

#3

When the `-XX:+HitachiVerboseGC` option is specified, this option must also be specified.



## Chapter

---

# 9. Adaptor Definition Files

---

This chapter describes the adaptor definition files that must be created in order to use the standard adaptors.

If you use custom adaptors, there is no need to create adaptor definition files.

- 9.1 Format of adaptor definition file explanations
- 9.2 Notes about creating adaptor definition files
- 9.3 List of adaptor definition files
- 9.4 Adaptor command definition file (AgentManagerDefinition.xml)
- 9.5 Adaptor configuration definition file (AdaptorCompositionDefinition.xml)
- 9.6 Common definition in the adaptor configuration definition file
- 9.7 Adaptor group definition in the adaptor configuration definition file
- 9.8 Adaptor definition in the adaptor configuration definition file
- 9.9 CB definition in the adaptor configuration definition file
- 9.10 CB definitions for input and output in the adaptor configuration definition file
- 9.11 CB definitions for data editing in the adaptor configuration definition file
- 9.12 CB definitions for sending and receiving in the adaptor configuration definition file
- 9.13 Coding examples for an adaptor configuration definition file

---

## 9.1 Format of adaptor definition file explanations

---

This section describes the format used to explain the adaptor definition files. The following items are provided to explain each file.

**(1) Format**

Shows the definition's specification format.

**(2) File name**

Indicates the file name.

**(3) File storage location**

Shows the file's storage location.

**(4) Details of definition**

Provides the details of the tags to be defined.

**(5) Example**

Presents an example of each definition.

*Reference note:*

Note that some items are not provided for all files. For some files, additional file-specific information is provided.



---

## 9.2 Notes about creating adaptor definition files

---

The following items should be noted about creating adaptor definition files.

- Specify adaptor definition files in the XML1.0 format. For the specifications for this format, see the XML specifications by W3C (*Extensible Markup Language (XML) 1.0*).
- Any special characters (symbols) you use in adaptor definition files must be sanitized. The table below shows the replacement character strings for special characters that must be sanitized:

Special character that must be sanitized	Replacement character string
<	&lt;
>	&gt;
&	&amp;
"	&quot;
'	&apos; or &#39;

---

### 9.3 List of adaptor definition files

---

The table below lists and describes the adaptor definition files. For details about each file, see the section indicated in the table.

*Table 9-1:* List of adaptor definition files

No.	Definition file	Overview	Section
1	Adaptor command definition file (AgentManagerDefinition.xml)	If you use RMI connection, you must create this file for each working directory. Specify in this file the port number to be used for RMI connection.	9.4
2	Adaptor configuration definition file (AdaptorCompositionDefinition.xml)	You must create this file for each working directory. Specify in this file such information as the configuration of adaptor groups and the callback processing to be performed by the standard adaptors.	Sections beginning with 9.5

## 9.4 Adaptor command definition file (AgentManagerDefinition.xml)

Specify in the adaptor command definition file the port number to be used for RMI connection. You create this file only if you use RMI connection. If you use in-process connection, there is no need to create this file.

### (1) Format

```
<AgentManagerDefinition port="port-number" />
```

### (2) File name

AgentManagerDefinition.xml

### (3) File storage location

You store this file in the following directory:  
*working-directory*\conf\xml\

### (4) Details of definition

AgentManagerDefinition tag (definition of port number for RMI connection)

You specify this definition only once.

```
port="port-number"
```

Specifies as an integer from 1 to 65535 the port number to be used to connect standard adaptors in the RMI connection mode. If this information is omitted, 20420 is assumed.

We recommend that you do not use port numbers 1 through 1023 (which are commonly used port numbers).

### (5) Namespace URI

The namespace URI for the adaptor command definition file is represented in the following format:

```
http://www.hitachi.co.jp/soft/xml/sdp/adaptor/definition/  
agentmanager
```

### (6) Example

```
<?xml version="1.0" encoding="UTF-8"?>  
<agtmgr:AgentManagerDefinition  
  xmlns:agtmgr="http://www.hitachi.co.jp/soft/xml/sdp/adaptor/definition/  
agentmanager"  
  port="20420"/>
```

---

## 9.5 Adaptor configuration definition file (AdaptorCompositionDefinition.xml)

---

Specify in the adaptor configuration definition file the configuration of the standard adaptors, such as the composition of adaptor groups and connectors.

This section provides an overview of the adaptor configuration definition file and discusses the namespace URI for the adaptor configuration definition file. The details about the individual definitions are provided in the subsequent sections. For coding examples of an adaptor configuration definition file, see *9.13 Coding examples for an adaptor configuration definition file*.

### 9.5.1 Overview of the adaptor configuration definition file (AdaptorCompositionDefinition.xml)

#### (1) Format

The format is shown in the subsections for the individual definitions.

#### (2) File name

`AdaptorCompositionDefinition.xml`

#### (3) File storage location

You store this file in the following directory:  
`working-directory\conf\xml\`

#### (4) Details of definition

The definitions in the adaptor configuration definition file have a hierarchical structure. The following figure shows the structure of the adaptor configuration definition file.

Figure 9-1: Structure of the adaptor configuration definition file



The individual definitions that have a hierarchical structure have parent-child relationships. For example, an input adaptor definition is a *child element* of an adaptor group definition and is defined within the tag for the adaptor group definition. Similarly, the adaptor group definition is the *parent element* of the input adaptor definition.

The table below lists and describes the definitions in the adaptor configuration definition file. For details about each definition, see the section indicated in the table.

Table 9-2: List of definitions in the adaptor configuration definition file

No.	Definition		Description	Section
1	Common definition (CommonDefinition)	Adaptor trace definition (AdaptorTraceDefinition tag)	Defines information common to all standard adaptors.	9.6
2	Adaptor group definition	In-process group definition (InprocessGroupDefinition tag)	Defines input adaptors or output adaptors that are to be started in the in-process connection mode.	9.7

No.	Definition		Description	Section		
3		RMI group definition (RMIGroupDefinition tag)	Defines input adaptors or output adaptors that are to be started in the RMI connection mode.			
4	Adaptor definition	Input adaptor definition (InputAdaptorDefinition tag)	Defines the callback that constitutes an input adaptor. You specify this definition for each input adaptor.	9.8		
5		Output adaptor definition (OutputAdaptorDefinition tag)	Defines the callback that constitutes an output adaptor. You specify this definition for each output adaptor.			
6	CB definition	CB definition for input (InputCBDefinition tag)	File input connector definition (FileInputConnectorDefinition tag)	Defines the file input connector processing that is used for reading files.	9.9, 9.10	
7			HTTP packet input connector definition (HttpPacketInputConnectorDefinition tag)	Defines the HTTP packet input connector processing that is used for reading HTTP packets.		
8		CB definition for output (OutputCBDefinition tag)	File output connector definition (FileOutputConnectorDefinition tag)	Defines the file output connector processing used for file output.		
9			Dashboard output connector definition (DashboardOutputConnectorDefinition tag)	Defines the dashboard output connector processing used for dashboard output.		
10		CB definition for editing (DataEditCBDefinition tag)	Format conversion definition (FormatDefinition tag)	Defines the format conversion processing that is performed in order to use file input or output.		9.9, 9.11
11			Mapping definition (MappingDefinition tag)	Defines mapping processing.		
12	Filter definition (FilterDefinition tag)		Defines record filtering processing.			

No.	Definition		Description	Section
13		Record extraction definition (RecordExtractionDefinition tag)	Defines record extraction processing.	
14	CB definition for sending (SendCBDefinition tag)	Input stream definition (streamInfo tag)	Defines the input streams to which input adaptors are to connect.	9.9, 9.12
15	CB definition for receiving (ReceiveCBDefinition tag)	Output stream definition (streamInfo tag)	Defines the output streams to which output adaptors are to connect.	

### (5) Order of CB definitions

The order of the CB definitions is not the same for the input and the output adaptor definitions. The following shows the order of the CB definitions for each.

Order of CB definitions for input adaptor definition:

1. CB definition for input (such as file input connector definition)
2. CB definition for editing (such as format conversion definition)
3. CB definition for sending (such as input stream definition)

Order of CB definitions for output adaptor definition:

1. CB definition for receiving (such as output stream definition)
2. CB definition for editing (such as format conversion definition)
3. CB definition for output (such as file output connector definition)

You specify the CB definitions for editing depending on the functions to be used, as described below.

- The information to be specified in a CB definition for editing depends on the type of data to be input and output by the standard adaptors. For example, a format conversion definition is specified only for file input or output.
- Filter definition and record extraction definition are optional and are specified only as necessary.

For details about the callback processing defined in a CB definition, see *10. Details About Definitions in the Definition Files*.

## 9.5.2 Adaptor configuration definition file namespaces

The *namespace URIs* for an adaptor configuration definition file are represented in the following format:

`http://www.hitachi.co.jp/soft/xml/sdp/adaptor/xxx~`

In this format, *xxx~* corresponds to an individual definition in the adaptor configuration definition file. You must declare each namespace so that it matches a definition you specify. The table below shows the correspondences between the definitions and the namespaces that are to be specified in *xxx~*.

*Table 9-3: Correspondences between definitions and namespaces*

No.	Definition	Namespace
1	Adaptor configuration definition file	definition
2	Common definition	definition/common
3	Adaptor trace definition	
4	In-process group definition	definition/adaptor
5	RMI group definition	
6	Input adaptor definition	
7	Output adaptor definition	
8	CB definition for input	definition/callback
9	CB definition for output	
10	CB definition for editing	
11	CB definition for sending	
12	CB definition for receiving	
13	File input connector definition	definition/callback/FileInputConnectorDefinition
14	HTTP packet input connector definition	definition/callback/HttpPacketInputConnectorDefinition
15	File output connector definition	definition/callback/FileOutputConnectorDefinition



<b>No.</b>	<b>Definition</b>	<b>Namespace</b>
16	Dashboard output connector definition	definition/callback/DashboardOutputConnectorDefinition
17	Format conversion definition	definition/callback/FormatDefinition
18	Mapping definition	definition/callback/MappingDefinition
19	Filter definition	definition/callback/FilterDefinition
20	Record extraction definition	definition/callback/RecordExtractionDefinition

## 9.6 Common definition in the adaptor configuration definition file

This section discusses the common definition in the adaptor configuration definition file (`AdaptorCompositionDefinition.xml`). The table below lists and describes the definitions in the common definition.

*Table 9-4:* List of definitions in the common definition

No.	Definition in the common definition	Parent element	Subsection
1	Common definition ( <code>CommonDefinition</code> tag)	Adaptor configuration definition	9.6.1
2	Adaptor trace definition ( <code>AdaptorTraceDefinition</code> tag)	Common definition	9.6.2

### 9.6.1 Common definition

You specify this definition only once. This definition is mandatory.

#### (1) Format

```
<CommonDefinition>
  adaptor-trace-definition
</CommonDefinition>
```

#### (2) Details of definition

`CommonDefinition` tag (all definition information)

Defines all common definition information.

*adaptor-trace-definition*

For details about the adaptor trace definition, see *9.6.2 Adaptor trace definition*.

### 9.6.2 Adaptor trace definition

You specify the adaptor trace definition (`AdaptorTraceDefinition` tag) as a child element of the common definition (`CommonDefinition` tag) described in *9.6.1 Common definition*.

#### (1) Format

```
<AdaptorTraceDefinition trace="{ON|OFF}"/>
```

**(2) Details of definition**

AdaptorTraceDefinition tag (all definition information)

Defines all adaptor trace definition information.

```
trace=" {ON|OFF} "
```

Specifies whether adaptor traces are to be output. When this information is omitted, OFF is assumed.

The permitted values are as follows:

- ON  
Outputs adaptor traces.
- OFF  
Does not output adaptor traces.

## 9.7 Adaptor group definition in the adaptor configuration definition file

This section discusses the adaptor group definition (`InprocessGroupDefinition` or `RMIGroupDefinition` tag) in the adaptor configuration definition file (`AdaptorCompositionDefinition.xml`). The table below lists and describes the definitions in the adaptor group definition.

Table 9-5: List of definitions in the adaptor group definition

No.	Definition in the adaptor group definition	Parent element	Subsection
1	In-process group definition ( <code>InprocessGroupDefinition</code> tag)	Adaptor configuration definition file	9.7.1
2	RMI group definition ( <code>RMIGroupDefinition</code> tag)		9.7.2

### 9.7.1 In-process group definition

You define in the in-process group definition (`InprocessGroupDefinition` tag) the input adaptors or output adaptors that are to be started in the in-process connection mode.

You specify this definition only once. You omit this definition if you do not use in-process connection to connect standard adaptors and the SDP server. If you omit this definition, you must specify an RMI group definition.

#### (1) Format

```
<InprocessGroupDefinition name="adaptor-group-name"
  dashboardPortNo="RMI-server's-port-number">
  adaptor-definition
</InprocessGroupDefinition>
```

#### (2) Details of definition

`InprocessGroupDefinition` tag (all definition information)

Defines all in-process group definition information.

`name="adaptor-group-name"`

Specifies a name for identifying the adaptor group, as 1 to 32 single-byte alphanumeric characters and the underscore (`_`). This name must begin with a single-byte alphabetic character. This attribute cannot be omitted.

The adaptor group name specified here must match the adaptor group name

specified for the file name in the in-process connection property file.

```
dashboardPortNo="RMI-server's-port-number"
```

Specifies the RMI server's port number that is to be used by the dashboard output connector, as an integer from 1 to 65535. If this attribute is omitted, 20421 is assumed.

*adaptor-definition*

For details about the adaptor definition, see *9.8 Adaptor definition in the adaptor configuration definition file*.

## 9.7.2 RMI group definition

You define in the RMI group definition (`RMIGroupDefinition` tag) the input adaptors or output adaptors that are to be started in the RMI connection mode.

You specify this definition only once. You omit this definition if you do not use RMI to connect standard adaptors and the SDP server. If you omit this definition, you must specify an in-process group definition.

### (1) Format

```
<RMIGroupDefinition name="adaptor-group-name"
  dashboardPortNo="RMI-server's-port-number" >
  adaptor-definition
</RMIGroupDefinition>
```

### (2) Details of definition

`RMIGroupDefinition` tag (all definition information)

Defines all RMI group definition information.

```
name="adaptor-group-name"
```

Specifies a name for identifying the adaptor group, as 1 to 32 single-byte alphanumeric characters and the underscore (`_`). This name must begin with a single-byte alphabetic character. This attribute cannot be omitted.

```
dashboardPortNo="RMI-server's-port-number"
```

Specifies the RMI server's port number that is to be used by the dashboard output connector, as an integer from 1 to 65535. If this attribute is omitted, 20421 is assumed.

*adaptor-definition*

For details about the adaptor definition, see *9.8 Adaptor definition in the adaptor configuration definition file*.

## 9.8 Adaptor definition in the adaptor configuration definition file

This section discusses the adaptor definition in the adaptor configuration definition file (`AdaptorCompositionDefinition.xml`).

You specify an adaptor definition as a child element of the adaptor group definition (`InprocessGroupDefinition` or `RMIGroupDefinition` tag) described in 9.7 *Adaptor group definition in the adaptor configuration definition file*.

The table below lists and describes the definitions in the adaptor definition.

Table 9-6: List of definitions in the adaptor definition

No.	Definition in the adaptor definition	Parent element	Subsection
1	Input adaptor definition ( <code>InputAdaptorDefinition</code> tag)	Adaptor group definition	9.8.1
2	Output adaptor definition ( <code>OutputAdaptorDefinition</code> tag)		9.8.2

### 9.8.1 Input adaptor definition

You specify an input adaptor definition as a child element of the adaptor group definition (`InprocessGroupDefinition` or `RMIGroupDefinition` tag) described in 9.7 *Adaptor group definition in the adaptor configuration definition file*.

You can specify a maximum of 64 input adaptor definitions. This definition is optional.

#### (1) Format

```
<InputAdaptorDefinition name="adaptor-name"
  interval="adaptor-execution-interval"
  charCode="{SJIS|MS932|EUC-JP|UTF-8|UTF-8-BOM|UTF-16BE|UTF-16BE-BOM|UTF-16LE|UTF-16LE-BOM}"
  lineFeed="{CR_LF|LF}">
  CB-definition
</InputAdaptorDefinition>
```

#### (2) Details of definition

`InputAdaptorDefinition` tag (all definition information)

Defines all input adaptor definition information.

`name="adaptor-name"`

Specifies a name for identifying the input adaptor, as 1 to 100 single-byte alphanumeric characters and the underscore (`_`). This name must begin with

a single-byte alphabetic character. This attribute cannot be omitted. This adaptor name must be unique within the adaptor group definition.

```
interval="adaptor-execution-interval"
```

Specifies the interval (in milliseconds) during which the input adaptor is not executed, as an integer from 0 to 60000.

The input adaptor's processing is stopped for the specified interval following the time the final callback defined in the input adaptor is finished until the time the first callback is started. If 0 is specified, the input adaptor's processing is not stopped. If this attribute is omitted, 0 is assumed.

```
charCode="{SJIS|MS932|EUC-JP|UTF-8|UTF-8-BOM|UTF-16BE|UTF-16BE-BOM|UTF-16LE|UTF-16LE-BOM}"
```

Specifies the character encoding to be used at the input source. If this attribute is omitted, MS932 is assumed.

The table below shows the correspondences between the character encodings and the `charCode` attribute value.

Character encoding	charCode attribute value
Shift JIS	SJIS
MS932	MS932
EUC	EUC-JP
UTF-8 (without BOM)	UTF-8
UTF-8 (with BOM)	UTF-8-BOM
UTF-16 (without big endian BOM)	UTF-16BE
UTF-16 (with big endian BOM)	UTF-16BE-BOM
UTF-16 (without little endian BOM)	UTF-16LE
UTF-16 (with little endian BOM)	UTF-16LE-BOM

```
lineFeed="{CR_LF|LF}"
```

Specifies the linefeed code to be used at the input source. If this attribute is omitted, CR\_LF is assumed.

The permitted values are as follows:

- CR\_LF

Treats the combination of a CR (carriage return) and an LF (linefeed) as the linefeed code.

- LF

Treats an LF (linefeed) as the linefeed code.

### *CB-definition*

You can specify the following CB definitions:

- CB definition for input
- CB definition for editing
- CB definition for sending

For details about the CB definitions, see *9.9 CB definition in the adaptor configuration definition file*.

## 9.8.2 Output adaptor definition

You specify an output adaptor definition as a child element of the adaptor group definition (`InprocessGroupDefinition` or `RMIGroupDefinition` tag) described in *9.7 Adaptor group definition in the adaptor configuration definition file*.

You can specify a maximum of 64 output adaptor definitions. This definition is optional.

### (1) Format

```
<OutputAdaptorDefinition name="adaptor-name"
  interval="adaptor-execution-interval"

  charCode="{SJIS|MS932|EUC-JP|UTF-8|UTF-8-BOM|UTF-16BE|UTF-16BE-BOM|UTF-16LE|UTF-16LE-BOM}"
  lineFeed="{CR_LF|LF}">
  CB-definition
</OutputAdaptorDefinition>
```

### (2) Details of definition

`OutputAdaptorDefinition` tag (all definition information)

Defines all output adaptor definition information.

`name="adaptor-name"`

Specifies a name for identifying the output adaptor, as 1 to 100 single-byte alphanumeric characters and the underscore (`_`). This name must begin with a single-byte alphabetic character. This attribute cannot be omitted. This adaptor name must be unique within the adaptor group definition.

`interval="adaptor-execution-interval"`

Specifies the interval (in milliseconds) during which the output adaptor is not executed, as an integer from 0 to 60000.



The output adaptor's processing is stopped for the specified interval following the time the last callback defined in the output adaptor is finished until the time the first callback is started. If 0 is specified, the output adaptor's processing is not stopped. If this attribute is omitted, 0 is assumed.

```
charCode="{SJIS|MS932|EUC-JP|UTF-8|UTF-8-BOM|UTF-16BE|UTF-16BE-BOM|UTF-16LE|UTF-16LE-BOM}"
```

Specifies the character encoding to be used at the output destination. If this attribute is omitted, MS932 is assumed. For the correspondences between the character encodings and the `charCode` attribute, see the description of the `charCode` attribute in *9.8.1 Input adaptor definition*.

```
lineFeed="{CR_LF|LF}"
```

Specifies the linefeed code to be used at the output destination. If this attribute is omitted, CR\_LF is assumed.

The permitted values are as follows:

- CR\_LF  
Treats a combination of a CR (carriage return) and an LF (linefeed) as the linefeed code.
- LF  
Treats an LF (linefeed) as the linefeed code.

#### *CB-definition*

You can specify the following CB definitions:

- CB definition for receiving
- CB definition for editing
- CB definition for output

For details about the CB definitions, see *9.9 CB definition in the adaptor configuration definition file*.

## 9.9 CB definition in the adaptor configuration definition file

This section discusses the CB definition in the adaptor configuration definition file (`AdaptorCompositionDefinition.xml`).

You define in the CB definition the callbacks that perform data input and output, data editing, and tuple send and receive processing with the standard adaptors.

You specify a CB definition as a child element of one of the following definitions:

- Input adaptor definition (`InputAdaptorDefinition` tag) discussed in 9.8.1 *Input adaptor definition*
- Output adaptor definition (`OutputAdaptorDefinition` tag) discussed in 9.8.2 *Output adaptor definition*

The table below lists and describes the definitions in the CB definition.

*Table 9-7: List of definitions in the CB definition*

No.	Definition in the CB definition	Parent element	Subsection
1	CB definition for input ( <code>InputCBDefinition</code> tag)	Input adaptor definition	9.9.1
2	CB definition for output ( <code>OutputCBDefinition</code> tag)	Output adaptor definition	9.9.2
3	CB definition for editing ( <code>DataEditCBDefinition</code> tag)	Input adaptor definition or output adaptor definition	9.9.3
4	CB definition for sending ( <code>SendCBDefinition</code> tag)	Input adaptor definition	9.9.4
5	CB definition for receiving ( <code>ReceiveCBDefinition</code> tag)	Output adaptor definition	9.9.5

Note that the order of the CB definitions is not the same for the input and the output adaptor definitions. For the order of the CB definitions, see 9.5.1(5) *Order of CB definitions*.

### 9.9.1 CB definition for input

You specify a CB definition for input (`InputCBDefinition` tag) as a child element of an input adaptor definition (`InputAdaptorDefinition` tag) as discussed in 9.8.1 *Input adaptor definition*.

**(1) Format**

```
<InputCBDefinition class="class-name"
  name="callback-name"
  interval="callback-execution-interval">
  input-connector-definition
</InputCBDefinition>
```

**(2) Details of definition**

InputCBDefinition tag (all definition information)

Defines all CB definition for input information.

`class="class-name"`

Specifies the name of the class in which the function for the CB definition for input has been implemented. This value is fixed.

The class name depends on the type of input connector definition specified in *input-connector-definition*. The following table shows the class name to be specified.

Type of input connector definition	Value of class attribute
File input connector definition	jp.co.Hitachi.soft.sdp.adaptor.callback.io.FileInputCBImpl
HTTP packet input connector definition	jp.co.Hitachi.soft.sdp.adaptor.callback.io.packetinput.HttpPacketInputCBImpl

`name="callback-name"`

Specifies a callback name for identifying the function, as 1 to 100 single-byte alphanumeric characters and the underscore (\_). This name must begin with a single-byte alphabetic character. This attribute cannot be omitted. You can specify for this callback name the same value as in another CB definition, but we recommend that you specify a unique callback name because this name is output as identification information for the input adaptor's processing in the adaptor trace information and message logs.

`interval="callback-execution-interval"`

Specifies a callback execution interval (in milliseconds), as an integer from 0 to 60000. Once the callback has executed, the input adaptor's processing is terminated following the specified interval. If you specify 0, the input adaptor's processing is not terminated. If you omit this attribute, 0 is set.

*input-connector-definition*

You can specify the following CB definitions:

- File input connector definition
- HTTP packet input connector definition

For details about these CB definitions, see *9.10 CB definitions for input and output in the adaptor configuration definition file*.

## 9.9.2 CB definition for output

You specify a CB definition for output (`OutputCBDefinition` tag) as a child element of an output adaptor definition (`OutputAdaptorDefinition` tag) discussed in *9.8.2 Output adaptor definition*.

### (1) Format

```
<OutputCBDefinition class="class-name"
  name="callback-name"
  interval="callback-execution-interval">
  output-connector-definition
</OutputCBDefinition>
```

### (2) Details of definition

`OutputCBDefinition` tag (all definition information)

Defines all CB definition for output information.

`class="class-name"`

Specifies the name of the class in which the function for the CB definition for output has been implemented. This value is fixed.

The class name depends on the type of output connector definition specified in *output-connector-definition*. The following table shows the class name to be specified.

Type of output connector definition	Value of class attribute
File output connector definition	<code>jp.co.Hitachi.soft.sdp.adaptor.callback.io.FileOutputCBImpl</code>
Dashboard output connector definition	<code>jp.co.Hitachi.soft.sdp.adaptor.callback.io.dashboard.DashboardOutputCBImpl</code>

`name="callback-name"`

Specifies a callback name for identifying the function, as 1 to 100 single-byte alphanumeric characters and the underscore (`_`). This name must begin with a single-byte alphabetic character. This attribute cannot be omitted. You can specify for this callback name the same value as in another CB definition, but we recommend that you specify a unique callback name because this name

is output as identification information for the output adaptor's processing in the adaptor trace information and message logs.

```
interval="callback-execution-interval"
```

Specifies a callback execution interval (in milliseconds), as an integer from 0 to 60000. Once the callback has executed, the output adaptor's processing is terminated following the specified interval. If you specify 0, the output adaptor's processing is not terminated. If you omit this attribute, 0 is set.

#### *output-connector-definition*

You can specify the following CB definitions:

- File output connector definition
- Dashboard output connector definition

For details about these CB definitions, see *9.10 CB definitions for input and output in the adaptor configuration definition file*.

### 9.9.3 CB definition for editing

You specify a CB definition for editing (`DataEditCBDefinition` tag) as a child element of the following definitions:

- Input adaptor definition (`InputAdaptorDefinition` tag) discussed in *9.8.1 Input adaptor definition*
- Output adaptor definition (`OutputAdaptorDefinition` tag) discussed in *9.8.2 Output adaptor definition*

#### (1) Format

```
<DataEditCBDefinition class="class-name"
  name="callback-name"
  interval="callback-execution-interval">
  CB-definition-for-data-editing
</DataEditCBDefinition>
```

#### (2) Details of definition

`DataEditCBDefinition` tag (all definition information)

Defines all CB definition for editing information.

```
class="class-name"
```

Specifies the name of the class in which the function for the CB definition for editing has been implemented. This value is fixed.

The class name depends on the type of CB definition for data editing specified in *CB-definition-for-data-editing*. The class name to be specified

might also depend on whether it is specified in an input adaptor definition or an output adaptor definition. The following table shows the class name to be specified.

Type of CB definition for data editing	Value of class attribute
Format conversion definition	<p>The value depends on whether this definition is specified in an input adaptor definition or an output adaptor definition:</p> <ul style="list-style-type: none"> <li>Value specified in input adaptor definition <code>jp.co.Hitachi.soft.sdp.adaptor.callback.dataedit.formattranslate.InputFormatTranslatorCBImp</code></li> <li>Value specified in output adaptor definition <code>jp.co.Hitachi.soft.sdp.adaptor.callback.dataedit.formattranslate.OutputFormatTranslatorCBImp</code></li> </ul>
Mapping definition	<p>The value depends on whether this definition is specified in an input adaptor definition or an output adaptor definition:</p> <ul style="list-style-type: none"> <li>Value specified in input adaptor definition <code>jp.co.Hitachi.soft.sdp.adaptor.callback.dataedit.mapping.InputMappingCBImp</code></li> <li>Value specified in output adaptor definition <code>jp.co.Hitachi.soft.sdp.adaptor.callback.dataedit.mapping.OutputMappingCBImp</code></li> </ul>
Filter definition	<p>The following value is always specified, whether this definition is specified in an input adaptor definition or an output adaptor definition: <code>jp.co.Hitachi.soft.sdp.adaptor.callback.dataedit.filter.FilterCBImp</code></p>
Record extraction definition	<p>A record extraction definition cannot be specified in an output adaptor definition. In an input adaptor definition, specify the following value: <code>jp.co.Hitachi.soft.sdp.adaptor.callback.dataedit.recordextract.RecordExtractionCBImp</code></p>

`name="callback-name"`

Specifies a callback name for identifying the function, as 1 to 100 single-byte alphanumeric characters and the underscore (\_). This name must begin with a single-byte alphabetic character. This attribute cannot be omitted. You can specify for this callback name the same value as in another CB definition, but we recommend that you specify a unique callback name because this name is output as identification information for the input or output adaptor's processing in the adaptor trace information and message logs.

`interval="callback-execution-interval"`

Specifies a callback execution interval (in milliseconds), as an integer from

0 to 60000. Once a callback has executed, the input or output adaptor's processing is terminated following the specified interval. If you specify 0, the input or output adaptor's processing is not terminated. If you omit this attribute, 0 is set.

#### *CB-definition-for-data-editing*

You can specify the following CB definitions:

- Format conversion definition
- Mapping definition
- Filter definition
- Record extraction definition (cannot be specified in an output adaptor definition)

For details about these CB definitions, see *9.11 CB definitions for data editing in the adaptor configuration definition file*.

### 9.9.4 CB definition for sending

You specify a CB definition for sending (`SendCBDefinition` tag) as a child element of an input adaptor definition (`InputAdaptorDefinition` tag) discussed in *9.8.1 Input adaptor definition*.

#### (1) Format

```
<SendCBDefinition class="class-name"
  name="callback-name"
  interval="callback-execution-interval">
  input-stream-definition
</SendCBDefinition>
```

#### (2) Details of definition

`SendCBDefinition` tag (all definition information)

Defines all CB definition for sending information.

`class="class-name"`

Specifies the name of the class in which the function for the CB definition for sending has been implemented. This value is fixed. You must specify the following class name:

```
jp.co.Hitachi.soft.sdp.adaptor.callback.sendreceive.SendCBImpl
```

`name="callback-name"`

Specifies a callback name for identifying the function, as 1 to 100 single-byte

alphanumeric characters and the underscore (\_). This name must begin with a single-byte alphabetic character. This attribute cannot be omitted. You can specify for this callback name the same value as in another CB definition, but we recommend that you specify a unique callback name because this name is output as identification information for the input adaptor's processing in the adaptor trace information and message logs.

```
interval="callback-execution-interval"
```

Specifies a callback execution interval (in milliseconds), as an integer from 0 to 60000. Once a callback has executed, the input adaptor's processing is terminated following the specified interval. If you specify 0, the input adaptor's processing is not terminated. If you omit this attribute, 0 is set.

*input-stream-definition*

For details about the input stream definition, see *9.12.1 Input stream definition*.

## 9.9.5 CB definition for receiving

You specify a CB definition for receiving (ReceiveCBDefinition tag) as a child element of an output adaptor definition (OutputAdaptorDefinition tag) discussed in *9.8.2 Output adaptor definition*.

### (1) Format

```
<ReceiveCBDefinition class="class-name"
  name="callback-name"
  interval="callback-execution-interval">
  output-stream-definition
</ReceiveCBDefinition>
```

### (2) Details of definition

ReceiveCBDefinition tag (all definition information)

Defines all CB definition for receiving information.

```
class="class-name"
```

Specifies the name of the class in which the function for the CB definition for receiving has been implemented. This value is fixed. You must specify the following class name:

```
jp.co.Hitachi.soft.sdp.adaptor.callback.sendreceive.ReceiveCBImpl
```

```
name="callback-name"
```

Specifies a callback name for identifying the function, as 1 to 100 single-byte alphanumeric characters and the underscore (\_). This name must begin with



a single-byte alphabetic character. This attribute cannot be omitted. You can specify for this callback name the same value as in another CB definition, but we recommend that you specify a unique callback name because this name is output as identification information for the output adaptor's processing in the adaptor trace information and message logs.

`interval="callback-execution-interval"`

Specifies a callback execution interval (in milliseconds), as an integer from 0 to 60000. Once a callback has executed, the output adaptor's processing is terminated following the specified interval. If you specify 0, the output adaptor's processing is not terminated. If you omit this attribute, 0 is set.

*output-stream-definition*

For details about the output stream definition, see *9.12 CB definitions for sending and receiving in the adaptor configuration definition file*.

## 9.10 CB definitions for input and output in the adaptor configuration definition file

This section discusses the CB definitions for input and output in the adaptor configuration definition file (`AdaptorCompositionDefinition.xml`).

You specify an input connector definition as a child element of a CB definition for input (`InputCBDefinition` tag) discussed in *9.9.1 CB definition for input*, and you specify an output connector definition as a child element of a CB definition for output (`OutputCBDefinition` tag) discussed in *9.9.2 CB definition for output*.

The table below lists and describes the definitions in the CB definitions for input and output.

*Table 9-8:* List of definitions in the CB definitions for input and output

No.	Definition in CB definition for input or CB definition for output		Parent element	Subsection
1	Input connector definition	File input connector definition ( <code>FileInputConnectorDefinition</code> tag)	CB definition for input	9.10.1
2		HTTP packet input connector definition ( <code>HttpPacketInputConnectorDefinition</code> tag)		9.10.2
3	Output connector definition	File output connector definition ( <code>FileOutputConnectorDefinition</code> tag)	CB definition for output	9.10.3
4		Dashboard output connector definition ( <code>DashboardOutputConnectorDefinition</code> tag)		9.10.4

### 9.10.1 File input connector definition

You specify a file input connector definition (`FileInputConnectorDefinition` tag) as a child element of a CB definition for input (`InputCBDefinition` tag).

For details about file input processing, see *10.2 File input*.

**(1) Format**

```

<FileInputConnectorDefinition>
  <input readType="{ BATCH | REAL_TIME } "
    compositionType="{ WRAP_AROUND | ANTI_WRAP_AROUND } "
    interval="monitoring-interval"
    retryCount="monitoring-retries-count"
    readOrder="{ DEFINED | MODIFIED } "
    readUnit="reading-unit" />
  <file path="input-path-name"
    name="input-files"
    messageLog="{ ON | OFF } " />
</FileInputConnectorDefinition>

```

**(2) Details of definition**

`FileInputConnectorDefinition` tag (all definition information)

Defines all file input connector definition information. You specify this definition only once.

`input` tag (input definition)

Defines information about the input file read processing and monitoring processing. You specify this definition only once.

`readType="{ BATCH | REAL_TIME } "`

Specifies how to read input files. If this attribute is omitted, `REAL_TIME` is assumed.

The permitted values are as follows:

- `BATCH`  
Reads input files in the batch processing mode.
- `REAL_TIME`  
Reads input files in the real-time processing mode.

`compositionType="{ WRAP_AROUND | ANTI_WRAP_AROUND } "`

Specifies the structure of input files. If this attribute is omitted, `ANTI_WRAP_AROUND` is assumed.

The permitted values are as follows:

- `WRAP_AROUND`  
Reads input files in the wraparound mode.

Note that if you specify `BATCH` in the `readType` attribute, `WRAP_AROUND` cannot be specified. Similarly, if you specify `REAL_TIME` in the `readType` attribute and `DEFINED` in the

`readOrder` attribute, `WRAP_AROUND` cannot be specified.

- `ANTI_WRAP_AROUND`

Reads input files in the non-wraparound mode.

Note that if you specify `MODIFIED` in the `readOrder` attribute, `ANTI_WRAP_AROUND` cannot be specified.

`interval="monitoring-interval"`

If you specified `REAL_TIME` in the `readType` attribute, this attribute specifies as an integer from 0 to 1000000 the interval (in milliseconds) at which monitoring of the input file storage directory is to be performed in order to determine whether or not a new file to be input has been created. If you specify 0, monitoring is performed continuously. If this attribute is omitted, 1000 is assumed.

`retryCount="monitoring-retries-count"`

Specifies as an integer from 0 to 1000 a retries count for monitoring the input file storage directory. If this attribute is omitted, 100 is assumed.

When the number of retries reaches the value of this attribute, the `KFSP46203-E` warning message is output indicating that there is no file in the input file storage directory waiting to be input and then monitoring is resumed.

A monitoring retry for each file to be input is performed at that point. When a target file is read, the retries count is reset to 0. Therefore, the retries count threshold is checked for each input file and a warning message is output for each input file.

`readOrder="{DEFINED|MODIFIED}"`

Specifies the order in which input files are to be read. If this attribute is omitted, `DEFINED` is assumed.

- `DEFINED`

Reads input files in the order their names are specified in the `name` attribute of the `file` tag.

- `MODIFIED`

Reads input files in the order of the input file update times.

`readUnit="reading-unit"`

Specifies as an integer from 1 to 10000 the number of records that are be read by the file input connector at one time. If this attribute is omitted, 1 is assumed.

`file` tag (input file definition)

Defines information about the input files. You specify this definition only once.

```
path="input-path-name"
```

Specifies the path name of the input file storage directory, as 1 to 160 single-byte characters. This attribute cannot be omitted.

The following characters are permitted for this attribute:

Single-byte alphanumeric characters, !, #, \$, %, &, ', (, ), =, ~, ` , {, }, +, -, ^, ., \_, @, \, /, :, single-byte space

```
name="input-files"
```

Specifies the files to be input, either as file names or as sequence numbers. This attribute cannot be omitted.

- Specifying file names

Specify each file name as a path relative to the input file storage directory specified in the `path` attribute. You specify multiple file names by delimiting the names with the comma (,). Note that no subdirectory can be specified. Express each input file name as 1 to 60 single-byte characters.

A specified file name cannot be a reserved device name (such as `NUL`). If a reserved device name is specified, operation is not guaranteed.

- Specifying sequence numbers

Specify the sequence numbers of files that are stored in the input file storage directory specified in the `path` attribute. For details about the specification method, see (4) *Specifying sequence numbers for input files*.

Express an input file name as 1 to 1,024 single-byte characters, excluding \, /, and .:

The following characters are permitted for this attribute:

Single-byte alphanumeric characters, !, #, \$, %, &, ', (, ), =, ~, ` , {, }, +, -, ^, ., \_, @, single-byte space

```
messageLog=" {ON|OFF} "
```

Specifies whether or not the start time and input file name are to be output to the message log when input read processing begins for a file. If this attribute is omitted, `OFF` is assumed.

The permitted values are as follows:

- ON

Outputs the start time and input file name to the message log.

- OFF

Does not output the start time and input file name to the message log.

### (3) Example

```
<?xml version="1.0" encoding="UTF-8"?>
<root:AdaptorCompositionDefinition
  xmlns:ficon="http://www.hitachi.co.jp/soft/xml/sdp/adaptor/definition/callback/
  FileInputConnectorDefinition">
  <!-- Omitted -->

  <!-- CB definition for input -->
  <cb:InputCBDefinition
    class="jp.co.Hitachi.soft.sdp.adaptor.callback.io.FileInputCBImpl" name="inputer1">
    <!-- File input connector definition -->
    <ficon:FileInputConnectorDefinition>
      <!-- Input definition -->
      <ficon:input readType="REAL_TIME" interval="600000"
        retryCount="100" readUnit="1"/>
      <!-- Input file definition -->
      <ficon:file path="C:\tmp\" name="a[1-100].txt"/>
    </ficon:FileInputConnectorDefinition>
  </cb:InputCBDefinition>
```

### (4) Specifying sequence numbers for input files

This subsection describes the format used to specify sequence numbers in place of input file names in the name attribute in the `file` tag.

- Sequence number specification format

```
prefix-string sequence-number-1 intermediate-string sequence-number-2 suffix string
```

The following explains each value.

*prefix-string, intermediate-string, suffix string*

Express each of these values as a string of single-byte characters.

*sequence-number-1, sequence-number-2*

Express these values in the format *minimum-value-maximum-value* or *minimum-value-* enclosed in [ and ]. Express *minimum-value* and *maximum-value* as 1 to 8 single-byte numeric characters.

- Rules for a sequence number specification
  - Only one file can be specified in a sequence number specification.
  - Neither [ nor ] can be specified as the prefix string, intermediate string, or suffix string.

- The minimum value cannot be omitted from a sequence number specification. If the maximum value is omitted, 99999999 is assumed as the maximum value.
- A specified sequence number must satisfy the conditions *minimum-value* ≤ *maximum-value* and the number of digits in *maximum-value* ≥ number of digits in *minimum-value*.
- The files specified by the sequence numbers are input in the following order:
  1. File with the smallest *sequence-number-1*
  2. File with the smallest *sequence-number-2*

For example, the file a1b2.txt is input before the file a1b2.txt, a2b1.txt, and the file a2b1.txt is input before the file a2b2.txt, a2b1.txt.

- There must be at least one non-numeric character in the intermediate string.
- The file input connector reads file names sequentially from the minimum value to the maximum value of the sequence numbers even when the input files' sequence numbers are not consecutive. For example, when the sequence numbers are 1, 2, 4, where 3 is missing, the file input connector reads 1, 2, and 4 in this order.

■ Examples of sequence number specification

The table below shows examples of sequence number specification.

Example	Processing result	Order of files to be input during normal operation
a[1-100].txt	Y	a1.txt, a2.txt, ... (omitted) ..., a9.txt, a10.txt, a11.txt, a12.txt, ... (omitted) ..., a99.txt, a100.txt
a[01-100].txt	Y	a01.txt, a02.txt, ... (omitted) ..., a09.txt, a10.txt, a11.txt, a12.txt, ... (omitted) ..., a99.txt, a100.txt
a[001-100].txt	Y	a001.txt, a002.txt, ... (omitted) ..., a009.txt, a010.txt, a011.txt, a012.txt, ... (omitted) ..., a099.txt, a100.txt
a[01-10]_[1-100].txt	Y	a01_1.txt, a01_2.txt, ... (omitted) ..., a01_9.txt, a01_10.txt, a01_a11.txt, ... (omitted) ..., a01_99.txt, a01_100.txt, a02_1.txt, a02_2.txt, ... (omitted) ..., a02_9.txt, a01_10.txt, a02_a11.txt, ... (omitted) ..., a02_99.txt, a02_100.txt, ... (omitted) ..., a10_1.txt, a10_2.txt, ... (omitted) ..., a10_9.txt, a10_10.txt, a10_a11.txt, ... (omitted) ..., a10_99.txt, a10_100.txt
a[5-3].txt	N <sup>#1</sup>	--
a[001-9].txt	N <sup>#2</sup>	--

Example	Processing result	Order of files to be input during normal operation
a[1-005].txt	Y	a1.txt, a2.txt, a3.txt, a4.txt, a5.txt
a.txt, b[1-3].txt	N <sup>#3</sup>	--
a[1-3].txt, b[4-6].txt	N <sup>#3</sup>	--

Legend:

Y: Processed normally

N: Results in an error

--: Not applicable

#1

This results in an error because the minimum value is greater than the maximum value.

#2

This results in an error because the number of digits in the minimum value is greater than the number of digits in the maximum value.

#3

This results in an error because only one file can be specified in a sequence number specification.

### 9.10.2 HTTP packet input connector definition

You specify an HTTP packet input connector definition (`HttpPacketInputConnectorDefinition` tag) as a child element of a CB definition for input (`InputCBDefinition` tag).

For details about HTTP packet input processing, see *10.3 HTTP packet input*.



**(1) Format**

```

<HttpPacketInputConnectorDefinition>
  <input buffersize="I/O-buffer-size"
    assemblingtime="packet-segment-assembly-time">
    <packetdata
      globalheader="global-header-area-size"
      packetheader="packet-header-area-size"
      packetoffset="offset-for-packet-data-size-area"
      packetlength="packet-data-size-area-size"
      timeoffset="offset-for-timestamp-area" />
    <command path="command-path-name"
      parameter="command-parameter" />
  </input>
  <output unit="maximum-output-unit">
    <record name="record-name" type="{REQUEST|RESPONSE}">
      <field name="field-name" />
    </record>
  </output>
</HttpPacketInputConnectorDefinition>

```

**(2) Details of definition**

HttpPacketInputConnectorDefinition tag (all definition information)

Defines all HTTP packet input connector definition information. You specify this definition only once.

input tag (input definition)

Defines the HTTP packet input or output information. You specify this definition only once.

buffersize="I/O-buffer-size"

Specifies as an integer from 1 to 12288 the maximum number of HTTP packets that can be stored in the input buffer or the maximum number of common records that can be stored in the output buffer. If this attribute is omitted, 4096 is assumed.

assemblingtime="packet-segment-assembly-time"

Specifies as an integer from 1 to 5000 the packet segment assembly time (in milliseconds). If this attribute is omitted, 2000 is assumed.

Packet segments are packet data that has been segmented in TCP hierarchy according to the maximum segment length (MSS: maximum segment size). If the data collected by the HTTP packet input connector is packet segments, packets are assembled on the basis of the TCP protocol header information. The packet segment assembly time means the time required to link the received packet segments to obtain a complete packet since a new packet segment arrived. Each time packet segments are assembled into a packet, the

counter is reset. Note that the packet data segmented in the IP hierarchy is not assembled.

If packet segments cannot be assembled within the amount of time specified here, those packet segments undergoing assembly processing are discarded. The most recent time information in the packet segment used for assembly is used as the timestamp.

#### packetdata tag (HTTP packet definition)

Defines the format of the HTTP packets to be acquired. You specify this definition only once.

`globalheader="global-header-area-size"`

Specifies the size (in bytes) of the global header area, as an integer from 1 to 128. If this attribute is omitted, 24 is assumed.

`packetheader="packet-header-area-size"`

Specifies the size (in bytes) of the packet header area, as an integer from 9 to 128. If this attribute is omitted, 16 is assumed.

`packetoffset="offset-for-packet-data-size-area"`

Specifies the offset (in bytes) from the beginning of the packet header to the packet data size area, as an integer from 0 to 127. If this attribute is omitted, 8 is assumed.

`packetlength="packet-data-size-area-size"`

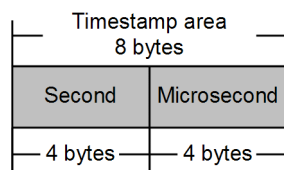
Specifies the size (in bytes) of the packet data size area, as an integer from 1 to 4. If this attribute is omitted, 4 is assumed.

`timeoffset="offset-for-timestamp-area"`

Specifies the offset (in bytes) from the beginning of the packet header to the timestamp area, as an integer from 0 to 120. If this attribute is omitted, 0 is assumed.

The timestamp area, which is located in the packet header area, stores the timestamp obtained when the packet analyzer captured the HTTP packet. The following figure shows the format of the timestamp area.

Figure 9-2: Format of timestamp area



`command` tag (command definition)

Defines the start command for the packet analyzer that is to be used. You specify this definition only once.

Stream Data Platform - AF supports WinDump as the packet analyzer. For details about the information to be defined here when WinDump is used, see (4) *Information to be specified in the command tag when WinDump is used.*

`path="command-path-name"`

Specifies the absolute path of the packet analyzer start command, as 1 to 100 single-byte characters (ASCII codes 32 to 126).

`parameter="command-parameter"`

Specifies a parameter to be passed to the packet analyzer start command, as 1 to 100 single-byte characters (ASCII codes 32 to 126).

`output` tag (output definition)

Defines the output information for common records when the HTTP packet input connector converts HTTP packets to common records. You specify this definition only once.

`unit="maximum-output-unit"`

Specifies the maximum number of output units (in records) for common records, as an integer from 1 to 1000. If this attribute is omitted, 100 is assumed.

`record` tag (record definition)

Defines record information for common records. You may specify 1 or 2 record definitions.

`name="record-name"`

Specifies a name for a common record, as 1 to 100 single-byte alphanumeric characters and the underscore (`_`). This record name must begin with a single-byte alphabetic character. This attribute cannot be omitted. This record name must be unique within the `record` tag.

`type="{REQUEST|RESPONSE}"`

Specifies the type of the common record. This attribute cannot be omitted.

The permitted values are as follows:

- "REQUEST"

Indicates a request record. This is a type of common record used to store data generated during request transmission from client to host in HTTP protocol communication.

- "RESPONSE"

Indicates a response record. This is a type of common record used to store data generated during response transmission from host to client in HTTP protocol communication.

`field` tag (field definition)

Defines information about the fields that constitute the common record. You may specify 1 to 16 field definitions.

`name="field-name"`

Specifies a field name.

Specify for a field name the identifier of data that is to be extracted from HTTP packets. The data corresponding to this identifier is stored as the field's value.

Each field name must be unique within the record definition.

The identifiers that can be specified as field names depend on the type of common record specified in the `type` attribute in the `record` tag. For details about the identifiers that can be specified as field names, see (5) *Identifiers that can be specified as field names*.

Extracted data is converted to the Java data type and then stored as field values. For details about the Java data types for the data that is stored as field values, see (6) *Java data types that are stored as field values*.

**(3) Example**

```

<?xml version="1.0" encoding="UTF-8"?>
<root:AdaptorCompositionDefinition
  xmlns:hpicon="http://www.hitachi.co.jp/soft/xml/sdp/adaptor/definition/callback/
  HttpPacketInputConnectorDefinition">
  <!-- Omitted -->

  <!-- CB definition for input -->
  <cb:InputCBDefinition

  class="jp.co.Hitachi.soft.sdp.adaptor.callback.io.packetinput.HttpPacketInputCBImp1"
  name="inputer2">
    <!-- HTTP packet input connector definition -->
    <hpicon:HttpPacketInputConnectorDefinition>
      <!-- Input definition -->
      <hpicon:input buffersize="096" assemblingtime="2000">
        <!-- HTTP packet definition -->
        <hpicon:packetdata globalheader="24" packetheader="16" packetoffset="8"
          packetlength="4" timeoffset="0"/>
        <!-- Command definition -->
        <hpicon:command path="C:\Program Files\WinDump\WinDump.exe"
          parameter="-i 1 -s 2048 -w - -n &quot;tcp port 80 and host 133.145.224.19&quot;"/>
      >
    </hpicon:input>
    <!-- Output definition -->
    <hpicon:output unit="100">
      <!-- Record definition -->
      <hpicon:record name="RECORD1" type="REQUEST">
        <!-- Field definition -->
        <hpicon:field name="SEND_IP"/>
        <hpicon:field name="RECEIVE_IP"/>
        <hpicon:field name="SEND_PORT"/>
        <hpicon:field name="RECEIVE_PORT"/>
        <hpicon:field name="MESSAGE_TYPE"/>
        <hpicon:field name="TARGET_URI"/>
      </hpicon:record>
    </hpicon:output>
  </hpicon:HttpPacketInputConnectorDefinition>
</cb:InputCBDefinition>

```

**(4) Information to be specified in the command tag when WinDump is used**

This subsection discusses the information to be specified in the `command` tag when WinDump is used as the packet analyzer.

The example presented here uses WinDump version 3.9.5. For details about the WinDump start command, see the WinDump documentation.

The HTTP packet input connector supports the WinDump start command (WinDump.exe) specified in the following format.

```

WinDump.exe Δ -i Δ network-device-number Δ -s Δ internal-buffer-size Δ -w Δ - Δ -n Δ "tcp Δ port Δ port-n
umber Δ and Δ host Δ IP-address"

```

**Legend:**

Δ: Single-byte space. It might be permissible to omit parameter delimiters (single-byte spaces) in the WinDump start command, but the single-byte spaces cannot be omitted from the `parameter` attribute in the `command` tag.

The following explains each value in the format.

`WinDump.exe`

This is the WinDump start command. Specify the absolute path of `WinDump.exe` in the `path` attribute in the `command` tag.

`-i Δ network-device-number`

This option specifies the number of the network device connected to the target computer that is to be analyzed. Specify this option and value in the `parameter` attribute in the `command` tag.

`-s Δ internal-buffer-size`

This option specifies the size (in bytes) of the internal buffer for storing the captured packet data. HTTP requires a larger packet size than TCP; normally, 2,048 bytes is sufficient. Specify this option and value in the `parameter` attribute in the `command` tag.

`-w Δ -`

This option specifies a file or the standard output as the output destination of the captured packets. If you use an HTTP packet input connector, specify `-w Δ -` because packets are output to the standard output. Specify this option and value in the `parameter` attribute in the `command` tag.

`-n Δ "tcp Δport Δport-number Δand Δhost ΔIP-address`

This option specifies the capture filter in the filter format of the packet capture library (`libpcap`). If you use an HTTP packet input connector, specify the port number used with the HTTP protocol and the IP address of the computer to be analyzed. Specify this option and value in the `parameter` attribute in the `command` tag.

Note that a double quotation mark (") is treated as a special character, so code it as `&quot;` ; .

**(5) Identifiers that can be specified as field names**

The table below lists and describes the identifiers that can be specified as field names in the `name` attribute in the `field` tag.

Table 9-9: Identifiers that can be specified as field name

No.	Identifier	Data	Description	Protocol	Whether or not specifiable according to record type	
					Request	Response
1	TIME	Time <sup>#1</sup>	Time packet data arrived	--	Y	Y
2	PACKET_LENGTH	Packet size <sup>#2</sup>	Size of packet data (bytes)	--	Y	Y
3	SEND_MAC	Sending MAC address	MAC address at the packet sending end	Ethernet	Y	Y
4	SEND_IP	Sending IP address	IP address at the packet sending end	IP	Y	Y
5	RECEIVE_IP	Receiving IP address	IP address at the packet receiving end	IP	Y	Y
6	SEND_PORT	Sending port number	Port number at the packet sending end	TCP	Y	Y
7	RECEIVE_PORT	Receiving port number	Port number at the packet receiving end	TCP	Y	Y
8	MESSAGE_TYPE	Message type	Request or Response	HTTP	Y	Y
9	METHOD_NAME	Method information	Method information, such as GET and POST	HTTP	Y	N
10	TARGET_URI	URI information <sup>#3</sup>	Target URI information	HTTP	Y	N
11	REFERER	Referer <sup>#3</sup>	Link source URI information	HTTP	Y	N
12	COOKIE	Cookie <sup>#3 #4</sup>	Cookie information <sup>#5</sup>	HTTP	Y	Y
13	STATUS_CODE	Status code	Result of request processing	HTTP	N	Y
14	CONNECTION	Connection	Connection persistency information	HTTP	Y	Y
15	CONTENT_LENGTH	Content-Length	Contents size (bytes)	HTTP	Y	Y
16	CONTENT_TYPE	Content-Type	Contents type	HTTP	Y	Y

No.	Identifier	Data	Description	Protocol	Whether or not specifiable according to record type	
					Request	Response
17	MESSAGE_BODY	Message body <sup>#3</sup>	Real data	HTTP	Y <sup>#6</sup>	N

Legend:

Y: Identifier can be specified

N: identifier cannot be specified

--: Not applicable

#1

The time is obtained from the timestamp in the packet header.

#2

The packet size is the sum of the sizes of the HTTP message start line, header size, and value referenced by the HTTP header "Content-Length". If there is no "Content-Length", the value referenced by "Content-Length" is 0.

#3

A character string in percent-encoding is decoded in UTF-8.

#4

The method for acquiring cookie data is not the same for request records and response records.

For a request record, the data referenced by the HTTP header "Cookie" is acquired.

For a response record, the data referenced by the HTTP header "Set-Cookie" is acquired. The data referenced by the HTTP headers "Cookie2" and "Set-Cookie2" is not acquired.

#5

The cookie information might contain multiple cookies. If you want to treat each cookie as a field, specify `regexsubstring` in the `function` attribute in the `map` tag (mapping definition) and acquire a character string using a regular expression.

#6

You can obtain the message body only when the method information is `POST`, the media type in `Content-Type` is `text` (regardless of the value of subtype), and



Content - Length exists.

### (6) Java data types that are stored as field values

Each protocol data item extracted according to the identifier specified in the `name` attribute in the `field` tag is converted to a Java data type, as shown in the table below. During conversion to the Java data type, if the value of the protocol data is greater than the permitted maximum value, only up to the maximum value is stored as the field value. If the data corresponding to the specified identifier is not found in the protocol data, the null character is stored as the field value for the `String` type and `-1` is stored for the `Integer` type.

Table 9-10: Java data types that are stored as field values

No.	Data	Protocol	Java data type	Value range
1	Time <sup>#</sup>	--	Timestamp	1970/01/01 00:00:00.000000 to 2261/12/31 23:59:59.999999
2	Packet size	--	Integer	0 to 2147483647
3	Sending MAC address	Ethernet	String	17 characters (00:00:00:00:00:00 to FF:FF:FF:FF:FF:FF)
4	Sending port number	TCP	Integer	0 to 65535
5	Receiving port number	TCP	Integer	
6	Sending IP address	IP	String	<ul style="list-style-type: none"> <li>• In IPv4: 7 to 15 characters (0.0.0.0 to 255.255.255.255)</li> <li>• In IPv6: 40 characters (0000:0000:0000:0000:0000:0000:0000:0000 to FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF:FFFF)</li> </ul>
7	Receiving IP address	IP	String	
8	Data type	HTTP	String	7 or 8 characters (Request or Response)
9	Method information	HTTP	String	1 to 127 characters (such as GET or CONNECT)
10	URI information	HTTP	String	1 to 255 characters
11	Referer	HTTP	String	
12	Cookie	HTTP	String	
13	Status code	HTTP	String	3 characters (such as 200, 404)

No.	Data	Protocol	Java data type	Value range
14	Connection	HTTP	String	1 to 127 characters (close or Keep-Alive)
15	Content-Length	HTTP	Integer	0 to 2147483647
16	Content-Type	HTTP	String	3 to 255 characters
17	Message body	HTTP	String	0 to 2,048 characters

Legend:

--: Not applicable

#

When you specify time data in the field, you must specify at least 6 digits including the significant digits for the CQL data type (TIMESTAMP) in the query definition file.

### 9.10.3 File output connector definition

You specify a file output connector definition (`FileOutputConnectorDefinition` tag) as a child element of a CB definition for output (`OutputCBDefinition` tag).

For details about file output processing, see *10.6 File output*.

#### (1) Format

```
<FileOutputConnectorDefinition>
  <output compositionType="{WRAP_AROUND|ANTI_WRAP_AROUND}"
    maxNumber="maximum-output-files-count"
    maxSize="maximum-output-file-size" />
  <file path="output-path-name"
    prefix="prefix-name"
    addDate="{ON|OFF}"
    extension="extension" />
</FileOutputConnectorDefinition>
```

#### (2) Details of definition

`FileOutputConnectorDefinition` tag (all definition information)

Defines all file output connector definition information. You specify this definition only once.

`output` tag (output definition)

Defines information about the output processing on output files. You specify this definition only once.

```
compositionType="{WRAP_AROUND|ANTI_WRAP_AROUND}"
```

Specifies the structure of the output files. Available structures are wraparound and non-wraparound. With both structures, if a specified output file name already exists, the existing file will be overwritten. If this attribute is omitted, WRAP\_AROUND is assumed.

The permitted values are as follows:

- WRAP\_AROUND

Outputs the files in the wraparound mode.

In this mode, records are output until the number of files specified in the `maxNumber` attribute is reached and then the first output file created is overwritten. During normal operation, specify WRAP\_AROUND in order to avoid erroneous system operation due to a shortage of disk space.

If you specify WRAP\_AROUND, you must also specify OFF in the `addDate` attribute in the `file` tag.

- ANTI\_WRAP\_AROUND

Outputs the files in the non-wraparound mode.

In this mode, records are output until the number of files specified in the `maxNumber` attribute is reached and then the standard adaptor stops outputting records.

```
maxNumber="maximum-output-files-count"
```

Specifies the maximum number of output files, as an integer from 1 to 100000. If this attribute is omitted, 256 is assumed.

```
maxSize="maximum-output-file-size"
```

Specifies the number of record rows to be written in one file, as an integer from 1 to 10000. This attribute cannot be omitted.

`file` tag (output file definition)

Defines information about the output files. You specify this definition only once.

The output files are created according to this definition. For the naming rules for output files, see (4) *Naming rules for output files*.

```
path="output-path-name"
```

Specifies the path name of the directory to which the files are to be output, as 1 to 160 single-byte characters. If the specified directory does not exist, it will be created. This attribute cannot be omitted.

The following characters are permitted for this attribute:

Single-byte alphanumeric characters, !, #, \$, %, &, ', (, ), =, ~, ` , {, }, +, -, ^, ., \_, @, \, /, :, single-byte space

`prefix="prefix-name"`

Specifies a prefix for the file names, as 1 to 60 single-byte characters. This must be a path relative to the output path name specified in the `path` attribute. This attribute cannot be omitted.

The characters \, /, and : cannot be specified.

The following characters are permitted for this attribute:

Single-byte alphanumeric characters, !, #, \$, %, &, ', (, ), =, ~, ` , {, }, +, -, ^, ., \_, @, single-byte space

A specified file name cannot be a reserved device name (such as NUL). If a reserved device name is specified, operation is not guaranteed. If an attempt is made to open a file for which a reserved device name has been specified, the KFSP46211-E message will be displayed, resulting in an error.

`addDate="{ON|OFF}"`

Specifies whether the date and time (in the format *hhmmss\_MMDD\_YYYY*) are to be added to the output file names. If this attribute is omitted, OFF is assumed.

The permitted values are as follows:

- ON

Adds the date and time.

- OFF

Does not add the date or time.

`extension="extension"`

Specifies the extension to be added to the output file names. If this attribute is omitted, files without an extension are output.

The following characters are permitted for this attribute:

Single-byte alphanumeric characters, !, #, \$, %, &, ', (, ), =, ~, ` , {, }, +, -, ^, ., \_, @, single-byte space

**(3) Example**

```

<?xml version="1.0" encoding="UTF-8"?>
<root:AdaptorCompositionDefinition
  xmlns:focon="http://www.hitachi.co.jp/soft/xml/sdp/adaptor/definition/callback/
  FileOutputConnectorDefinition">
  <!-- Omitted -->

  <!-- CB definition for output -->
  <cb:OutputCBDefinition
    class="jp.co.Hitachi.soft.sdp.adaptor.callback.io.FileOutputCBImpl"
    name="outputer1">
    <!-- File output connector definition -->
    <focon:FileOutputConnectorDefinition>
      <!-- Output definition -->
      <focon:output compositionType="WRAP_AROUND" maxNumber="100" maxSize="10"/>
      <!-- Output file definition -->
      <focon:file path="D:\home\" prefix="out" addDate="OFF" extension="csv"/>
    </focon:FileOutputConnectorDefinition>
  </cb:OutputCBDefinition>

```

**(4) Naming rules for output files**

Output files are created according to the definition specified in the `file` tag.

The following shows the format of the names of output files:

```
' ['prefix']' < ['date-and-time_']' > ['sequence-number']' < .extension >
```

Legend:

[ ]: An item enclosed in square brackets must be specified as is, including the brackets.

< >: An item enclosed in these symbols is optional.

The following explains each value of the file name.

*prefix*

The value specified in the `prefix` attribute is used.

*date-and-time*

The date and time are specified in the format *hhmmss\_MMDD\_YYYY*:

- *hh*: Hour (00 to 23)
- *mm*: Minute (00 to 59)
- *ss*: Second (00 to 59)
- *MM*: Month (01 to 12)
- *DD*: Day (01 to 31)

- *YYYY*: Year (4-digit calendar year)

To specify whether or not the date and time are to be added to the output file name, use the `addDate` attribute. If you specify `ON` in the `addDate` attribute, the date and time are added; if you specify `OFF`, the date and time are not added.

#### *sequence-number*

This is the order in which output files are created. The sequence numbers begin with 1 and end with the value specified in the `maxNumber` attribute of the `output` tag. The sequence numbering also resets to 1 when the standard adaptors are restarted.

When the date and time are output to the output file names, the sequence numbering continues even though the year-month-day changes. The following shows an example.

Example:

- `path` attribute value: `D:\home\`
- `prefix` attribute value: `a`
- `addDate` attribute value: `ON`
- `extension` attribute value: `csv`

The table below shows the file creation numbers and the created file names in this example.

Creation number	Created file name
1	D:\home\a235015_0706_2009_1.csv
2	D:\home\a235959_0706_2009_2.csv
3	D:\home\a000130_0707_2009_3.csv

Note that if `WRAP_AROUND` is specified in the `compositionType` attribute in the `output` tag and `ON` is specified in the `addDate` attribute, the files are different even if they have the same sequence numbers because the date and time are added.

#### *extension*

The value specified in the `extension` attribute is used.

### 9.10.4 Dashboard output connector definition

You specify a dashboard output connector definition (`DashboardOutputConnectorDefinition` tag) as a child element of a CB definition for output (`OutputCBDefinition` tag).

For details about dashboard output processing, see *10.7 Dashboard output*.

### (1) Format

```
<DashboardOutputConnectorDefinition
  MaxNum="maximum-number-of-records-to-be-retained"
  Record="record-name"
  ReadRecordRemoveFlag="{ON|OFF}" >
  <RecordHoldTime
    DateReference="{CURRENT_DATE|LAST_UPDATE}"
    RecordTime="record-retention-period"
    DateFieldPosition="time-field-number" />
  <DataProcessingDefinition
    Name="{HistoryRecorder|NoDataProcessing}"
    <HistoryRecorder/>
    <NoDataProcessing/>
  </DataProcessingDefinition>
</DashboardOutputConnectorDefinition>
```

### (2) Details of definition

DashboardOutputConnectorDefinition tag (all definition information)

Defines all dashboard output connector definition information. You specify this definition only once.

MaxNum= "maximum-number-of-records-to-be-retained"

Specifies the maximum number of records to be retained in the record area, as an integer from 1 to 10000. If this attribute is omitted, 1000 is assumed.

When the number of records exceeds the value specified in this attribute, records are deleted sequentially beginning with the oldest record.

Record= "record-name"

Specifies a record to be output to the dashboard. This attribute cannot be omitted.

ReadRecordRemoveFlag= "{ON|OFF}"

Specifies whether the record acquired by Dashboard Server is to be deleted from the dashboard output connector. If this attribute is omitted, OFF is assumed.

The permitted values are as follows:

- ON  
Deletes record.
- OFF  
Does not delete record.

**RecordHoldTime** tag (definition of record retention period)

Defines the period of time during which the record is to be retained when records are deleted from the dashboard output connector based on the retention period. If this attribute is omitted, record deletion based on the retention period is not performed.

A record is deleted when the following condition is satisfied:

```
(Reference time in the DateReference attribute - retention period in the
RecordTime attribute)
> Time in the field with the DateFieldPosition attribute
```

For example, if the reference time is 10:00:10, retention period is 5 seconds, and time in the field is 10:00:04, that record is deleted.

```
DateReference="{CURRENT_DATE|LAST_UPDATE}"
```

Specifies the reference time for determining record deletion. This attribute cannot be omitted.

The permitted values are as follows:

- CURRENT\_DATE

Uses the current system time as the reference time.

- LAST\_UPDATE

Uses the last time a tuple was received as the reference time.

```
RecordTime="record-retention-period"
```

Specifies the record's retention period (in seconds), as an integer from 0 to 86400. This attribute cannot be omitted.

```
DateFieldPosition="time-field-number"
```

Specifies the number of the field in the record that contains the time, as an integer from 1 to 3000. This attribute cannot be omitted.

**DataProcessingDefinition** tag (definition of data processing)

Defines how to process the dashboard-display data that is output by the dashboard output connector. If this definition is omitted, data is not processed.

```
Name="{HistoryRecorder|NoDataProcessing}"
```

Specifies whether or not to process the dashboard-display data. This attribute cannot be omitted.

The permitted values are as follows:



- `HistoryRecorder`

Processes dashboard-display data. If you want to display charts, specify `HistoryRecorder`. When dashboard-display data is processed, all the records that meet the maximum number of records to be retained and the record retention period are retained as the dashboard-display data.

If you specify `HistoryRecorder`, specify an empty `HistoryRecorder` tag under the `DataProcessingDefinition` tag.

- `NoDataProcessing`

Does not process dashboard-display data. If you do not need to process the dashboard-display data, specify `NoDataProcessing`. When dashboard-display data is not processed, any existing dashboard-display data is deleted each time a record is acquired from the dashboard output connector.

If you specify `NoDataProcessing`, specify an empty `NoDataProcessing` tag under the `DataProcessingDefinition` tag.

#### `HistoryRecorder` tag (log retention definition)

If you specified `HistoryRecorder` in the `Name` attribute in the `DataProcessingDefinition` tag, specify an empty `HistoryRecorder` tag under the `DataProcessingDefinition` tag.

If you specified `NoDataProcessing` in the `Name` attribute, there is no need to specify this tag.

#### `NoDataProcessing` tag (no-processing output definition)

If you specified `NoDataProcessing` in the `Name` attribute in the `DataProcessingDefinition` tag, specify an empty `NoDataProcessing` tag under the `DataProcessingDefinition` tag.

If you specified `HistoryRecorder` in the `Name` attribute, there is no need to specify this tag.

**(3) Example**

```

<?xml version="1.0" encoding="UTF-8"?>
<root:AdaptorCompositionDefinition
xmlns:docon="http://www.hitachi.co.jp/soft/xml/sdp/adaptor/definition/callback/
DashboardOutputConnectorDefinition">
<!-- Omitted -->

<!-- CB definition for output -->
<cb:OutputCBDefinition
class="jp.co.Hitachi.soft.sdp.adaptor.callback.io.dashboard.DashboardOutputCBImp1"
name="outputer2">
  <!-- Dashboard output connector definition -->
  <docon:DashboardOutputConnectorDefinition MaxNum="1000" Record="RECORD2"
ReadRecordRemoveFlag="OFF">
    <!-- Definition of record retention period -->
    <docon:RecordHoldTime DateReference="LAST_UPDATE"
RecordTime="300" DateFieldPosition="1"/>
    <!-- Definition of data processing -->
    <docon:DataProcessingDefinition Name="NoDataProcessing">
      <docon:NoDataProcessing/>
    </docon:DataProcessingDefinition>
  </docon:DashboardOutputConnectorDefinition>
</cb:OutputCBDefinition>

```

## 9.11 CB definitions for data editing in the adaptor configuration definition file

This section discusses the CB definitions for data editing in the adaptor configuration definition file (`AdaptorCompositionDefinition.xml`).

You specify the CB definitions for data editing as child elements of a CB definition for editing (`DataEditCBDefinition` tag) described in *9.9.3 CB definition for editing*.

The table below lists and describes the definitions that constitute the CB definitions for data editing.

*Table 9-11: List of definitions constituting the CB definitions for data editing*

No.	Definition constituting a CB definition for data editing	Parent element	Subsection
1	Format conversion definition ( <code>FormatDefinition</code> tag)	CB definition for editing	9.11.1
2	Mapping definition ( <code>MappingDefinition</code> tag)		9.11.2
3	Filter definition ( <code>FilterDefinition</code> tag)		9.11.3
4	Record extraction definition ( <code>RecordExtractionDefinition</code> tag)		9.11.4

### 9.11.1 Format conversion definition

You specify a format conversion definition (`FormatDefinition` tag) as a child element of a CB definition for editing (`DataEditCBDefinition` tag) described in *9.9.3 CB definition for editing*.

For details about format conversion processing, see *10.2.2(3) Format conversion* or *10.6.2(4) Format conversion*.

**(1) Format**

```

<FormatDefinition ioType="{ INPUT | OUTPUT } ">
  <common>
    <unmatchedFormat>{ IGNORE | WARNING | ERROR }
  </unmatchedFormat>
    <format timestampFormat="format-number" year="start-year" month="start-month" />
  </common>
  <records>
    <record name="record-name" exp="record-structure"
      timestampFormat="format-number" year="start-year" month="start-month">
      <field name="field-name"

type="{ INT | SHORT | BYTE | LONG | BIG_DECIMAL | FLOAT | DOUBLE | STRING | DATE | TIME | TIMESTAMP } "
      pattern="pattern" />
    </record>
  </records>
</FormatDefinition>

```

**(2) Details of definition**

**FormatDefinition tag (all definition information)**

Defines all format conversion definition information. You specify this definition only once.

`ioType= " { INPUT | OUTPUT } "`

Specifies the type of standard adaptors being specified in this definition. This attribute cannot be omitted.

The permitted values are as follows:

- INPUT

Specifies that format conversion is being defined for input adaptors.

- OUTPUT

Specifies that format conversion is being defined for output adaptors.

**common tag (common definition)**

Defines common format conversion definition information. You specify this definition only once.

**unmatchedFormat tag (definition of records that do not match the conversion pattern)**

Specifies how to handle a record that does not match the format conversion pattern.

You specify this definition only once. This definition is optional. If you omit this definition, `ERROR` is assumed.

The permitted values are as follows:

- IGNORE  
Ignores the detected record and resumes standard adaptor processing.
- WARNING  
Outputs a warning message and resumes standard adaptor processing.
- ERROR  
Outputs an error message and terminates the standard adaptor.

`format` tag (format definition)

Defines the character string representation for the `TIMESTAMP` data type. You can specify this definition only once. If you do not use a character string representation for the `TIMESTAMP` data type, you can omit this definition.

`timestampformat="format-number"`

Specifies the format number of the `TIMESTAMP` data type, as an integer from 1 to 4. If this attribute is omitted, 1 is assumed.

Note that if you specify the `timestampformat` attribute in both the `format` tag and the `records` tag, the specification in the `records` tag takes effect.

The table below lists and describes the format numbers that can be specified.

Format number	Format of character string representation	Format
1 (default value)	<i>year-month-day</i> <i>hour:minute:second.millisecond</i> (1 to 9 digits)	<i>yyyy-MM-dd</i> <i>HH:mm:ss.fffffff</i>
2 <sup>#1</sup>	<i>month-name<sup>#2</sup> day hour:minute:second</i>	<i>MMM dd HH:mm:ss</i>
3	<i>year/month/day</i> <i>hour:minute:second.millisecond</i> (3 digits)	<i>yyyy/MM/dd HH:mm:ss.SSS</i>
4	<i>day/month-name<sup>#2</sup>/year:hour:minute:second</i>	<i>dd/MMM/yyyy:HH:mm:ss</i>

#1

Format 2 does not contain the year. If you specify format 2 for input adaptors, you should use the `year` and `month` attributes to specify the start year and month. If you specify format 2 for output adaptors, neither the `year` nor the `month` attribute can be specified.

If neither the `year` nor the `month` attribute is specified, the system time (year and month) at the time the standard adaptor is started becomes the start year and month for the `TIMESTAMP` type.

#2

This is the 3-letter month name in English (JAN, FEB, MAR, APR, MAY, JUN, JUL, AUG, SEP, OCT, NOV, DEC).

`year="start-year"`

If you specified 2 in the `timestampformat` attribute, use this attribute to specify the start year for the `TIMESTAMP` data type. Express the year as an integer from 1970 to 2261. This attribute cannot be specified for output adaptors.

For details about specification of the start year and month for the `TIMESTAMP` data type, see (4) *Specifying the start year and month for the `TIMESTAMP` data type*.

`month="start-month"`

If you specified 2 in the `timestampformat` attribute, use this attribute to specify the start month for the `TIMESTAMP` data type. Express the month as an integer from 1 to 12. This attribute cannot be specified for output adaptors.

For details about specification of the start year and month for the `TIMESTAMP` data type, see (4) *Specifying the start year and month for the `TIMESTAMP` data type*.

`records` tag (record group definition)

Defines record group information. You specify this definition only once. This definition is mandatory.

`record` tag (record definition)

Defines information about a record. You can specify a maximum of 10 record definitions. This definition is mandatory.

`name="record-name"`

Specifies a name for identifying this record information, as 1 to 100 single-byte alphanumeric characters and the underscore (`_`). This name must begin with a single-byte alphabetic character.

This record name must be unique within the `records` tag. This attribute cannot be omitted.

`exp="record-structure"`

Specifies the fields that constitute the record, as 1 to 1,000,000 characters.

You can specify in the record structure a delimiter that is specific to this record. You can also specify a different delimiter for each field. This attribute cannot be omitted.

For the format of the record structure, see (5) *Format of record structure*.

`timestampformat="format-number"`

Specifies the format number of the `TIMESTAMP` data type, as an integer from 1 to 4. If this attribute is omitted, the `timestampformat` attribute value in the `format` tag is assumed.

Note that if you specify the `timestampformat` attribute in both the `format` tag and the `records` tag, the specification in the `records` tag takes effect.

For the values that can be specified as the format numbers and their meanings, see the description of the `timestampformat` attribute in the `format` tag.

`year="start-year"`

If you specified 2 in the `timestampformat` attribute, use this attribute to specify the start year for the `TIMESTAMP` data type. Express the year as an integer from 1970 to 2261. This attribute cannot be specified for output adaptors.

For details about specification of the start year and month for the `TIMESTAMP` data type, see (4) *Specifying the start year and month for the `TIMESTAMP` data type*.

`month="start-month"`

If you specified 2 in the `timestampformat` attribute, use this attribute to specify the start month for the `TIMESTAMP` data type. Express the month as an integer from 1 to 12. This attribute cannot be specified for output adaptors.

For details about specification of the start year and month for the `TIMESTAMP` data type, see (4) *Specifying the start year and month for the `TIMESTAMP` data type*.

`field` tag (field definition)

Defines field information. You can specify a maximum of 3,000 field definitions. This definition is mandatory.

`name="field-name"`

Specifies a name for identifying a field and its information, as 1 to 100 single-byte alphanumeric characters and the underscore (`_`). This name must begin with a single-byte alphabetic character.

This field name must be unique within the `records` tag. This attribute cannot be omitted.

```
type=" { INT | SHORT | BYTE | LONG | BIG_DECIMAL | FLOAT | DOUBLE | STRING |  
DATE | TIME | TIMESTAMP } "
```

Specifies the data type of the field, which must correspond to a Java data type. This attribute cannot be omitted.

For the values that can be specified in this attribute and the corresponding Java and CQL data types, see (6) *Values of the type attribute and the corresponding Java and CQL data types*.

```
pattern="pattern"
```

If you specified `STRING` in the `type` attribute, specify a pattern for the field value using a regular expression. Because the `java.util.regex.Pattern` class is used to analyze the regular expression, you must specify a regular expression that is within the range supported by the `java.util.regex.Pattern` class. The dollar sign (\$) cannot be used.

If you specified a value other than `STRING` in the `type` attribute or are specifying this definition for output adaptors, this attribute cannot be specified; if it is specified in such a case, an error results.

You can specify a constant for the pattern. When a constant is specified for the pattern, that constant will be treated as the field's value.

For the rules for representing the data types that are applied when this attribute is omitted, see (7) *Rules for representing data types*.



**(3) Example**

```

<?xml version="1.0" encoding="UTF-8"?>
<root:AdaptorCompositionDefinition
xmlns:form="http://www.hitachi.co.jp/soft/xml/sdp/adaptor/definition/callback/
FormatDefinition">
<!-- Omitted -->

<!-- CB definition for editing -->
<cb:DataEditCBDefinition
class="jp.co.Hitachi.soft.sdp.adaptor.callback.dataedit.formattranslate.InputFormatT
ranslatorCBImpl" name="editor1">
  <!-- Format conversion definition -->
  <form:FormatDefinition ioType="INPUT">
    <form:common/>
    <!-- Record group definition -->
    <form:records>
      <form:record name="R1" exp="($_F1),($_F2),($_F3),($_F4),($_F5)">
        <!-- Field definition -->
        <form:field name="F1" type="INT"/>
        <form:field name="F2" type="STRING" pattern="^[,]*"/>
        <form:field name="F3" type="STRING" pattern="[A-Z]+.[A-Z]+"/>
        <form:field name="F4" type="INT"/>
        <form:field name="F5" type="INT"/>
      </form:record>
    </form:records>
  </form:FormatDefinition>
</cb:DataEditCBDefinition>

```

**(4) Specifying the start year and month for the *TIMESTAMP* data type**

If you specify format number 2 in the `timestampformat` attribute in the `format` tag for input adaptors and you specify both `year` and `month` attributes in the `format` tag, the start year and month for a field of the *TIMESTAMP* data type are determined as follows:

- First input record immediately following startup of the input adaptor
 

The value specified in the `year` and `month` attributes in the `format` tag become the start year and month for the *TIMESTAMP*-type field.
- Second and subsequent input records
 

The start year and month are determined using the year and month for the *TIMESTAMP*-type field in the previous input record as the reference year-month.

For the second and subsequent input records, the start year and month are determined by comparing the month value in a field of the input record with the reference year-month, as described below.

- When the month value in a field of the input record  $\geq$  month immediately preceding the reference year-month
 

The year of the reference year-month becomes the year value for the field. If the month immediately preceding the reference year-month is December and the

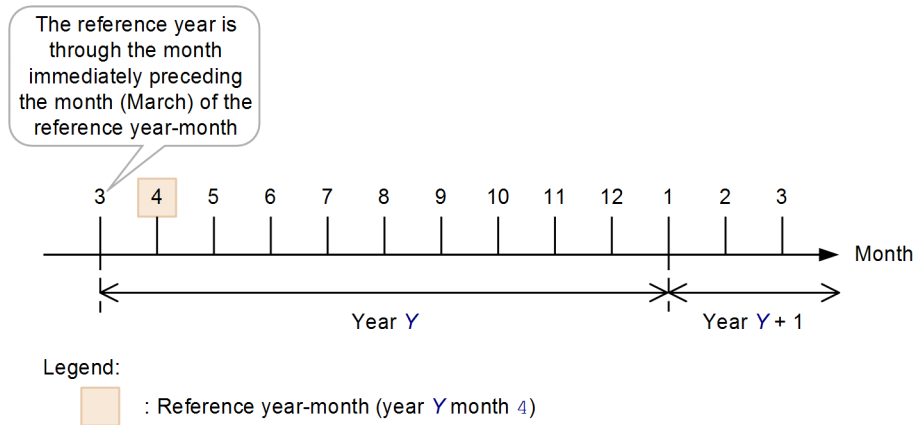
month value in the field is December, the year immediately preceding the reference year-month becomes the year value for the field.

- When the month value in a field of the input record < month immediately preceding the reference year-month

The year immediately following the reference year-month becomes the year value for the field. If the month immediately preceding the reference year-month is December, the year of the reference year-month becomes the year value for the field.

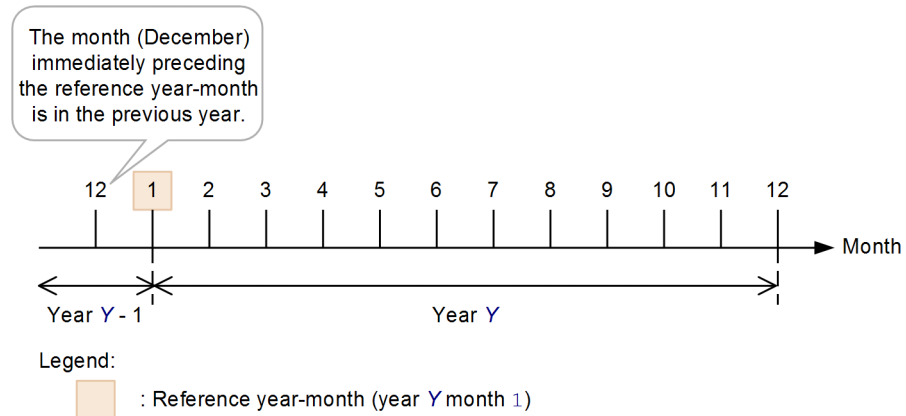
The figure below shows an example where the reference year-month is set to year  $Y$  and month 4. If the month value in a field of the input record is in the range from 3 to 12, the year value for the field becomes  $Y$ . If the month value is 1 or 2, the year value for the field becomes  $Y + 1$ .

Figure 9-3: Example when the reference year-month is set to year  $Y$  and month 4



The figure below shows an example where the reference year-month is set to year  $Y$  and month 1. If the month value in a field of the input record is 12, the year value for the field becomes  $Y - 1$ . If the month value is in the range from 1 to 11, the year value for the field becomes  $Y$ .

Figure 9-4: Example when the reference year-month is set to year Y and month 1



### (5) Format of record structure

This subsection discusses the format of the record structure that is specified in the `exp` attribute in the `record` tag.

Format of record structure:

```
{ [delimiter] (§field-name) [delimiter]
  | [delimiter] (§field-name) [delimiter] . . . }
```

The following explains each value.

#### *delimiter*

Specifies the delimiter used to separate the fields in this record. When the value of the `ioType` attribute in the `FormatDefinition` tag is `INPUT`, you can use a regular expression to specify a character string. To use a character that has a special meaning in regular expressions ( `(`, `)`, `[`, `]`, `.`, `*`, `?`, `+`, `^`, or `$`), you must use the backslash (`\`) as an escape character.

When the value of the `ioType` attribute in the `FormatDefinition` tag is `OUTPUT`, a regular expression cannot be used, so there is no need for an escape character to handle characters that have a special meaning in regular expressions.

#### *field-name*

Specifies the name of an individual field in this record.

Rules for specifying the record structure:

- There is no need to specify the double-quotation mark (`"`) for the delimiter.

- The specification of each field name must begin with `($_` and end with `)`.
- You must specify the names of all fields that constitute this record.
- Specify in the `field` tag the names of the fields that constitute this record in the same order, from left to right, that the fields are to be arranged in the record.
- The same field name cannot be specified more than once. The name of a field that is not a part of this record cannot be specified.
- For special characters, their replacement characters must be specified. For details about the replacement characters, see the table of special characters (symbols) and replacement characters in *9.2 Notes about creating adaptor definition files*.

Examples of record structure specifications:

Example 1:

- There are five fields, F1 through F5.
- The fields are delimited by the comma (,).

This record's structure is specified as follows:

```
"($_F1),($_F2),($_F3),($_F4),($_F5)"
```

Example 2:

- There are three fields, F1 through F3.
- The record begins with `<`.
- The delimiter between fields F1 and F2 is `> MSG`.
- The delimiter between fields F2 and F3 is a single-byte space.

This record's structure is specified as follows:

```
"&lt;($_F1)&gt; MSG($_F2) ($_F3)"
```

In this example, `<` and `>` are replaced with `&lt;` and `&gt;`, respectively, because `<` and `>` are special characters.

### **(6) Values of the type attribute and the corresponding Java and CQL data types**

The table below shows the correspondences between the values of the `type` attribute in the `field` tag and the Java and CQL data types.

For the rules for representing the data types, see the rules for data types for the `pattern` attribute in the `field` tag. For details about the CQL data types, see the manual *uCosminexus Stream Data Platform - Application Framework Application*

*Development Guide.**Table 9-12: Values of the type attribute and the corresponding Java and CQL data types*

No.	Data type (value of type attribute)	Classification	Data type	Java data type	CQL data type
1	INT	Numeric data	4-byte integer	Primitive type <code>int</code>	INT [ EGER ]
2	SHORT		2-byte integer	Primitive type <code>short</code>	SMALLINT
3	BYTE		1-byte integer	Primitive type <code>byte</code>	TINYINT
4	LONG		8-byte integer	Primitive type <code>long</code>	BIGINT
5	BIG_DECIMAL		Fixed-point number	java.math.BigDecimal class	DEC [IMAL] #1
6					NUMERIC #1
7	FLOAT		4-byte real number	Primitive type <code>float</code>	REAL
8	DOUBLE		8-byte real number	Primitive type <code>double</code>	FLOAT DOUBLE
9	STRING	Character data	Character string	java.lang.String class	CHAR [ACTER] #3
10					CHAR [ACTER] (n) #4 VARCHAR (p) #5
11	DATE	Date/time data	Date (year-month-day)	java.sql.Date class	DATE
12	TIME		Time (hour-minute-second)	java.sql.Time class	TIME
13	TIMESTAMP		Date and time (year-month-day + hour-minute-second + millisecond)	java.sql.Timestamp class	TIMESTAMP #6
14					TIMESTAMP [ (q) ] #7

#1

The number of digits is assumed to be 15. If the number of digits exceeds 15, an error occurs when tuples are sent.

#2

$m$  is a positive integer in the range  $1 \leq m \leq 38$ . If the number of digits exceeds  $m$ , an error occurs when tuples are sent.

#3

The number of characters is assumed to be 1. If the number of characters exceeds 1, an error occurs when tuples are sent.

#4

$n$  is a positive integer in the range  $1 \leq n \leq 255$ . If the number of characters exceeds  $n$ , an error occurs when tuples are sent.

#5

$p$  is a positive integer in the range  $1 \leq p \leq 32767$ . If the number of characters exceeds  $p$ , an error occurs when tuples are sent.

#6

*year-month-day + hour-minute-second + millisecond* (3 digits) is assumed. If a precision higher than milliseconds is specified, a tuple send error occurs.

#7

$q$  is an integer in the range  $0 \leq q \leq 9$  and indicates the fraction part of a second. If a precision higher than  $q$  is specified, a tuple send error occurs.

**(7) Rules for representing data types**

The table below describes the rules for representing data types when the `pattern` attribute is omitted from the `field` tag.

Table 9-13: Rules for representing data types when the pattern attribute is omitted from the field tag

No.	Data type (value of type attribute)	Pattern for data type (regular expression)	Description	Whether or not changeable#
1	INT	"[-]{0,1}[0-9]+"	In this character string pattern, one minus sign (-) or no minus signs appears at the beginning, and a number in the range from 0 to 9 appears once or more.	N
2	SHORT			N
3	BYTE			N
4	LONG			N
5	BIG_DECIMAL	"[-]{0,1}[0-9]+\.[0-9]+"	In this character string pattern: <ul style="list-style-type: none"> <li>• One minus sign (-) or no minus signs appears at the beginning.</li> <li>• A number in the range from 0 to 9 appears once or more.</li> <li>• A number in the range from 0 to 9 appears once or more immediately after the period (.).</li> </ul>	N
6	FLOAT	"[-]{0,1}[0-9]+\.[0-9]+"	In this character string pattern: <ul style="list-style-type: none"> <li>• One minus sign (-) or no minus signs appears at the beginning.</li> <li>• A number in the range from 0 to 9 appears once or more.</li> <li>• A number in the range from 0 to 9 appears once or more immediately after the period (.).</li> </ul>	N
7	DOUBLE			N
8	STRING	"[^, ;]*"	In this character string pattern, a character other than the space, comma (,), or semicolon (;) appears repeatedly. If the data type is STRING, you can change (specify) the pattern. Example of changing (specifying) the pattern: This example uses only the period (.) as a delimiter: " [ ^ . ] + "	Y

No.	Data type (value of type attribute)	Pattern for data type (regular expression)	Description	Whether or not changeable#
9	DATE	"[0-9]{1,4}-[0-9]{1,2}-[0-9]{1,2}"	<p>This is a character string pattern in the format <i>yyyy-mm-dd</i>. The permitted characters are shown below; only the number of digits is checked.</p> <ul style="list-style-type: none"> <li>• <i>yyyy</i> Numeric characters from 0 to 9999</li> <li>• <i>mm</i> Numeric characters from 0 to 99</li> <li>• <i>dd</i> Numeric characters from 0 to 99</li> </ul> <p>If the values of <i>mm</i> and <i>dd</i> are outside of the applicable range for <i>MM</i> and <i>DD</i>, respectively, they are converted to the regular year-month-day according to the <code>java.sql.Date</code> specifications.</p> <p>Examples:</p> <p style="margin-left: 40px;">2001-13-12 → 2002-01-12 0000-01-12 → 0001-01-12</p>	N
10	TIME	"[0-9]{1,2}:[0-9]{1,2}:[0-9]{1,2}"	<p>This is a character string pattern in the format <i>hh:mm:ss</i>. The permitted characters are shown below; only the number of digits is checked.</p> <ul style="list-style-type: none"> <li>• <i>hh</i> Numeric characters from 0 to 99</li> <li>• <i>mm</i> Numeric characters from 0 to 99</li> <li>• <i>ss</i> Numeric characters from 0 to 99</li> </ul> <p>If the values of <i>hh</i>, <i>mm</i>, and <i>ss</i> are outside of the applicable range for <i>HH</i>, <i>MM</i>, and <i>SS</i>, respectively, they are converted to the regular time according to the <code>java.sql.Time</code> specifications.</p> <p>Example:</p> <p style="margin-left: 40px;">16:22:66 → 16:23:06</p>	N
11	TIMESTAMP	The pattern depends on the value of the <code>timestampformat</code> attribute in the <code>records</code> tag, as described below.	The pattern depends on the value of the <code>timestampformat</code> attribute in the <code>records</code> tag, as described below.	N



No.	Data type (value of type attribute)	Pattern for data type (regular expression)	Description	Whether or not changeable#
		<p>When timestampformat attribute value is 1: " [0-9]{1,4}-[0-9]{1,2}-[0-9]{1,2}:[0-9]{1,2}:[0-9]{1,2} \.[0-9]{1,9}"</p> <p>When timestampformat attribute value is 2: " [A-Za-z]{3} [0-9]{1,2}:[0-9]{1,2}:[0-9]{1,2}:[0-9]{1,2}"</p>	<p>When the value of the timestampformat attribute is 1: This is a character string pattern in the format <i>yyyy-mm-dd hh:mm:ss.fffffff</i>. The permitted characters are shown below; only the number of digits is checked.</p> <ul style="list-style-type: none"> <li>• <i>yyyy</i> Numeric characters from 0 to 9999</li> <li>• <i>mm</i> Numeric characters from 0 to 99</li> <li>• <i>dd</i> Numeric characters from 0 to 99</li> <li>• <i>hh</i> Numeric characters from 0 to 99</li> <li>• <i>mm</i> Numeric characters from 0 to 99</li> <li>• <i>ss</i> Numeric characters from 0 to 99</li> <li>• <i>fffffff</i> Numeric characters from 0 to 99999999</li> </ul> <p>When the value of the timestampformat attribute is 2: This is a character string pattern in the format <i>MMM dd HH:mm:ss</i>. The permitted characters are shown below; only the number of digits is checked.</p> <ul style="list-style-type: none"> <li>• <i>MMM</i> Alphabetic characters A to Z and a to z (3-letter name of the month in English)</li> <li>• <i>dd</i> Numeric characters from 0 to 99</li> <li>• <i>HH</i> Numeric characters from 0 to 99</li> <li>• <i>mm</i> Numeric characters from 0 to 99</li> <li>• <i>ss</i> Numeric characters from 0 to 99</li> </ul>	

No.	Data type (value of type attribute)	Pattern for data type (regular expression)	Description	Whether or not changeable#
		When timestampformat attribute value is 3: "[0-9]{1,4}/ [0-9]{1,2}/ [0-9]{1,2}[ ]+[0-9]{1,2}:[0-9]{ 1,2}\.[0-9]{3}"	When the value of the timestampformat attribute is 3: This is a character string pattern in the format <i>yyyy/MM/dd</i> <i>HH:mm:ss.SSS</i> . The permitted characters are shown below; only the number of digits is checked. <ul style="list-style-type: none"> <li>• <i>yyyy</i>              Numeric characters from 0 to              9999</li> <li>• <i>MM</i>              Numeric characters from 0 to 99</li> <li>• <i>dd</i>              Numeric characters from 0 to 99</li> <li>• <i>HH</i>              Numeric characters from 0 to 99</li> <li>• <i>mm</i>              Numeric characters from 0 to 99</li> <li>• <i>ss</i>              Numeric characters from 0 to 99</li> <li>• <i>SSS</i>              Numeric characters from 000 to              999</li> </ul>	

No.	Data type (value of type attribute)	Pattern for data type (regular expression)	Description	Whether or not changeable <sup>#</sup>
		When timestampformat attribute value is 4: "[0-9]{1,2}/ [A-Za-z]{3}/ [0-9]{1,4}:[0-9] ]{1,2}:[0-9]{1, 2}:[0-9]{1,2}"	When the value of the timestampformat attribute is 4: This is a character string pattern in the format <i>dd/MMM/</i> <i>yyyy:HH:mm:ss</i> . The permitted characters are shown below; only the number of digits is checked. <ul style="list-style-type: none"> <li>• <i>dd</i>                Numeric characters from 0 to 99</li> <li>• <i>MMM</i>                Alphabetic characters A to Z and                a to z (3-letter name of the                month in English)</li> <li>• <i>yyyy</i>                Numeric characters from 0 to                9999</li> <li>• <i>HH</i>                Numeric characters from 0 to 99</li> <li>• <i>mm</i>                Numeric characters from 0 to 99</li> <li>• <i>ss</i>                Numeric character from 0 to 99</li> </ul> If the values of <i>mm</i> , <i>dd</i> , <i>hh</i> , <i>mm</i> , and <i>ss</i> are outside of the applicable range for MM (month), DD, HH, MM (minute), and SS, respectively, they are converted to the regular year-month-day-time according to the java.sql.Timestamp specifications. Example: 2001-13-10 16:22:66.101 → 2002-01-10 16:23:06.101	

Legend:

Y: Changeable

N: Not changeable

#

Indicates whether or not the character string pattern of each data type can be changed by specifying the `pattern` attribute.

## 9.11.2 Mapping definition

You specify a mapping definition (`MappingDefinition` tag) as a child element of a CB definition for editing (`DataEditCBDefinition` tag) described in 9.9.3 *CB definition for editing*.

For details about mapping processing, see 10.2.2(4) *Mapping* or 10.6.2(3) *Mapping*.

### (1) Format

```
<MappingDefinition ioType="{ INPUT|OUTPUT}" >
  <source>
    <streams>
      <stream name="stream-name"
        querygroup="query-group-name" >
        <column name="column-name"
          type="{ INT|SHORT|BYTE|LONG|BIG_DECIMAL|FLOAT|DOUBLE|STRING
|DATE|TIME|TIMESTAMP}"/>
        </stream>
      </streams>
    </source>
    <target>
      <streams>
        <stream name="stream-name"
          querygroup="query-group-name" >
          <column name="column-name"
            type="{ INT|SHORT|BYTE|LONG|BIG_DECIMAL|FLOAT|DOUBLE|STRING
|DATE|TIME|TIMESTAMP}"/>
          </stream>
        </streams>
      <records>
        <record name="record-name">
          <field name="field-name"
            type="{ INT|SHORT|BYTE|LONG|BIG_DECIMAL|FLOAT|DOUBLE|STRING |DATE|TIME|TIMESTAMP}"/>
          </records>
        </target>
      <intermediate>
        <mappings source="source-name"
          querygroup="query-group-name"
          target="target-name" >
          <map source="source-path-expression"
            function="built-in-function-name" argument1="argument-1"
            argument2="argument-2" target="target-path-expression"/>
          </mappings>
        </intermediate>
      </MappingDefinition>
```

### (2) Details of definition

`MappingDefinition` tag (all definition information)

Defines all mapping conversion definition information. You specify this definition only once.

`ioType="{ INPUT | OUTPUT }"`

Specifies the type of standard adaptors being specified in this definition. This attribute cannot be omitted.

The permitted values are as follows:

- INPUT

Specifies that mapping is being defined for input adaptors.

- OUTPUT

Specifies that mapping is being defined for output adaptors.

`source` tag (mapping source definition)

Defines information about the streams at the mapping source.

As shown in the table below, the information to be specified in this tag depends on the value of the `ioType` attribute in the `MappingDefinition` tag and what is being mapped.

Value of <code>ioType</code> attribute	What is being mapped	Information to be specified in the source tag
INPUT	Mapping between record and stream	Specify an empty-element tag
	Mapping between records	
OUTPUT	Mapping between record and stream	Specify the <code>streams</code> tag
	Mapping between records	Specify an empty-element tag

`target` tag (mapping target definition)

Defines information about the streams or records at the mapping target.

As shown in the table below, the information to be specified in this tag depends on the value of the `ioType` attribute in the `MappingDefinition` tag and what is being mapped.

Value of <code>ioType</code> attribute	What is being mapped	Information to be specified in the target tag
INPUT	Mapping between record and stream	Specify the <code>streams</code> tag
	Mapping between records	Specify the <code>records</code> tag
OUTPUT	Mapping between record and stream	Specify an empty-element tag

Value of <code>ioType</code> attribute	What is being mapped	Information to be specified in the target tag
	Mapping between records	Specify the <code>records</code> tag

**streams** tag (stream group definition)

Defines under the `source` or `target` tag information about the group of streams at the mapping source or target, respectively, when mapping is being performed between record and stream.

If you specify this definition, you specify it only once.

**stream** tag (stream definition)

Defines information about a stream at the mapping source or target.

You can specify a maximum of 1,024 stream definitions. This definition is mandatory.

`name="stream-name"`

Specifies the name of a stream, as 1 to 100 single-byte alphanumeric characters and the underscore (`_`). This name must begin with a single-byte alphabetic character. The specified pair of a stream name and a query group name must be unique within the `streams` tag. This attribute cannot be omitted.

`querygroup="query-group-name"`

Specifies the name of the query group, as 1 to 64 single-byte alphanumeric characters and the underscore (`_`). This name must begin with a single-byte alphabetic character. The specified pair of a stream name and a query group name must be unique within the `streams` tag. This attribute cannot be omitted.

**column** tag (column definition)

Defines information about the columns in the stream at the mapping source or target.

You can specify a maximum of 3,000 column definitions. This definition is mandatory.

`name="column-name"`

Specifies the name of a column, as 1 to 100 single-byte alphanumeric characters and the underscore (`_`). This name must begin with a single-byte alphabetic character. The specified column name must be unique within the column definition in the `streams` tag. This attribute cannot be omitted.

```
type=" { INT | SHORT | BYTE | LONG | BIG_DECIMAL | FLOAT | DOUBLE | STRING |
DATE | TIME | TIMESTAMP } "
```

Specifies the column's data type, which must correspond to a CQL data type.

For details about the values for this attribute and the corresponding CQL data types, see *9.11.1(6) Values of the type attribute and the corresponding Java and CQL data types*.

#### records tag (record group definition)

Defines under the `target` tag information about a group of records at the mapping target when mapping is performed between records.

If you specify this definition, you specify it only once.

#### record tag (record definition)

Defines information about a record at the mapping target. You can specify a maximum of 1,024 record definitions. This definition is mandatory.

```
name="record-name"
```

Specifies the name of a record, as 1 to 100 single-byte alphanumeric characters and the underscore (`_`). This record name must begin with a single-byte alphabetic character.

You can specify this attribute only once for a record. This attribute cannot be omitted. This record name must be unique within the corresponding standard adaptor.

#### field tag (field definition)

Defines information about the fields in the record at the mapping target. You can specify a maximum of 3,000 field definitions. This definition is mandatory.

```
name="field-name"
```

Specifies the name of a field, as 1 to 100 single-byte alphanumeric characters and the underscore (`_`). This field name must begin with a single-byte alphabetic character. You can specify this attribute only once for a field. This attribute cannot be omitted. This field name must be unique within the corresponding record.

```
type=" { INT | SHORT | BYTE | LONG | BIG_DECIMAL | FLOAT | DOUBLE | STRING |
DATE | TIME | TIMESTAMP } "
```

Specifies the field's data type, which must corresponds to a Java data type. This attribute cannot be omitted.

For details about the values for this attribute and the corresponding CQL data types, see *9.11.1(6) Values of the type attribute and the corresponding Java and CQL data types*.

**intermediate tag (intermediate mapping definition)**

Defines information about mapping from the source to the target.

You specify this definition only once.

**mappings tag (mapping group definition)**

Defines mapping information.

You can specify a maximum of 1,024 mapping definitions. This definition is mandatory.

`source="source-name"`

Specifies a source stream name or record name, as 1 to 100 single-byte alphanumeric characters and the underscore (`_`). This attribute cannot be omitted.

As shown in the table below, the information to be specified in this attribute depends on the value of the `ioType` attribute in the `MappingDefinition` tag and what is being mapped.

Value of <code>ioType</code> attribute	What is being mapped	Information to be specified in the source attribute
INPUT	Mapping between record and stream	Specify the record name at the mapping source
	Mapping between records	
OUTPUT	Mapping between record and stream	Specify the stream name that was specified in the <code>name</code> attribute in the <code>stream</code> tag
	Mapping between records	Specify the record name at the mapping source

`querygroup="query-group-name"`

Specifies, as 1 to 64 single-byte alphanumeric characters and the underscore (`_`), the name of the query group to which the source stream or target stream belongs when mapping is performed between record and stream. This name must begin with a single-byte alphabetic character. The specified pair of a query group name and a source name or the specified pair of a query group name and a target name must be unique within the `mappings` tag.

As shown in the table below, the information to be specified in this attribute depends on the value of the `ioType` attribute in the `MappingDefinition` tag and what is being mapped.



Value of ioType attribute	What is being mapped	Information to be specified in the querygroup attribute
INPUT	Mapping between record and stream	Specify the query group name in the <code>streams</code> tag under the <code>target</code> tag
	Mapping between records	Do not specify
OUTPUT	Mapping between record and stream	Specify the query group name in the <code>streams</code> tag under the <code>source</code> tag
	Mapping between records	Do not specify

`target="target-name"`

Specifies the stream name or record name at the mapping target, as 1 to 100 characters. The specified target name must be unique within the `mappings` tag. This attribute cannot be omitted.

As shown in the table below, the information to be specified in this attribute depends on the value of the `ioType` attribute in the `MappingDefinition` tag and what is being mapped.

Value of ioType attribute	What is being mapped	Information to be specified in the target attribute
INPUT	Mapping between record and stream	Specify the stream name that was specified in the <code>name</code> attribute in the <code>stream</code> tag
	Mapping between records	Specify the record name that was specified in the <code>name</code> attribute in the <code>record</code> tag
OUTPUT	Mapping between record and stream	Specify the record name at the mapping target
	Mapping between records	Specify the record name that was specified in the <code>name</code> attribute in the <code>record</code> tag

`map` tag (mapping information definition)

Defines mapping information. The order of the instances of this tag must match the order of the stream or record components for the mapping target. You can specify a maximum of 3,000 mapping information definitions. This definition is mandatory.

`source="source-path-expression"`

Specifies as a source path expression the name of a source field, as 1 to 1,024 characters. This attribute cannot be omitted.

As shown in the table below, the information to be specified in this attribute depends on the value of the `ioType` attribute in the `MappingDefinition` tag and what is being mapped.

Value of <code>ioType</code> attribute	What is being mapped	Information to be specified in the source attribute
INPUT	Mapping between record and stream	Specify the name of a field that belongs to the record whose name was specified in the <code>source</code> attribute in the <code>mappings</code> tag
	Mapping between records	
OUTPUT	Mapping between record and stream	Specify the name of a field that belongs to the stream whose name was specified in the <code>source</code> attribute in the <code>mappings</code> tag
	Mapping between records	Specify the name of a field that belongs to the record whose name was specified in the <code>source</code> attribute in the <code>mappings</code> tag

`function="built-in-function-name"`

When you perform mapping between records, you can use *built-in functions* in the `function` attribute in the `map` tag to acquire character strings or a time from a common record at the source and apply them to a common record at the target.

You specify in this attribute the name of the built-in function to be used to acquire a character string or time. Note that this attribute cannot be specified for mapping between record and stream (mapping to a common record in the case of input streams, or mapping from a common record in the case of output streams).

The values permitted for this attribute are as follows:

- `regexsubstring`

Specify this value to acquire a partial character string in regular expression from the input character string. The return value is a character string of the `String` type.

When you specify `regexsubstring`, specify arguments in the

`argument1` and `argument2` attributes.

- `getTupleTime`

This value is permitted only for output adaptors. Specify it to acquire the time in a tuple that is output from the output stream by the output adaptor. The return value is a time of the `TIMESTAMP` data type (in milliseconds).

When you specify `getTupleTime`, there is no need to specify the `argument1` or `argument2` attribute.

Note that `getTupleTime` cannot be specified when the value of the `ioType` attribute in the `MappingDefinition` tag is `INPUT`.

- `subTime`

Specify this value to obtain the time difference between the time values in two fields. The smaller time value is subtracted from the larger time value to obtain the difference. The return value is a time of the `LONG` type (in milliseconds).

When you specify `subTime`, specify arguments in the `argument1` and `argument2` attributes.

`argument1="argument-1"`

Specifies the first argument of the built-in function specified in the `function` attribute.

- When `regexsubstring` is specified in the `function` attribute

Specifies the name of the `String`-type field that stores the input character string, expressed as 1 to 100 single-byte alphanumeric characters and the underscore (`_`).

- When `subTime` is specified in the `function` attribute

Specifies the name of the `TIMESTAMP`-type field that stores the time, expressed as 1 to 100 single-byte alphanumeric characters and the underscore (`_`).

`argument2="argument-2"`

Specifies the second argument of the built-in function specified in the `function` attribute.

- When `regexsubstring` is specified in the `function` attribute

Specify in the format shown below the condition for extracting a partial character string, as 2 to 1,024 characters:

<code>[regular-expression] (" [regular-expression] ") " [regular-expression]</code>
---

Because the `java.util.regex.Pattern` class is used to analyze the regular expression, you must specify a regular expression within the range supported by the `java.util.regex.Pattern` class.

If the input character string in the field specified in *argument-1* matches the character string in the regular expression specified in *argument-2*, the character string that corresponds to the part enclosed in ( and ) is returned (`String` type). If they do not match, the null character is returned.

- When `subTime` is specified in the `function` attribute

Specify the name of the `TIMESTAMP`-type field that stores the time, expressed as 1 to 100 single-byte alphanumeric characters and the underscore (`_`).

`target="target-path-expression"`

Specifies as a target path expression the name of a target field, expressed as 1 to 100 single-byte alphanumeric characters and the underscore (`_`). This specified field name must be unique within the `mappings` tag. This attribute cannot be omitted.

The value of this attribute depends on the value of the `ioType` attribute in the `MappingDefinition` tag, as described below.

- When `INPUT` is specified in the `ioType` attribute

Specify the name of a field that belongs to the named target stream or record.

- When `OUTPUT` is specified in the `ioType` attribute

Specify the name of a field that belongs to the named target record.

### (3) Example

- Mapping between record and stream for input records

The following shows a coding example for performing mapping between record and stream by an input adaptor.

```

<?xml version="1.0" encoding="UTF-8"?>
<root:AdaptorCompositionDefinition
  <!-- Omitted -->
  xmlns:map="http://www.hitachi.co.jp/soft/xml/sdp/adaptor/definition/callback/
MappingDefinition">
  <!-- Omitted -->

  <!-- CB definition for editing -->
  <cb:DataEditCBDefinition
class="jp.co.Hitachi.soft.sdp.adaptor.callback.dataedit.mapping.InputMappingCBImpl"
name="editor1">
  <!-- Mapping definition -->
  <map:MappingDefinition ioType="INPUT">
    <map:source/>
    <map:target>
      <map:streams>
        <map:stream name="S1" querygroup="QG1">
          <map:column name="id" type="INT"/>
          <map:column name="time" type="STRING"/>
          <map:column name="name" type="STRING"/>
          <map:column name="val" type="INT"/>
        </map:stream>
      </map:streams>
    </map:target>
    <map:intermediate>
      <map:mappings source="R1" querygroup="QG1" target="S1">
        <map:map source="F1" target="id"/>
        <map:map source="F2" target="time"/>
        <map:map source="F3" target="name"/>
        <map:map source="F5" target="val"/>
      </map:mappings>
    </map:intermediate>
  </map:MappingDefinition>
</cb:DataEditCBDefinition>

```

#### ■ Mapping between records for input records

The following shows a coding example for performing mapping between records by an input adaptor. This example uses built-in functions in the `function` attribute to acquire a character string in a regular expression and the difference between time values.

```

<?xml version="1.0" encoding="UTF-8"?>
<root:AdaptorCompositionDefinition
  <!-- Omitted -->
  xmlns:map="http://www.hitachi.co.jp/soft/xml/sdp/adaptor/definition/callback/
  MappingDefinition">
  <!-- Omitted -->

  <!-- CB definition for editing -->
  <cb:DataEditCBDefinition
  class="jp.co.Hitachi.soft.sdp.adaptor.callback.dataedit.mapping.InputMappingCBImpl"
  name="editor1">
  <!-- Mapping definition -->
  <map:MappingDefinition ioType="INPUT">
    <map:source/>
    <map:target>
      <map:records>
        <map:record name="RECORD1">
          <map:field name="id" type="INT"/>
          <map:field name="F21" type="STRING"/>
          <map:field name="F22" type="LONG"/>
        </map:record>
      </map:records>
    </map:target>
    <map:intermediate>
      <map:mappings source="RECORD0" target="RECORD1">
        <map:map source="F1" target="id"/>
        <map:map function="regexsubstring" argument1="F11"
          argument2=".*Data=([0-9]+).*" target="F21"/>
        <map:map function="subTime" argument1="F12" argument2="F13" target="F22"/>
      </map:mappings>
    </map:intermediate>
  </map:MappingDefinition>
</cb:DataEditCBDefinition>

```

- Mapping between record and stream for output records

The following shows a coding example for performing mapping between record and stream by an output adaptor.

```

<?xml version="1.0" encoding="UTF-8"?>
<root:AdaptorCompositionDefinition
  <!-- Omitted -->
  xmlns:map="http://www.hitachi.co.jp/soft/xml/sdp/adaptor/definition/callback/
MappingDefinition">
  <!-- Omitted -->

  <!-- CB definition for editing -->
  <cb:DataEditCBDefinition
class="jp.co.Hitachi.soft.sdp.adaptor.callback.dataedit.mapping.InputMappingCBImpl"
name="editor1">
  <!-- Mapping definition -->
  <map:MappingDefinition ioType="INPUT">
    <map:source>
      <map:streams>
        <map:stream name="S1" querygroup="QG1">
          <map:column name="id" type="INT"/>
          <map:column name="time" type="TIMESTAMP"/>
          <map:column name="name" type="STRING"/>
        </map:stream>
      </map:streams>
    </map:source>
    <map:target/>
    <map:intermediate>
      <map:mappings source="S1" querygroup="QG1" target="RECORD1">
        <map:map source="id" target="F21"/>
        <map:map source="time" target="F22"/>
        <map:map source="name" target="F23"/>
      </map:mappings>
    </map:intermediate>
  </map:MappingDefinition>
</cb:DataEditCBDefinition>

```

#### ■ Mapping between records for output records

The following shows a coding example for performing mapping between records by an output adaptor. This example uses built-in functions in the `function` attribute to acquire a character string in a regular expression, as well as times in a tuple and the difference between the times.

```

<?xml version="1.0" encoding="UTF-8"?>
<root:AdaptorCompositionDefinition
  <!-- Omitted -->
  xmlns:map="http://www.hitachi.co.jp/soft/xml/sdp/adaptor/definition/callback/
MappingDefinition">
  <!-- Omitted -->

  <!-- CB definition for editing -->
  <cb:DataEditCBDefinition
class="jp.co.Hitachi.soft.sdp.adaptor.callback.dataedit.mapping.InputMappingCBImpl"
name="editor1">
  <!-- Mapping definition -->
  <map:MappingDefinition ioType="INPUT">
    <map:source/>
    <map:target>
      <map:records>
        <map:record name="RECORD1">
          <map:field name="F21" type="STRING"/>
          <map:field name="F22" type="TIMESTAMP"/>
          <map:field name="F23" type="LONG"/>
        </map:record>
      </map:records>
    </map:target>
    <map:intermediate>
      <map:mappings source="RECORD0" target="RECORD1">
        <map:map function="regexsubstring" argument1="F11"
          argument2=".*Data=([0-9]+).*" target="F21"/>
        <map:map function="getTupleTime" target="F22"/>
        <map:map function="subTime" argument1="F12" argument2="F13" target="F23"/>
      </map:mappings>
    </map:intermediate>
  </map:MappingDefinition>
</cb:DataEditCBDefinition>

```

### 9.11.3 Filter definition

You specify a filter definition (`FilterDefinition` tag) as a child element of a CB definition for editing (`DataEditCBDefinition` tag) described in 9.9.3 *CB definition for editing*.

For details about record filtering processing, see 10.4 *Record filtering*.

#### (1) Format

```

<FilterDefinition>
  <record source="name-of-record-to-be-filtered"
    conditionName="record-condition-name" condition="{AND|OR}">
    <field source="field-name"
      condition="{eq|ge|gt|le|lt|ne}" value="condition-value"/>
  </record>
</FilterDefinition>

```

#### (2) Details of definition

`FilterDefinition` tag (all definition information)



Defines all filter definition information. You specify this definition only once.

`record` tag (definition of record condition)

Defines a record condition. You can specify a maximum of 10 record condition definitions. This definition is mandatory.

A *record condition* is one of the conditions for selecting records to be filtered. For the record condition, specify the records to be filtered.

`source="name-of-record-to-be-filtered"`

Specifies, as 1 to 100 single-byte alphanumeric characters and the underscore (`_`), the name of a record to be filtered, which must be in the record format passed from the callback that was processed before the record filtering. This record name must begin with a single-byte alphabetic character.

You can specify this attribute only once.

`conditionName="record-condition-name"`

Specifies a name for the record condition, as 1 to 100 single-byte alphanumeric characters and the underscore (`_`). This name must begin with a single-byte alphabetic character.

You specify this definition only once. The specified name must be unique within the `FilterDefinition` tag.

`condition="{AND|OR}"`

Specifies the logical operation for the record condition. If this attribute is omitted, `AND` is assumed.

The permitted values are as follows:

- `AND`

Outputs as the results the records that contain a field that satisfies all the field conditions.

- `OR`

Outputs as the results the records that contain a field that satisfies any of the field conditions.

`field` tag (definition of field condition)

Specifies a field condition. You can specify a maximum of 10 field condition definitions. This definition is mandatory.

A field condition is one of the conditions for selecting records to be filtered. You specify for a field condition the name of a field in the records to be filtered, a comparison operator, and a condition value.

`source="field-name"`

Specifies the name of a field, which must have the record format specified for `source` in the `record` tag, expressed as 1 to 100 single-byte alphanumeric characters and the underscore (`_`). This field name must begin with a single-byte alphabetic character.

This attribute cannot be omitted.

`condition="{eq|ge|gt|le|lt|ne}"`

Specifies the value that represents the desired comparison operator for the field condition.

The specifiable values depend on whether character data or numeric data is specified in the `value` attribute in the `field` tag, as shown in the table below. If this attribute is omitted, `eq` is assumed.

Value of this attribute	Comparison operator	Value of value attribute	
		Character data	Numeric data
<code>eq</code>	<code>=</code>	Y	Y
<code>ge</code>	<code>&gt;=</code>	N	Y
<code>gt</code>	<code>&gt;</code>	N	Y
<code>le</code>	<code>&lt;=</code>	N	Y
<code>lt</code>	<code>&lt;</code>	N	Y
<code>ne</code>	<code>!=</code>	Y	Y

Legend:

Y: Can be specified

N: Cannot be specified

The table below explains the comparison operators.

Comparison operator	Usage example	Meaning
<code>=</code>	<code>A = B</code>	A is equal to B
<code>&gt;=</code>	<code>A &gt;= B</code>	A is equal to or greater than B
<code>&gt;</code>	<code>A &gt; B</code>	A is greater than B
<code>&lt;=</code>	<code>A &lt;= B</code>	A is equal to or less than B
<code>&lt;</code>	<code>A &lt; B</code>	A is less than B

Comparison operator	Usage example	Meaning
!=	A != B	A is not equal to B

value="*condition-value*"

Specifies the condition value for this field condition, expressed as character data or numeric data. The permitted characters for character and numeric data are shown below. This attribute cannot be omitted.

- For character data

You can specify 1 to 128 characters.

You can use a regular expression for the character string. Because the `java.util.regex.Pattern` class is used to analyze the regular expression, you must specify a regular expression that is within the range supported by the `java.util.regex.Pattern` class.

To use a character that has a special meaning in a regular expression, you must use the backslash (`\`) as an escape character.

The following are the characters that have a special meaning in regular expressions:

(, ), [, ], ., \*, ?, +, ^, \$

- For numeric data

You can specify an integer in the range from

-9223372036854775808 to 9223372036854775807.

**(3) Example**

```

<?xml version="1.0" encoding="UTF-8"?>
<root:AdaptorCompositionDefinition
  <!-- Omitted -->
  xmlns:filter="http://www.hitachi.co.jp/soft/xml/sdp/adaptor/definition/callback/
  FilterDefinition">
  <!-- Omitted -->

  <!-- CB definition for editing -->
  <cb:DataEditCBDefinition
  class="jp.co.Hitachi.soft.sdp.adaptor.callback.dataedit.filter.FilterCBImpl"
  name="filter1">
    <!-- Filter definition -->
    <filter:FilterDefinition>
      <!-- Definition of record condition -->
      <filter:record source="R1" conditionName="filterName1" condition="AND">
        <!-- Definition of field conditions -->
        <filter:field source="F11" condition="ge" value="1"/>
        <filter:field source="F11" condition="le" value="100"/>
        <filter:field source="F21" condition="eq" value=".*MIKE.*"/>
      </filter:record>
      <!-- Definition of record condition -->
      <filter:record source="R2" conditionName="filterName2" operator="OR">
        <!-- Definition of field conditions -->
        <filter:field source="F21" value="1"/>
        <filter:field source="F22" condition="ne" value=".*CATHY.*"/>
      </filter:record>
    </filter:FilterDefinition>
  </cb:DataEditCBDefinition>

```

**9.11.4 Record extraction definition**

You specify a record extraction definition (`RecordExtractionDefinition` tag) as a child element of a CB definition for editing (`DataEditCBDefinition` tag) described in *9.9.3 CB definition for editing*.

For details about record extraction processing, see *10.5 Record extraction*.

**(1) Format**

```

<RecordExtractionDefinition>
  <targetrecords>
    <targetrecord name="name-of-record-to-be-extracted">
      <record source="record-name"
        timeposition="time-field-name" condition="{AND|OR}">
        <field source="field-name"
          condition="{eq|ge|gt|le|lt|ne}" value="condition-value"/>
        </record>
      </targetrecord>
    </targetrecords>
  <extractions size="maximum-number-of-records-to-be-retained" timeout="{ON|OFF}"
    samerecord="{overwrite|delete}"
    <extraction name="extraction-condition-name" timelimit="timeout-value">
      <targets>
        <target sourceL="name-of-record-to-be-extracted"
          sourceR="name-of-record-to-be-extracted" condition="AND">
          <fieldcondition sourceL="field-name"
            condition="eq" sourceR="field-name" />
          </target>
        </targets>
      <extractrecord name="name-of-retrieved-record">
        <select source="name-of-record-to-be-extracted" />
      </extractrecord>
      <timeoutrecord name="timeout-record-name" />
    </extraction>
  </extractions>
</RecordExtractionDefinition>

```

**(2) Details of definition**

`RecordExtractionDefinition` tag (all definition information)

Defines all record extraction definition information. You specify this definition only once.

`targetrecords` tag (definition of extraction target condition group)

Defines information about a group of extraction target conditions. An extraction target condition is a single condition for selecting a record to be extracted; it consists of a record condition and field conditions. You specify for the record condition (`record` tag) the name of a record to be extracted, and you specify for the field conditions (`field` tag) field values in the record to be extracted.

`targetrecord` tag (definition of extraction target condition)

Defines an extraction target condition. You can specify a maximum of 10 of these definitions. This definition is mandatory.

`name="name-of-record-to-be-extracted"`

Specifies a name for an extracted record, as 1 to 100 single-byte alphanumeric characters and the underscore (`_`). This record name must

begin with a single-byte alphabetic character. You can specify this definition only once for an extracted record.

The record name specified in this attribute is also specified in the `source` attribute in the `select` tag and the `sourceL` and `sourceR` attributes in the `target` tag.

#### `record` tag (definition of record condition)

Defines a record condition. You can specify this definition only once.

`source="record-name"`

Specifies, as 1 to 100 single-byte alphanumeric characters and the underscore (`_`), the name of a record to be extracted, which must be in the record format passed from the callback that was processed before the record extraction. This attribute cannot be omitted.

`timeposition="time-field-name"`

Specifies a field of the `TIMESTAMP` data type that contains time information, as 1 to 100 single-byte alphanumeric characters and the underscore (`_`). This attribute cannot be omitted.

`condition="{AND|OR}"`

Specifies the logical operation for the record condition. If this attribute is omitted, `AND` is assumed.

The permitted values are as follows:

- `AND`

Outputs in the results the record if it contains a field that satisfies all the field conditions.

- `OR`

Outputs in the results the record if it contains a field that satisfies any of the field conditions.

#### `field` tag (definition of field condition)

Specifies a field condition. You can specify a maximum of 10 field condition definitions. This definition is mandatory.

Each field condition applies to a single field of the record to be extracted.

`source="field-name"`

Specifies the name of a field, which must have the record format specified for `source` in the `record` tag, expressed as 1 to 100 single-byte alphanumeric characters and the underscore (`_`). This attribute cannot be omitted.

```
condition="{eq|ge|gt|le|lt|ne}"
```

Specifies the value that represents the desired comparison operator for the field condition.

The specifiable values depend on whether character data or numeric data is specified in the `value` attribute in the `field` tag, as explained below. If this attribute is omitted, `eq` is assumed.

- When character data is specified  
You can specify `eq` or `ne`.
- When numeric data is specified  
You can specify `eq`, `ge`, `gt`, `le`, `lt`, or `ne`.

For the meanings of the permitted values, see the description of the `condition` attribute in the `field` tag in *9.11.3 Filter definition*.

```
value="condition-value"
```

Specifies the condition value for the field condition, expressed as character data or numeric data. The permitted characters for character and numeric data are shown below. This attribute cannot be omitted.

- For character data  
You can specify 1 to 128 characters.

You can use a regular expression for the character string. Because the `java.util.regex.Pattern` class is used to analyze the regular expression, you must specify a regular expression that is within the range supported by the `java.util.regex.Pattern` class.

To use a character that has a special meaning in a regular expression, you must use the backslash (`\`) as an escape character.

The following are the characters that have a special meaning in regular expressions:

```
(, ), [, ], ., *, ?, +, ^, $
```

- For numeric data  
You can specify an integer in the range from `-9223372036854775808` to `9223372036854775807`.

`extractions` tag (definition of extraction condition group)

Defines information about a group of extraction conditions. An extraction condition is one of the conditions for selecting a record to be extracted. This definition is mandatory.

`size="maximum-number-of-records-to-be-retained"`

Specifies the maximum number of records to be retained in the record buffer, as an integer from 1 to 1000000. If this attribute is omitted, 10000 is assumed.

`timeout="{ON|OFF}"`

Specifies whether timed-out records in the record buffer are to be output from the record buffer when the time specified in the `timelimit` attribute in the `extraction` tag has elapsed (such records are called *timeout records*).

If this attribute is omitted, OFF is assumed.

The permitted values are as follows:

- ON

Outputs the timeout records in the event of a timeout.

- OFF

Does not output the timeout records in the event of a timeout.

You must use the `timeoutrecord` tag to define a record name for the timeout records.

`samerecord="{overwrite|delete}"`

Specifies how to handle the record when the same record to be extracted is input more than once. If this attribute is omitted, `overwrite` is assumed.

The permitted values are as follows:

- `overwrite`

Processing depends on the location of the record existing in the record buffer that is determined to be the same as the record being input.

If it is the last record in the buffer, it is overwritten by the record being input. If it is not the last record in the buffer, all records in the record buffer are discarded and then the input record is retained.

- `delete`

Discards the record that was input.

`extraction` tag (definition of extraction condition)

Defines an extraction condition. You can specify a maximum of 10 of these definitions. This definition is mandatory.



`name="extraction-condition-name"`

Specifies a name for this extraction condition, as 1 to 100 single-byte alphanumeric characters and the underscore (`_`). This condition name must begin with a single-byte alphabetic character.

You can specify this definition only once.

`timelimit="timeout-value"`

Specifies a lifespan (in milliseconds) for records in the record buffer, as an integer from 1 to 3600000 (1 millisecond to 1 hour). When a record in the record buffer reaches the end of the specified lifespan, it becomes a timed-out record.

If this attribute is omitted, 10000 is assumed.

`targets` tag (definition of a group of conditions between records)

Defines a group of conditions between records.

A *condition between records* is one of the extraction conditions to be checked. It consists of a record input order (specified in the `target` tag) and a field name for comparing field values in records (specified in the `fieldcondition` tag).

`target` tag (definition of condition between records)

Specifies a record input order as a condition between records. You can specify a maximum of 9 of these definitions. This definition is mandatory.

You define in this definition a records input order by specifying record names in the `sourceL` and `sourceR` attributes. For example, the coding below specifies the records input order as `R1 → R2 → R3`. Records entered in this order will satisfy this records condition.

```

:
<rex:target sourceL="R1" sourceR="R2">
:
<rex:target sourceL="R2" sourceR="R3">
:

```

`sourceL="name-of-record-to-be-extracted"`

Specifies the record name for the left-hand operand of the condition between records, as 1 to 100 single-byte alphanumeric characters and the underscore (`_`). This record name must begin with a single-byte alphabetic character.

`sourceR="name-of-record-to-be-extracted"`

Specifies the record name for the right-hand operand of the condition between records, as 1 to 100 single-byte alphanumeric characters and the underscore (`_`). This record name must begin with a single-byte alphabetic

character.

condition="AND"

Specifies the logical operation for the condition between records. If this attribute is omitted, AND is assumed.

The only permitted value is as follows:

- AND

Outputs the records that satisfy all the conditions between records.

fieldcondition tag (field definition)

Specifies field names as part of the condition between records. You can specify a maximum of 10 field definitions. This definition is mandatory.

You specify in this definition the names of fields in the records specified in the sourceL and sourceR attributes in the target tag. For example, the coding below compares the value of field F11 in record R1 with the value of field F21 in record R2. If the value of field F11 in record R1 matches the value of field F21 in record R2, the condition between records is determined to be satisfied.

```

:
<rex:target sourceL="R1" sourceR="R2">
<rex:fieldcondition sourceL="F11" condition="eq" sourceR="F21"/>
:

```

sourceL="*field-name*"

Specifies the name of a field in the record that is specified in the sourceL attribute in the target tag, expressed as 1 to 100 single-byte alphanumeric characters and the underscore (\_). This field name must begin with a single-byte alphabetic character.

condition="eq"

Specifies the value that indicates the desired comparison operator for the record condition. For both character data and numeric data, the only permissible value is eq, which indicates the comparison operator =. If this attribute is omitted, eq is assumed.

sourceR="*field-name*"

Specifies the name of a field in the record that is specified in the sourceR attribute in the target tag, expressed as 1 to 100 single-byte alphanumeric characters and the underscore (\_). This field name must begin with a single-byte alphabetic character.

`extractrecord` tag (retrieved record definition)

Defines a retrieved record.

A *retrieved record* is a new record created by joining the records that satisfy the extraction conditions.

`name="name-for-retrieved-record"`

Specifies a name for the retrieved record, as 1 to 100 single-byte alphanumeric characters and underscore (`_`). This record name must begin with a single-byte alphabetic character, and must be unique within the standard adaptor.

`select` tag (definition of record to be extracted)

Defines a record to be joined in order to create the retrieved record. You can specify a maximum of 10 of these definitions. This definition is mandatory.

`source="name-of-record-to-be-extracted"`

Specifies a record name that was specified in a `name` attribute in the `targetrecord` tag, as 1 to 100 single-byte alphanumeric characters and underscore (`_`). This attribute cannot be omitted.

`timeoutrecord` tag (timeout record definition)

Defines the record to be output in the event of a timeout. This definition is mandatory when `ON` is specified in the `timeout` attribute in the `extractions` tag. This definition is ignored when `OFF` is specified in the `timeout` attribute.

`name="timeout-record-name"`

Specifies a name for the record that is to be output, as 1 to 100 single-byte alphanumeric characters and the underscore (`_`). This attribute cannot be omitted. The specified timeout record name must be unique within the standard adaptor.

**(3) Example**

```

<?xml version="1.0" encoding="UTF-8"?>
<root:AdaptorCompositionDefinition
  xmlns:rex ="http://www.hitachi.co.jp/soft/xml/sdp/adaptor/definition/callback/
RecordExtractionDefinition">
  <!-- Omitted -->

  <!-- CB definition for editing -->
  <cb:DataEditCBDefinition
class="jp.co.Hitachi.soft.sdp.adaptor.callback.dataedit.recordextract.RecordExtracti
onCBImpl" name="extractor1">
  <!-- Record extraction definition -->
  <rex:RecordExtractionDefinition>
    <!-- Definition of extraction target condition group -->
    <rex:targetrecords>
      <rex:targetrecord name="element1">
        <rex:record source="R1" timeposition="F11" condition="AND">
          <rex:field source="F12" condition="eq" value="MIKE"/>
          <rex:field source="F13" condition="gt" value="100"/>
        </rex:record>
      </rex:targetrecord>
      <rex:targetrecord name="element2">
        <rex:record source="R2" timeposition="F21" condition="OR">
          <rex:field source="F22" condition="eq" value="JAKE"/>
          <rex:field source="F23" condition="le" value="50"/>
        </rex:record>
      </rex:targetrecord>
    </rex:targetrecords>
    <!-- Definition of extraction condition group -->
    <rex:extractions size="100000" timeout="ON" samerecord="overwrite">
      <rex:extraction name="T1" timelimit="10000">
        <rex:targets>
          <rex:target sourceL="element1" sourceR="element2" condition="AND">
            <rex:fieldcondition sourceL="F13" condition="eq" sourceR="F23"/>
            <rex:fieldcondition sourceL="F14" condition="eq" sourceR="F24"/>
          </rex:target>
        </rex:targets>
        <!-- Retrieved record definition -->
        <rex:extractrecord name="ER1">
          <rex:select source="element1"/>
          <rex:select source="element2"/>
        </rex:extractrecord>
        <!-- Timeout record definition -->
        <rex:timeoutrecord name="TIMEOUT"/>
      </rex:extraction>
    </rex:extractions>
  </rex:RecordExtractionDefinition>
</cb:DataEditCBDefinition>

```

This coding example defines the following information.

- Record format subject to extraction

R1, R2

**■ Field structure for R1 and R2**

- For R1

F11 (TIMESTAMP type), F12 (STRING type), F13 (INT type), F14 (INT type)

- For R2

F21 (TIMESTAMP type), F22 (STRING type), F23 (INT type), F24 (INT type)

**■ Extraction target condition**

- For R1

F12 is MIKE, and F13 is greater than 100.

- For R2

F22 is JAKE, or F23 is 50 or less.

**■ Extraction conditions**

Records R1 and R2 are input in this order.

The value F13 in R1 is equal to the value of F23 in R2.

The value F14 in R1 is equal to the value of F24 in R2.

**■ Retrieved record**

A record that joins R1 and R2 is created.

**■ Timeout**

The timeout value for records is 10 seconds. If a timeout occurs, a record named TIMEOUT is output.

## 9.12 CB definitions for sending and receiving in the adaptor configuration definition file

This section discusses the CB definition for sending and the CB definition for receiving in the adaptor configuration definition file (`AdaptorCompositionDefinition.xml`).

You specify an input stream definition (`streamInfo` tag) as a child element of a CB definition for sending (`SendCBDefinition` tag) described in 9.9.4 *CB definition for sending*, and you specify an output stream definition (`streamInfo` tag) as a child element of a CB definition for receiving (`ReceiveCBDefinition` tag) described in 9.9.5 *CB definition for receiving*.

The table below lists and describes the CB definitions for sending and receiving.

Table 9-14: List of CB definitions for sending and receiving

No.	CB definition for editing	Parent element	Subsection
1	Input stream definition ( <code>streamInfo</code> tag)	CB definition for sending	9.12.1
2	Output stream definition ( <code>streamInfo</code> tag)	CB definition for receiving	9.12.2

### 9.12.1 Input stream definition

You specify in a CB definition for sending information about the input streams at the receiving end.

#### (1) Format

```
<streamInfo name="input-stream-name"
  querygroup="query-group-name" />
```

#### (2) Details of definition

You can specify a maximum of 1,024 input stream definitions. This definition is mandatory.

`streamInfo` tag (all definition information)

Defines all input stream definition information.

`name="input-stream-name"`

Specifies the name of an input stream, as 1 to 100 single-byte alphanumeric characters and the underscore (`_`). This name must begin with a single-byte alphabetic character. This input stream name must be unique within the CB

definition for sending. This attribute cannot be omitted.

Note that stream names specified in the `register stream` clause in the query definition file must be used as the input stream names specified in the adaptor configuration definition file. For details about the `register stream` clause, see the manual *uCosminexus Stream Data Platform - Application Framework Application Development Guide*.

```
querygroup="query-group-name"
```

Specifies a query group name, as 1 to 64 single-byte alphanumeric characters and the underscore (`_`). This name must begin with a single-byte alphabetic character. This attribute cannot be omitted.

## 9.12.2 Output stream definition

You specify in a CB definition for receiving information about the output streams at the sending end.

### (1) Format

```
<streamInfo name="output-stream-name"
  querygroup="query-group-name" />
```

### (2) Details of definition

You can specify a maximum of 1,024 output stream definitions. This definition is mandatory.

`streamInfo` tag (all definition information)

Defines all input stream definition information.

```
name="output-stream-name"
```

Specifies the name of an output stream, as 1 to 100 single-byte alphanumeric characters and the underscore (`_`). This name must begin with a single-byte alphabetic character and must be unique within the CB definition for receiving. This attribute cannot be omitted.

Note that query names specified in the `register query` clause in the query definition file must be used as the output stream names specified in the adaptor configuration definition file. For details about the `register query` clause, see the manual *uCosminexus Stream Data Platform - Application Framework Application Development Guide*.

```
querygroup="query-group-name"
```

Specifies a query group name, as 1 to 64 single-byte alphanumeric characters and the underscore (`_`). This name must begin with a single-byte alphabetic character. This attribute cannot be omitted.

## 9.13 Coding examples for an adaptor configuration definition file

This section presents coding examples for an adaptor configuration definition file.

### 9.13.1 Example 1

This subsection presents and explains the coding example for an adaptor configuration definition file that is provided in the following sample file:

*installation-directory\samples\file\conf\xml\AdaptorCompositionDefinition.xml*

#### (1) Coding example of adaptor configuration definition file

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- All Rights Reserved. Copyright (C) 2010, Hitachi, Ltd. -->
<root:AdaptorCompositionDefinition
  xmlns:root="http://www.hitachi.co.jp/soft/xml/sdp/adaptor/
definition"
  xmlns:cmn="http://www.hitachi.co.jp/soft/xml/sdp/adaptor/
definition/common"
  xmlns:adp="http://www.hitachi.co.jp/soft/xml/sdp/adaptor/
definition/adaptor"
  xmlns:cb="http://www.hitachi.co.jp/soft/xml/sdp/adaptor/
definition/callback"
  xmlns:ficon="http://www.hitachi.co.jp/soft/xml/sdp/adaptor/
definition/callback/FileInputConnectorDefinition"
  xmlns:focon="http://www.hitachi.co.jp/soft/xml/sdp/adaptor/
definition/callback/FileOutputConnectorDefinition"
  xmlns:form="http://www.hitachi.co.jp/soft/xml/sdp/adaptor/
definition/callback/FormatDefinition"
  xmlns:map="http://www.hitachi.co.jp/soft/xml/sdp/adaptor/
definition/callback/MappingDefinition">

  <!-- Common definition -->
  <cmn:CommonDefinition>
    <!-- Adaptor trace definition -->
    <cmn:AdaptorTraceDefinition trace="OFF"/>
  </cmn:CommonDefinition>

  <!-- ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ In-process group definition ↓ ↓ ↓ ↓ ↓ ↓
↓ ↓ -->
  <!-- In-process group definition 1 -->
  <adp:InprocessGroupDefinition name="InprocessAPTtest">
    <!-- ↓ ↓ ↓ ↓ ↓ ↓ ↓ Input adaptor definition (definable more than once) ↓ ↓
↓ ↓ ↓ ↓ -->
    <!-- Input adaptor definition 1 -->
```



```

<adp:InputAdaptorDefinition name="InputAdaptor1"
  interval="0" charCode="MS932" lineFeed="CR_LF">

  <!-- CB definition for input -->
  <cb:InputCBDefinition
class="jp.co.Hitachi.soft.sdp.adaptor.callback.io.FileInputCBImpl" name="inputer">
  <!-- Input connector definition -->
  <ficon:FileInputConnectorDefinition>
    <ficon:input readType="REAL_TIME" interval="1000"
retryCount="100" readUnit="1"/>
    <ficon:file path="C:\temp\input\Inprocess\"
name="Inprocess_Data1.csv" messageLog="OFF"/>
  </ficon:FileInputConnectorDefinition>
  </cb:InputCBDefinition>

  <!-- CB definition for data editing -->
  <cb:DataEditCBDefinition
class="jp.co.Hitachi.soft.sdp.adaptor.callback.dataedit.format
translate.InputFormatTranslatorCBImpl" name="editor1">
  <!-- Format conversion definition -->
  <form:FormatDefinition ioType="INPUT">
    <form:common>
      <form:unmatchedFormat>ERROR</form:unmatchedFormat>
    </form:common>
    <form:records>
      <form:record name="RECORD0" exp="($_name), ($_num)">
        <form:field name="name" type="STRING"/>
        <form:field name="num" type="LONG"/>
      </form:record>
    </form:records>
  </form:FormatDefinition>
  </cb:DataEditCBDefinition>

  <!-- CB definition for data editing -->
  <cb:DataEditCBDefinition
class="jp.co.Hitachi.soft.sdp.adaptor.callback.dataedit.mapping
.InputMappingCBImpl" name="editor3">
  <!-- Mapping definition -->
  <map:MappingDefinition ioType="INPUT">
    <map:source/>
    <map:target>
      <map:streams>
        <map:stream name="DATA0"
querygroup="Inprocess_QueryGroupTest">
          <map:column name="name" type="STRING"/>
          <map:column name="num" type="LONG"/>
        </map:stream>

```

```

    </map:streams>
  </map:target>
  <map:intermediate>
    <map:mappings source="RECORD0"
querygroup="Inprocess_QueryGroupTest" target="DATA0">
      <map:map source="name" target="name"/>
      <map:map source="num" target="num"/>
    </map:mappings>
  </map:intermediate>
</map:MappingDefinition>
</cb:DataEditCBDefinition>

  <!-- CB definition for sending -->
  <cb:SendCBDefinition
class="jp.co.Hitachi.soft.sdp.adaptor.callback.sendreceive.Send
dCBImpl" name="sender">
    <cb:streamInfo name="DATA0"
querygroup="Inprocess_QueryGroupTest"/>
  </cb:SendCBDefinition>

</adp:InputAdaptorDefinition>
<!-- ↑ ↑ ↑ ↑ ↑ ↑ Input adaptor definition (definable more than once) ↑ ↑
↑ ↑ ↑ ↑ -->

<!-- ↓ ↓ ↓ ↓ ↓ ↓ Output adaptor definition (definable more than once) ↓
↓ ↓ ↓ ↓ ↓ -->
<adp:OutputAdaptorDefinition name="OutputAdaptor1"
charCode="MS932" lineFeed="CR_LF">

  <!-- CB definition for receiving -->
  <cb:ReceiveCBDefinition
class="jp.co.Hitachi.soft.sdp.adaptor.callback.sendreceive.Rec
eiveCBImpl" name="receiver">
    <cb:streamInfo name="FILTER1"
querygroup="Inprocess_QueryGroupTest"/>
  </cb:ReceiveCBDefinition>

  <!-- CB definition for data editing -->
  <cb:DataEditCBDefinition
class="jp.co.Hitachi.soft.sdp.adaptor.callback.dataedit.mappin
g.OutputMappingCBImpl" name="editor1">
    <!-- Mapping definition -->
    <map:MappingDefinition ioType="OUTPUT">
      <map:source>
        <map:streams>
          <map:stream name="FILTER1"
querygroup="Inprocess_QueryGroupTest">
            <map:column name="name" type="STRING"/>

```

```

        <map:column name="num" type="LONG"/>
    </map:stream>
</map:streams>
</map:source>
<map:target/>
<map:intermediate>
    <map:mappings source="FILTER1"
querygroup="Inprocess_QueryGroupTest" target="RECORD1">
        <map:map source="name" target="name"/>
        <map:map source="num" target="num"/>
    </map:mappings>
</map:intermediate>
</map:MappingDefinition>
</cb:DataEditCBDefinition>

<!-- CB definition for data editing -->
<cb:DataEditCBDefinition
class="jp.co.Hitachi.soft.sdp.adaptor.callback.dataedit.format
translate.OutputFormatTranslatorCBImpl" name="editor3">
    <!-- Format conversion definition -->
    <form:FormatDefinition ioType="OUTPUT">
        <form:common/>
        <form:records>
            <form:record name="RECORD1" exp="($_name), ($_num)">
                <form:field name="name" type="STRING"/>
                <form:field name="num" type="LONG"/>
            </form:record>
        </form:records>
    </form:FormatDefinition>
</cb:DataEditCBDefinition>

<!-- CB definition for output -->
<cb:OutputCBDefinition
class="jp.co.Hitachi.soft.sdp.adaptor.callback.io.FileOutputCB
Impl" name="outputer">
    <!-- Output connector definition -->
    <focon:FileOutputConnectorDefinition>
        <focon:output compositionType="WRAP_AROUND" maxNumber="256"
maxSize="1000"/>
        <focon:file path="C:\temp\output\Inprocess\" prefix="out"
addDate="OFF" extension="csv"/>
    </focon:FileOutputConnectorDefinition>
</cb:OutputCBDefinition>
</adp:OutputAdaptorDefinition>
<!-- ↑ ↑ ↑ ↑ ↑ ↑ Output adaptor definition (definable more than once) ↑
↑ ↑ ↑ ↑ ↑ -->

</adp:InprocessGroupDefinition>

```

```

<!-- ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ In-process group definition ↑ ↑ ↑ ↑ ↑ ↑
↑ ↑ -->

</root:AdaptorCompositionDefinition>

```

## (2) Details of the example

This example uses in-process connection to connect standard adaptors and the SDP server. Input adaptor `InputAdaptor1` inputs files, and output adaptor `OutputAdaptor1` outputs the stream data summary analysis results to a file.

Input adaptor `InputAdaptor1` performs the processing shown in the table below. The definitions used in the adaptor configuration definition file are shown in parentheses.

Type of callback	Callback processing
Callback for input (CB definition for input)	File input (file input connector definition)
Callback for editing (CB definition for editing)	Format conversion (format conversion definition)
	Mapping between record and stream (mapping definition)
Callback for sending (CB definition for sending)	Tuple transmission (input stream definition)

The following describes each definition for input adaptor `InputAdaptor1`.

- Details of the file input connector definition
  - Input data storage location: `C:\temp\input\Inprocess\`
  - File name: `Inprocess_Data1.csv`
  - Read processing mode: Real-time mode
  - Read units: One row of a record is read at a time
- Details of the format conversion definition
  - The input record output by the file input connector is converted to a common record.
- Details of the mapping definition
  - During mapping between record and stream, the common record obtained after format conversion is associated with the common record that corresponds to the input stream format.
- Details of the input stream definition

The common record obtained after mapping is converted to a tuple and then is sent to input stream `DATA0` in query group `Inprocess_QueryGroupTest`.

Output adaptor `OutputAdaptor1` performs the processing shown in the table below. The definitions used in the adaptor configuration definition file are shown in parentheses.

Type of callback	Callback processing
Callback for receiving (CB definition for receiving)	Tuple reception (output stream definition)
Callback for editing (CB definition for editing)	Mapping between record and stream (mapping definition)
	Format conversion (format conversion definition)
Callback for output (CB definition for output)	Output to file (file output connector definition)

The following describes each definition for output adaptor `OutputAdaptor1`.

- Details of the output stream definition

A tuple is received from output stream `FILTER1` in query group `Inprocess_QueryGroupTest` and then is converted to a common record.

- Details of the mapping definition

During mapping between record and stream, the common record that corresponds to the output stream format is associated with the common record that is input during format conversion.

- Details of the format conversion definition

The common record obtained after mapping is converted to an output record.

- Details of the file output connector definition

The output record obtained after format conversion is output to the file under the following conditions:

File structure: Wraparound structure

Output data storage location: `C:\temp\output\Inprocess\`

File names and their order of output: `out1.csv, out2.csv, ..., out256.csv`

- Stream names specified in the adaptor configuration definition file

The stream names specified in the adaptor configuration definition file must match the names specified in the query definition file.

- For input stream

Stream name `DATA0` specified in the `register stream` clause in the query definition file becomes the input stream name that is specified in the adaptor configuration definition file.

- For output stream

Query name `FILTER1` specified in the `register query` clause in the query definition file becomes the output stream name that is specified in the adaptor configuration definition file.

The following shows the query definition file for this example:

```
register stream DATA0(name VARCHAR(10), num BIGINT);

register query FILTER1 ISTREAM(SELECT * FROM DATA0 [ROWS 1] WHERE DATA0.NUM <=
24);
```

For details about the query definition file, see the manual *uCosminexus Stream Data Platform - Application Framework Application Development Guide*.

### 9.13.2 Example 2

This subsection presents and explains the coding example for an adaptor configuration definition file that is provided in the following sample file:

*installation-directory*\samples\httppacket\conf\xml\AdaptorCompositionDefinition.xml

#### (1) Coding example of adaptor configuration definition file

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- All Rights Reserved. Copyright (C) 2010, Hitachi, Ltd. -->
<root:AdaptorCompositionDefinition
  xmlns:root="http://www.hitachi.co.jp/soft/xml/sdp/adaptor/
definition"
  xmlns:cmn="http://www.hitachi.co.jp/soft/xml/sdp/adaptor/
definition/common"
  xmlns:adp="http://www.hitachi.co.jp/soft/xml/sdp/adaptor/
definition/adaptor"
  xmlns:cb="http://www.hitachi.co.jp/soft/xml/sdp/adaptor/
definition/callback"
  xmlns:hpicon="http://www.hitachi.co.jp/soft/xml/sdp/adaptor/
definition/callback/HttpPacketInputConnectorDefinition"
  xmlns:map="http://www.hitachi.co.jp/soft/xml/sdp/adaptor/
definition/callback/MappingDefinition"
  xmlns:filter="http://www.hitachi.co.jp/soft/xml/sdp/adaptor/
definition/callback/FilterDefinition"
  xmlns:rex="http://www.hitachi.co.jp/soft/xml/sdp/adaptor/
definition/callback/RecordExtractionDefinition"
  xmlns:docon="http://www.hitachi.co.jp/soft/xml/sdp/adaptor/
definition/callback/DashboardOutputConnectorDefinition"
```

```

>

<!-- Common definition -->
<cmn:CommonDefinition>
  <!-- Adaptor trace definition -->
  <cmn:AdaptorTraceDefinition trace="OFF"/>
</cmn:CommonDefinition>

<!-- ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓ In-process group definition ↓ ↓ ↓ ↓ ↓ ↓ ↓ ↓
↓ ↓ -->
<!-- In-process group definition 1 -->
<adp:InprocessGroupDefinition name="InprocessAPTtest">
  <!-- ↓ ↓ ↓ ↓ ↓ ↓ Input adaptor definition (definable more than once) ↓ ↓
↓ ↓ ↓ ↓ -->
  <!-- Input adaptor definition 1 -->
  <adp:InputAdaptorDefinition name="InputAdaptor1" interval="0"
charCode="MS932" lineFeed="CR_LF">

    <!-- CB definition for input -->
    <cb:InputCBDefinition
class="jp.co.Hitachi.soft.sdp.adaptor.callback.io.packetinput.
HttpPacketInputCBImpl" name="inputer">
      <!-- Packet input connector definition -->
      <hpicon:HttpPacketInputConnectorDefinition>
        <hpicon:input buffersize="4096" assemblingtime="2000">
          <hpicon:packetdata globalheader="24" packetheader="16"
packetoffset="8" packetlength="4" timeoffset="0"/>
          <hpicon:command path="C:\Program
Files\WinDump\WinDump.exe" parameter=" -i 1 -s 2048 -w - -n
&quot;tcp port 80 or port 8080&quot;"/>
        </hpicon:input >
        <hpicon:output unit="100">
          <hpicon:record name="REQUEST" type="REQUEST" >
            <hpicon:field name="SEND_IP"/>
            <hpicon:field name="RECEIVE_IP"/>
            <hpicon:field name="SEND_PORT"/>
            <hpicon:field name="RECEIVE_PORT"/>
            <hpicon:field name="TARGET_URI"/>
            <hpicon:field name="TIME"/>
          </hpicon:record >
          <hpicon:record name="RESPONSE" type="RESPONSE" >
            <hpicon:field name="SEND_IP"/>
            <hpicon:field name="RECEIVE_IP"/>
            <hpicon:field name="SEND_PORT"/>
            <hpicon:field name="RECEIVE_PORT"/>
            <hpicon:field name="TIME"/>
          </hpicon:record >
        </hpicon:output >
      </hpicon:HttpPacketInputConnectorDefinition>
    </cb:InputCBDefinition>
  </adp:InputAdaptorDefinition>
</adp:InprocessGroupDefinition>

```

```

    </hpicon:output>
  </hpicon:HttpPacketInputConnectorDefinition>
</cb:InputCBDefinition>

  <!-- CB definition for data editing -->
  <cb:DataEditCBDefinition
class="jp.co.Hitachi.soft.sdp.adaptor.callback.dataedit.filter
.FilterCBImpl" name="editor1">
  <!-- Filter definition -->
  <filter:FilterDefinition>
    <!-- Record condition -->
    <filter:record source="REQUEST"
conditionName="filterName1" condition="AND">
      <!-- Field condition -->
      <filter:field source="TARGET_URI" condition="eq"
value="http://\www.*"/>
    </filter:record>
  </filter:FilterDefinition>
</cb:DataEditCBDefinition>

  <!-- CB definition for data editing -->
  <cb:DataEditCBDefinition
class="jp.co.Hitachi.soft.sdp.adaptor.callback.dataedit.record
extract.RecordExtractionCBImpl" name="editor2">

  <!-- Record extraction definition -->
  <rex:RecordExtractionDefinition>
    <!-- Definition of a group of records to be extracted -->
    <rex:targetrecords>
      <rex:targetrecord name="element1">
        <rex:record source="REQUEST" timeposition="TIME"
condition="AND">
          <rex:field source="SEND_PORT" condition="ne" value="0"/>
        </rex:record>
      </rex:targetrecord>
      <rex:targetrecord name="element2">
        <rex:record source="RESPONSE" timeposition="TIME"
condition="AND">
          <rex:field source="RECEIVE_PORT" condition="ne"
value="0"/>
        </rex:record>
      </rex:targetrecord>
    </rex:targetrecords>
    <!-- Extraction condition group definition -->
    <rex:extractions size="100000" timeout="OFF"
samerecord="overwrite">
      <rex:extraction name="BIND_PACKET" timelimit="10000">
        <rex:targets>

```



```

        <rex:target sourceL="element1" sourceR="element2"
condition="AND">
        <rex:fieldcondition sourceL="SEND_IP"      condition="eq"
sourceR="RECEIVE_IP"/>
        <rex:fieldcondition sourceL="RECEIVE_IP"
condition="eq" sourceR="SEND_IP"/>
        <rex:fieldcondition sourceL="SEND_PORT"    condition="eq"
sourceR="RECEIVE_PORT"/>
        <rex:fieldcondition sourceL="RECEIVE_PORT"
condition="eq" sourceR="SEND_PORT"/>
        </rex:target>
</rex:targets>
<!-- Retrieved record definition -->
<rex:extractrecord name="BINDRECORD">
    <rex:select source="element1"/>
    <rex:select source="element2"/>
</rex:extractrecord>
</rex:extraction>
</rex:extractions>
</rex:RecordExtractionDefinition>
</cb:DataEditCBDefinition>

<!-- CB definition for data editing -->
<cb:DataEditCBDefinition
class="jp.co.Hitachi.soft.sdp.adaptor.callback.dataedit.mapping.
InputMappingCBImpl" name="editor3">
    <!-- Mapping definition -->
    <map:MappingDefinition ioType="INPUT">
        <map:source/>
        <map:target>
            <map:records>
                <map:record name="RESULT">
                    <map:field name="SEND_IP"      type="STRING"/>
                    <map:field name="RECEIVE_IP"   type="STRING"/>
                    <map:field name="SEND_PORT"    type="INT"/>
                    <map:field name="RECEIVE_PORT" type="INT"/>
                    <map:field name="URI"          type="STRING"/>
                    <map:field name="SUBTIME"     type="LONG"/>
                    <map:field name="TIME"        type="TIMESTAMP"/>
                </map:record>
            </map:records>
        </map:target>
        <map:intermediate>
            <map:mappings source="BINDRECORD" target="RESULT">
                <map:map source="element1_SEND_IP"      target="SEND_IP"/>
                <map:map source="element1_RECEIVE_IP"
target="RECEIVE_IP"/>
                <map:map source="element1_SEND_PORT"

```

```

target="SEND_PORT"/>
  <map:map source="element1_RECEIVE_PORT"
target="RECEIVE_PORT"/>
  <map:map source="element1_TARGET_URI" target="URI"/>
  <map:map function="subTime" argument1="element1_TIME"
argument2="element2_TIME" target="SUBTIME"/>
  <map:map source="element2_TIME" target="TIME"/>
  </map:mappings>
</map:intermediate>
</map:MappingDefinition>
</cb:DataEditCBDefinition>

<!-- CB definition for data editing -->
<cb:DataEditCBDefinition
class="jp.co.Hitachi.soft.adaptor.callback.dataedit.mappin
g.InputMappingCBImpl" name="editor4">
  <!-- Mapping definition -->
  <map:MappingDefinition ioType="INPUT">
    <map:source/>
    <map:target>
      <map:streams>
        <map:stream name="s1"
querygroup="Inprocess_QueryGroupTest">
          <map:column name="sendip" type="STRING"/>
          <map:column name="receiveip" type="STRING"/>
          <map:column name="sendport" type="INT"/>
          <map:column name="receiveport" type="INT"/>
          <map:column name="uri" type="STRING"/>
          <map:column name="subtime" type="LONG"/>
          <map:column name="time" type="TIMESTAMP"/>
        </map:stream>
      </map:streams>
    </map:target>
    <map:intermediate>
      <map:mappings source="RESULT"
querygroup="Inprocess_QueryGroupTest" target="s1">
        <map:map source="SEND_IP" target="sendip"/>
        <map:map source="RECEIVE_IP" target="receiveip"/>
        <map:map source="SEND_PORT" target="sendport"/>
        <map:map source="RECEIVE_PORT" target="receiveport"/>
        <map:map source="URI" target="uri"/>
        <map:map source="SUBTIME" target="subtime"/>
        <map:map source="TIME" target="time"/>
      </map:mappings>
    </map:intermediate>
  </map:MappingDefinition>
</cb:DataEditCBDefinition>

```

```

    <!-- CB definition for sending -->
    <cb:SendCBDefinition
class="jp.co.Hitachi.soft.sdp.adaptor.callback.sendreceive.Send
dCBImpl" name="sender">
    <cb:streamInfo name="s1"
querygroup="Inprocess_QueryGroupTest"/>
    </cb:SendCBDefinition>
    </adp:InputAdaptorDefinition>
    <!-- ↑ ↑ ↑ ↑ ↑ ↑ Input adaptor definition (definable more than once) ↑ ↑
↑ ↑ ↑ ↑ -->

    <!-- ↓ ↓ ↓ ↓ ↓ ↓ Output adaptor definition (definable more than once) ↓
↓ ↓ ↓ ↓ ↓ -->
    <adp:OutputAdaptorDefinition name="OutputAdaptor1"
charCode="MS932" lineFeed="CR_LF">

    <!-- CB definition for receiving -->
    <cb:ReceiveCBDefinition
class="jp.co.Hitachi.soft.sdp.adaptor.callback.sendreceive.Rec
eiveCBImpl" name="receiver">
    <cb:streamInfo name="q1"
querygroup="Inprocess_QueryGroupTest"/>
    </cb:ReceiveCBDefinition>

    <!-- CB definition for data editing -->
    <cb:DataEditCBDefinition
class="jp.co.Hitachi.soft.sdp.adaptor.callback.dataedit.mappin
g.OutputMappingCBImpl" name="editor1">
    <!-- Mapping definition -->
    <map:MappingDefinition ioType="OUTPUT">
    <map:source>
    <map:streams>
    <map:stream name="q1"
querygroup="Inprocess_QueryGroupTest">
    <map:column name="sendip" type="STRING"/>
    <map:column name="receiveip" type="STRING"/>
    <map:column name="sendport" type="INT"/>
    <map:column name="receiveport" type="INT"/>
    <map:column name="uri" type="STRING"/>
    <map:column name="subtime" type="LONG"/>
    <map:column name="time" type="TIMESTAMP"/>
    </map:stream>
    </map:streams>
    </map:source>
    <map:target/>
    <map:intermediate>
    <map:mappings source="q1"
querygroup="Inprocess_QueryGroupTest" target="RECORD1">

```

```

        <map:map source="sendip"          target="SEND_IP"/>
        <map:map source="receiveip"      target="RECEIVE_IP"/>
        <map:map source="sendport"       target="SEND_PORT"/>
        <map:map source="receiveport"    target="RECEIVE_PORT"/>
        <map:map source="uri"            target="URI"/>
        <map:map source="subtime"        target="SUBTIME"/>
        <map:map source="time"           target="TIME"/>
    </map:mappings>
</map:intermediate>
</map:MappingDefinition>
</cb:DataEditCBDefinition>

<!-- CB definition for data editing -->
<cb:DataEditCBDefinition
class="jp.co.Hitachi.soft.sdp.adaptor.callback.dataedit.mappin
g.InputMappingCBImpl" name="editor2">
    <!-- Mapping definition -->
    <map:MappingDefinition ioType="OUTPUT">
        <map:source/>
        <map:target>
            <map:records>
                <map:record name="RECORD2" >
                    <map:field name="SEND_IP"          type="STRING"/>
                    <map:field name="RECEIVE_IP"       type="STRING"/>
                    <map:field name="SEND_PORT"        type="INT"/>
                    <map:field name="RECEIVE_PORT"     type="INT"/>
                    <map:field name="URI"              type="STRING"/>
                    <map:field name="SUBTIME"          type="LONG"/>
                    <map:field name="TIME"             type="TIMESTAMP"/>
                    <map:field name="GET_TUPLE_TIME"   type="TIMESTAMP"/>
                </map:record>
            </map:records>
        </map:target>
    </map:intermediate>
    <map:mappings source="RECORD1" target="RECORD2">
        <map:map source="SEND_IP"          target="SEND_IP"/>
        <map:map source="RECEIVE_IP"       target="RECEIVE_IP"/>
        <map:map source="SEND_PORT"        target="SEND_PORT"/>
        <map:map source="RECEIVE_PORT"     target="RECEIVE_PORT"/>
        <map:map source="URI"              target="URI"/>
        <map:map source="SUBTIME"          target="SUBTIME"/>
        <map:map source="TIME"             target="TIME"/>
        <map:map function="getTupleTime" target="GET_TUPLE_TIME"/>
    >
    </map:mappings>
</map:intermediate>
</map:MappingDefinition>
</cb:DataEditCBDefinition>

```

```

    <!-- CB definition for output -->
    <cb:OutputCBDefinition
class="jp.co.Hitachi.soft.sdp.adaptor.callback.io.dashboard.Da
shboardOutputCBImpl" name="outputer">
    <!-- Dashboard output connector definition -->
    <docon:DashboardOutputConnectorDefinition Record="RECORD2">
    <docon:RecordHoldTime
        DateReference="LAST_UPDATE"
            RecordTime="300"
            DateFieldPosition="8" />
    </docon:DashboardOutputConnectorDefinition>
    </cb:OutputCBDefinition>
</adp:OutputAdaptorDefinition>
<!-- ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ Output adaptor definition (definable more than once) ↑
↑ ↑ ↑ ↑ ↑ -->

</adp:InprocessGroupDefinition>
<!-- ↑ ↑ ↑ ↑ ↑ ↑ ↑ ↑ In-process group definition ↑ ↑ ↑ ↑ ↑ ↑
↑ ↑ -->

</root:AdaptorCompositionDefinition>

```

## (2) Details of the example

This example uses in-process connection to connect standard adaptors and the SDP server. Input adaptor `InputAdaptor1` joins corresponding request information and response information in HTTP packet information and obtains the communication time from the difference between their time values. Output adaptor `OutputAdaptor1` acquires HTTP communication information and uses the dashboard output connector to output the information for the past 5 minutes.

Input adaptor `InputAdaptor1` performs the processing shown in the table below. The definitions used in the adaptor configuration definition file are shown in parentheses.

Type of callback	Callback processing
Callback for input (CB definition for input)	HTTP packet input (HTTP packet input connector definition)
Callback for editing (CB definition for editing)	Record filtering (filter definition)
	Record extraction (record extraction definition)
	Mapping between records (mapping definition)
	Mapping between record and stream (mapping definition)

Type of callback	Callback processing
Callback for sending (CB definition for sending)	Tuple transmission (input stream definition)

The following describes each definition for input adaptor `InputAdaptor1`.

■ Details of the HTTP packet input connector definition

This example uses `WinDump` as the packet analyzer. A command parameter is used to specify the following information regarding the computer to be analyzed (referred to hereafter as the server):

- Network device number: 1
- Port number used for HTTP protocol: 80 or 8080

The HTTP packet input connector is used to assemble records from the packets received by the server (request packets) and the packets that are sent (response packets). Each record retains the following fields:

- Request code  
Sending IP address, receiving IP address, sending port number, receiving port number, URI information, and the time the server received the packet
- Response code  
Sending IP address, receiving IP address, sending port number, receiving port number, and the time the server sent the packet

■ Details of the filter definition

Only those records whose URI in the request code is `http://\www.*` are acquired (where `*` represents a string of any number of characters).

■ Details of the record extraction definition

A retrieved record is created by joining a request record and a response record that satisfy the following conditions:

- The sending IP address in the request record is equal to the receiving IP address in the response record.
- The receiving IP address in the request record is equal to the sending IP address in the response record.
- The sending port number in the request record is equal to the receiving port number in the response record.
- The receiving port number in the request record is equal to the sending port number in the response record.

- Details of the mapping definition

- During the first mapping (between records), a record containing the following fields is created from the joined record obtained by record extraction:

Sending IP address, receiving IP address, sending port number, receiving port number, URI information (excluding parameters), communication response time (difference between the time the server received the packet and the time the server sent the packet), and the time the server sent the packet

You obtain the communication response time by specifying `subTime` in the `function` attribute.

- During the second mapping (between record and stream), the common record obtained from the first mapping is associated with the common record corresponding to the input stream format.

- Details of the input stream definition

The common record obtained after mapping is converted to a tuple and then is sent to input stream `s1` in query group `Inprocess_QueryGroupTest`.

Output adaptor `OutputAdaptor1` performs the processing shown in the table below. The definitions used in the adaptor configuration definition file are shown in parentheses.

Type of callback	Callback processing
Callback for receiving (CB definition for receiving)	Tuple reception (output stream definition)
Callback for editing (CB definition for editing)	Mapping between record and stream (mapping definition)
	Mapping between records (mapping definition)
Callback for output (CB definition for output)	Output to dashboard (dashboard output connector definition)

The following describes each definition for output adaptor `OutputAdaptor1`.

- Details of the output stream definition

A tuple is received from output stream `q1` in query group `Inprocess_QueryGroupTest` and then is converted to a common record.

- Details of the mapping definition

- During the first mapping (between record and stream), the common record corresponding to the output stream format is associated with the common

record corresponding to the next callback format.

- During the second mapping (between records), a record containing the following fields is created from the record obtained from the first mapping:

Sending IP address, receiving IP address, sending port number, receiving port number, URI information (excluding parameters), communication response time (difference between the time the server received the packet and the time the server sent the packet), the time the server sent the packet, and the time in the tuple output from the output stream

You obtain the time in the tuple output from the output stream by specifying `getTupleTime` in the `function` attribute.

#### ■ Details of the dashboard output connector definition

The records obtained after mapping are output to the dashboard under the following conditions:

- The last time the dashboard output connector received a tuple is the reference time.
- Records whose time is within 5 minutes (300 seconds) from the reference time are retained.
- The time information in a tuple is contained in the 8th field (`GET_TUPLE_TIME`).
- The RMI server's port number is 20421 (this is not specified because it is the default).

The connection name used to display data by Dashboard Viewer is `InprocessAPTTest/OutputAdaptor1/outputer`.

#### ■ Stream names specified in the adaptor configuration definition file

The stream names specified in the adaptor configuration definition file must match the names specified in the query definition file.

- For input stream

Stream name `s1` specified in the `register stream` clause in the query definition file becomes the input stream name that is specified in the adaptor configuration definition file.

- For output stream

Query name `q1` specified in the `register query` clause in the query definition file becomes the output stream name that is specified in the adaptor configuration definition file.

The following shows the query definition file for this example:



```
REGISTER STREAM s1(sendip VARCHAR(15),receiveip VARCHAR(15),sendport
INTEGER,receiveport INTEGER,uri VARCHAR(255),subtime BIGINT,times
TIMESTAMP(9));

REGISTER QUERY q1 RSTREAM(SELECT * FROM s1[RANGE 5 MINUTE]);
```

For details about the query definition file, see the manual *uCosminexus Stream Data Platform - Application Framework Application Development Guide*.



## Chapter

---

# 10. Details About Definitions in the Definition Files

---

This chapter provides details of the definitions specified in 8. *SDP Server Definition Files* and 9. *Adaptor Definition Files*, including data input and output, data editing functions, overviews of the processing flows, processing flow details, and required settings.

- 10.1 Organization of this chapter
- 10.2 File input
- 10.3 HTTP packet input
- 10.4 Record filtering
- 10.5 Record extraction
- 10.6 File output
- 10.7 Dashboard output
- 10.8 Timestamp adjustment for tuples

## 10.1 Organization of this chapter

This chapter provides an overview of data processing using the standard adaptors and discusses the processing flows and the settings in the definition files that are required for data processing. It also discusses the timestamp adjustment performed by the SDP server for standard and custom adaptors in order to change the timestamp in tuples.

The table below lists the functions discussed in this chapter and the sections where they are explained.

*Table 10-1:* List of functions and corresponding sections

No.	Function		Input adaptors	Output adaptors	Section
1	Functions supported by standard adaptors	File input	Y	N	10.2
2		HTTP packet input	Y	N	10.3
3		Record filtering	Y	Y	10.4
4		Record extraction	Y	N	10.5
5		File output	N	Y	10.6
6		Dashboard output	N	Y	10.7
7	Timestamp adjustment for tuples		Y	N	10.8

Legend:

Y: Applicable

N: Not applicable

## 10.2 File input

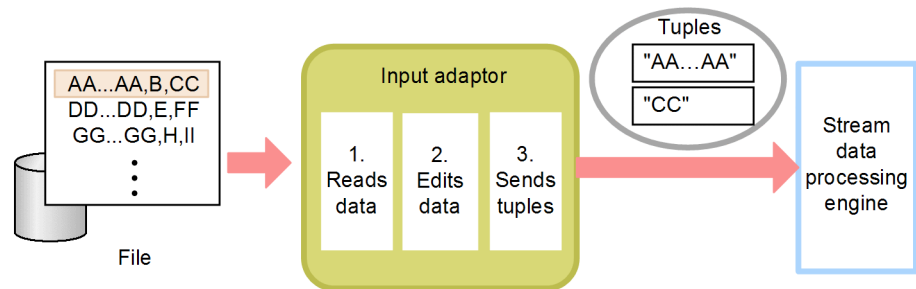
This section provides an overview of file input and discusses the flow of data processing and the settings in the definition files.

### 10.2.1 Overview of file input

File input processing involves reading data from files, such as error log and access log files, converting the data into a format that can be processed by the stream data processing engine, and then sending the data to the stream data processing engine. An input adaptor is used to perform file input processing.

The following figure provides an overview of file input.

*Figure 10-1: Overview of file input*

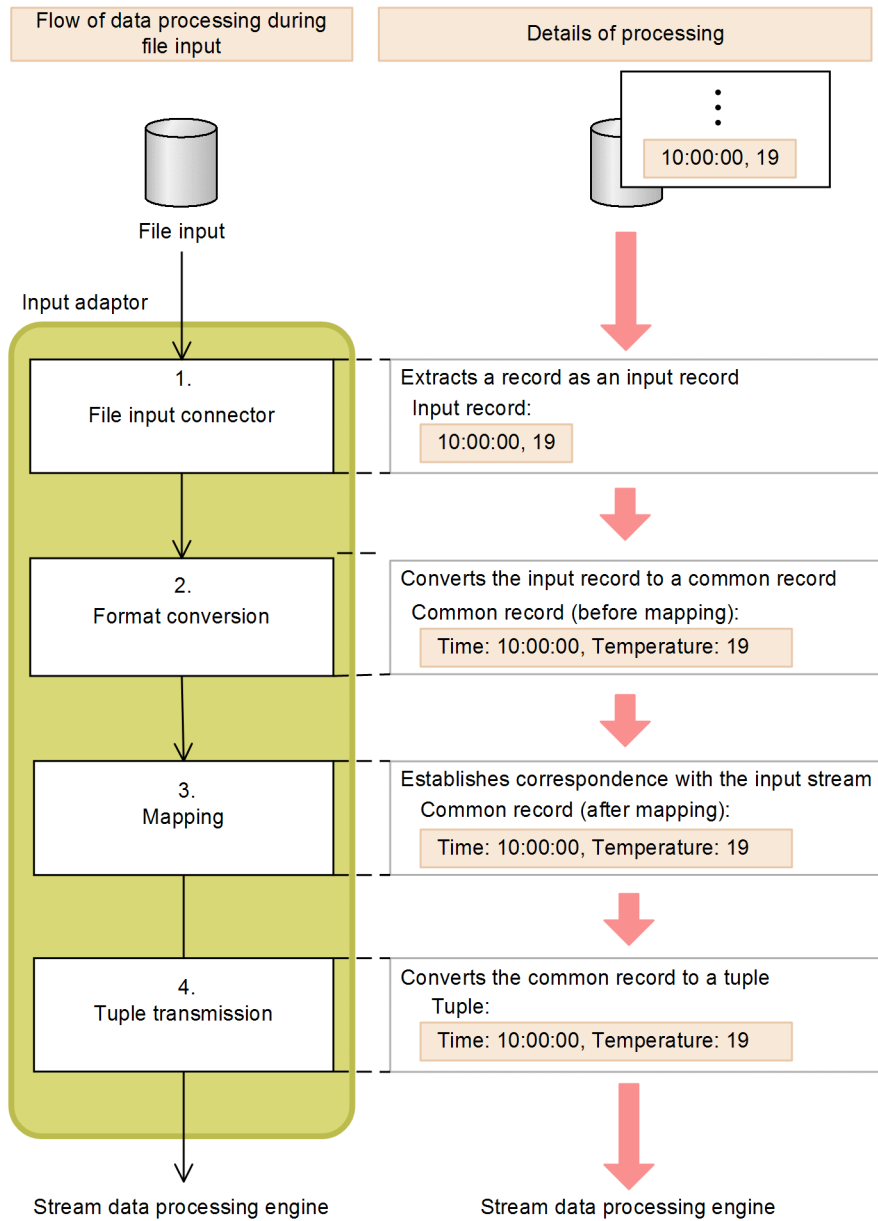


1. Reads data  
Uses a file input connector to read data from an input file.
2. Edits data  
Edits the data (performs format conversion and mapping).
3. Sends tuples  
Converts the edited data to tuples and sends the tuples to the stream data processing engine.

### 10.2.2 Flow of data processing during file input

The following figure shows the flow and details of data processing during file input by the input adaptor.

Figure 10-2: Flow and details of data processing during file input



For details about the data formats handled by an input adaptor, see (1) *Data formats handled by an input adaptor*. For details about the processing, see the subsections beginning with (2) *File input by a file input connector*.

**(1) Data formats handled by an input adaptor**

The data formats handled by an input adaptor are the input record and the common record. These data formats are discussed below.

*Input record*

An input record is a row of data acquired from an input file. The input adaptor handles one row of data as one input record.

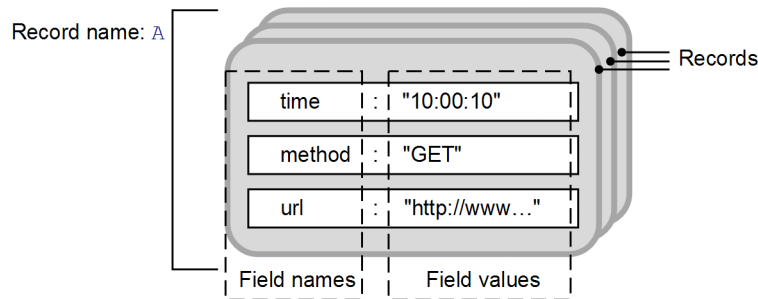
*Common record*

A common record is a set of data items consisting of multiple fields (field names and field values). A field name is a name assigned to a data segment (field) in the input record, and a field value is the value of the data segment.

A common record is in a standard data format that is handled internally by the input and output adaptors.

The standard adaptors manage multiple sets of a field name and a field value in the common record as a record structure and use a record name to identify each record structure.

Figure 10-3: Structure of a common record

**(2) File input by a file input connector**

You use the file input connector definition in the adaptor configuration definition file to define information about file input.

A *file input connector* reads data in rows from one or more files stored in the input file storage directory and converts the data into input records. One row of data read by the file input connector becomes one input record.

An input adaptor can read multiple input records at a time. The number of input records read by the file input connector is equal to the number of records that can be processed at one time for the format conversion, mapping, and tuple transmission operations.

This subsection discusses the types and structures of input files that are read by a file input connector.

### File types

An input file to be read by a file input connector must be a text file consisting of character data only. The records can be of variable length.

### File structure

The input files that are read by a file input connector have one of the structures described below.

File structure	Description	Prerequisite
Wraparound	Data is written to the files sequentially in the order the files were defined; there is a fixed number of input files. When all files are filled, data is written to the first file again.	The order of the files to which data is to be written is predetermined, and data is always written to the files in that order. To write data to a file that has become full, the file is first cleared of its data and then new data is written to it.
Non-wraparound	Data is written to the files in the order the files were defined; there is no fixed number of input files.	The order of the files to which data is to be written is predetermined, and data is always written to the files in that order.

The order of record and file creation must be chronological.

### File name

A file to be read by a file input connector is specified by its file name or sequence number in the `name` attribute in the `file` tag in the file input connector definition. For details about the `name` attribute, see *9.10.1 File input connector definition*.

### Order in which files are read

A file input connector reads files in the order the file names were specified in the `name` attribute in the `file` tag in the file input connector definition or in the order of the input file update times. You use the `readOrder` attribute in the `input` tag to specify which of these orders is to be used for reading files.

Note that if multiple input files have the same update time, the input order cannot be predicted.

### File read processing modes

The two modes of reading files by a file input connector are the *batch processing mode* and the *real-time processing mode*. The table below describes these processing modes. In both modes, the input adaptor stops automatically when read processing is completed.



Processing mode	Details of read processing	
	Reading method	Read processing completion condition
Batch processing mode	When the input adaptor starts, the input file storage directory is checked for any input files. If there are any input files, they are read.	Read processing is completed when either of the following conditions is satisfied: <ul style="list-style-type: none"> <li>• When the input adaptor starts, there are no unread files in the input file storage directory.</li> <li>• All files specified in the definition have been read.</li> </ul>
Real-time processing mode	The input file storage directory is monitored periodically at a specified interval and input files are read whenever they are detected. The input file storage directory monitoring interval and a monitoring count are specified in the <code>input</code> tag in the file input connector definition. When the specified monitoring count is exceeded, a warning message is issued and processing resumes.	Read processing is completed when either of the following conditions is satisfied: <ul style="list-style-type: none"> <li>• When input files are specified by sequence number: The file with the last sequence number has been read.</li> <li>• When input files are specified by file name: All files specified in the definition have been read.</li> </ul>

If an input file is copied to or moved to the input file storage directory while the file input connector is running, the file input connector goes onto standby for one second and then resumes file read processing. If the file input connector cannot resume file read processing after five consecutive standby operations, it issues the `KFSP46200-E` message.

If a new input file is created or data is added to an existing input file in the input file storage directory while the file input connector is running, the newly created file or the added records are not read.

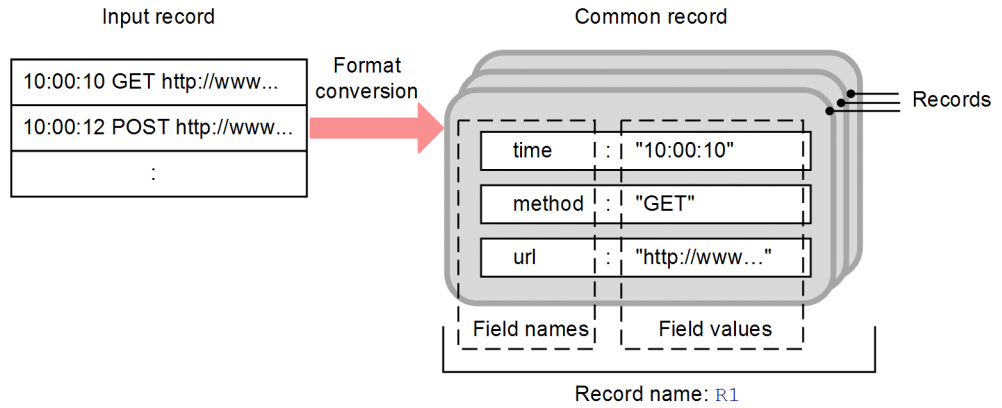
### (3) *Format conversion*

You define information about *format conversion* in the format conversion definition in the adaptor configuration definition file.

Format conversion involves segmenting an input record into fields and then converting the fields into a common record.

The figure below shows an example of format conversion from input record to common record. In this example, the input record consists of three fields delimited by the space, the format of field 1 is converted to the `TIME` type, and the format of fields 2 and 3 is converted to the character string type.

*Figure 10-4:* Example of format conversion from input record to common record



The table below shows the structure of the common record in the above example and the tags that are specified in the format conversion definition.

Record structure	Tag
Record name: R1 Record structure: (\$_time) Δ (\$_method) Δ (\$_url)	record tag (record definition)
Field name: time, Type: TIME	field tag (field definition)
Field name: method, Type: STRING	
Field name: url, Type: STRING	

Legend:

Δ: Single-byte space

For details about the data types that can be converted and the settings for the structure of common records, see *9.11.1 Format conversion definition*.

Format conversion enables you to define multiple record structures. When multiple record structures are defined, the input adaptor selects automatically the corresponding record structure and performs format conversion.

When multiple record structures are defined, the input adaptor uses the following methodology to select the appropriate record structure:

1. The input adaptor checks the structure of an input record read by the file input connector to determine whether it matches a record structure specified in the `record` tag in the format conversion definition (it compares the input record

structure against the `record` tag's record structures in the order the record structures were defined). The input adaptor selects the first record structure it detects that matches the input record.

2. If no matching record structure is found, the input adaptor discards the input record. In such a case, the input adaptor does one of the following, as specified in the `unmatchedFormat` tag in the format conversion definition:
  - Resumes processing.
  - Issues a warning message and resumes processing.
  - Issues an error message and terminates the input adaptor.

*Reference note:*

When format conversion is completed, you can perform record filtering, record extraction, and mapping between records, as necessary (in any order).

For details about record filtering, see *10.4 Record filtering*. For details about record extraction, see *10.5 Record extraction*. For details about mapping between records, see *(4) Mapping*.

#### **(4) Mapping**

You specify information about *mapping* in the mapping definition in the adaptor configuration definition file.

The two types of mapping are *mapping between record and stream* and *mapping between records*. The table below provides an overview of these types of mapping.

*Table 10-2: Overview of mapping*

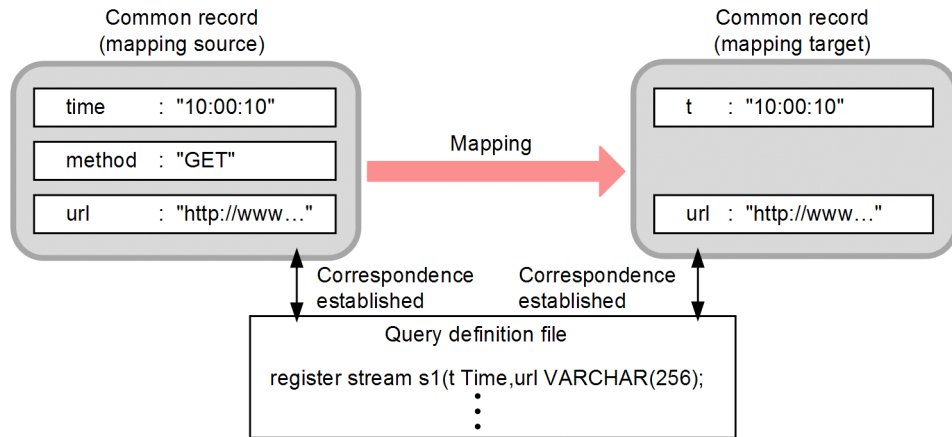
No.	Type of mapping	Description
1	Mapping between record and stream	A common record output by the callback before mapping (mapping source) is associated with a common record based on the input stream format (mapping target). Mapping between record and stream is always performed before tuple transmission.
2	Mapping between records	A common record output by the callback before mapping (mapping source) is edited and converted to a target common record. If necessary, mapping between records is performed after format conversion, but before mapping between record and stream. You can use this type of mapping to change field names in the source common record or to delete fields that are not needed for the next callback processing. You can also use built-in functions <sup>#</sup> to obtain character strings and time values from source common records and apply them to target common records. You can specify multiple definitions for mapping between records.

#

You use the `function` attribute in the `map` tag in the adaptor configuration definition file to specify the built-in functions that can be used for mapping between records. For details about the built-in functions that can be specified in the `function` attribute, see *9.11.2 Mapping definition*.

The figure below shows an example of mapping between record and stream by an input adaptor. This example maps the fields `time` and `url`, which are required for input stream `s1`, to the schema of the input stream and then converts them to a common record (mapping target).

*Figure 10-5: Example of mapping between record and stream by an input adaptor*



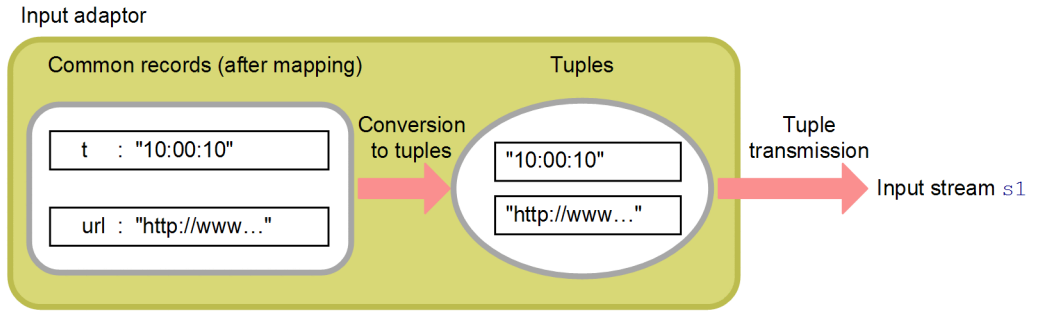
**(5) Tuple transmission**

You define information about *tuple transmission* in the input stream definition in the adaptor configuration definition file.

The common records are converted to tuples based on the mapping results, and the tuples are then sent to the input stream according to the input stream definition.

The figure below shows an example of a tuple transmission from the input adaptor to the input stream. This example sends tuples to input stream `s1`.

Figure 10-6: Example of tuple transmission from the input adaptor



### 10.2.3 File input settings

File input settings must be specified in the adaptor configuration definition file. File input settings are specified in the following CB definitions in the input adaptor definition in the adaptor configuration definition file:

- CB definition for input  
Define in the file input connector definition information about the input connector. For details about this definition, see *9.9.1 CB definition for input* and *9.10.1 File input connector definition*.
- CB definition for editing  
Define in the format conversion definition the data format for the input data, and define mapping information in the mapping definition. For details about these definitions, see *9.9.3 CB definition for editing*, *9.11.1 Format conversion definition*, and *9.11.2 Mapping definition*.
- CB definition for sending  
Define in the input stream definition information about the input stream to which the input adaptor connects. For details about this definition, see *9.9.4 CB definition for sending* and *9.12.1 Input stream definition*.

## 10.3 HTTP packet input

This section provides an overview of HTTP packet input and discusses the flow of data processing and the settings in the definition files.

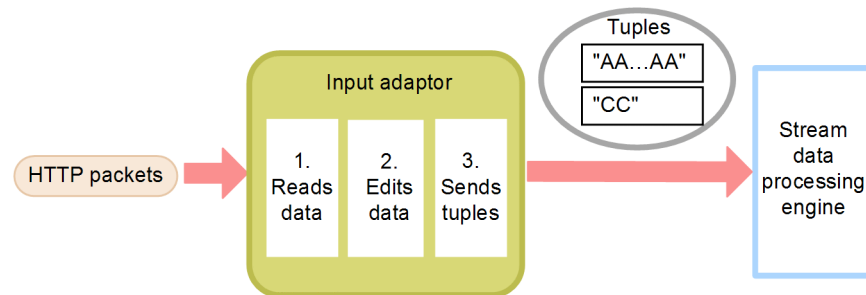
### 10.3.1 Overview of HTTP packet input

HTTP packet input processing involves receiving HTTP packet data with the standard input, converting the data into a format that can be processed by the stream data processing engine, and then sending the data to the stream data processing engine.

An input adaptor is used to perform HTTP packet input processing.

The following figure provides an overview of HTTP packet input.

Figure 10-7: Overview of HTTP packet input



#### 1. Reads data

Uses a packet input connector to read data obtained from the packet analyzer with the standard output, analyzes the obtained HTTP packet data, then converts the data.

#### 2. Edits data

Edits the data (performs mapping).

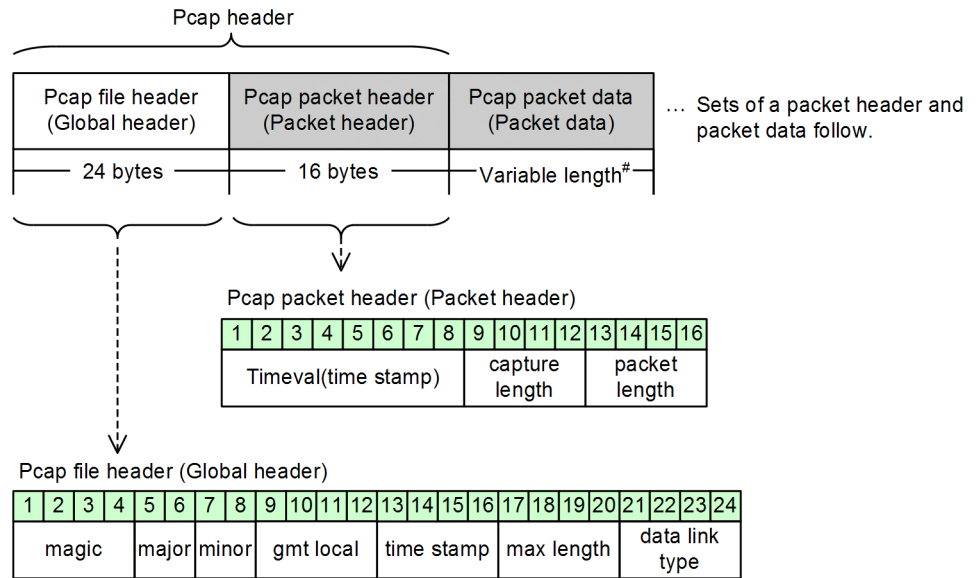
#### 3. Sends tuples

Converts the edited data to tuples and sends them to the SDP server.

To input HTTP packets, you require a packet analyzer (WinDump) that can output packets in *Pcap format*. For details about the WinDump version, see the Release Notes for Stream Data Platform - AF.

The HTTP packet input function can handle only HTTP packets output in Pcap format, which is shown in the figure below.

Figure 10-8: Pcap format



#: The size is obtained from the packet header.

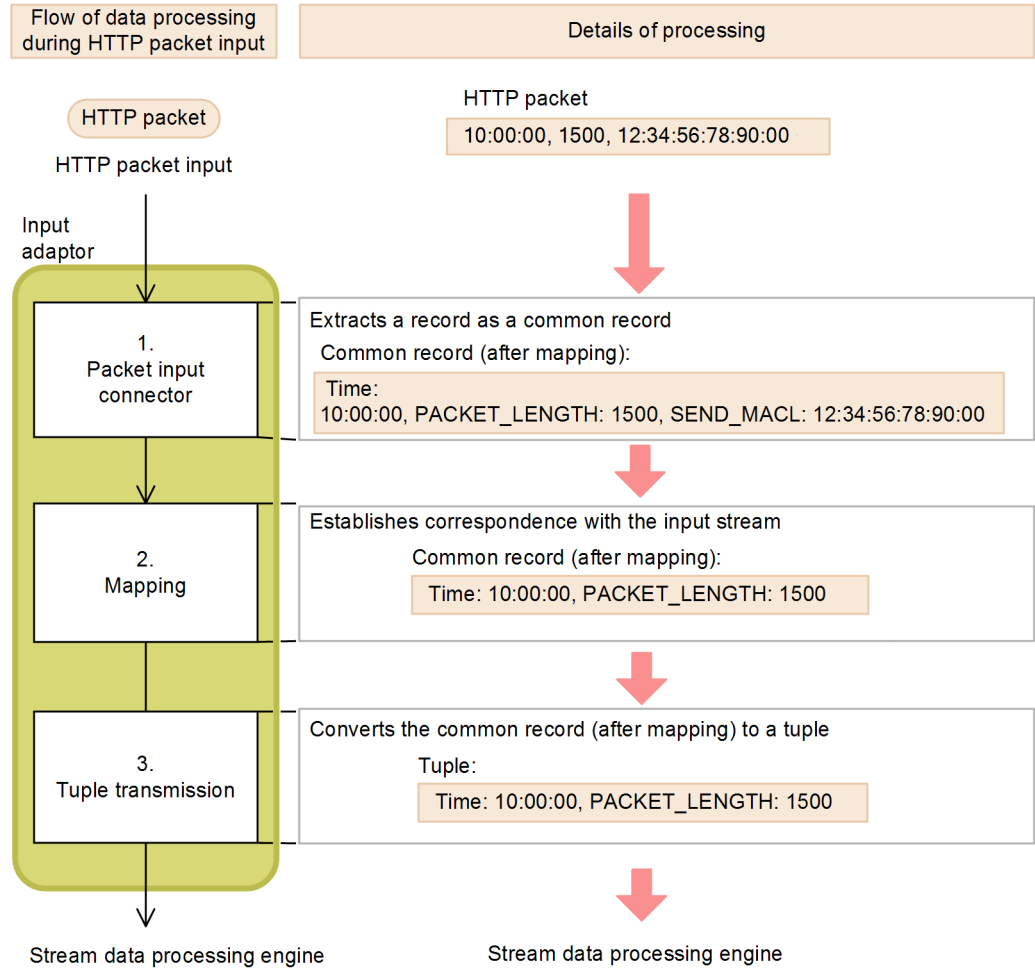
In the Pcap format, a Pcap header is added to the packet part that flows in the network. In the Pcap header, the first header created is the global header created after WinDump starts. After the global header is created, repeated sets of a Pcap packet header (packet header) and Pcap packet data (packet data) are created.

You specify information about the Pcap format or about some other HTTP packet format in the HTTP packet input connector definition in the adaptor configuration definition file.

### 10.3.2 Flow of data processing during HTTP packet input

The following figure shows the flow and details of data processing during HTTP packet input by an input adaptor.

Figure 10-9: Flow and details of data processing during HTTP packet input

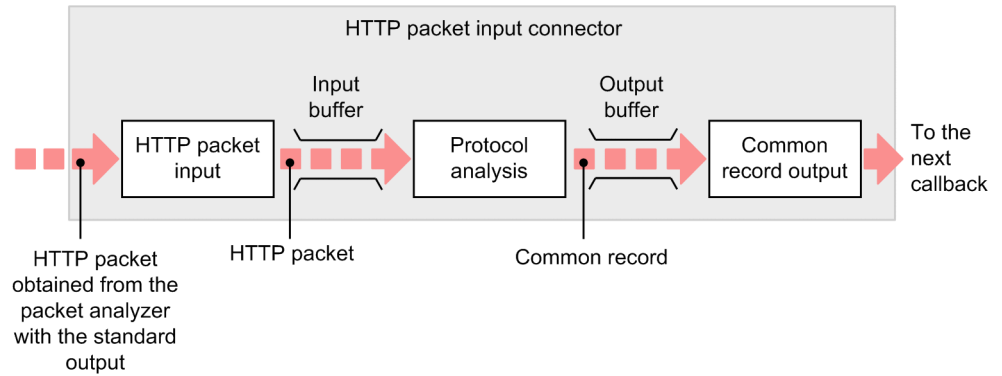


This subsection discusses HTTP packets input by an HTTP packet input connector. For details about mapping and the tuple transmission, see *10.2.2(4) Mapping* and *10.2.2(5) Tuple transmission*. For details about the data formats handled by an input adaptor, see *10.2.2(1) Data formats handled by an input adaptor*.

The following figure provides an overview of HTTP packet input by an HTTP packet input connector.



Figure 10-10: Overview of HTTP packet input by an HTTP packet input connector



#### 1. HTTP packet input

The *HTTP packet input connector* obtains with the standard input the HTTP packets output by the packet analyzer with the standard output. The obtained HTTP packets are stored in the input buffer.

#### 2. Protocol analysis

The HTTP packet input connector analyzes the protocol in the stored data and converts the analyzed protocol data to common records. Note that HTTP packet input is performed asynchronously with protocol analysis. The common records obtained after conversion are stored in the output buffer.

#### 3. Common record output

The HTTP packet input connector outputs the common records from the output buffer to the next callback.

### 10.3.3 HTTP packet input settings

HTTP packet input settings must be specified in the adaptor configuration definition file. HTTP packet input settings are specified in the following CB definitions in the input adaptor definition in the adaptor configuration definition file:

- CB definition for input

Define in the HTTP packet input connector definition information about the HTTP packet input connector. For details about this definition, see *9.9.1 CB definition for input* and *9.10.2 HTTP packet input connector definition*.

- CB definition for editing

Define mapping information in the mapping definition. For details about this definition, see *9.9.3 CB definition for editing* and *9.11.2 Mapping definition*.

- CB definition for sending

Define in the input stream definition information about the input stream to which the input adaptor connects. For details about this definition, see *9.9.4 CB definition for sending* and *9.12.1 Input stream definition*.

## 10.4 Record filtering

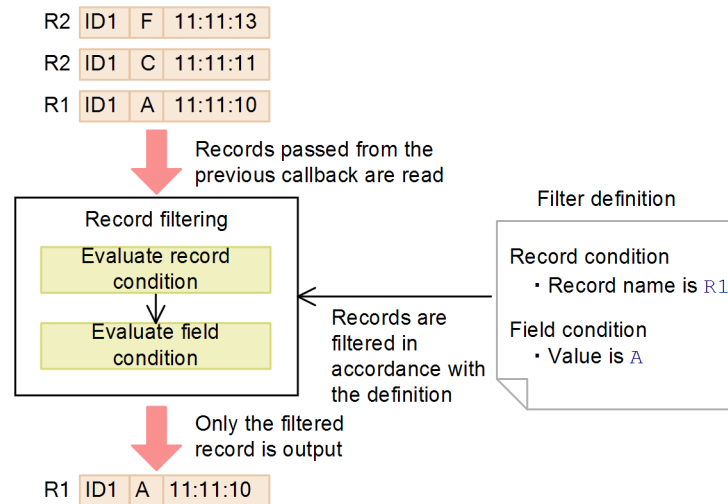
This section provides an overview of record filtering and discusses the flow of data processing and the required settings.

### 10.4.1 Overview of record filtering

When you wish to process only specific records during stream data processing, you can use a filter as an editing callback.

The following figure provides an overview of record filtering.

Figure 10-11: Overview of record filtering



Record filtering involves evaluating the input records against the information specified in the filter definition in the adaptor configuration definition file in order to determine the records that satisfy the specified conditions. The target records are selected and output based on the record format and field values in the records. For details about record filtering processing, see *10.4.2 Flow of data processing during record filtering*.

#### Format of input and output data for filtering

The format of the input and output data for record filtering is the common record.

#### Timing of filtering

Filtering is performed as an editing callback in the input or output adaptor.

In the input adaptor, filtering is performed after data input or format conversion by the HTTP packet input connector. In the output adaptor, filtering is performed after the mapping that follows tuple reception.

For the timing for performing the editing callback, see 2.5.2 *Evaluating the data editing methods*.

## 10.4.2 Flow of data processing during record filtering

Record filtering involves evaluating record and field conditions. You specify the following information for the record and field conditions:

- *Record condition*

Specify in the `record` tag the name of a record to be filtered.

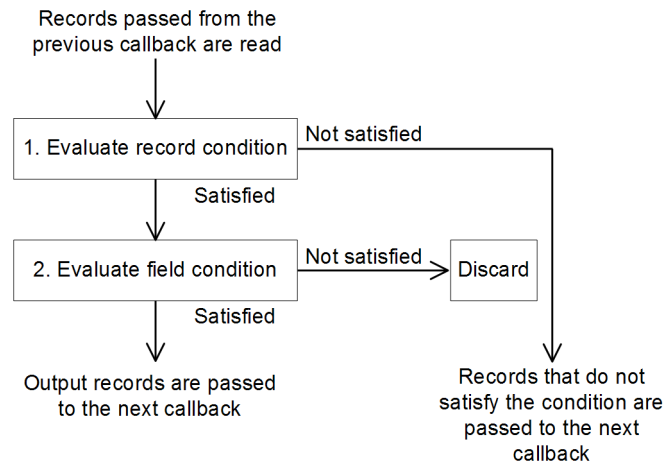
- *Field condition*

Specify in the `field` tag a field name, a comparison operator, and a condition value for the records to be filtered.

For example, if you want to set as a condition that the record name is R1, the field name is F1, and this field's value is greater than 100, you would specify `source="R1"` as the record name in the record condition and `source="F1" condition="gt" value="100"` as the field condition.

The following figure shows the flow of data processing during record filtering.

*Figure 10-12:* Flow of data processing during record filtering



1. Evaluate the record condition

Determines whether the record name in the input record matches the record name specified in the record condition.

- Record that satisfies the record condition

A record that satisfies the record condition is passed on to the field condition evaluation.

- Record that does not satisfy the record condition

A record that does not satisfy the record condition is output as is and passed to the next callback.

## 2. Evaluate the field condition

Determines whether the field value in the record that satisfied the record condition matches the field value specified in the field condition.

- Record that satisfies the field condition

A record that satisfies both the record condition and the field condition is output. The output record is passed to the next callback.

- Record that does not satisfy the field condition

A record that does not satisfy the condition is discarded at this point.

If records are discarded, the number of discarded records is output to the adaptor trace information. For details about the adaptor trace information, see *6.3.3 Details of adaptor trace information*.

### 10.4.3 Record filtering settings

Record filtering settings must be specified in the adaptor configuration definition file. Record filtering settings are specified in the following CB definition in the input or output adaptor definition in the adaptor configuration definition file:

- CB definition for editing

Define in the filter definition the record and field conditions for filtering records. For details about this definition, see *9.9.3 CB definition for editing* and *9.11.3 Filter definition*.

## 10.5 Record extraction

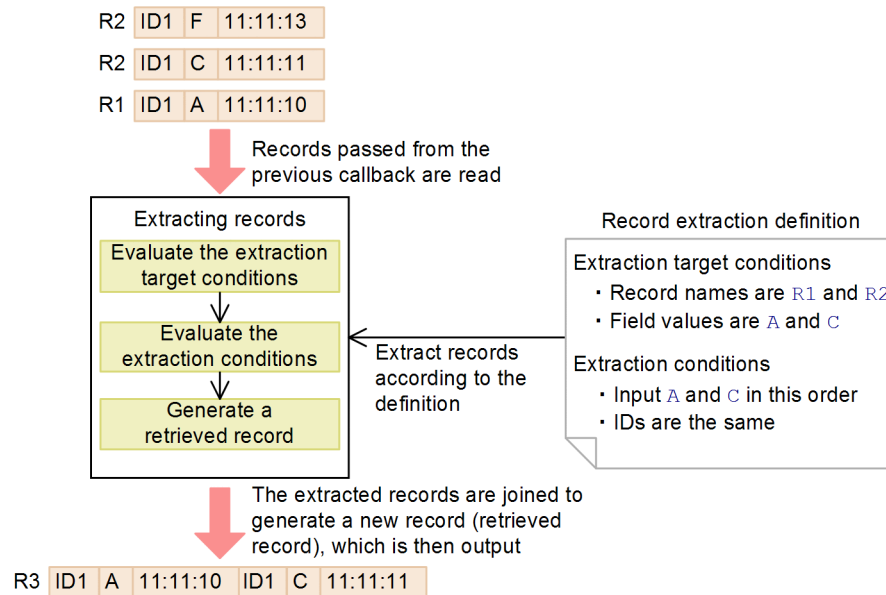
This section provides an overview of record extraction and discusses the flow of data processing and the required settings.

### 10.5.1 Overview of record extraction

Record extraction is used as an editing callback to extract specific records and then assemble them into a new record.

The following figure provides an overview of record extraction.

Figure 10-13: Overview of record extraction



Record extraction involves evaluating records to determine the records that satisfy conditions specified in the record extraction definition in the adaptor configuration definition file. The target records are extracted based on the specified criteria, such as record format, field values in records, and record input order, and the resulting records are joined to generate a new record, which is then output. The record that is generated by joining other records is called a *retrieved record*. For details about record extraction processing, see *10.5.2 Flow of data processing during record extraction*.

#### Format of input and output data for record extraction

The format of the input and output data for record extraction is the common record.

### Timing of record extraction

Record extraction is executed as an editing callback in the input adaptor. Record extraction is performed after data input or format conversion by the HTTP packet input connector.

Note that an output adaptor cannot use record extraction.

For the timing for performing the editing callback, see *2.5.2 Evaluating the data editing methods*.

### Example of record extraction

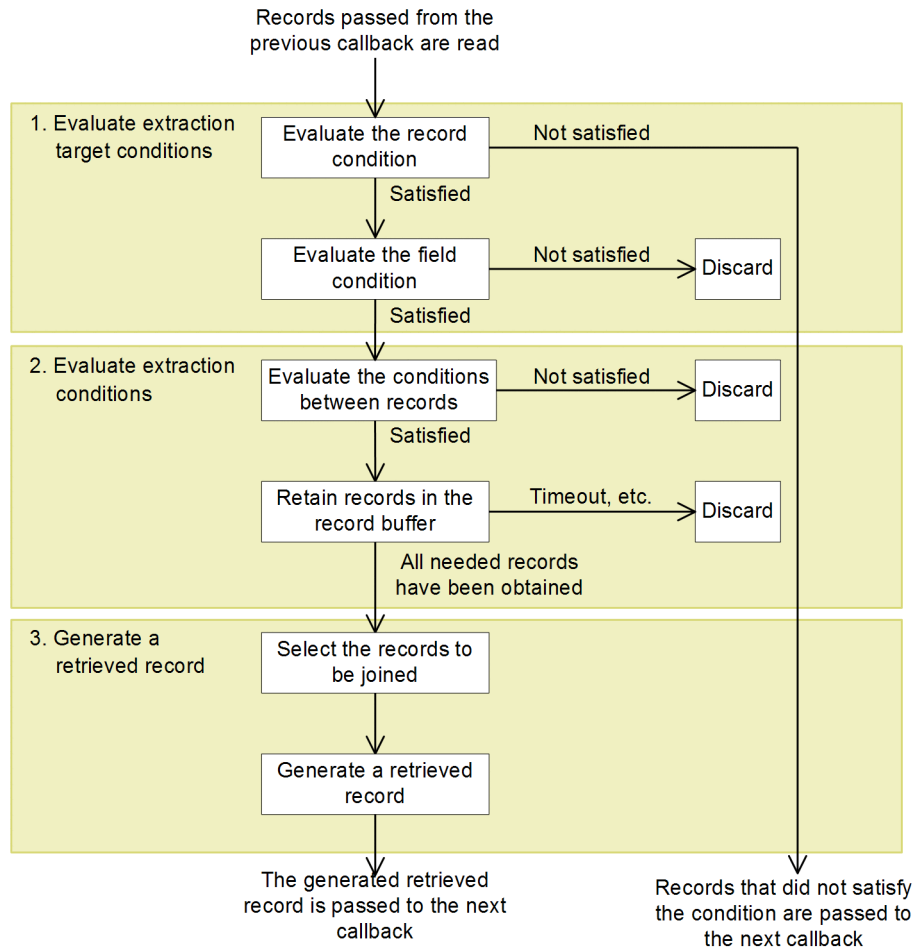
If you combine record extraction and the HTTP packet input connector described in *10.3 HTTP packet input*, you can measure the elapsed time from HTTP packet request to response. In this case, the record extraction function extracts records with the request and response packets that have the same sending IP address and port number and the same receiving IP address and port number, then joins the extracted records. You can determine the time from request to response by comparing the time values in the request and response packets in the joined record.

You can also perform record extraction on joined records to extract only the records with a specified URL, then join the records whose URL changes in a specified order. This would enable you to monitor a user's site change status from the URL information in the HTTP packets.

## 10.5.2 Flow of data processing during record extraction

In record extraction, evaluation of extraction target conditions and extraction conditions and joining of extracted records occur before a new record (retrieved record) is generated. The following figure shows the flow of data processing during record extraction.

Figure 10-14: Flow of data processing during record extraction



Details of the processing in 1 through 3 in this figure are provided below. For details about the tags and attributes in the record extraction definition in the adaptor configuration definition file that are discussed below, see *9.11.4 Record extraction definition*.

### (1) Evaluating the extraction target conditions

This processing involves evaluating the extraction target conditions in the input records.

An *extraction target condition* is one of the types of conditions used to select records to be extracted. You specify an extraction target condition in the `targetrecord` tag in the record extraction definition.



For the extraction target conditions, specify a record condition and a field condition.

- *Record condition*

Specify in the `record` tag the name of a record to be extracted.

- *Field condition*

Specify in the `field` tag the name of a field in the record to be extracted, a comparison operator, and a condition value.

For example, to specify the record name `R1`, field name `F1`, and field value `MIKE` as the conditions, specify `source="R1"` for the record name in the record condition and `source="F1" condition="eq" value="MIKE"` in the field condition.

Evaluating the extraction target conditions involves evaluating the record condition and field condition according to the extraction target conditions specified in the record extraction definition.

1. Evaluate the record condition

The record extraction function determines whether the record name in the input record matches the record name specified in the record condition.

- Record satisfying the record condition  
Passed on to the field condition evaluation processing.
- Record that does not satisfy the record condition  
Output as is and passed on to the next callback.

2. Evaluate the field condition

The record extraction function determines whether the field value in the record that satisfies the record condition matches the field value specified in the field condition.

- Record satisfying the field condition  
Passed on to the extraction condition evaluation processing because this record satisfies the extraction target condition. For details about extraction condition evaluation processing, see (2) *Evaluating the extraction conditions*.
- Record that does not satisfy the field condition  
Discarded at this point.

## **(2) Evaluating the extraction conditions**

The extraction condition is evaluated for the records that satisfy the extraction target condition.

An *extraction condition* is one of the types of conditions used to select records to be

extracted. You specify an extraction condition in the `extraction` tag in the record extraction definition.

For the extraction condition, you specify the maximum number of records that can be retained in the record buffer and a condition between records. For the *condition between records*, you specify the record input order and the field values to be compared between records. The `targets` tag is used to specify the condition between records.

Evaluating the extraction conditions involves evaluating the condition between records according to the extraction conditions specified in the record extraction definition. The records that satisfy the condition between records are retained in the record buffer until the extraction conditions are satisfied. The flow of evaluating the extraction conditions is described below.

1. Evaluate the conditions between records

The record extraction function evaluates the records that satisfied the extraction target conditions to determine whether they also satisfy the following conditions between records:

- Whether the record input orders match
- Whether the field values match between records

The records that satisfy these conditions are retained in the record buffer, while the records that do not satisfy the conditions are discarded.

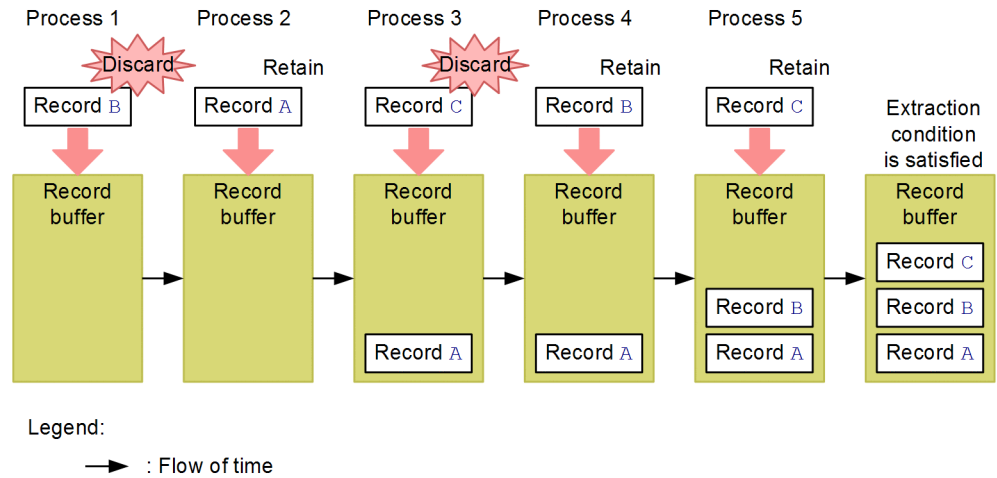
2. Retain records in the record buffer

The records that satisfy the conditions between records are retained in the record buffer. When all applicable records are obtained after the conditions between records are checked, the extraction condition is satisfied and the generation of the retrieved record is then performed. For details about the retrieved record generation processing, see (3) *Generating a retrieved record*.

The following figure shows an example of the flow of retaining and discarding records when the record input order is specified.

*Figure 10-15:* Example flow of retaining and discarding records when the record input order is specified

Specification of record input order: Record A → Record B → Record C



This example specifies the record input order as Record A → Record B → Record C. As a result, the input records are retained or discarded as follows:

- Process 1: Record B that was read while there were no records in the record buffer is discarded because it does not satisfy the input order.
- Process 2: Record A that was read while there were no records in the record buffer is retained because it satisfies the input order.
- Process 3: Record C that was read after record A is discarded because it does not satisfy the input order.
- Process 4: Record B that was read after record A is retained because it satisfies the input order.
- Process 5: Record C that was read after record A and record B is retained because it satisfies the input order. The extraction condition is now satisfied because all the records were input in the order of Record A → Record B → Record C.

If records are discarded, the number of discarded records is output to the adaptor trace information. For details about the adaptor trace information, see [6.3.3 Details of adaptor trace information](#).

During the extraction condition evaluation processing, records are also discarded if their time information is out of sequence or a timeout occurs on a record retained in the record buffer. These situations are discussed below.

- When a record is out of sequence based on its time information  
If an input record's time information shows that it has been received out of sequence chronologically, the record is discarded.
- When the time information is the same in multiple records  
If multiple records with identical time information are input, the extraction condition is evaluated in the order the records were input.
- When a timeout occurs on a record retained in the record buffer

The record extraction function enables you to specify a record lifespan as a condition for determining a timeout on records retained in the record buffer. You can specify the record lifespan in the `timelimit` attribute in the `extraction` tag.

If the extraction condition has not been satisfied when the specified lifespan is reached, the expired record results in a timeout and is discarded from the record buffer.

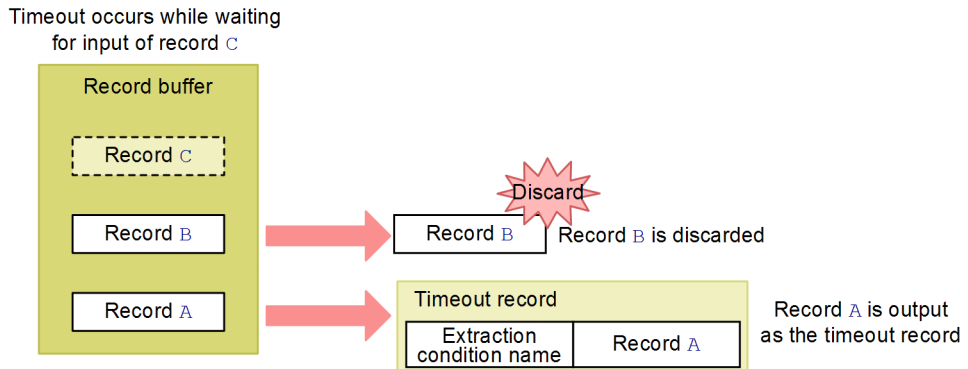
When output of records resulting in a timeout has been specified in the `timeout` attribute in the `extractions` tag, only the first record of the affected records in the record buffer is output.

This record that is output in the event of a timeout is called the *timeout record*. A field containing the extraction condition name is added to the timeout record (the extraction condition name to be used is specified in the `name` attribute in the `extraction` tag).

The following figure shows the flow of record processing in the event of a timeout.

Figure 10-16: Flow of record processing in the event of a timeout

Specification of record input order: Record A → Record B → Record C



The status of this example is as follows:

- If all three records A, B, and C are obtained, they can be passed on to the retrieved record generation processing because the extraction conditions are all satisfied.
- If a timeout occurs, a timeout record is output.
- Here, a timeout occurs while waiting for input of record C.

As a result, record B is discarded and record A is output as the timeout record.

- When the number of records retained in the record buffer exceeds the maximum value

The record extraction function enables you to specify for management purposes a maximum number of records that can be retained in the record buffer. You specify in the `size` attribute in the `extractions` tag the maximum number of records to be retained. When the number of records to be retained in the record buffer exceeds the specified value, all records are discarded from the record buffer.

- When the same record is input more than once

The record extraction function regards as identical any number of records that all satisfy the extraction target condition and contain the field values specified in the condition between records. All such records become "same records." If the same record is input more than once, the following processing takes place according to the information specified in the `samerecord` attribute in the `extractions` tag:

- When `overwrite` is specified in the `samerecord` attribute

Processing depends on the location of the record existing in the record buffer that is determined to be the same as the record being input.

If it is the last record in the buffer, it is overwritten by the record being input. If it is not the last record in the buffer, all records in the record buffer are discarded and then the input record is retained.

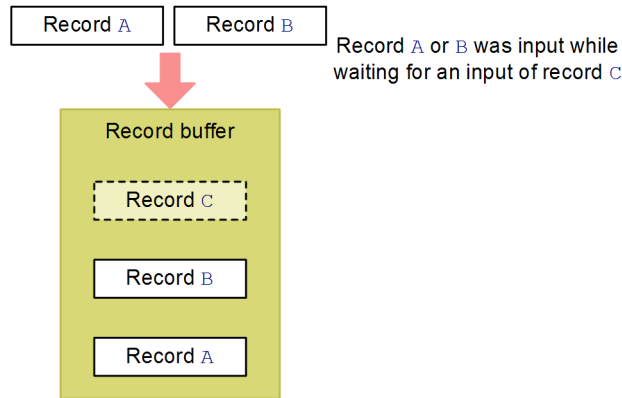
- When `delete` is specified in the `samerecord` attribute

The input record is discarded.

The following figure shows an example of the processing when the same record is input.

*Figure 10-17: Processing when the same record is input*

Specification of record input order: Record A → Record B → Record C



The status of this example is as follows:

- Once the three records A, B, and C have all been obtained, they can be passed on for retrieved record generation processing because the extraction conditions are all satisfied.
- `overwrite` is specified in the `samerecord` attribute.
- Record A or B was input while waiting for an input of record C.

In this example, the processing depends on whether record A or B is input as described below.

- When record A is input  
Records A and B in the record buffer are discarded and the record A that was just input is retained.
- When record B is input  
The record B existing in the record buffer is overwritten by the record B that was just input.

### **(3) Generating a retrieved record**

Generation of a retrieved record involves joining the records that satisfy the extraction conditions to generate a retrieved record. The flow of retrieved record generation is described below.

1. Select the records to be joined

Based on the extraction condition, the record extraction function selects the records to be joined to generate a retrieved record. You specify the records to be

joined in the source attribute in the `select` tag.

## 2. Generate a retrieved record

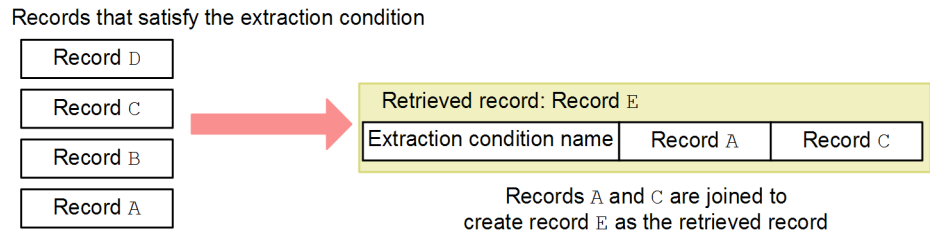
The records selected in step 1 are joined to generate a retrieved record.

A field containing the extraction condition name specified in the `name` attribute in the `extraction` tag is added to the generated retrieved record.

- Field in which the extraction condition name is to be placed: `condition`
- Other fields: `extraction-target-record-name_field-name`

The following figure shows an example of a retrieved record.

*Figure 10-18: Example of a retrieved record*



This example joins records A and C to generate a new record E as the retrieved record.

### 10.5.3 Record extraction settings

Record extraction settings must be specified in the adaptor configuration definition file. Record extraction settings are specified in the following CB definition in the input adaptor definition in the adaptor configuration definition file:

- CB definition for editing

In the record extraction definition, define information required for extracting records, such as an extraction target condition and an extraction condition. For details about the definition, see *9.9.3 CB definition for editing* and *9.11.4 Record extraction definition*.

## 10.6 File output

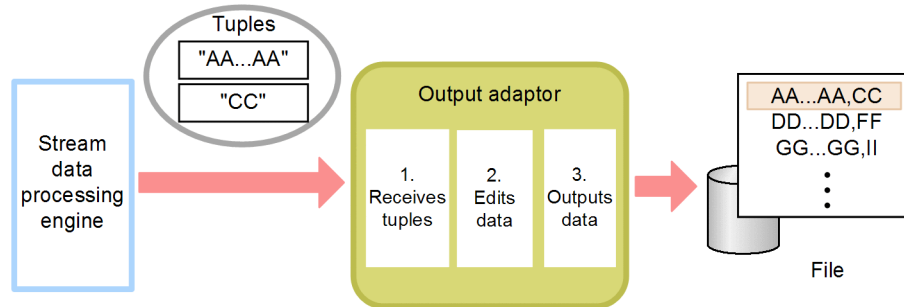
This section provides an overview of file output and discusses the flow of data processing and the settings in the definition files.

### 10.6.1 Overview of file output

File output processing involves converting data processed by the stream data processing engine so that it conforms to a specified output format, and then outputting the data. An output adaptor is used to output data to a file.

The following figure provides an overview of file output.

Figure 10-19: Overview of file output



#### 1. Receives tuples

Receives tuples containing the summary analysis results of stream data processed by the stream data processing engine.

#### 2. Edits data

Edits the received data (performs data mapping and format conversion).

#### 3. Outputs data

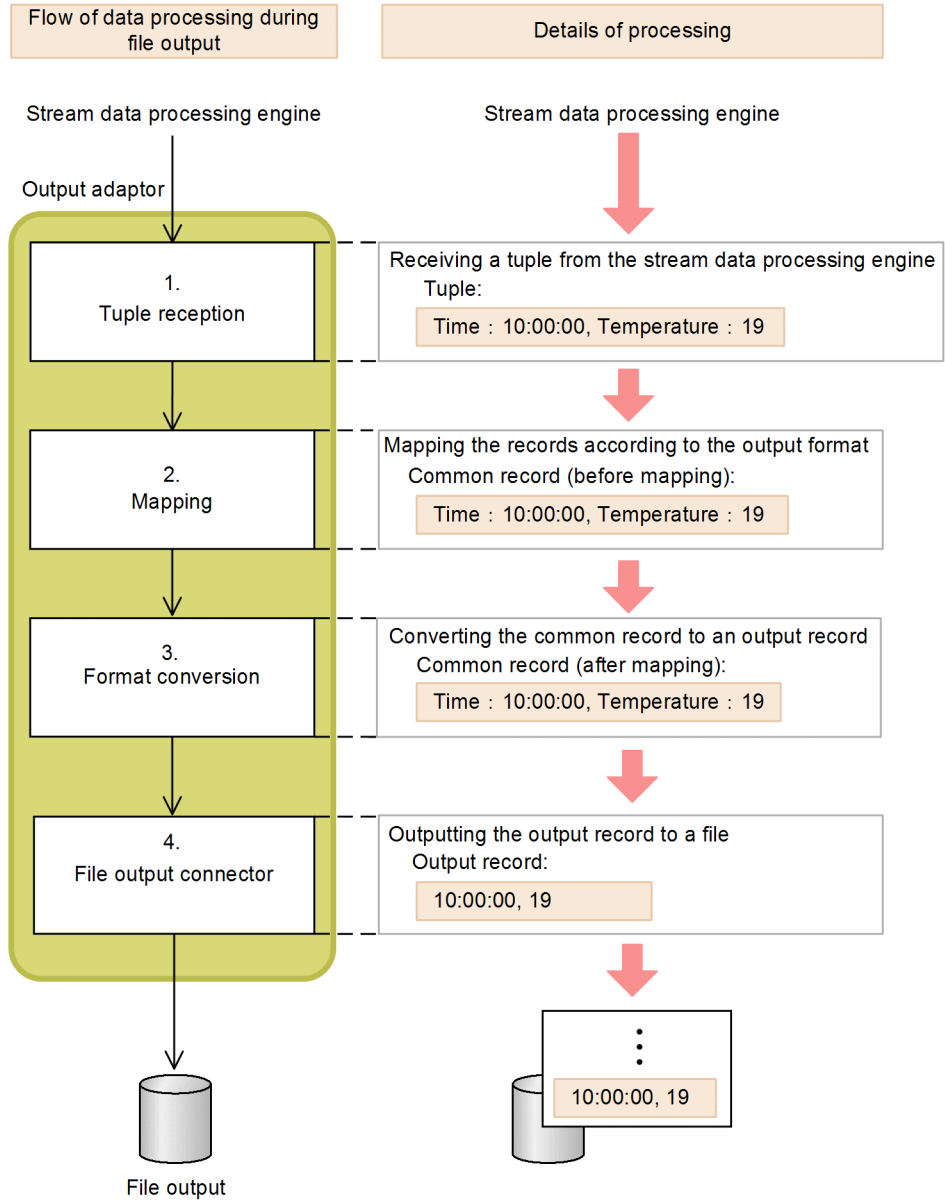
Converts the edited data to conform to a specified output format and then uses a file output connector to output the data.

### 10.6.2 Flow of data processing during file output

The following figure shows the flow and details of data processing during file output by an output adaptor.



Figure 10-20: Flow and details of data processing during file output



For details about the data formats handled by an output adaptor, see (1) *Data formats handled by an output adaptor*. For details about the processing, see the subsections beginning with (2) *Tuple reception*.

**(1) Data formats handled by an output adaptor**

The data formats handled by an output adaptor are the common record and the output record. The common record is the same as the format handled by input adaptors. For details about the common record, see the description of the common record in *10.2.2(1) Data formats handled by an input adaptor*. This subsection discusses the output record.

*Output record*

An output record is a row of data that is output to an output file. An output adaptor treats one row of data as one output record.

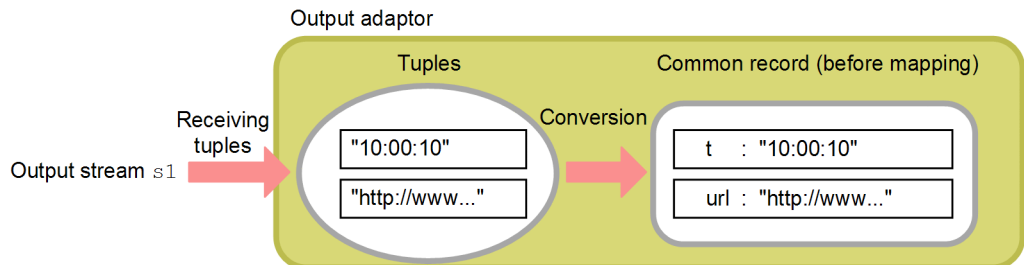
**(2) Tuple reception**

You define information about *tuple reception* in the output stream definition in the adaptor configuration definition file.

Tuple reception involves receiving tuples sent from the output stream and converting them to common records.

The figure below shows an example of tuple reception from the output stream using an output adaptor. This example receives tuples from output stream *s1*.

*Figure 10-21: Example of tuple reception by an output adaptor*

**(3) Mapping**

You specify information about *mapping* in the mapping definition in the adaptor configuration definition file.

The two types of mapping are *mapping between record and stream* and *mapping between records*. The table below provides an overview of these types of mapping.

Table 10-3: Overview of mapping

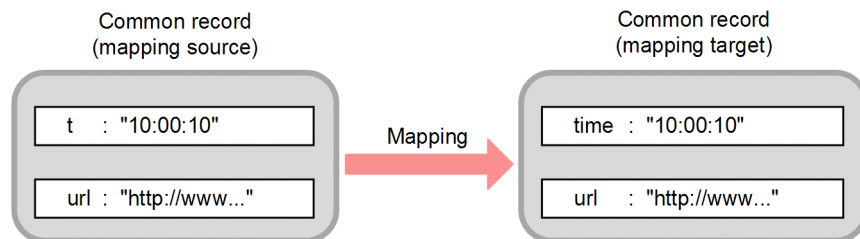
No.	Type of mapping	Description
1	Mapping between record and stream	A common record (mapping source) that was obtained during tuple reception is converted to a common record (mapping target) according to the output format. Mapping between record and stream is always performed after tuple reception.
2	Mapping between records	A common record obtained from mapping between record and stream is edited to obtain a target common record. If necessary, mapping between records is performed after the mapping between record and stream but before format conversion. You can use this type of mapping to change field names in the source common record or to delete fields that are not needed for the next callback processing. You can also use built-in functions <sup>#</sup> to obtain character strings and time values from source common records and apply them to target common records. You can specify multiple definitions for mapping between records.

#

You use the `function` attribute in the `map` tag in the adaptor configuration definition file to specify the built-in functions that can be used for mapping between records. For details about the built-in functions that can be specified in the `function` attribute, see *9.11.2 Mapping definition*.

The figure below shows an example of mapping using an output adaptor. This example converts the fields `t` and `url` to the fields `time` and `url` according to the adaptor's output format and then converts them to the target common record.

Figure 10-22: Example of mapping between record and stream using an output adaptor



*Reference note:*

After performing mapping between record and stream, you can perform mapping between records and record filtering, as necessary (in any order).

For details about record filtering, see *10.4 Record filtering*.

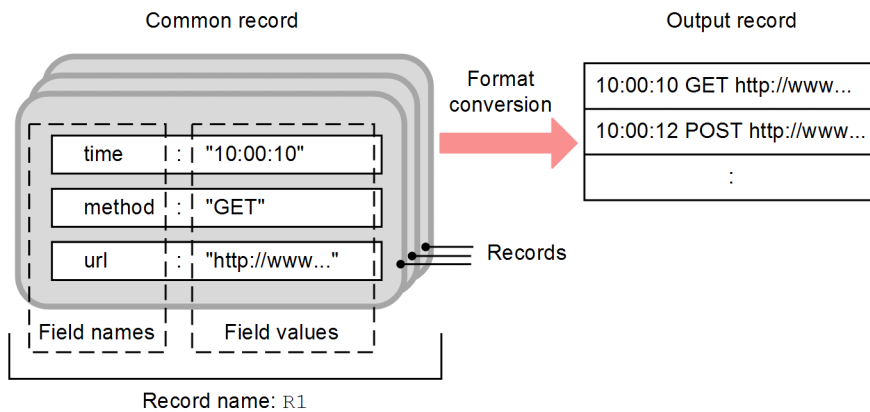
**(4) Format conversion**

You define information about *format conversion* in the format conversion definition in the adaptor configuration definition file.

Format conversion involves conversion from a common record to an output record.

The figure below shows an example of format conversion from common record to output record. This example converts a common record that has field values of the TIME and character string types to an output format that consists of three fields delimited by the space.

*Figure 10-23:* Example of format conversion from common record to output record



The table below shows the structure of the common record in the above example and the tags that are specified in the format conversion definition.

*Table 10-4:* Definition files to be specified and record structure

Record structure	Tag
Record name: R1 Record structure: (\$_time) Δ (\$_method) Δ (\$_url)	record tag (record definition)
Field name: time, Type: TIME	field tag (field definition)
Field name: method, Type: STRING	
Field name: url, Type: STRING	

Legend:

Δ: Single-byte space

For details about the data types that can be converted and the settings for the structure of common records, see *9.11.1 Format conversion definition*

### (5) Data output by the file output connector

You use the file output connector definition in the adaptor configuration definition file to define information about file output.

A *file output connector* outputs obtained output records to the defined output directory.

The table below describes the structures for the files that can be output by a file output connector.

File structure	Description
Wraparound	Files are created according to the output file creation rules specified in the file output connector definition. When the number of files reaches the maximum value, the first file used is overwritten.
Non-wraparound	Files are created according to the output file creation rules specified in the file output connector definition. When the number of files reaches the maximum value, the output adaptor stops record output and discards the remaining records.

### 10.6.3 File output settings

File output settings must be specified in the adaptor configuration definition file. File output settings are specified in the following CB definitions in the output adaptor definition in the adaptor configuration definition file:

- CB definition for output  
Define in the file output connector definition information about the output connector. For details about this definition, see *9.9.2 CB definition for output* and *9.10.3 File output connector definition*.
- CB definition for editing  
Define in the format conversion definition the data format for the output data, and define mapping information in the mapping definition. For details about these definitions, see *9.9.3 CB definition for editing*, *9.11.1 Format conversion definition*, and *9.11.2 Mapping definition*.
- CB definition for receiving  
Define in the output stream definition information about the output stream to which the output adaptor connects. For details about this definition, see *9.9.5 CB definition for receiving* and *9.12.2 Output stream definition*.

## 10.7 Dashboard output

This section provides an overview of dashboard output and discusses the flow of data processing and the settings in the definition files.

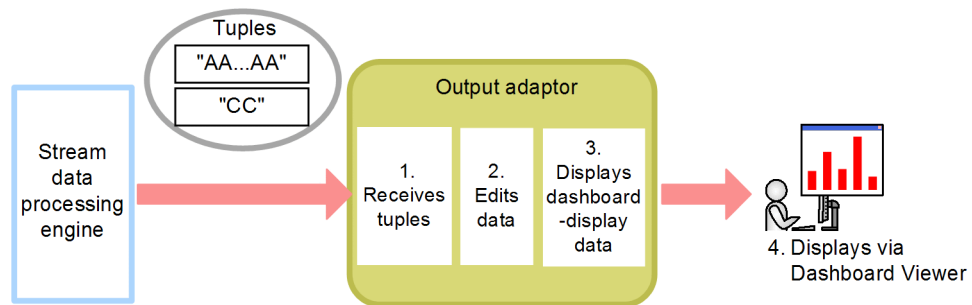
### 10.7.1 Overview of dashboard output

The dashboard output function converts data processed by the stream data processing engine to *dashboard-display data* and then sends it to the *Dashboard Server* on Flex Dashboard where stream data summary analysis results can be displayed as bar or broken line graphs. This function also uses *Dashboard Viewer* to display the dashboard-display data obtained by Dashboard Server.

The figure below provides an overview of dashboard output.

Note that Internet Explorer and Flash Player are required in order to use Dashboard Viewer to display stream data summary analysis results.

Figure 10-24: Overview of dashboard output



#### 1. Receives tuples

Receives tuples containing summary analysis results of the stream data processed by the stream data processing engine.

#### 2. Edits data

Performs mapping on the received data.

#### 3. Displays dashboard-display data

Uses the dashboard output connector to convert the edited data to dashboard-display data and displays the data (also deletes old data).

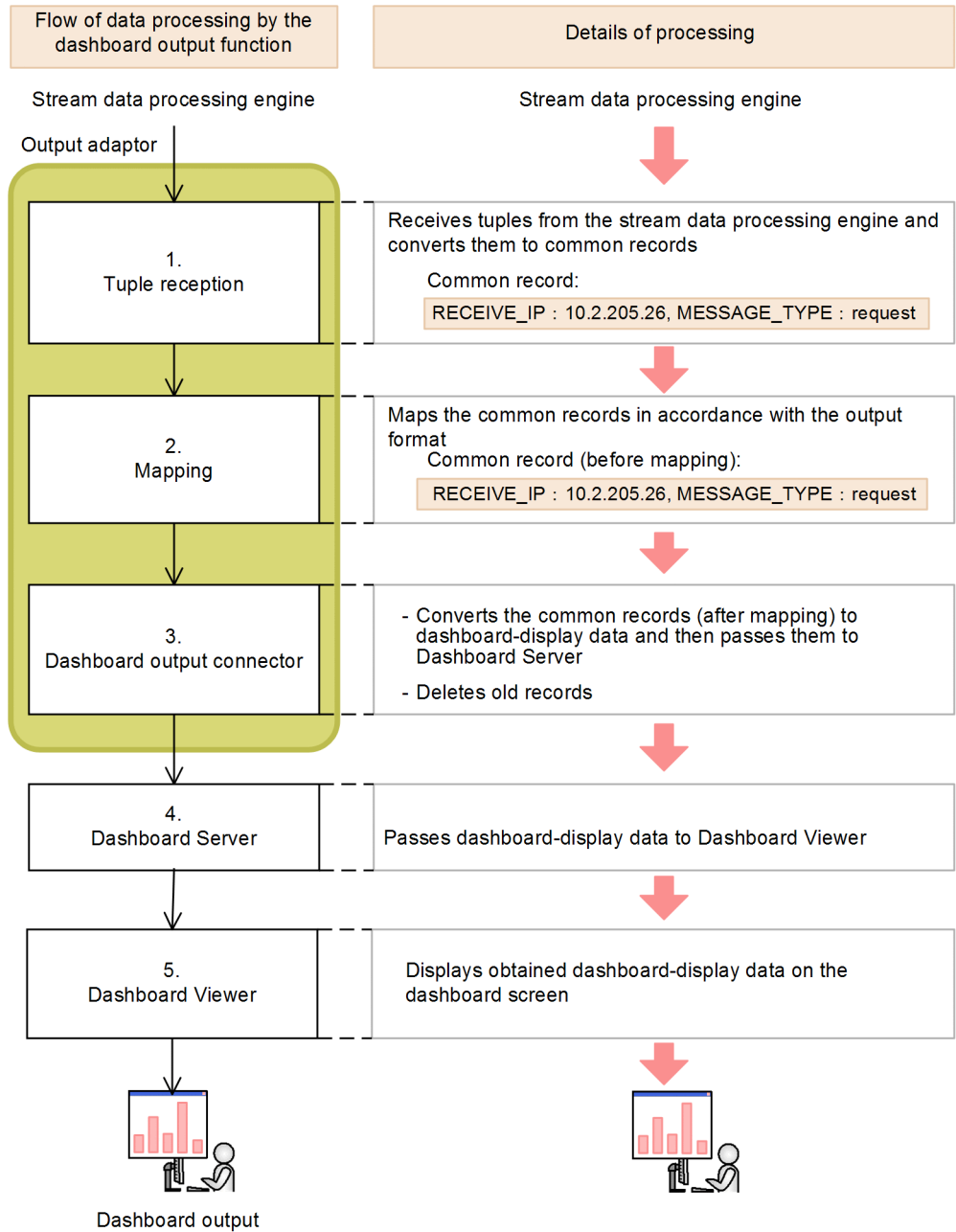
#### 4. Displays via Dashboard Viewer

Uses Dashboard Viewer to display dashboard-display data obtained from a dashboard output connector.

## 10.7.2 Flow of data processing during dashboard output

The following figure shows the flow and details of data processing during dashboard output.

Figure 10-25: Flow and details of data processing during dashboard output



This subsection describes the flow of data processing from output of



dashboard-display data by a dashboard output connector to display of dashboard-display data by Dashboard Viewer. For details about tuple reception and mapping, see *10.6.2(2) Tuple reception* and *10.6.2(3) Mapping*. For details about the data formats handled by an output adaptor, see *10.6.2(1) Data formats handled by an output adaptor*.

**(1) Using a dashboard output connector to convert data to dashboard-display data**

You specify information about data conversion to dashboard-display data by a dashboard output connector in the dashboard output connector definition in the adaptor configuration definition file.

A dashboard output connector obtains the common records after mapping, converts them to dashboard-display data according to the dashboard output connector definition, then outputs the data to the *RMI communication area*. You can use Dashboard Server to acquire the dashboard-display data in the RMI communication area.

**(2) Using a dashboard output connector to delete records**

You specify information about record deletion by a dashboard output connector in the dashboard output connector definition in the adaptor configuration definition file.

Dashboard-display data is deleted from the dashboard output connector's record area according to the record deletion condition specified in the dashboard output connector definition. You do this so that the dashboard-display data obtained by the dashboard output connector will not cause a memory shortage in the dashboard output connector.

When a dashboard output connector deletes dashboard-display data, it outputs to the adaptor trace information the date and time of deletion and the number of deleted dashboard-display data records.

The available record deletion methods are explained below:

*Deletion based on the record retention period*

When a new record is added to a dashboard output connector, this method deletes those records in the record area whose time value is older than a reference time minus a retention period. For example, if the reference time is 10:00:10 and the retention period is 5 seconds, a record whose time value is 10:00:04 is deleted.

A record is not deleted if its time value is equal to the reference time minus the retention period.

This deletion method is based on the record retention period and assumes that the records are in ascending chronological order of the time of record arrival in the record area. If the records are not sorted in this order, a record will not be deleted if it arrives after a record whose time value is more recent than the reference time minus the retention period even though its time value is older than the reference

time minus the retention period.

The precision used to compare the time values is milliseconds. A value smaller than a millisecond is discarded.

You specify this deletion method in the `DateReference`, `RecordTime`, and `DateFieldPosition` attributes in the `RecordHoldTime` tag in the dashboard output connector definition. For details about these attributes, see *9.10.4 Dashboard output connector definition*.

#### *Deletion based on the maximum number of records to be retained*

When new records are being added to a dashboard output connector, if the number of records in the record area exceeds a set maximum number of records to be retained, this method deletes records until the number of records in the record area equals the maximum number of records to be retained. In such a case, records are deleted in the order in which they were added to the record area, starting with the oldest record.

You specify this deletion method in the `MaxNum` attribute in the `DashboardOutputConnectorDefinition` tag in the dashboard output connector definition. For details about this attribute, see *9.10.4 Dashboard output connector definition*.

#### *Deletion of acquired records*

When Dashboard Viewer acquires dashboard-display data from a single Dashboard Server, this method deletes records that have already been acquired.

You specify this method in the `ReadRecordRemoveFlag` attribute in the `DashboardOutputConnectorDefinition` tag in the dashboard output connector definition. For details about this attribute, see *9.10.4 Dashboard output connector definition*.

### **(3) Using Dashboard Server to acquire dashboard-display data**

You specify information about acquisition of dashboard-display data by Dashboard Server in the adaptor group definition.

Dashboard Server accesses the RMI communication area of the dashboard output connector and acquires the dashboard-display data that has been output.

To access the RMI communication area, Dashboard Server specifies the port number of the RMI server. You define the port number of the RMI server in the `dashboardPortNo` attribute in the in-process group definition or RMI group definition in the adaptor group definition. For details about this attribute, see *9.7.1 In-process group definition* or *9.7.2 RMI group definition*.

### **(4) Displaying dashboard-display data via Dashboard Viewer**

Dashboard Viewer acquires dashboard-display data from Dashboard Server and then

displays it in a dashboard window.

Clients can view dashboard-display data as stream data summary analysis results by accessing the applicable URL from a Web browser and downloading Dashboard Viewer.

For details about how to display stream data summary analysis results on a dashboard, see *4.5 Displaying analysis results on a dashboard*.

### 10.7.3 Dashboard output settings

Dashboard output settings must be specified in the adaptor configuration definition file. Dashboard output settings are specified in the following adaptor group definition, CB definition for editing, and CB definition for output in the output adaptor definition in the adaptor configuration definition file:

- Adaptor group definition
 

Specify the port number of the RMI server in the in-process group definition or RMI group definition. For details about this definition, see *9.7.1 In-process group definition* or *9.7.2 RMI group definition*.
- CB definition for editing
 

Define mapping information in the mapping definition. For details about this definition, see *9.9.3 CB definition for editing* and *9.11.2 Mapping definition*.
- CB definition for output
 

Define information about the dashboard output connector in the dashboard output connector definition. For details about this definition, see *9.9.2 CB definition for output* and *9.10.4 Dashboard output connector definition*.

---

## 10.8 Timestamp adjustment for tuples

---

This section discusses the scope of timestamp adjustment, the range of times to be adjusted, and how to adjust the time.

### 10.8.1 Scope of timestamp adjustment

In the data source mode, the tuples must be input to the input stream in ascending order of the time values set in the tuples. When multiple input adaptors are used to send tuples, tuples may arrive at the SDP server out of sequence in terms of the chronological order of the time values. In such a case, a tuple whose time value is out of sequence is discarded by the SDP server.

Use of the *timestamp adjustment function* enables you to re-sort the tuples that have arrived at the SDP server so that the ones that arrived out of sequence will be input to the input stream.

The timestamp adjustment function is supported when the SDP server's time control method is set to the data source mode. The timestamp adjustment function is applied to each input stream.

### 10.8.2 Range of times to be adjusted

The *timestamp adjustment function* adjusts the time in those tuples whose time value is within a *range of times to be adjusted*. The range of times to be adjusted is determined by a *reference time* and a *length of time to be adjusted*. The period from the reference time to the point in time that is earlier than the reference time by the length of time to be adjusted constitutes the range of times to be adjusted by the timestamp adjustment function. This means that if the reference time changes, the range of times to be adjusted also changes.

#### *Reference time*

This is the base time used to adjust the time in tuples. The most recent time information in the tuples that have been sent to the input stream by the input adaptor after a query group started becomes the reference time for that input stream. When a tuple containing a more recent time arrives, the reference time changes to that time.

#### *Length of time to be adjusted*

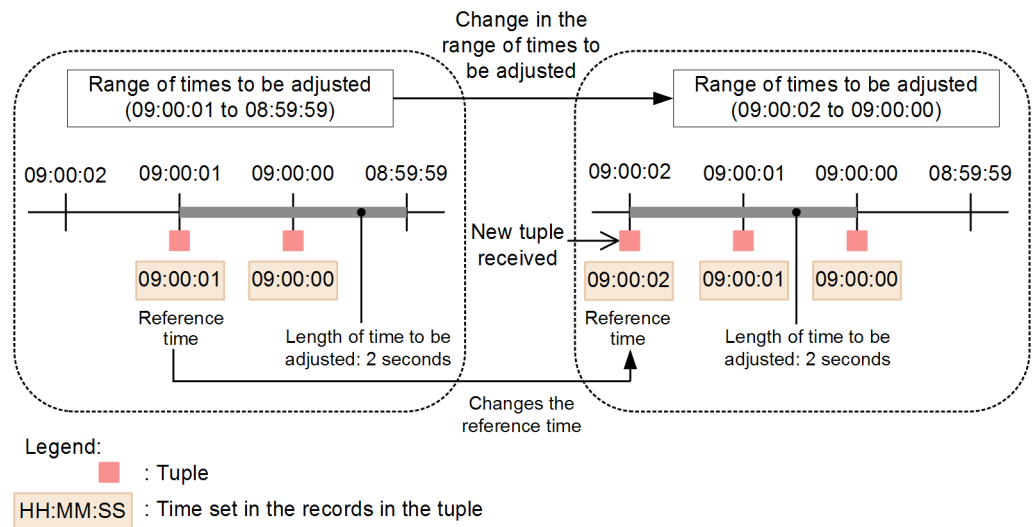
This is the period of time to be adjusted. It is determined by the *unit of time* and the *time adjustment range*. For the unit of time, you can choose *sec* (seconds), *msec* (milliseconds), or *usec* (microseconds). The time adjustment range is a period of time in the selected units.

You specify the unit of time and the time adjustment range in the `stream.timestampAccuracy` parameter of the system configuration property

file, query group property file, or stream property file. For details about the `stream.timestampAccuracy` parameter, see 8.6 *System configuration property file (system\_config.properties)*.

The following figure shows an example in which the range of times to be adjusted changes. In this example, `sec` is specified as the unit of time and 2 is specified as the time adjustment range.

Figure 10-26: Example in which the range of times to be adjusted changes



In this example, the reference time is set to 09:00:01, which is the time set in the tuple that has the most recent time information among the tuples that have arrived at the SDP server. The range of times to be adjusted is from 09:00:01 to 08:59:59, because the length of time to be adjusted is 2 seconds.

Then, a new tuple is sent from the input adaptor, and its time information is 09:00:02, which becomes the most recent time. Therefore, the reference time changes. As a result, the range of times to be adjusted changes to 09:00:02 to 09:00:00.

### 10.8.3 How to adjust the time

The timestamp adjustment function uses the following procedure to adjust time information in tuples:

1. Sets the timestamp

The function timestamps a tuple that has arrived at the SDP server. To do this, it first discards from the time value set in the records in the tuple any portion of the time value that is smaller than the specified unit of time. It then sets the resulting value as the timestamp for the tuple. For example, if `sec` is specified as the unit

of time, any value smaller than a second, such as a milliseconds value, is discarded from the time value and the resulting value (in seconds) is set as the timestamp.

2. Holds tuples

The function holds tuples whose timestamp falls within the range of times to be adjusted.

It also discards any tuple whose timestamp is earlier than the range of times to be adjusted.

3. Inputs tuples to the input stream

Those tuples among the tuples that are being held that are now outside the range of times to be adjusted because of a change in the range are input to the input stream.

When the timestamp adjustment function holds tuples and inputs them to the input stream, the method used to adjust the tuples' time information and the order in which tuples are input to the input stream depend on the reference time, the time set in the records in the tuples, and the specified unit of time. This subsection discusses how to adjust a tuple's time and the order in which tuples are input to the input stream.

**(1) How to adjust a tuple's time**

This subsection describes how to adjust a tuple's time in the following four cases:

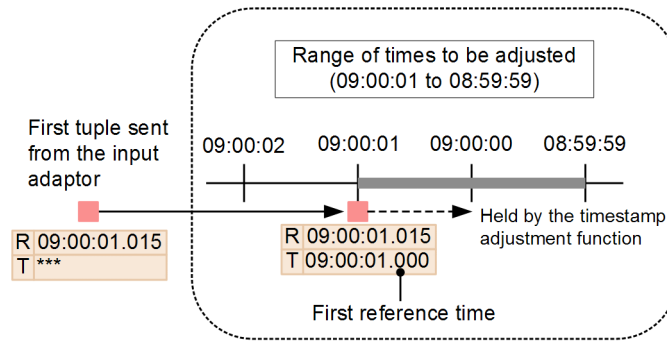
- Tuple is first tuple obtained after a query group has started
- Time set in records in the tuple is after (subsequent to) the current reference time
- Time set in records in the tuple is within the range of times to be adjusted
- Time set in records in the tuple is earlier than the range of times to be adjusted

Tuple is first tuple obtained after a query group has started

The time set in records in the tuple that was sent from the input adaptor is set as the reference time.

The figure below shows an example of the first tuple obtained after a query group has started. In this example, `sec` is specified for the unit of time and 2 is specified for the length of time to be adjusted.

Figure 10-27: Example of the first tuple obtained after a query group has started



Legend:

■ : Tuple

R HH:MM:SS.mmm : Time set in records in the tuple

T HH:MM:SS.mmm : Timestamp set by the timestamp adjustment function

When the first tuple (with 09:00:01:015 set as the time in the records) is sent from the input adaptor, the timestamp adjustment function sets the reference time to 09:00:01.

In this case, the time adjustment range is 09:00:01-08:59:59. This first tuple is held in the timestamp adjustment function as a tuple with the timestamp 09:00:01.

Time set in records in the tuple is after (subsequent to) the current reference time

If the time set in records in the tuple is after the current reference time, the tuple's time is adjusted with the following procedure:

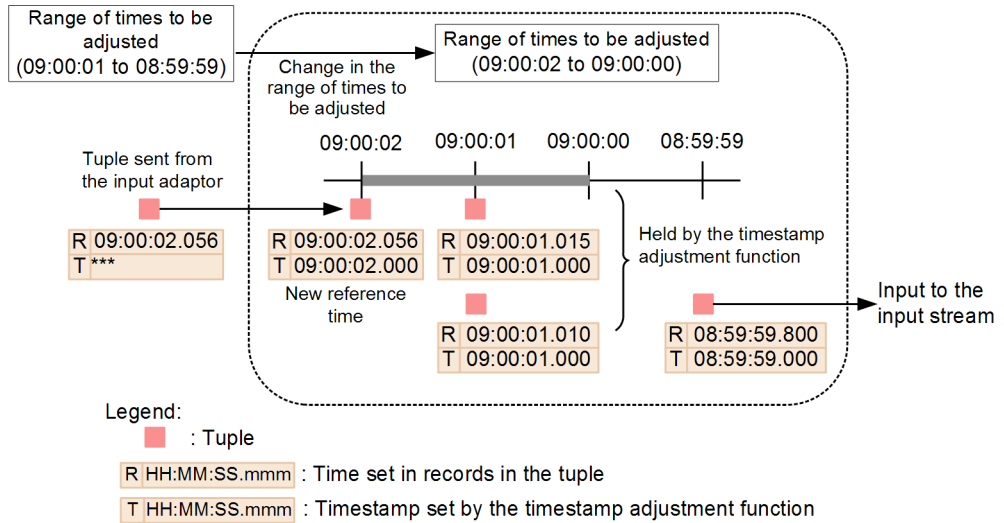
1. The new reference time is set based on the time information in the records in the tuple that has been sent. As a result, the range of times to be adjusted changes.

This tuple is held in the timestamp adjustment function.

2. If there are tuples that are being held by the timestamp adjustment function because their time was within the range of times to be adjusted based on the previous reference time, but their time is no longer within the range of times to be adjusted because of the change to the new reference time, those tuples are input to the input stream in chronological order of their timestamps. If multiple of these tuples have the same timestamp value, they are input to the input stream in the order they arrived at the SDP server.

The figure below shows an example of when the time set in records in a tuple is after the current reference time. In this example, `sec` is specified for the unit of time and 2 is specified for the length of time to be adjusted.

Figure 10-28: Example of when the time set in the records in a tuple is after (subsequent to) the current reference time



In this example, the reference time is 09 : 00 : 01 and the range of times to be adjusted is 09 : 00 : 01 - 08 : 59 : 59. When a new tuple with 09 : 00 : 02 : 056 set as the time in the records is sent from the input adaptor, the reference time changes to 09 : 00 : 02 because this tuple's time is subsequent to the existing reference time. As a result, the range of times to be adjusted changes to 09 : 00 : 02 - 09 : 00 : 00.

This new tuple is held by the timestamp adjustment function with a timestamp of 09 : 00 : 02.

The tuple with the timestamp 08 : 59 : 59 is input to the input stream because it is not within the new time adjustment range.

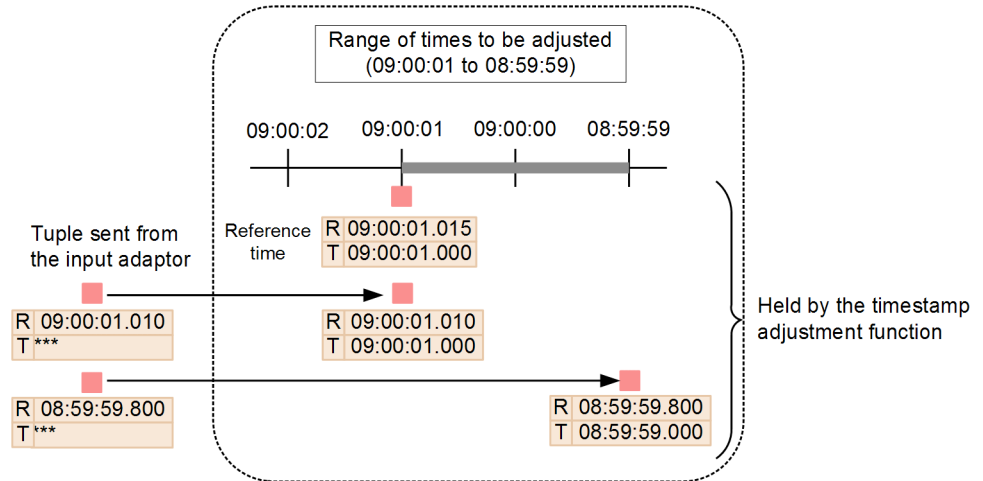
#### Time set in records in the tuple is within the range of times to be adjusted

If the time set in records in the tuple sent from the input adaptor is within the range of times to be adjusted, the tuple is held by the timestamp adjustment function.

The figure below shows an example of when the time set in records in a tuple is within the range of times to be adjusted. In this example, `sec` is specified for the unit of time and 2 is specified for the length of time to be adjusted.



Figure 10-29: Example of when the time set in records in a tuple is within the range of times to be adjusted



Legend:

■ : Tuple

R HH:MM:SS.mmm : Time set in records in the tuple

T HH:MM:SS.mmm : Timestamp set by the timestamp adjustment function

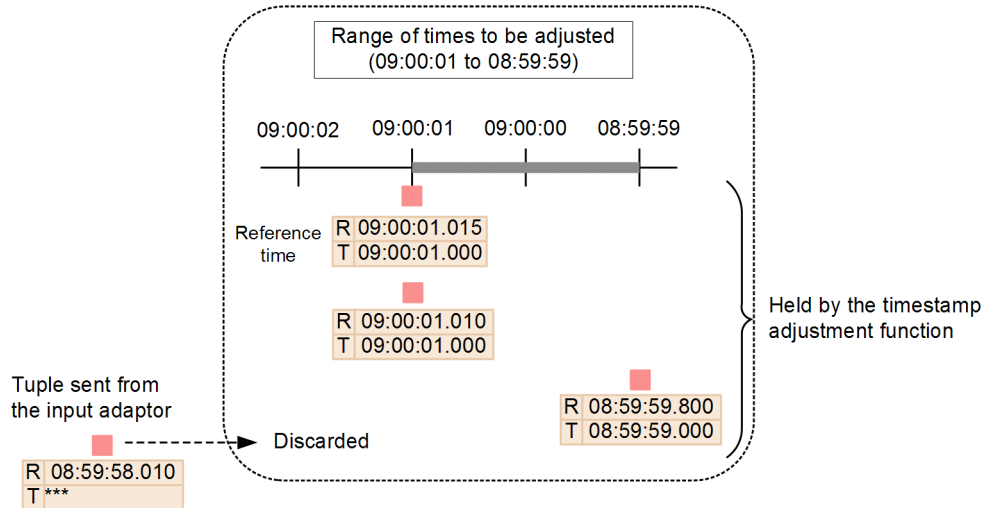
In this example, the reference time is 09:00:01 and the range of times to be adjusted is 09:00:01-08:59:59. When a new tuple with 09:00:01:010 set as the time in the records is sent from the input adaptor, it is held by the timestamp adjustment function with a timestamp of 09:00:01 because the time set in its records is within the range of times to be adjusted. Similarly, if a tuple with 08:59:59.800 set as the time in its records is sent from the input adaptor, it will be held by the timestamp adjustment function with a timestamp of 08:59:59.

Time set in records in the tuple is earlier than the range of times to be adjusted

If the time set in records in the tuple sent from the input adaptor is before the range of times to be adjusted, the timestamp adjustment function discards the tuple and outputs a log to that effect to the tuple log.

The figure below shows an example of when the time set in records in a tuple is earlier than the range of times to be adjusted. In this example, `sec` is specified for the unit of time and 2 is specified for the length of time to be adjusted.

*Figure 10-30:* Example of when the time set in records in a tuple is earlier than the range of times to be adjusted



Legend:

■ : Tuple

R HH:MM:SS.mmm : Time set in records in the tuple

T HH:MM:SS.mmm : Timestamp set by the timestamp adjustment function

In this example, the reference time is 09:00:01 and the range of times to be adjusted is 09:00:01-08:59:59. When a new tuple with 08:59:58:010 set as the time in its records is sent from the input adaptor, that tuple is discarded because the time set in its records is earlier than the range of times to be adjusted. A log to that effect is then sent to the tuple log.

## (2) Order in which tuples are input to the input stream

Tuples whose time information has been adjusted by the timestamp adjustment function are input to the input stream in ascending order of the timestamps set by the timestamp adjustment function. In the case of multiple tuples with the same timestamp value, they are input to the input stream in the order they arrived at the SDP server.

If tuples have different units of time and ranges to be adjusted by the timestamp adjustment function, the order in which they are input to the input stream depends on several factors.

This subsection presents examples for different combinations of the unit of time and the range of times to be adjusted in order to discuss the timestamp set in tuples by the timestamp adjustment function and the order in which tuples would be input to the input stream.

The examples presented here assume that tuples A through E shown in the table below are sent from the input adaptor:

Tuple	Order sent from input adaptor	Time set in records in the tuple
A	1	2009/03/01 12:15:22:345678901
B	2	2009/03/01 12:15:22:123456789
C	3	2009/03/01 12:15:23:123456789
D	4	2009/03/01 12:15:22:890123456
E	5	2009/03/01 12:15:24:123456789

When `sec` is specified as the unit of time and 1 is specified for the range of times to be adjusted:

The table below shows the timestamps set in these tuples by the timestamp adjustment function and the order in which the tuples are input to the input stream when `sec` is specified as the unit of time and 1 is specified for the range of times to be adjusted.

Note that when `sec` is specified as the unit of time, the timestamp is expressed in whole seconds and any fractional part of a second is discarded.

Tuple	Input order	Timestamp set by timestamp adjustment function
A	1	2009/03/01 12:15:22:000000000
B	2	2009/03/01 12:15:22:000000000
C	4	2009/03/01 12:15:23:000000000
D	3	2009/03/01 12:15:22:000000000
E	5	2009/03/01 12:15:24:000000000

In this example, all tuples are input to the input stream because they all fall within the time adjustment range.

When `msec` is specified as the unit of time and 999 is specified for the range of times to be adjusted:

The table below shows the timestamps set in these tuples by the timestamp adjustment function and the order in which the tuples are input to the input stream when `msec` is specified as the unit of time and 999 is specified for the range of times to be adjusted.

Note that when `msec` is specified as the unit of time, the timestamp is expressed

in whole milliseconds and any fractional part of a millisecond is discarded.

<b>Tuple</b>	<b>Input order</b>	<b>Timestamp set by timestamp adjustment function</b>
A	2	2009/03/01 12:15:22:345000000
B	1	2009/03/01 12:15:22:123000000
C	4	2009/03/01 12:15:23:123000000
D	3	2009/03/01 12:15:22:890000000
E	5	2009/03/01 12:15:24:123000000

In this example, all tuples are input to the input stream because they are all within the time adjustment range.

When `usec` is specified as the unit of time and 999 is specified for the range of times to be adjusted:

The table below shows the timestamps set in these tuples by the timestamp adjustment function and the order in which the tuples are input to the input stream when `usec` is specified as the unit of time and 999 is specified for the range of times to be adjusted.

Note that when `usec` is specified as the unit of time, the timestamp is expressed in whole microseconds and any fractional part of a microsecond is discarded.

<b>Tuple</b>	<b>Input order</b>	<b>Timestamp set by timestamp adjustment function</b>
A	1	2009/03/01 12:15:22:345678000
B	Discarded	None set
C	2	2009/03/01 12:15:23:123456000
D	Discarded	None set
E	3	2009/03/01 12:15:24:123456000

In this example, once tuple A's timestamp becomes the reference time, tuple B is discarded because its time is earlier than this reference time and is outside the range of times to be adjusted. Similarly, tuple D is discarded because its time is earlier than that of tuple C and is also outside the range of times to be adjusted.

When `sec` is specified as the unit of time and 0 is specified for the range of times to be adjusted:

The table below shows the timestamps set in these tuples by the timestamp adjustment function and the order in which the tuples are input to the input stream when `sec` is specified as the unit of time and 0 is specified for the range of times

to be adjusted.

Note that when `sec` is specified as the unit of time, the timestamp is expressed in whole seconds and any fractional part of a second is discarded.

Tuple	Input order	Timestamp set by timestamp adjustment function
A	1	2009/03/01 12:15:22:000000000
B	2	2009/03/01 12:15:22:000000000
C	3	2009/03/01 12:15:22:000000000
D	Discarded	None set
E	4	2009/03/01 12:15:23:000000000

In this example, only the reference time is within the time adjustment range. When tuple C is sent, its timestamp becomes the reference time and tuple D is discarded because its time is earlier than the reference time.

#### 10.8.4 Processing after tuples have been input

The timestamp adjustment function changes the reference time when the time information set in records in the tuples sent from the input adaptor changes and then inputs to the input stream input-eligible tuples it has placed on hold. Once the tuple input operation from the input adaptor is completed, the timestamp adjustment function's reference time will no longer be subject to change, which means that it will not be possible for tuples still being held by the function to be input to the input stream.

However, the `putEnd` method of the `StreamInput` object can be used to input to the input stream the tuples that are still being held by the timestamp adjustment function after a tuple input operation has been completed. If you are using a custom adaptor, you can input to the input stream these tuples held by the function by issuing the `putEnd` method from a data transmission application. If you are using a standard adaptor, the `putEnd` method is issued upon completion of the tuple input operation.

When the `putEnd` method is issued, the following statuses occur in the timestamp adjustment function:

1. Stops accepting new tuples

The timestamp adjustment function goes onto a status in which it no longer accepts new tuples.

When the `putEnd` method has been issued for all input streams within the query group, the timestamp adjustment function prepares to resume accepting tuples.

2. Resumes accepting new tuples

When the timestamp adjustment function completes its preparations, it starts

accepting new tuples again. Once the timestamp adjustment function resumes accepting tuples, it goes onto the same status as when the query group started.

You can check the status of the timestamp adjustment function by using the `isStarted` method of the `StreamInput` object. If you issue the `isStarted` method when the timestamp adjustment function is in status 1 above, `false` (function has not resumed accepting tuples) is returned. If you issue the `isStarted` method when the timestamp adjustment function is in status 2 above, `true` (function has resumed accepting tuples) is returned.

### 10.8.5 Operation when the query group shuts down

The tuple processing performed by the timestamp adjustment function when the query group is terminated depends on the query group's status (normal termination, forced termination, or shutdown).

The table below shows the tuple processing performed by the timestamp adjustment function in response to these query group statuses.

*Table 10-5:* Tuple processing by the timestamp adjustment function when the query group is terminated

No.	Query group's status	Processing by timestamp adjustment function		
		Stops accepting new tuples	Inputs to the input stream all tuples being held by the function	Discards all tuples being held by the function
1	Normal termination	Y	Y	N
2	Forced termination	Y	N	Y
3	Shutdown	Y	N	Y

Legend:

Y: Performed

N: Not performed

### 10.8.6 Tuple retention period

The timestamp adjustment function places tuples on hold in order to adjust their time. If too many tuples are on hold, a memory shortage might occur. To achieve stable resource utilization throughout entire system, you can set a maximum number of tuples that can be kept on hold by the timestamp adjustment function.

When a new tuple is sent from the input adaptor, the timestamp adjustment function adjusts the time and checks the number of tuples that are currently being held. When

the number of tuples on hold has reached the maximum value, the timestamp adjustment function stops placing any new tuples on hold and shuts down the query group. When a query group is shut down, all its tuples being held by the function are discarded.

You specify the maximum number of tuples that can be held by the timestamp adjustment function in the `stream.maxKeepTupleCount` parameter in the system configuration property file, query group property file, or stream property file. For details about the `stream.maxKeepTupleCount` parameter, see *8.6 System configuration property file (system\_config.properties)*.

### 10.8.7 Selecting tuples by filtering

By filtering tuple record values using a predefined conditional expression, you can select the tuples to be held by the timestamp adjustment function, thereby reducing the number of tuples to be placed on hold.

You use the `stream.filterMode` parameter in the query group property file or stream property file to specify whether or not tuples are to be filtered by the timestamp adjustment function. A conditional expression for filtering is specified in the `stream.filterCondition` parameter. For details about the `stream.filterMode` and `stream.filterCondition` parameters, see *8.6 System configuration property file (system\_config.properties)*.

### 10.8.8 Timestamp adjustment settings

Settings for the timestamp adjustment function are required in the following definition files:

- System configuration property file, query group property file, or stream property file
  - `stream.timestampAccuracy` parameter  
Specifies the unit of time and the time adjustment range that are to be used by the timestamp adjustment function
  - `stream.maxKeepTupleCount` parameter  
Specifies the maximum number of tuples to be held by the timestamp adjustment function.
- Query group property file or stream property file
  - `stream.filterMode` parameter  
Specifies whether or not tuples are to be filtered by the timestamp adjustment function.
  - `stream.filterCondition` parameter  
Specifies a conditional expression for filtering tuples by the timestamp

adjustment function.



## Chapter

---

# 11. Details of Flex Dashboard Settings

---

This chapter provides the details of the settings required in order to output data processed by the stream data processing engine to Flex Dashboard.

- 11.1 Organization of this chapter
- 11.2 Dashboard Server internal settings file (usrconf.properties)
- 11.3 Dashboard Viewer window layout file
- 11.4 Example definitions for dashboard output

---

## 11.1 Organization of this chapter

---

This chapter discusses the settings required in order to output stream data summary analysis results to a dashboard, shows the organization of the windows, and provides an example of the definition files.

You must specify Dashboard Server settings, as well as Dashboard Viewer settings of Flex Dashboard, in order to output data to a dashboard. The table below lists the sections to be referenced to learn about the Flex Dashboard settings and to see an example of the definition files.

*Table 11-1:* Flex Dashboard settings and sections to be referenced

No.	Flex Dashboard settings	Section
1	Dashboard Server internal settings file ( <code>usrconf.properties</code> )	11.2
2	Dashboard Viewer window layout file	11.3
3	Example of definitions for dashboard output	11.4

## 11.2 Dashboard Server internal settings file (usrconf.properties)

To use Dashboard Server, you must specify settings in the Dashboard Server internal settings file (`usrconf.properties`). This section discusses how to create a Dashboard Server internal settings file, explains the keys that can be set, and provides a coding example.

### (1) How to create the file

We recommend that you use the sample file provided during installation to create a server internal settings file. The sample file is stored in the directory shown below. Copy the file and then edit the copy.

`installation-directory\samples\httppacket\web\containers\uCSDPAF_Server\usrconf\`

The only keys in the sample file that you should edit are the `REFRESH_INTERVAL` and `RETRY_NUM` keys; no other keys should be edited.

Store the created server internal settings file in the following directory:

`installation-directory\psb\CC\web\containers\uCSDPAF_Server\usrconf\`

### (2) Keys that can be set

The table below lists and describes the keys that can be set in the server internal settings file.

Table 11-2: Keys that can be set in the server internal settings file

No.	Key name	Description	Default value
1	<code>REFRESH_INTERVAL</code>	Specifies the interval (in milliseconds) at which Dashboard Server is to access the dashboard output connector and refresh the dashboard-display data, as an integer from 0 to 600000.	2000
2	<code>RETRY_NUM</code>	Specifies the number of retries that will be permitted when Dashboard Server's refreshing of the dashboard-display data fails, as an integer from 0 to 100.	0

You specify these settings in the server internal settings file in the Java property format (`key=value`).

### (3) Example

The following shows an example of specifying the settings in the server internal settings file:

## 11. Details of Flex Dashboard Settings

```
REFRESH_INTERVAL=2000  
RETRY_NUM=0
```

This example sets the refresh interval to 2000 (milliseconds) and the retry count to 0.

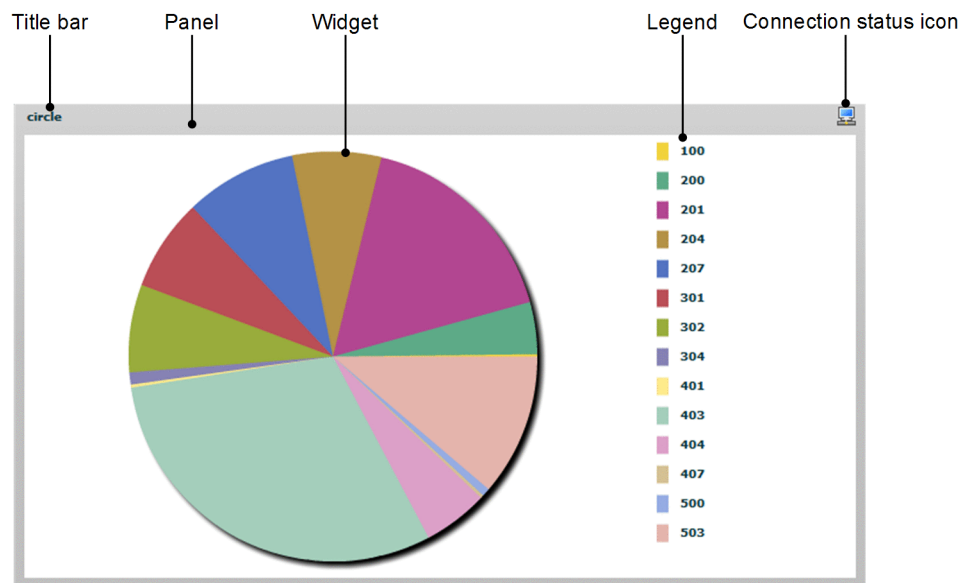
## 11.3 Dashboard Viewer window layout file

You can change the layout of the data (such as charts) that is displayed on the dashboard by editing a Dashboard Viewer window layout file.

This section discusses the configuration of a Dashboard Viewer window that can be changed by editing its Dashboard Viewer window layout file, explains details of a Dashboard Viewer window layout file, and lists the types of dashboard-display data handled by Dashboard Viewer.

### 11.3.1 Configuration of Dashboard Viewer window

The figure below shows the configuration of a Dashboard Viewer window that can be changed by editing its Dashboard Viewer window layout file. The maximum window size that Dashboard Viewer can display is 1,920 pixels x 1,200 pixels.



#### Widget

A widget displays the analysis result of stream data processing with the specified format. The types of widgets that are available include bar graphs, pie charts, scatter charts, line graphs, and tables. Note that only one widget can be displayed on a canvas.

The following discusses the data display order for widgets and how to change from a widget to a different URL.

**Data display order**

- Bar graph, pie chart, and broken line graph

Data on the abscissa is sorted in ascending order.

- Table

Data in column 1 is sorted in ascending order.

You sort the data in the column by clicking the column header. Each time the column header is clicked, the sort order toggles between ascending order and descending order. There is no second-level sorting, so if the sort column contains multiple items with the same value, these data items are not further sorted.

**Changing the URL**

Double-clicking in the displayed widget displays a new window containing information about a target URL that has been specified in the Dashboard Viewer window layout file. The table below describes how to change the URL for each type of widget.

*Table 11-3:* How to change the URL for each type of widget

No.	Widget type	How to change the URL
1	Bar graph	<ul style="list-style-type: none"> <li>• Double-click a bar in the bar graph.</li> <li>• If a legend is displayed, single-click the legend.</li> </ul>
2	Pie chart	<ul style="list-style-type: none"> <li>• Click a sector in the pie chart.</li> <li>• If a legend is displayed, single-click the legend.</li> </ul>
3	Scatter chart	If a legend is displayed, single-click the legend.
4	Broken line graph	If a legend is displayed, single-click the legend.
5	Table	Double-click a row in the table.

**Panel**







The panel area displays a widget and title bar. The panel is displayed behind the widget.

**Title bar**

The title bar area displays the title for the widget.

**Connection status icon**

The connection status icon indicates the widget's connection status. The table below shows the icons and describes the connection status each icon indicates.

No.	Icon	Status	Description
1		Connection successful	Displayed when widget data has been acquired successfully.
2		Connecting	Displayed when Dashboard Viewer is starting or before data is received after connection has been re-established.
3		Connection canceled	Displayed when acquisition of widget data has been canceled.
4		Connection failed	Displayed when data acquisition has failed.
5		Error occurred	Displayed when there is an inconsistency between the definition in the Dashboard Viewer window layout file and the received data.
6		Not connected	Displayed when a required connection-related item has not been defined in the Dashboard Viewer window layout file.

If you click the icon for status 1 or 2, acquisition of widget data is canceled. In such a case, the data in the window is placed in the status when the icon is clicked.

If you click the icon for status 3 or 4, connection establishment is retried. In such a case, the icon changes to the status 2 icon.

#### Legend

The legend area provides a legend for the widget. A legend is displayed when legend display is set to ON in the Dashboard Viewer window layout file. If there is not enough room for all of the legend, a portion of it might not be displayed.

### 11.3.2 Details of a Dashboard Viewer window layout file

This subsection discusses how to create a Dashboard Viewer window layout file, shows the specification format, and explains details of the definition. It also discusses the handling of data when dashboard-display data is displayed with Dashboard Viewer.

#### (1) How to create the file

The name of a Dashboard Viewer window layout file is *window-name.xml*. You can specify for *window-name* any desired name consisting of 1 to 16 single-byte alphanumeric characters.

We recommend that you use the sample file provided during installation to create each Dashboard Viewer window layout file. Copy the following sample file and then edit the copy:

*installation-directory\samples\httppacket\web\containers\uCSDPAF\_Serv*

```
er\layout\dashboard.xml
```

Store the created Dashboard Viewer window layout file in the following directory:

```
installation-directory\psb\CC\web\containers\uCSDPAF_Server\layout\#
#
```

The user must create the `layout` directory.

The following notes apply to creating a Dashboard Viewer window layout file.

- The character encoding used in a Dashboard Viewer window layout file must be UTF-8.
- Single-byte spaces, tabs, and linefeed codes at the beginning or end of the value specified in a tag are ignored.
- Any tag that is not described in (3) *Details of the definition* is ignored.
- If a tag is omitted, the default value is assumed.

## (2) Specification format

The following shows an example of the specification format for a Dashboard Viewer window layout file.

```
<dashboardDisplay>
  <connectionSetting>
    <serverName>localhost</serverName>
    <portNo>20421</portNo>
  </connectionSetting>
  <widgetGroup>
    <widget>
      <kind>table</kind>
      <dataSetting>
        <interval>5000</interval>
        <data1>
          <dataName>InprocessAPTTest/OutputAdaptor1/outputer</dataName>
          <filter>
            <columnName>SUBTIME</columnName>
            <condition>ge</condition>
            <value>100</value>
          </filter>
        </data1>
      </dataSetting>
      <dataDisplay>
        <threshold>
          <columnName>SUBTIME</columnName>
          <condition>ge</condition>
          <value>5000</value>
        </threshold>
      </dataDisplay>
    </widget>
  </widgetGroup>
</dashboardDisplay>
```



**(3) Details of the definition**

`dashboardDisplay` tag (all definition information)

Defines the entire interface for a dashboard window.

`connectionSetting` tag (definition of the connection-target server)

Defines information about the connection-target server.

`serverName` tag

Specifies the name of the connection-target SDP server, as 0 to 255 single-byte alphanumeric characters. The default is the null character. When the null character is specified, no connection with a server will be established.

`portNo` tag

Specifies the port number of the connection-target SDP server, as a single-byte integer in the range from 0 to 65535. The default is the null character. When the null character is specified, no connection with a server will be established.

`widgetGroup` tag (definitions of widgets)

Defines the widgets.

`widget` tag (definition of a widget)

Defines a widget.

`kind` tag

Specifies the type of widget to be displayed. The default value is `table`. The permitted values are as follows:

- `bar`  
Vertical bar graph
- `circle`  
Pie chart
- `x-y`  
Scatter chart
- `trend`  
Broken line graph
- `table`  
Table

`widgetSetting` tag (definition of panel for the widget)

Defines information about the panel for the widget.

`title` tag

Specifies a title for the widget, as a string of 0 to 32 characters. The default is the null character.

`border` tag (definition of a border for the widget)

Defines a border for the widget.

`Color` tag

Specifies a color (RGB color) for the widget's border. The default value is `192,192,192` (gray).

`bg` tag (definition of a background for the widget)

Defines a background for the widget.

When `table` is specified in the `kind` tag under the `widget` tag, no background color or background image will be displayed.

`Color1` tag

Specifies a background color (RGB color) for the widget. The default value is `255,255,255` (white).

`bgImage` tag

Specifies the URL of an image file that is to be displayed as the background for the widget, as 0 to 255 single-byte alphanumeric characters beginning with `http://`. The default is the null character.

You can specify an image file that is in GIF or JPG format. If the specified file contains animation, the operation is not guaranteed. If the size of the widget differs from the size of the image, the image is displayed centered in the widget; the image is not enlarged or reduced.

When the null character is specified or if the specified image cannot be displayed, the background color is set.

`dataSetting` tag (definition of data settings for the widget)

Defines information about setting the data for the widget.

`interval` tag

Specifies the interval to be used as Dashboard Viewer's display refresh interval and as Dashboard Viewer's data acquisition interval (the interval for acquiring data from Dashboard Server), as a single-byte integer in the range from 1 to 60000 (representing milliseconds). The default value is 3000.

Note that immediately after a startup or re-connection, it might take about twice as much time as the specified refresh interval before data is displayed.

`data1` to `data5` tags (definition of series of data items)

Defines the data to be displayed. For each data series, a maximum of 1,024 data items can be retained. If the number of data items in a series exceeds 1,024, the excess items are discarded sequentially beginning with the first data item that arrived at Dashboard Server.

When the value of the `kind` tag under the `widget` tag is `bar`, `circle`, or `table`, any values in `data2` through `data5` tags are not used.

`legendName` tag

Specifies the name that is to be displayed as this data's legend, as a string of 0 to 30 characters. The default is the null character.

When the value of the `kind` tag under the `widget` tag is `bar`, `circle`, or `table`, any specified legend name is not displayed.

`dataName` tag

Specifies the connection name of the data that is to be obtained from Dashboard Server, as 0 to 255 single-byte alphanumeric characters. The default is the null character.

When the null character is specified, a data connection is not used. If the specified connection name is not found, an error occurs.

When the specified connection name is not found or connection fails after Dashboard Viewer has started, the connection status icon changes to the connection-failed icon.

`column1` to `column2` tags

Specifies the columns to be displayed in the data obtained from Dashboard Server, as 0 to 255 single-byte alphanumeric characters. The default is the null character.

The values depend on the value of the `kind` tag under the `widget` tag:

- `bar`  
Specify the abscissa (ID) in the `column1` tag and the ordinate (value) in the `column2` tag.
- `circle`  
Specify a sector (ID) in the `column1` tag and the size of the sector (value) in the `column2` tag.
- `x-y`

Specify the abscissa (value of x-axis) in the `column1` tag and the ordinate (value of y-axis) in the `column2` tag.

- `trend`

Specify the abscissa (time) in the `column1` tag and the ordinate (value) in the `column2` tag.

- `table`

No values are used.

When the value of the `kind` tag is `bar`, `circle`, `x-y`, or `trend` and the null character is specified in the `column1` or `column2` tag, data is not displayed.

If any data has been acquired since Dashboard Viewer started, but the data does not contain a specified column, the `KFSP46954-E` error message is displayed. If the data type is not correct, the `KFSP46955-E` error message is displayed.

#### `color` tag

When the value of the `kind` tag under the `widget` tag is `x-y` or `trend`, specifies a color (RGB color) to be used to display the data. The default value is `0, 0, 0` (black).

#### `filter` tag (filter definition)

Defines information about a filter for the acquired data. You can set only one filter for each data.

##### `columnName` tag

Specifies the name of the column whose data is to be filtered, as 0 to 255 single-byte alphanumeric characters. The default is the null character.

The data that does not satisfy the filter condition defined in the `condition` tag is hidden. If the null character is specified, no data is hidden.

If any data has been acquired since Dashboard Viewer started, but the data does not contain the specified column, the `KFSP46954-E` error message is displayed. If the data type is not correct, the `KFSP46955-E` error message is displayed.

##### `condition` tag

Specifies the relationship between the column value and the value specified in the `value` tag. The default value is `eq`.

The permitted values depend on the data type of the value in the column specified in the `columnName` tag, as described below:

- When a string-type (`STRING`) column is specified in the `columnName`

`tag`

The permitted values are `eq (=)` and `ne (!=)`.

- When an integer-type (`BYTE`, `SHORT`, `INT`, or `LONG`) column is specified in the `columnName` tag

The permitted values are `lt (>)`, `gt (<)`, `le (>=)`, `ge (<=)`, `eq (=)`, and `ne (!=)`.

If any data has been acquired since Dashboard Viewer started, but the value specified for the condition is invalid, the `KFSP46961-E` error message is displayed.

`value` tag

Specifies the value to be compared.

The permitted value depends on the data type of the value in the column specified in the `columnName` tag, as described below:

- When a string-type (`STRING`) column is specified in the `columnName` tag

Specify 0 to 255 single-byte alphanumeric characters.

- When an integer-type (`BYTE`, `SHORT`, `INT`, or `LONG`) column is specified in the `columnName` tag

Specify a 32-bit integer-type value, in the range from `-2147483648` to `2147483647`.

If any data has been acquired since Dashboard Viewer started, but the value specified for the condition is invalid, the `KFSP46961-E` error message is displayed.

A value must be specified in the `value` tag.

`dataDisplay` tag (definition of data display)

Defines information about how data is displayed in the widget.

`threshold` tag (definition of threshold)

Defines a threshold value for the data that is to be displayed.

`columnName` tag

Specifies the column name for which a threshold is to be specified, as 0 to 255 single-byte alphanumeric characters. The default is the null character.

When the value of the `kind` tag under the `widget` tag is `table`, all characters in the row that satisfies the specified threshold value are displayed in red. If the null character is specified, display highlighting is disabled.

If any data has been acquired since Dashboard Viewer started, but the table data to be displayed does not contain the specified column, the KFSP46954-E error message is displayed. If the data type is not correct, the KFSP46955-E error message is displayed.

#### condition tag

Specifies the relationship between the column value and the value specified in the `value` tag. The default value is `eq`.

The permitted values depend on the data type of the value in the column specified in the `columnName` tag, as described below:

- When a string-type (STRING) column is specified in the `columnName` tag

The permitted values are `eq (=)` and `ne (!=)`.

- When an integer-type (BYTE, SHORT, INT, or LONG) column is specified in the `columnName` tag

The permitted values are `lt (>)`, `gt (<)`, `le (>=)`, `ge (<=)`, `eq (=)`, and `ne (!=)`.

If the value specified for the condition is invalid, the KFSP46961-E error message is displayed.

#### value tag

Specifies the value to be compared.

The permitted value depends on the type of value in the column specified in the `columnName` tag, as described below:

- When a string-type (STRING) column is specified in the `columnName` tag

Specify 0 to 255 single-byte alphanumeric characters.

- When an integer-type (BYTE, SHORT, INT, or LONG) column is specified in the `columnName` tag

Specify a 32-bit integer-type value, in the range from -2147483648 to 2147483647.

If the value specified for the condition is invalid, the KFSP46961-E error message is displayed.

A value must be specified in the `value` tag.

#### xAxis tag (definition of x-axis)

Defines the x-axis settings.

**max tag**

When the value of the `kind` tag under the `widget` tag is `x-y`, specifies the maximum value for the data to be plotted in the x-axis direction, expressed as the null character or a 32-bit integer-type value in the range from -2147483648 to 2147483647. The default is the null character.

If the specified maximum value (`max tag` value) is equal to or less than the specified minimum value (`min tag` value), the KFSP46961-E error message is displayed.

If the null character is specified for the maximum value, the scale is changed automatically.

If the value of the `kind` tag under the `widget` tag is not `x-y`, any specified maximum value is not set for the x-axis.

**min tag**

When the value of the `kind` tag under the `widget` tag is `x-y`, specifies the minimum value for the data to be plotted in the x-axis direction, expressed as the null character or a 32-bit integer-type value in the range from -2147483648 to 2147483647. The default is the null character.

If the specified maximum value (`max tag` value) is equal to or less than the specified minimum value (`min tag` value), the KFSP46961-E error message is displayed.

If the null character is specified for the minimum value, the scale is changed automatically.

If the value of the `kind` tag under the `widget` tag is not `x-y`, any specified minimum value is not set for the x-axis.

**yAxis tag (definition of y-axis)**

Defines the y-axis settings.

**max tag**

When the value of the `kind` tag under the `widget` tag is `x-y` or `trend`, specifies the maximum value for the data to be plotted in the y-axis direction, expressed as the null character or a 32-bit integer-type value in the range from -2147483648 to 2147483647. The default is the null character.

If the specified maximum value (`max tag` value) is equal to or less than the specified minimum value (`min tag` value), the KFSP46961-E error message is displayed.

If the null character is specified for the maximum value, the scale is changed automatically.

If the value of the `kind` tag under the `widget` tag is neither `x-y` nor `trend`, any specified maximum value is not set for the y-axis.

`min` tag

When the value of the `kind` tag under the `widget` tag is `x-y` or `trend`, specifies the minimum value for the data to be plotted in the y-axis direction, expressed as the null character or a 32-bit integer-type value in the range from -2147483648 to 2147483647. The default is the null character.

If the specified maximum value (`max` tag value) is equal to or less than the specified minimum value (`min` tag value), the KFSP46961-E error message is displayed.

If the null character is specified for the minimum value, the scale is changed automatically.

If the value of the `kind` tag under the `widget` tag is neither `x-y` nor `trend`, any specified minimum value is not set for the y-axis.

`xTimeRange` tag (definition of time axis)

Defines settings for the x-axis (time axis).

`range` tag

When the value of the `kind` tag under the `widget` tag is `trend`, specifies the amount of time to be plotted on the x-axis for the displayed data, expressed as a single-byte integer in the range from 1 to 86400 (representing seconds). The specified range of time must fit within a single window. The default value is 300.

If the value of the `kind` tag under the `widget` tag is not `trend`, any specified time is not set for the x-axis.

`SliderFlag` tag

When the value of the `kind` tag under the `widget` tag is `trend`, specifies whether or not a slider is to be displayed for the time axis (x-axis). The default value is `OFF`.

The permitted values (in single-byte upper-case letters) are as follows:

- `ON`  
Displays a slider.
- `OFF`  
Does not display a slider.

If the value of the `kind` tag under the `widget` tag is not `trend`, the slider display setting for the time axis (x-axis) is disabled.



`tableProperties` tag (table definition)

Defines settings for a table.

`columnHeaderList` tag (definition of column headers)

When the value of the `kind` tag under the `widget` tag is `table`, defines information about the column headers that are to be displayed.

`column` tag

Defines information about a column that is to be displayed.

If more than 16 `column` tags are specified, the `KFSP46966-E` error message is displayed.

`actualName` tag

Specifies the name of the column whose data is to be displayed, as 0 to 255 single-byte alphanumeric characters; this must be a column contained in the received data. The default is the null character.

When the null character is specified, data is not sorted.

If any data has been acquired since Dashboard Viewer started, but the data does not contain the specified column, the `KFSP46954-E` error message is displayed.

`displayName` tag

Specifies the name to be displayed for this column in the column header, as a string of 0 to 32 characters. The default is the null character.

`legend` tag (legend definition)

Defines information about a legend.

`displayFlag` tag

Specifies whether or not a legend is to be displayed on the right side of a chart. The default value is `OFF`.

The permitted values (in single-byte upper-case letters) are as follows:

- `ON`

Displays a legend.

- `OFF`

Does not display a legend.

The content of the legend that will be displayed depends on the value of the `kind` tag under the `widget` tag:

- `bar`

Displays a list of the names that are displayed on the abscissa.

- `circle`

Displays a list of the names that are assigned to the sectors in the pie chart.

- `x-y`

Displays the value specified in the `legendName` tag for each of the `data1` through `data5` tags.

- `trend`

Displays the value specified in the `legendName` tag for each of the `data1` through `data5` tags.

- `table`

Does not display a legend, even when `ON` is specified.

#### `width` tag

Specifies the width of the legend area in pixels, as a single-byte integer in the range from 1 to 2048; or, specifies the width of the legend area as a single-byte integer in the range from 1 to 100 followed by the percent sign (%). A value specified using single-byte integers and % is treated as the ratio of the legend's width to the widget's width. If the legend cannot be displayed based on the specified value, the legend is resized so that it is displayable.

#### `jumpToURL` tag (definition of URL change)

Defines information about a URL change.

#### `URL` tag

Specifies a destination URL, as 0 to 128 single-byte alphanumeric characters beginning with `http://`. The default is the null character.

If the null character is specified, no URL change will be performed.

#### `parameter1` tag

Specifies the name of argument 1 in the information contained in the selected data that is to be passed to the destination URL, as 0 to 16 single-byte alphanumeric characters. The default is the null character.

If the null character is specified, no information will be passed to the destination URL.

#### `value1` tag

Specifies the value set in argument 1 in the information contained in the selected data that is to be passed to the destination URL, as 0 to 16

single-byte alphanumeric characters. The default is the null character.

The permitted values (the value that can be passed to the destination URL) depends on the value of the `kind` tag under the `widget` tag:

- `bar`  
Specifies the value of `ID` (abscissa).
- `circle`  
Specifies the value of `ID` (abscissa).
- `x-y`  
Specifies the value of `legend`.
- `trend`  
Specifies the value of `legend`.
- `table`  
Specifies the values of `column1` through `column16`.

When `table` is specified in the `kind` tag under the `widget` and any data has been acquired since Dashboard Viewer started, but the value of `[n]` in `column [n]` is greater than the actual number of columns to be displayed, the KFSP46961-E error message is displayed. If the type of `column [n]` is not a character string, the KFSP46955-E error message is displayed.

#### (4) Handling of types by Dashboard Viewer

When Dashboard Viewer is used to display dashboard-display data, Dashboard Viewer handles the types of the dashboard-display data as shown in the table below.

No.	Type that can be specified in the dashboard output connector definition	Handling of type by Dashboard Viewer	Value range
1	BYTE	Integer	$-2^{31}$ to $(2^{31})-1$
2	SHORT		
3	INT		
4	LONG	Integer	$-2^{53}$ to $2^{53}$ #1
5	FLOAT	Double-precision real number#2	$1.79769313486231 \times (10^{308})$ to $-1.79769313486231 \times (10^{308})$
6	DOUBLE		

No.	Type that can be specified in the dashboard output connector definition	Handling of type by Dashboard Viewer	Value range
7	BIG_DECIMAL	Character string type	0 to 1,024 characters <sup>#3</sup>
8	STRING		
9	DATE	Time type <sup>#4</sup>	Maximum <sup>#1</sup> : 1970-01-01 at 00:00:0.000
10	TIME		Minimum <sup>#1</sup> : 9999-12-31 at 23:59:59.999
11	TIMESTAMP		

#1

If a received value is less than the minimum value, it is replaced with the minimum value. If a received value is greater than the maximum value, it is replaced with the maximum value.

#2

The value range complies with IEEE 754. Rounding of values does not occur between the dashboard output connector and Dashboard Viewer.

#3

If a received character string is longer than the maximum length, it is truncated at the maximum length. A double-byte character is treated as a single character.

#4

The following shows the display format for the `TIMESTAMP` type:

```
YYYY-MM-DD hh:mm:ss.SSS
```

*YYYY*: Year

*MM*: Month

*DD*: Day

*hh*: Hour

*mm*: Minute

*ss*: Second

*SSS*: Millisecond

## 11.4 Example definitions for dashboard output

This section presents example definitions of adaptor configuration definition files for displaying the data listed below on a dashboard and provides details of the example.

- Displaying only the most recent stream data summary analysis results
- Displaying stream data summary analysis results including historical data

This section discusses output adaptor definition. For details about input adaptors, see the description of input adaptors in *9.13.2 Example 2*.

### 11.4.1 Output adaptor definition (displaying the most recent data)

To display the most recent stream data summary analysis results on a dashboard as a bar graph, pie chart, scatter chart, or table, you specify definitions in such a manner that only the most recent `rstream` calculation results based on the query definition file are displayed.

This subsection presents a coding example of an output adaptor definition and then discusses the details of the example.

#### (1) Coding example

```
<adp:OutputAdaptorDefinition name="OutputAdaptor1"
charCode="MS932" lineFeed="CR_LF">

    <!-- CB definition for receiving -->
    <cb:ReceiveCBDefinition
class="jp.co.Hitachi.soft.sdp.adaptor.callback.sendreceive.Rec
eiveCBImpl" name="receiver">
        <cb:streamInfo name="QUERY"
querygroup="Inprocess_QueryGroupTest"/>
    </cb:ReceiveCBDefinition>

    <!-- CB definition for data editing -->
    <cb:DataEditCBDefinition
class="jp.co.Hitachi.soft.sdp.adaptor.callback.dataedit.mappin
g.OutputMappingCBImpl" name="editor1">
        <!-- Mapping definition -->
        <map:MappingDefinition ioType="OUTPUT">
            <map:source>
                <map:streams>
                    <map:stream name="QUERY"
querygroup="Inprocess_QueryGroupTest">
                        <map:column name="sendip"    type="STRING"/>
                        <map:column name="subtime"    type="LONG"/>
                    </map:stream>
                </map:streams>
            </map:source>
        </map:MappingDefinition>
    </cb:DataEditCBDefinition>
</adp:OutputAdaptorDefinition>
```

```

        </map:streams>
    </map:source>
    <map:target/>
    <map:intermediate>
        <map:mappings source="QUERY"
querygroup="Inprocess_QueryGroupTest" target="RECORD1">
            <map:map source="sendip"    target="SEND_IP"/>
        >
            <map:map source="subtime"    target="SUBTIME"/>
        >
        </map:mappings>
    </map:intermediate>
    </map:MappingDefinition>
</cb:DataEditCBDefinition>

    <!-- CB definition for data editing -->
    <cb:DataEditCBDefinition
class="jp.co.Hitachi.soft.sdp.adaptor.callback.dataedit.mappin
g.InputMappingCBImpl" name="editor2">
        <!-- Mapping definition -->
        <map:MappingDefinition ioType="OUTPUT">
            <map:source/>
            <map:target>
                <map:records>
                    <map:record name="RECORD2" >
                        <map:field name="SEND_IP"    type="STRING"/>
                        <map:field name="SUBTIME"    type="LONG"/>
                    >
                        <map:field name="GET_TUPLE_TIME"
type="TIMESTAMP"/>
                    </map:record>
                </map:records>
            </map:target>
            <map:intermediate>
                <map:mappings source="RECORD1" target="RECORD2">
                    <map:map source="SEND_IP"    target="SEND_IP"/>
                    <map:map source="SUBTIME"    target="SUBTIME"/>
                    <map:map function="getTupleTime"
target="GET_TUPLE_TIME"/>
                </map:mappings>
            </map:intermediate>
        </map:MappingDefinition>
    </cb:DataEditCBDefinition>

    <cb:OutputCBDefinition
class="jp.co.Hitachi.soft.sdp.adaptor.callback.io.dashboard.Da
shboardOutputCBImpl" name="outputer">
        <!-- Dashboard output connector definition -->

```

```

    <docon:DashboardOutputConnectorDefinition
Record="RECORD2" >
    <docon:RecordHoldTime DateReference="LAST_UPDATE"
RecordTime="0" DateFieldPosition="3" />
    <docon:DataProcessingDefinition Name="HistoryRecorder" >
        <docon:HistoryRecorder/>
    </docon:DataProcessingDefinition>
    </docon:DashboardOutputConnectorDefinition>
</cb:OutputCBDefinition>

</adp:OutputAdaptorDefinition>

```

## (2) Details of the example

This example defines that output adaptor `OutputAdaptor1` displays stream data summary analysis results on a dashboard.

This output adaptor performs the processing described in the table below (where parentheses enclose a definition name in the adaptor configuration definition file).

Type of callback	Callback processing
Callback for receiving (CB definition for receiving)	Tuple reception (output stream definition)
Callback for editing (CB definition for editing)	Mapping between record and stream (mapping definition)
	Mapping between records (mapping definition)
	Format conversion (format conversion definition)
Callback for sending (CB definition for sending)	Dashboard output (dashboard output connector definition)

The details of each definition for `OutputAdaptor1` are provided below.

- Details of the output stream definition

Receives tuples from the output stream `QUERY` of the query group `Inprocess_QueryGroupTest` and then converts them to common records.

- Details of the mapping definition

Performs mapping between record and stream in order to associate the common records corresponding to the output stream format with the common records input during format conversion.

Also performs mapping between records in order to obtain a tuple's time (by specifying `getTupleTime` in the `function` attribute in the `map` tag) so that the records containing the most recent results of `rstream` can be identified, and then outputs the data to the common records.

- Details of the format conversion definition

Converts the common records output during mapping to output records.

- Details of the dashboard output connector definition

Specifies 0 as the record retention period (`RecordTime` attribute in the `RecordHoldTime` tag) for the output records obtained during format conversion so that all records are deleted except those containing the most recent results.

- Stream name specified in the adaptor configuration definition file

The stream name specified in the adaptor configuration definition file must match the name specified in the query definition file.

- For input stream

The stream name `STREAM` specified in the `register stream` clause in the query definition file becomes the input stream name that is specified in the adaptor configuration definition file.

- For output stream

The query name `QUERY` specified in the `register query` clause in the query definition file becomes the output stream name that is specified in the adaptor configuration definition file.

The query definition file used in this example is shown below. This example obtains the maximum value for the send/receive time (`subtime`) over the past one minute for each receiving IP address (`sendip`) and then outputs the result by `rstream`.

```
register stream STREAM(sendip VARCHAR(15),receiveip VARCHAR(15),sendport
INT,receiveport INT,uri VARCHAR(255),subtime BIGINT,times TIMESTAMP(9));

register query QUERY rstream(
  select STREAM.sendip as sendip, max(STREAM.subtime) as subtime
  from STREAM[RANGE 1 MINUTE]
  group by STREAM.sendip);
```

For details about the query definition file, see the manual *uCosminexus Stream Data Platform - Application Framework Application Development Guide*.

### 11.4.2 Output adaptor definition (displaying data including historical data)

To display on a chart (such as a broken line graph) results that include historical data from a point in the past to the current time, you specify definitions in such a manner that the results of `rstream` that cover a period of  $n$  seconds from the current time are displayed.



In this example, you specify  $n$  as the record retention period (`RecordTime` attribute in the `RecordHoldTime` tag) in the dashboard output connector definition in the output adaptor definition, and you define the field for storing the results of `getTupleTime` in such a manner that it holds the results obtained from  $n$  seconds in the past through the current time.

For another example of an output adaptor definition and details of the definition, see *11.4.1 Output adaptor definition (displaying the most recent data)*.



---

# Appendix

---

## A. Reference Material for This Manual

---

## A. Reference Material for This Manual

---

This appendix provides reference information, including various conventions, for this manual.

### A.1 Related publications

This manual is part of a related set of manuals. The manuals in the set are listed below (with the manual numbers):

- *uCosminexus Stream Data Platform - Application Framework Description* (3020-3-V01(E))

This manual provides an overview and a basic understanding of Stream Data Platform - AF.

We recommend reading this manual to learn about the features and system configurations of Stream Data Platform - AF, and to acquire the basic knowledge needed to set up and operate a system.

- *uCosminexus Stream Data Platform - Application Framework Application Development Guide* (3020-3-V03(E))

This manual explains how to code CQL for use in analyzing data with Stream Data Platform - AF, and how to create custom adaptors.

We recommend reading this manual to learn about writing CQL code for achieving analysis objectives, and to learn about using APIs to create custom adaptors.

- *uCosminexus Stream Data Platform - Application Framework Messages* (3020-3-V04(E))

This manual explains the messages output by Stream Data Platform - AF.

We recommend referring to this manual if necessary when a message is output.

### A.2 Conventions: Abbreviations for product names

This manual uses the following abbreviations for product names and Java-related terms:

Abbreviation	Full name or meaning
jar	Java™ Archive
Java	Java™
JavaVM	Java Virtual Machine

Abbreviation	Full name or meaning
JDK	Java Development Kit
Stream Data Platform - AF	uCosminexus Stream Data Platform - Application Framework

### A.3 Conventions: Acronyms

This manual also uses the following acronyms:

Acronym	Full name or meaning
AP	application program
API	application programming interface
CB	callback
CPU	central processing unit
CQL	Continuous Query Language
GMT	Greenwich Mean Time
HTTP	Hyper Text Transfer Protocol
IEEE	Institute of Electrical and Electronic Engineers
I/O	input/output
IP	Internet Protocol
OS	operating system
RMI	Remote Method Invocation
TCP	Transmission Control Protocol
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
XML	Extensible Markup Language

### A.4 Conventions: KB, MB, GB, and TB

This manual uses the following conventions:

- 1 KB (kilobyte) is 1,024 bytes.
- 1 MB (megabyte) is 1,024<sup>2</sup> bytes.

A. Reference Material for This Manual

- 1 GB (gigabyte) is  $1,024^3$  bytes.
- 1 TB (terabyte) is  $1,024^4$  bytes.

---

# Index

---

## Symbols

%SDP\_INSTALL\_DIR% 51

## A

abbreviations for products 446

acronyms 447

adaptor 11

    evaluating, to be used 23

    starting (custom adaptor) 74

    starting (standard adaptor) 73

    status change 74

    terminating (custom adaptor) 83

    terminating (standard adaptor) 82

adaptor command definition file 253

    namespace URI for 253

adaptor configuration definition file 254

    coding example for 346

    list of definitions in 255

    namespace 258

    structure of 255

adaptor configuration, evaluating 23

adaptor definition 264

adaptor definition file 14, 249

    creating 60

    format of explanation 250

    notes about creating 251

    whether or not necessary to create 60

adaptor definition files, list of 252

adaptor group 16

    checking status of 75

    checking whether active or inactive 75

    configuration of 16

adaptor group definition 262

adaptor trace definition 260

adaptor trace information 103

    details of 111

AdaptorCompositionDefinition.xml 254

AdaptorTraceDefinition tag 261

administrator user 55

    registering 55

AgentManagerDefinition.xml 253

API trace information, details of 110

## B

batch processing mode 370

built-in function 324

## C

callback 25

CB definition 268

    for editing 271

    for input 268

    for output 270

    for receiving 274

    for sending 273

    order of 257

child element 255

class library trace function option 245

command tag, information to be specified when

    WinDump is used 287

command, format of explanation 134

commands, list of 135

common definition 260

common record 369

CommonDefinition tag 260

condition between records 388

conditional expression, coding rules for 224

connection mode

    between adaptor and SDP server 14

    between standard adaptor and SDP server 17

conventions

    abbreviations for products 446

    acronyms 447

    diagrams iii

    fonts and symbols i

    KB, MB, GB, and TB 447

- version numbers iii
- CQL data type 310
- custom adaptor 11
  - creating 47
  - evaluation item when using 24
  - starting 74
  - terminating 83
- D**
- dashboard
  - displaying analysis results on 70, 87
  - modifying 95
  - setting up 62
- dashboard output 29, 400
  - example definition for 439
- dashboard output connector definition 296
- Dashboard Server
  - canceling registration of 95
  - internal settings file 64, 421
  - registering 62
  - setting up 62
  - shutting down 88
  - starting 87
- Dashboard Viewer
  - changing window 95
  - configuration of window 423
  - displaying analysis results using 87
  - editing window 65
  - setting up 64
  - URL of 65
  - window layout file 423
- dashboard-display data 400
  - output settings for 62
- dashboard.war 64
- DashboardOutputConnectorDefinition tag 297
- data
  - evaluating editing methods 26
  - evaluating input methods 26
  - evaluating output methods 29
  - timing of editing 28
- data reception application 11
- data source mode 21
- data transmission application 11
- data transmission, checking result of 76

- data type, rule for representing 312
- DataEditCBDefinition tag 271
- definition file, details about definition in 365
- deleting records
  - acquired records 404
  - based on record retention period 403
  - based on the maximum number of records to be retained 404
- detailed time output option 244
- diagram conventions iii
- directory structure 49

**E**

- engine.initialQueueSize 196
- engine.maxQueueSize 197
- engine.watchQueueSize.threshold 197
- environment variable, specifying 63
- error handling, data to be collected in event of error 99
- error report file 101
- extended function option in event of OutOfMemoryError 245
- extended thread dump function option 243
- extended verbosegc function option 244
- extraction condition 387
  - evaluating 387
- extraction target condition 386
  - evaluating 386

**F**

- field condition 382, 387
  - evaluating 383
- field name, identifier that can be specified as 288
- file input 26, 367
- file input connector 369
- file input connector definition 276
- file output 29, 394
- file output connector 399
- file output connector definition 292
- FileInputConnectorDefinition tag 277
- FileOutputConnectorDefinition tag 292
- filter definition 330
- FilterDefinition tag 330
- font conventions i



format conversion 27, 371, 398  
 format conversion definition 301  
 FormatDefinition tag 302

## G

garbage collection interval 185, 190  
 GB meaning 447

## H

Hitachi JavaVM log file option 243  
 Hitachi trace common library, directory for 49  
 Hitachi Web Server error log file 105  
 HTTP packet input 26, 376  
 HTTP packet input connector 379  
 HTTP packet input connector definition 282  
 HttpPacketInputConnectorDefinition tag 283  
 httpsd.conf 63

## I

in-process connection 14  
 in-process connection property file 239  
 in-process group 17  
   configuration of 17  
 in-process group definition 262  
 in-process-connection adaptor  
   starting 155  
   terminating 161  
 InprocessGroupDefinition tag 262  
 input adaptor 11  
 input adaptor definition 264  
 input file, specifying sequence number for 280  
 input record 369  
 input stream definition 344  
 input stream queue 12  
 InputAdaptorDefinition tag 264  
 InputCBDefinition tag 269  
 installation directory, structure of 49

## J

Java data type 310  
   that is stored as field value 291  
 JavaVM memory space  
   option for specifying ratio of 242

  option for specifying size 242  
 JavaVM options file  
   for RMI connection 187  
   for SDP server 182  
 JavaVM options, list of 242  
 jvm\_client\_options.cfg 187  
 jvm\_options.cfg 182

## K

KB meaning 447

## L

length of time to be adjusted 406  
 local variable information output function option 246  
 log file 102  
   details of 106  
   size of 106  
 log file output property file 241  
 logger.console.abnormal.enabled 197  
 logger.console.enabled 197  
 logger.properties 241  
 logger.serverMessageFileCount 241  
 logger.serverMessageMaxFileSize 241  
 logger.serverTraceFileCount 241  
 logger.serverTraceMaxFileSize 241

## M

manual, organization of 5  
 mapping 27, 373  
   between record and stream 373, 396  
   between records 373, 396  
 mapping definition 318  
 MappingDefinition tag 318  
 MB meaning 447  
 memory requirement, estimating 33  
 memory requirements, estimating  
   for standard adaptors 38  
   for stream data processing engine 34  
 message log 106  
 mod\_jk.conf 64  
 mon.process\_exp\_time 198

**N**

namespace URI 258  
 naming rule for output file 295

**O**

operating environment, setting up 55, 148  
 OS statistics 103  
 OS status 103  
 output adaptor 11  
 output adaptor definition 266  
   example (displaying data including historical data) 442  
   example (displaying the most recent data) 439  
 output file, naming rule for 295  
 output record 396  
 output stream definition 345  
 output stream queue 12  
 OutputAdaptorDefinition tag 266  
 OutputCBDefinition tag 270

**P**

parent element 255  
 Pcap format 376  
 prerequisite program, preparing 46  
 PRF daemon 87, 88  
 PRFSPOOL 63  
 property file, changing settings in 92  
 protocol analysis 379

**Q**

query 12  
   re-executing 78  
   using 70, 78  
 query definition file 13  
   changing 93  
   creating 59  
 query group 12  
   deleting 138  
   displaying status of 81, 143  
   forced termination of 84  
   modifying 92  
   normal termination of 83

  registering 72, 136  
   starting 73, 139  
   terminating 83, 141  
   using 70, 78

query group property file 209  
 query re-execution 78  
   how to perform 80  
   permitted range of 79  
 query.decimalMaxPrecision 198  
 query.decimalRoundingMode 198  
 querygroup.cqlFilePath 214  
 querygroup.sleepOnOverStore 199, 215  
 querygroup.sleepOnOverStoreRetryCount 199, 215

**R**

real-time processing mode 370  
 ReceiveCBDefinition tag 274  
 record condition 382, 387  
   evaluating 382  
 record extraction 27, 384  
 record extraction definition 334  
 record filtering 27, 381  
 record structure, format of 309  
 RecordExtractionDefinition tag 335  
 reference time 406  
 REFRESH\_INTERVAL 421  
 relation 84  
   discarding 84  
 retrieved record 384  
   generating 392  
 RETRY\_NUM 421  
 RMI connection 14  
 RMI group 17  
   configuration of 18  
 RMI group definition 263  
 RMI-connection adaptor  
   starting 152  
   terminating 159  
 rmi.serverPort 200  
 RMIGroupDefinition tag 263

**S**

SDP server 12  
   evaluating number of 21

- evaluating time control method of 21
  - forced termination of 86
  - normal termination of 86
  - shutting down 85
  - starting 72, 150
  - stopping 157
- SDP server definition file 13, 177
  - creating 56
  - format of explanation 178
  - notes about creating 179
  - whether or not necessary to create 56
- SDP server definition files, list of 180
- SDP\_CLASS\_PATH 183, 188
- SDP\_CLASSLIB\_TRACE 183, 188
- SDP\_CLASSLIB\_TRACE\_LINESIZE 183, 188
- SDP\_GC 183, 188
- SDP\_GC\_PRINT\_CAUSE 183, 188
- SDP\_INITIAL\_MEM\_SIZE 183, 188
- SDP\_JVM\_LOG 183, 188
- SDP\_JVM\_LOG\_FILE\_SIZE 183, 188
- SDP\_LOCALS\_IN\_STACK\_TRACE 183, 189
- SDP\_LOCALS\_SIMPLE\_FORMAT 184, 189
- SDP\_MAX\_MEM\_SIZE 184, 189
- SDP\_MAX\_PERM\_SIZE 184, 189
- SDP\_NEW\_RATIO 184, 189
- SDP\_OOM\_STACK\_TRACE 184, 189
- SDP\_OUTPUT\_MILLI\_TIME 184, 189
- SDP\_PERM\_SIZE 184, 189
- SDP\_SYS\_OPT 184
- SDP\_THRD\_DUMP 184, 189
- SDP\_TRUE\_TYPE\_IN\_LOCALS 184, 189
- SDP\_USER\_OPT 185, 190
- sdpcql 136
- sdpcqldel 138
- sdpcqlstart 139
- sdpcqlstop 141
- sdpls 143
- sdpsetup 148
- sdpstart 150
- sdpstartap 152
- sdpstartinpro 155
- sdpstop 157
- sdpstopap 159
- sdpstopinpro 161
- sdptpls 163
- sdptplput 169
- sdptreed 173
- SendCBDefinition tag 273
- server mode 21
- standard adaptor 11
  - evaluating processing to be performed by 25
  - evaluation item when using 23
  - flow of processing performed by 25
  - relationship with stream queue (stream) 19
  - starting 73
  - terminating 82
- standard error output 101
- standard output 101
- Stream Data Platform - AF
  - flow from introduction to operation of 2
  - installing 46
- stream data processing engine 11
  - evaluating 21
  - modifying 91
- stream data processing system
  - components of 11
  - number of components that can be placed in 15
  - program configuration in 10
- stream data, summary analysis scenario for 12
- stream property file 227
- stream.filterCondition 216, 231
- stream.filterMode 216, 232
- stream.freeInputQueueSizeThreshold 200, 217, 232
- stream.freeInputQueueSizeThresholdOutputMessage 201, 217, 233
- stream.maxKeepTupleCount 201, 218, 233
- stream.propertyFiles 218
- stream.streamName 234
- stream.timestampAccuracy 202, 219, 234
- stream.timestampMode 203, 220
- stream.timestampPosition 203, 220, 235
- stream.tupleLogMode 204, 221
- streamInfo tag 344, 345
- summary analysis scenario
  - evaluating 30
  - for stream data 12
- symbol conventions i

## Index

### system

- design of 7
  - modifying 89
  - problem during operation 127
  - shutting down 71, 82
  - starting 70
- system configuration 45
- evaluating 10
  - flow of 46
- system configuration property file 192
- system operation 67
- flow of 68
- system\_config.properties 192

## T

- TB meaning 447
- thread dump 103
- details of 122
- time control method, evaluating SDP server's 21
- timeout record 390
- timestamp adjustment
- for tuple 406
  - scope of 406
  - setting 417
- timestamp adjustment function 406
- operation when query group shuts down 416
  - processing after tuples have been input 415
  - selecting tuples by filtering 417
- TIMESTAMP data type, specifying start year and month for 307
- tpl.backupFileCount 204, 221, 235
- tpl.bufferCount 204, 221, 236
- tpl.bufferSize 205, 222, 236
- tpl.fileCount 205, 222, 236
- tpl.fileSize 205, 222, 236
- tpl.outputLevel 205, 223, 237
- tpl.outputTrigger 206, 223, 237
- tpl.useOverwrite 206, 223, 238
- trace file 102
- trace information, editing 173
- trace log 106
- trc.api.bufferCount 207
- trc.api.bufferSize 207
- trc.api.fileCount 207

- trc.api.fileSize 207
- trc.api.ioBufferSize 207
- trc.api.outputTrigger 207
- trc.api.useOverwrite 207
- troubleshooting 97
- tuple
- evaluating adjustment of timestamp in 21
  - how to adjust time 408
  - reloading 169
  - retention period 416
  - size of one 34
  - timestamp adjustment for 406
- tuple information, displaying 163
- tuple log 103
- details of 118
- tuple reception 396
- tuple transmission 374

## U

- user\_app.adaptor-group-name-or-adaptor-name.properties 239
- user\_app.classname 240
- user\_app.classpath\_dir 240
- usrconf.cfg 64
- usrconf.properties 64, 421

## V

- version number conventions iii

## W

- Web container server, setting up 62
- Web server, setting up 62
- workers.properties 64
- working directory 15, 52, 55
- creating 55
  - evaluating configuration for executing query groups 31
  - structure of 52